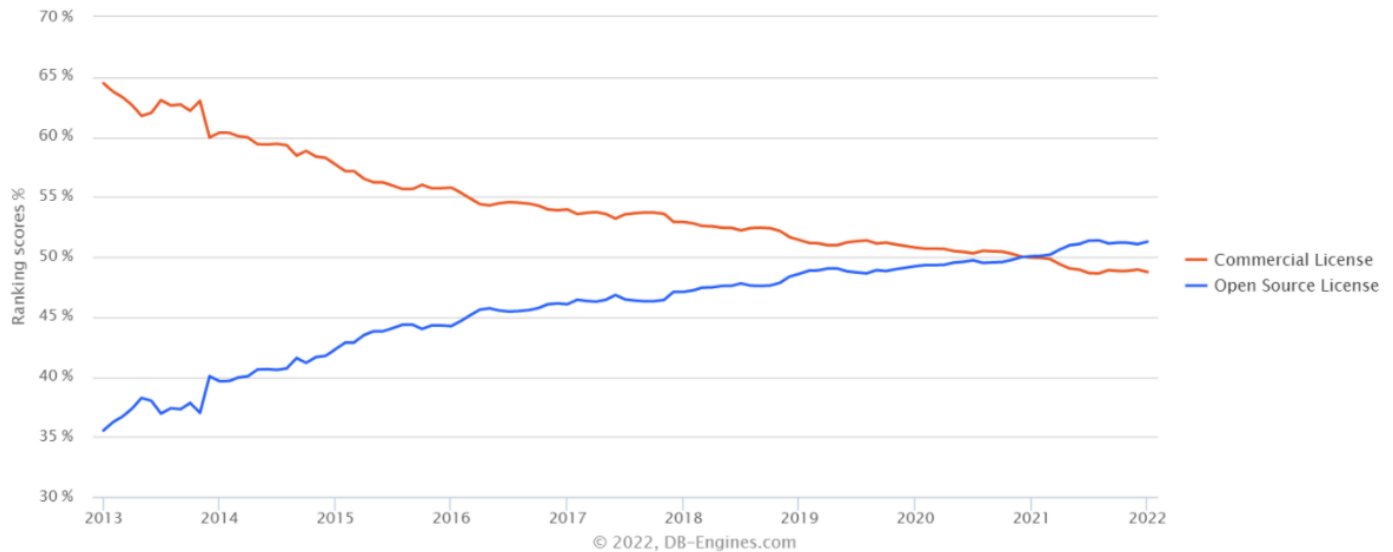


Lecture 03 - Select

Jan 25

Open Source Used More

Popularity trend



Open source databases overtook closed source databases in popularity in early 2021 (DB-engines.com, [link](#))

```
Select [ Projected Columns ]
From Table ...
    join to ...
Where ...
Group By Column List
Having [ where on grouped data ]
Ordr By [ Columns ]
```

Let's use our table from last time

```
1: \c l02
2: create table vote_by_county (
3:     id                serial primary key,
4:     year              int default 2021,
5:     state             text default '--',      -- irritatingly all upper case.
6:     state_uc          text default '--',
7:     state_po          varchar(2) default '--', -- Incorrectly Named Column!
8:     county_name       text default '--',      -- irritatingly all upper case.
9:     county_name_uc    text default '--',
10:    county_fips        int default 0,
11:    office             text default 'unk',
12:    candidate          text default 'unk',
13:    candidate_uc       text default 'unk',
14:    party              text default 'unk',
```

```

15:     candidatevotes int default 0,
16:     totalvotes     int default 0,
17:     version         int,
18:     vote_mode       text
19: );

```

and do some selects with the projected columns.

```

select id, year
  from vote_by_county
;

```

we can rename a column

```

select id, year as "Year of Our Lord"
  from vote_by_county
;

```

we can pick different columns

File: 01.sql

```

1: select id, year, state, county_name as "county"
2:     from vote_by_county
3: ;

