Database Management Systems

Module 11: SQL Examples

Partha Pratim Das

Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

ppd@cse.iitkgp.ac.in

Week Recap

- Basic notions of Relational Database Models
  - Attributes and their types
  - Mathematical structure of relational model
  - Schema and Instance
  - Keys, primary as well as foreign
- Relational algebra with operators
- Relational query language
  - DDL (Data Definition)
  - DML (Basic Query Structure)
- Detailed understanding of basic query structure
- Set operations, null values, and aggregation

Module Objectives

- To recap various basic SQL features through example workout

Module Outline

- Examples of basic SQL

Select distinct

- From the *classroom* relation in the figure, find the names of buildings in which every individual classroom has capacity less than 100 (removing the duplicates).

building	*room_number*	*capacity*
Packard	101	500
Painter	514	10
Taylor	3128	70
Watson	100	30
Watson	120	50

Figure: *classroom* relation

- Query:

  select distinct *building*
  from *classroom*
  where *capacity* < 100;

- Output :

building
Painter
Taylor
Watson

Select distinct

- From the *classroom* relation in the figure, find the names of buildings in which every individual classroom has capacity less than 100 (removing the duplicates).

building	*room_number*	*capacity*
Packard	101	500
Painter	514	10
Taylor	3128	70
Watson	100	30
Watson	120	50

Figure: *classroom* relation

- Query:

  select distinct *building*
  from *classroom*
  where *capacity* < 100;

- Output :

```
postgres=# select distinct building
from classroom
where capacity < 100;
 building
----------
 Brown
 Power
 Taylor
 Polya
 Grace
 Fairchild
 Nassau
 Bronfman
 Whitman
 Alumni
 Gates
 Lamberton
 Main
 Saucon
 Chandler
 Garfield
 Rathbone
 Lambeau
 Painter
(19 rows)
```

building
Painter
Taylor
Watson

- From the *classroom* relation in the figure, find the names of buildings in which every individual classroom has capacity less than 100 (without removing the duplicates).

building	*room_number*	*capacity*
Packard	101	500
Painter	514	10
Taylor	3128	70
Watson	100	30
Watson	120	50

Figure: *classroom* relation

  - Query:

    select all *building*
    from *classroom*
    where *capacity* < 100;

  - Outpu

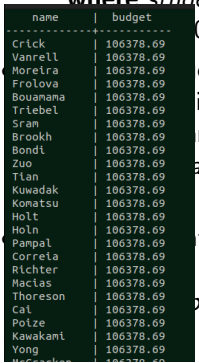
```
postgres=# select all  building from classroom where capacity < 100;
 building
----------
 Lamberton
 Chandler
 Fairchild
 Nassau
 Grace
 Lamberton
 Painter
 Alumni
 Alumni
 Drown
 Saucon
 Whitman
 Saucon
 Bronfman
 Polya
 Gates
 Gates
 Main
 Taylor
 Power
 Garfield
 Rathbone
 Power
 Main
 Lambeau
 Chandler
(26 rows)
```

- Note that duplicate retention is the default an to skip *all* immediately after *select*.

Cartesian Product

- Find the list of all students of departments which have a budget $< \$0.1 million$

  select *name, budget*
  from *student, department*
  where *student.dept_name* = *department.dept_name* **and** 00000;



name	budget
Crick	106378.69
Vanrell	106378.69
Moreira	106378.69
Frolova	106378.69
Bouamama	106378.69
Triebel	106378.69
Sram	106378.69
Brookh	106378.69
Bondi	106378.69
Zuo	106378.69
Tian	106378.69
Kuwadak	106378.69
Komatsu	106378.69
Holt	106378.69
Holn	106378.69
Pampal	106378.69
Correia	106378.69
Richter	106378.69
Macias	106378.69
Thoreson	106378.69
Cai	106378.69
Poize	106378.69
Kawakami	106378.69
Yong	106378.69

ery first generates every possible student-ir, which is the Cartesian product of stu-rtment. Then, it filters all the rows with *ame* = *department.dept_name* **and** *budget* <

ttribute *dept_name* in the resulting table are the rela *pt_name*)

name	budget
Brandt	50000.00
Peltier	70000.00
Levy	70000.00
Sanchez	80000.00
Snow	70000.00
Aoi	85000.00
Bourikas	85000.00
Tanaka	90000.00

```
postgres=#
postgres=# select name, budget from student, department
where student.dept_name = department.dept_name and
budget < 200000;
postgres=#
```

Rename AS Operation

- The same query in the previous slide can be framed by renaming the tables as shown below.