```

Output:

id	year	state	county
16	2000	Alabama	Bibb
1992	2000	Georgia	Troup
4932	2000	Michigan	Ingham
5704	2000	Mississippi	Perry
12884	2004	Arkansas	Pope
...			
72592	2020	Wyoming	Sheridan
72593	2020	Wyoming	Sheridan
72594	2020	Wyoming	Sublette
72595	2020	Wyoming	Sublette
72596	2020	Wyoming	Sublette
72597	2020	Wyoming	Sublette
72598	2020	Wyoming	Sweetwater
72599	2020	Wyoming	Sweetwater
72600	2020	Wyoming	Sweetwater
72601	2020	Wyoming	Sweetwater
72602	2020	Wyoming	Teton
72603	2020	Wyoming	Teton
72604	2020	Wyoming	Teton
72605	2020	Wyoming	Teton
72606	2020	Wyoming	Uinta
72607	2020	Wyoming	Uinta

72608		2020		Wyoming		Uinta
72609		2020		Wyoming		Uinta
72610		2020		Wyoming		Washakie
72611		2020		Wyoming		Washakie
72612		2020		Wyoming		Washakie
72613		2020		Wyoming		Washakie
72614		2020		Wyoming		Weston
72615		2020		Wyoming		Weston
72616		2020		Wyoming		Weston
72617		2020		Wyoming		Weston

(72617 rows)

How about sorting the data

File: 02.sql

```

1: select id, year, state, county_name
2:     from vote_by_county
3:     order by county_name, state
4: ;

```

Output:

id		year		state		county_name
----- ----- ----- -----						
28679		2008		South Carolina		Abbeville
38030		2012		South Carolina		Abbeville
38029		2012		South Carolina		Abbeville
38028		2012		South Carolina		Abbeville
28677		2008		South Carolina		Abbeville
9148		2000		South Carolina		Abbeville
9147		2000		South Carolina		Abbeville
9146		2000		South Carolina		Abbeville
9145		2000		South Carolina		Abbeville
19328		2004		South Carolina		Abbeville
19326		2004		South Carolina		Abbeville
19327		2004		South Carolina		Abbeville
28678		2008		South Carolina		Abbeville
...						
66634		2020		Texas		Zavala
66635		2020		Texas		Zavala
48759		2016		Texas		Zavala
66633		2020		Texas		Zavala
47714		2016		South Dakota		Ziebach
69518		2020		South Dakota		Ziebach
47713		2016		South Dakota		Ziebach
47712		2016		South Dakota		Ziebach
69516		2020		South Dakota		Ziebach
69517		2020		South Dakota		Ziebach
38362		2012		South Dakota		Ziebach
38363		2012		South Dakota		Ziebach
29012		2008		South Dakota		Ziebach
9589		2000		South Dakota		Ziebach
29011		2008		South Dakota		Ziebach
29010		2008		South Dakota		Ziebach
19661		2004		South Dakota		Ziebach
19660		2004		South Dakota		Ziebach

```

9590 | 2000 | South Dakota | Ziebach
19659 | 2004 | South Dakota | Ziebach
9591 | 2000 | South Dakota | Ziebach
38361 | 2012 | South Dakota | Ziebach
9592 | 2000 | South Dakota | Ziebach
(72617 rows)

```

you can only sort by the columns that you have in the *projected columns*.

you can use the column position

File: 03.sql

```

1: select id, year, state, county_name
2:     from vote_by_county
3:     order by 4, 3
4: ;