  select *S.name* **as** *studentname*, *budget* **as** *deptbudget*
  from *student* **as** *S*, *department* **as** *D*
  where *S.dept_name* = *D.dept_name* **and** *budget* < 100000;

- The above query renames the relation *student* **as** *S* and the relation *department* **as** *D*
- It also displays the attribute *name* as StudentName and *budget* as DeptBudget.
- Note that the budget attribute does not have any prefix because it occurs only in the department relation.

```
postgres=# select name as studentname, budget as dept_budget from student, d
epartment
where student.dept_name = department.dept_name and
```

studentname	*deptbudget*
Brandt	50000.00
Peltier	70000.00
Levy	70000.00
Sanchez	80000.00
Snow	70000.00
Aoi	
Bour	
Tana	

```
studentname | dept_budget
------------+-------------
Crick       |   106378.69
Vanrell     |   106378.69
Moreira     |   106378.69
Frolova     |   106378.69
Bouamama    |   106378.69
Triebel     |   106378.69
Sram        |   106378.69
Brookh      |   106378.69
Bondi       |   106378.69
Zuo         |   106378.69
Tian        |   106378.69
```

- From the *instructor* and *department* relations in the figure, find out the names of all instructors whose department is Finance or whose department is in any of the following buildings: Watson, Taylor.

instructor

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

department

dept_name	*building*	*budget*
Biology	Watson	90000
Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

- Query:

```
select name
from instructor I, department D
where D.dept_name = I.dept_name
    and (I.dept_name = 'Finance'
        or building in ('Watson','Taylor'));
```

- Output:

name
Srinivasan
Wu
Einstein
Gold
Katz
Singh
Crick
Brandt
Kim

Where: AND and OR


```
postgres=# select name from instructor as  I, department as D
where D.dept_name = I.dept_name
and (I.dept_name = 'Finance'
or building in ('Watson','Taylor'));
   name
-----------
 Pingr
 Arias
 Mingoz
 Choll
 Arinb
 Gutierrez
 Romero
 Atanassov
(8 rows)
```

Comp. Sci.	Taylor	100000
Elec. Eng.	Taylor	85000
Finance	Painter	120000
History	Painter	50000
Music	Packard	80000
Physics	Watson	70000

Singh
Crick
Brandt
Kim

- From the *course* relation in the figure, find the titles of all courses whose *course_id* has three alphabets indicating the department.

course_id	*title*	*dept_name*	*credits*
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

Figure: *course* relation

  - Query:

    select *title*
    from *course*
    where *course_id* **like** '___-%';

  - Output:

title
Intro. to Biology
Genetics
Computational Biology
Investment Banking
World History
Physical Principles

- The *course_id* of each department has either 2 or 3 alphabets in the beginning, followed by a hyphen and then followed by a 3-digit number. The above query returns the names of those departments that have 3 alphabets in the beginning.

```
postgres=# select title, course_id  from course where course_id like '___%'
imit 10;
         title          | course_id
------------------------+-----------
 C  Programming         | 787
 The Music of Donovan   | 238
 Electron Microscopy    | 608
 International Finance  | 539
 Greek Tragedy          | 278
 Greek Tragedy          | 972
 Virology               | 391
 Compiler Design        | 814
 Geology                | 272
 Mobile Computing       | 612
(10 rows)
```

Figure: course relation

Physical Principles

- The *course_id* of each department has either 2 or 3 alphabets in the beginning, followed by a hyphen and then followed by a 3-digit number. The above query returns the names of those departments that have 3 alphabets in the beginning.

Order By

- From the *student* relation in the figure, obtain the list of all students in alphabetic order of departments and within each department, in decreasing order of total credits.

ID	*name*	*dept_name*	*tot_cred*
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: *student* relation

  - The list is first sorted in alphabetic order of dept name.
  - Within each dept, it is sorted in decreasing order of total credits.

  - Query:

    select *name, dept_name, tot_cred*
    from *student*
    order by *dept_name* **ASC**, *tot_cred* **DESC**;

  - Output:

name	*dept_name*	*tot_cred*
Tanaka	Biology	120
Zhang	Comp. Sci.	102
Brown	Comp. Sci.	58
Williams	Comp. Sci.	54
Shankar	Comp. Sci.	32
Bourikas	Elec. Eng.	98
Aoi	Elec. Eng.	60
Chavez	Finance	110
Brandt	History	80
Sanchez	Music	38
Peltier	Physics	56
Levy	Physics	46
Snow	Physics	0

```
postgres=# select name, dept_name , tot_cred from student order by dept_name
 ASC , tot_cred DESC limit 30;
    name     | dept_name  | tot_cred
-------------+------------+----------
 Sauer       | Accounting |      128
 Kothari     | Accounting |      125
 Greenbaum   | Accounting |      124
 Lepp        | Accounting |      121
 Shishkin    | Accounting |      120
 Bricker     | Accounting |      116
 Sin         | Accounting |      115
 Giralt      | Accounting |      114
 Holloway    | Accounting |      113
 Fok         | Accounting |      113
 Coddington  | Accounting |      110
 Patne       | Accounting |      105
 Philippe    | Accounting |      105
 Bertranp    | Accounting |      105
 Afim        | Accounting |      100
 Jr          | Accounting |      100
 Sarnak      | Accounting |      100
 Wrzesz      | Accounting |       99
 Kereth      | Accounting |       96
 Ashmi       | Accounting |       95
 Marlet      | Accounting |       91
 Pottos      | Accounting |       90
 Heilprin    | Accounting |       88
 McDonald    | Accounting |       87
 Sudirm      | Accounting |       84
 Nagal       | Accounting |       83
 Akaiw       | Accounting |       82
 Chien       | Accounting |       81
 Baer        | Accounting |       81
 Lanfr       | Accounting |       78
(30 rows)
```


f all students in alphabetic order of departments
l credits.

name, dept_name, tot_cred
student
by *dept_name* **ASC**, *tot_cred* **DESC**;

name	*dept_name*	*tot_cred*
Tanaka	Biology	120
Zhang	Comp. Sci.	102
Brown	Comp. Sci.	58
Williams	Comp. Sci.	54
Shankar	Comp. Sci.	32
Bourikas	Elec. Eng.	98
Aoi	Elec. Eng.	60
Chavez	Finance	110
Brandt	History	80
Sanchez	Music	38
Peltier	Physics	56
Levy	Physics	46
Snow	Physics	0

- From the *teaches* relation in the figure, find the IDs of all courses taught in the Fall or Spring of 2018.

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-351	1	Spring	2018
45565	CS-101	1	Spring	2018
45565	CS-319	1	Spring	2018
76766	BIO-101	1	Summer	2017
76766	BIO-301	1	Summer	2018
83821	CS-190	1	Spring	2017
83821	CS-190	2	Spring	2017
83821	CS-319	2	Spring	2018
98345	EE-181	1	Spring	2017

Figure: *teaches* relation

Note: We can use **distinct** to remove duplicates.

- Query:

  select *course_id*
  from *teaches*
  where *semester* **in** *('Fall','Spring')*
  and *year*=2018;

- Output:

course_id
CS-315
FIN-201
MU-199
HIS-351
CS-101
CS-319
CS-319

.

```
postgres=# select course_id from teaches where semester in ('Fall', 'Spring'
) and year = 2011;
 course_id
-----------
(0 rows)

postgres=# select * from teaches ;
postgres=# select Distinct year  from teaches ;
 year
------
 2001
 2005
 2003
 2010
 2002
 2008
 2009
 2006
 2007
 2004
(10 rows)

postgres=# select course_id from teaches where semester in ('Fall', 'Spring'
) and year = 2010;
 course_id
-----------
 270
 493
 679
 415
 867
 443
 843
 735
 476
 313
 692
(11 rows)
```

nd the IDs of all courses taught in the Fall or

uery:

select *course_id*
from *teaches*
where *semester* **in** *('Fall','Spring')*
and *year*=2018;

utput:

course_id
CS-315
FIN-201
MU-199
HIS-351
CS-101
CS-319
CS-319

Set Operations: union

- For the same question in the previous slide, we can find the solution using **union** operator as follows.

ID	*course_id*	*sec_id*	*semester*	*year*
10101	CS-101	1	Fall	2017
10101	CS-315	1	Spring	2018
10101	CS-347	1	Fall	2017
12121	FIN-201	1	Spring	2018
15151	MU-199	1	Spring	2018
22222	PHY-101	1	Fall	2017
32343	HIS-351	1	Spring	2018
45565	CS-101	1	Spring	2018
45565	CS-319	1	Spring	2018
76766	BIO-101	1	Summer	2017
76766	BIO-301	1	Summer	2018
83821	CS-190	1	Spring	2017
83821	CS-190	2	Spring	2017
83821	CS-319	2	Spring	2018
98345	EE-181	1	Spring	2017

Figure: *teaches* relation

- Query:

```
select course_id
from teaches
where semester='Fall'
    and year=2018
union
select course_id
from teaches
where semester='Spring'
    and year=2018
```

- Output:

course_id
CS-101
CS-315
CS-319
FIN-201
HIS-351
MU-199

- Note that **union** removes all duplicates. If we use **union all** instead of **union**, we get the same set of tuples as in previous slide.

```
postgres=# select course_id from teaches where semester='Fall' and year =2006

union
select course_id from teaches where semester='Spring'and year =2004;
 course_id
-----------
 362
 867
 376
 795
 959
 561
 760
 747
 960
 571
 626
 239
(12 rows)
```

in previous slide.

FIN-201
HIS-351
MU-199

Set Operations (2): intersect

- From the *instructor* relation in the figure, find the names of all instructors who taught in either the Computer Science department or the Finance department and whose salary is < 80000.

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

Figure: *instructor* relation

  - Query:

```
select name
from instructor
where dept_name in ('Comp. Sci.','Finance')
intersect
select name
from instructor
where salary < 80000;
```

  - Output:

name
Srinivasan
Katz

- Note that the same can be achieved using the query:
select *name* **from** *instructor* **where** *dept_name* **in**('Comp. Sci.', 'Finance') **and** *salary* < 80000;

```
postgres=# select name from instructor where salary < 80000;
      name
------------------
 Pingr
 Murata
 Konstantinides
 Queiroz
 Bertolino
 Hau
 Soisalon-Soininen
 Moreira
 Lembr
 Choll
 Valtchev
 Arinb
 Bawa
 Vicentino
 Dusserre
 Desyl
 DAgostino
 Ullman
 Morris
 Yin
 Pimenta
 Gutierrez
 Romero
 Kean
 Tung
(25 rows)

postgres=# select name from instructor where dept_name in ('Comp. Sci.','Fina
nce');
   name
----------
 Mingoz
 Bondi
 Bourrier
(3 rows)
```

```
(5 rows)

postgres=# select Distinct dept_name from instructor;
  dept_name
-------------
 Accounting
 Pol. Sci.
 Marketing
 Finance
 English
 Geology
 Cybernetics
 Physics
 Athletics
 Statistics
 Astronomy
 Languages
 Comp. Sci.
 Psychology
 Biology
 Elec. Eng.
 Mech. Eng.
(17 rows)

postgres=# select name from instructor where dept_name in ('Comp. Sci.','Fina
nce')
intersect
select name from instructor where salary < 80000;
 name
------
(0 rows)
```

Set Operations (3): except

- From the *instructor* relation in the figure, find the names of all instructors who taught in either the Computer Science department or the Finance department and whose salary is either ≥ 90000 or ≤ 70000.

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

Figure: *instructor* relation

  - Note that the same can be achieved using the query given below:

```
select name from instructor
where dept_name in('Comp. Sci.', 'Finance')
and (salary >= 90000 or salary <= 70000);
```

  - Query:

```
select name
from instructor
where dept_name in ('Comp. Sci.','Finance')
except
select name
from instructor
where salary < 90000 and salary > 70000;
```

  - Output:

name
Srinivasan
Brandt
Wu

Aggregate functions: avg

- From the *classroom* relation given in the figure, find the names and the average capacity of each building whose average capacity is greater than 25.

building	*room_number*	*capacity*
Packard	101	500
Painter	514	10
Taylor	3128	70
Watson	100	30
Watson	120	50

Figure: *classroom* relation

- Query:

  select *building*, **avg** (*capacity*)
  from *classroom*
  group by *building*
  having avg (*capacity*) > 25;

- Output:

building	*avg*
Taylor	70.00
Packard	500.00
Watson	40.00

```
postgres=# select building, avg (capacity )from classroom group by building h
aving avg (capacity ) > 25;
 building  |         avg
-----------+----------------------
 Taylor    |  93.0000000000000000
 Polya     |  28.0000000000000000
 Grace     |  34.0000000000000000
 Fairchild |  27.0000000000000000
 Nassau    |  92.0000000000000000
 Whitman   |  76.0000000000000000
 Alumni    |  36.5000000000000000
 Gates     |  37.5000000000000000
 Stabler   | 113.0000000000000000
 Main      |  26.0000000000000000
 Saucon    |  49.3333333333333333
 Garfield  |  59.0000000000000000
 Rathbone  |  60.0000000000000000
 Lambeau   |  51.0000000000000000
 Painter   |  97.0000000000000000
(15 rows)
```

Aggregate functions (2): min

- From the *instructor* relation given in the figure, find the least salary drawn by any instructor among all the instructors.

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

- Query:

  select min(*salary*) **as** *least_salary*
  from *instructor*;

- Output:

least_salary
40000.00

```
postgres=# select min(salary ) as least_salary from instructor ;
 least_salary
--------------
     32241.56
(1 row)

postgres=#
```

Aggregate functions (3): max

- From the *student* relation given in the figure, find the maximum credits obtained by any student among all the students.

ID	*name*	*dept_name*	*tot_cred*
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: *student*

- Query:

  select max(*tot_cred*) **as** *max_credits*
  from *student*;

- Output:

max_credits
120

```
postgres=# select max(tot_cred) as max_credits
from student;
 max_credits
-------------
         129
(1 row)

postgres=#
```

Aggregate functions (4): count

- From the *section* relation given in the figure, find the number of courses run in each building.

course_id	*sec_id*	*semester*	*year*	*building*	*room_number*	*time_slot_id*
BIO-101	1	Summer	2017	Painter	514	B
BIO-301	1	Summer	2018	Painter	514	A
CS-101	1	Fall	2017	Packard	101	H
CS-101	1	Spring	2018	Packard	101	F
CS-190	1	Spring	2017	Taylor	3128	E
CS-190	2	Spring	2017	Taylor	3128	A
CS-315	1	Spring	2018	Watson	120	D
CS-319	1	Spring	2018	Watson	100	B
CS-319	2	Spring	2018	Taylor	3128	C
CS-347	1	Fall	2017	Taylor	3128	A
EE-181	1	Spring	2017	Taylor	3128	C
FIN-201	1	Spring	2018	Packard	101	B
HIS-351	1	Spring	2018	Painter	514	C
MU-199	1	Spring	2018	Packard	101	D
PHY-101	1	Fall	2017	Watson	100	A

Figure: *section* relation

- Query:

```
select building,
       count(course_id) as course_count
from section
group by building;
```

- Output:

building	*course_count*
Taylor	5
Packard	4
Painter	3
Watson	3

Aggregate functions (4): count


```
                                                    ^
postgres=# select building,count(course_id) as course_count from section grou
p by building ;
 building  | course_count
-----------+--------------
 Main      |            5
 Saucon    |           17
 Lamberton |           13
 Drown     |            3
 Nassau    |            2
 Power     |            8
 Chandler  |            4
 Garfield  |            1
 Bronfman  |            2
 Rathbone  |            1
 Whitman   |            4
 Lambeau   |            3
 Taylor    |           15
 Alumni    |            4
 Polya     |            4
 Fairchild |            7
 Stabler   |            2
 Gates     |            5
(18 rows)

postgres=#
```

ses run in each

as *course_count*

course_count
5
4
3
3

Aggregate functions (5): sum

- From the *course* relation given in the figure, find the total credits offered by each department.

course_id	*title*	*dept_name*	*credits*
BIO-101	Intro. to Biology	Biology	4
BIO-301	Genetics	Biology	4
BIO-399	Computational Biology	Biology	3
CS-101	Intro. to Computer Science	Comp. Sci.	4
CS-190	Game Design	Comp. Sci.	4
CS-315	Robotics	Comp. Sci.	3
CS-319	Image Processing	Comp. Sci.	3
CS-347	Database System Concepts	Comp. Sci.	3
EE-181	Intro. to Digital Systems	Elec. Eng.	3
FIN-201	Investment Banking	Finance	3
HIS-351	World History	History	3
MU-199	Music Video Production	Music	3
PHY-101	Physical Principles	Physics	4

Figure: *course* relation

- Query:

  select *dept_name*,
  sum(*credits*) **as** *sum_credits*
  from *course*
  group by *dept_name*;

- Output:

dept_name	*sum_credits*
Finance	3
History	3
Physics	4
Music	3
Comp. Sci.	17
Biology	11
Elec. Eng.	3

Aggregate functions (5): sum


```
postgres=# select dept_name,sum(credits) as sumcredits from course group by d
ept_name;
  dept_name  | sumcredits
-------------+------------
 Astronomy   |         37
 Civil Eng.  |         34
 Comp. Sci.  |         37
 Athletics   |         30
 History     |         29
 Psychology  |         44
 Accounting  |         40
 Geology     |         33
 Math        |         34
 Pol. Sci.   |         23
 Biology     |         23
 Finance     |         49
 Elec. Eng.  |         28
 Languages   |         37
 Statistics  |         21
 English     |         28
 Cybernetics |         67
 Mech. Eng.  |         40
 Marketing   |         20
 Physics     |         38
(20 rows)
```

s offered by each

me,

s) **as** *sum_credits*

name;

name	*sum_credits*
ce	3
y	3
ts	4
	3
. Sci.	17
y	11
Eng.	3

Module Summary

- SQL Examples have been practiced for
  - Select
  - Cartesian Product / as
  - Where: and / or
  - String Matching
  - Order by
  - in
  - Set Operations: union, intersect, except
  - Aggregate Functions: avg, min, max, count, sum