```

Output:

id	year	state	county_name
28679	2008	South Carolina	Abbeville
38030	2012	South Carolina	Abbeville
38029	2012	South Carolina	Abbeville
38028	2012	South Carolina	Abbeville
28677	2008	South Carolina	Abbeville
9148	2000	South Carolina	Abbeville
9147	2000	South Carolina	Abbeville
9146	2000	South Carolina	Abbeville
9145	2000	South Carolina	Abbeville
19328	2004	South Carolina	Abbeville
19326	2004	South Carolina	Abbeville
19327	2004	South Carolina	Abbeville
28678	2008	South Carolina	Abbeville
47379	2016	South Carolina	Abbeville
...			
38362	2012	South Dakota	Ziebach
38363	2012	South Dakota	Ziebach
29012	2008	South Dakota	Ziebach
9589	2000	South Dakota	Ziebach
29011	2008	South Dakota	Ziebach
29010	2008	South Dakota	Ziebach
19661	2004	South Dakota	Ziebach
19660	2004	South Dakota	Ziebach
9590	2000	South Dakota	Ziebach
19659	2004	South Dakota	Ziebach
9591	2000	South Dakota	Ziebach
38361	2012	South Dakota	Ziebach
9592	2000	South Dakota	Ziebach

(72617 rows)

you can ascending or descending sort

File: 04.sql

```

1: select id, year, state, county_name
2:     from vote_by_county
3:     order by 4 desc, 3 asc
4: ;

```

Output:

id	year	state	county_name
9591	2000	South Dakota	Ziebach
19659	2004	South Dakota	Ziebach
19660	2004	South Dakota	Ziebach
19661	2004	South Dakota	Ziebach
38361	2012	South Dakota	Ziebach
38362	2012	South Dakota	Ziebach
38363	2012	South Dakota	Ziebach
29012	2008	South Dakota	Ziebach
29011	2008	South Dakota	Ziebach
9589	2000	South Dakota	Ziebach
29010	2008	South Dakota	Ziebach
9592	2000	South Dakota	Ziebach
9590	2000	South Dakota	Ziebach
47714	2016	South Dakota	Ziebach
47713	2016	South Dakota	Ziebach
47712	2016	South Dakota	Ziebach
69518	2020	South Dakota	Ziebach
69517	2020	South Dakota	Ziebach
69516	2020	South Dakota	Ziebach
48761	2016	Texas	Zavala
66637	2020	Texas	Zavala
66633	2020	Texas	Zavala
...			
9148	2000	South Carolina	Abbeville
38030	2012	South Carolina	Abbeville
38029	2012	South Carolina	Abbeville
38028	2012	South Carolina	Abbeville
28677	2008	South Carolina	Abbeville
28678	2008	South Carolina	Abbeville
28679	2008	South Carolina	Abbeville
19326	2004	South Carolina	Abbeville
19327	2004	South Carolina	Abbeville

(72617 rows)

You can apply functions and operators to the columns. In this case I will add 10 to the year and concatenate, || the state and county.

File: 05.sql

```

1: select id, year + 10 as "x", state||', '||county_name as "Location"
2:     from vote_by_county
3:     order by 3
4: ;

```

Output:

id	x	Location
31169	2022	Alabama, Autauga
31168	2022	Alabama, Autauga
31167	2022	Alabama, Autauga
1	2010	Alabama, Autauga
21818	2018	Alabama, Autauga
21817	2018	Alabama, Autauga
21816	2018	Alabama, Autauga
12465	2014	Alabama, Autauga
12466	2014	Alabama, Autauga
12467	2014	Alabama, Autauga
4	2010	Alabama, Autauga
3	2010	Alabama, Autauga
2	2010	Alabama, Autauga
40519	2026	Alabama, Autauga
50527	2030	Alabama, Autauga
40518	2026	Alabama, Autauga
50525	2030	Alabama, Autauga
...		
49853	2026	Wyoming, Uinta
40502	2022	Wyoming, Uinta
40504	2022	Wyoming, Washakie
72610	2030	Wyoming, Washakie
72611	2030	Wyoming, Washakie
72612	2030	Wyoming, Washakie
72613	2030	Wyoming, Washakie
49856	2026	Wyoming, Washakie
40503	2022	Wyoming, Washakie
49855	2026	Wyoming, Washakie
49854	2026	Wyoming, Washakie
40505	2022	Wyoming, Washakie
12448	2010	Wyoming, Washakie
12447	2010	Wyoming, Washakie
12445	2010	Wyoming, Washakie
31152	2018	Wyoming, Washakie
31153	2018	Wyoming, Washakie
21803	2014	Wyoming, Washakie
21802	2014	Wyoming, Washakie
31154	2018	Wyoming, Washakie
21801	2014	Wyoming, Washakie
12446	2010	Wyoming, Washakie
12452	2010	Wyoming, Weston
12449	2010	Wyoming, Weston
31156	2018	Wyoming, Weston
31155	2018	Wyoming, Weston
12451	2010	Wyoming, Weston
21806	2014	Wyoming, Weston
21805	2014	Wyoming, Weston
21804	2014	Wyoming, Weston
12450	2010	Wyoming, Weston
31157	2018	Wyoming, Weston
49859	2026	Wyoming, Weston
72616	2030	Wyoming, Weston
40507	2022	Wyoming, Weston
49858	2026	Wyoming, Weston
49857	2026	Wyoming, Weston
72615	2030	Wyoming, Weston

```

40506 | 2022 | Wyoming, Weston
72617 | 2030 | Wyoming, Weston
72614 | 2030 | Wyoming, Weston
40508 | 2022 | Wyoming, Weston
(72617 rows)

```

Single quotes denote string constants. Double quotes denote things like tables and column names. If you want an upper-lower case or a table name or column name with blanks then you have to quote it with double quotes.

Lot's of stuff will fail if you put blanks in your table names. This is worse than putting blanks in your file names. Python will crash with blanks in file names.

We will be using more than one table in queries. To do this we need to tell SQL the table. That is done with a table-alias. In this case *t1*.

File: 06.sql

```

1: select t1.id, t1.year + 10 as "x", t1.state||', '||t1.county_name as "Location"
2:     from vote_by_county as t1
3:     order by 3 asc
4: ;

```

There are lots of builtin functions that you can use and all sorts of arithmetic operators in PostgreSQL that can be applied to the projected columns. You can also write your own functions and pass data from the projected columns to the function and get back results.

Languages for functions include PG/SQL - the built in PostgreSQL language and others like JavaScript, Lua, C, C++, Go etc. I use PG/SQL and C for processing. JavaScript is 5 to 10x slower than PG/SQL. C is 10x faster than PG/SQL. These are rough numbers. I am a big fan of Go but I haven't used it for stored-procedure/functions yet in PostgreSQL.

Using a C function requires re-loading and re-starting the database - so it is hard.

Get the list of all of the candidates in the data.

```

1: select distinct candidate from vote_by_county ;

```

Find out what counties where voted in:

```

1:
2: -- Find the set of counties that voted for "Sleepy Joe" in 2020
3:
4: select t1.state, t1.county_name
5:     from vote_by_county as t1
6:     where t1.year = 2020
7:           and t1.candidate = 'Joseph R Biden Jr'
8:     order by state, county_name
9: ;
10:
11: select t1.state, t1.county_name

```

```
12:     from vote_by_county as t1
13:     where t1.year = 2020
14:           and t1.candidate = 'Donald J Trump'
15:     order by state, county_name
16: ;
17:
```

The question is who won? To find that we have to find the candidate that had the most votes in each county. We need to use the "max" in a county.

```
1:
2: -- Find the set of counties that voted for "Sleepy Joe" in 2020
3:
4: select t1.state, t1.county_name
5: from vote_by_county as t1
6: where t1.year = 2020
7:       and t1.candidate = 'Joseph R Biden Jr'
8:       and t1.candidatevotes = (
9:         select max(t2.candidatevotes) as max_votes
10:        from vote_by_county as t2
11:        where t2.state = t1.state
12:              and t2.county_name = t1.county_name
13:      )
14:     order by state, county_name
15: ;
16:
17:
```

Operators

<https://www.postgresql.org/docs/9.0/functions.html>

There are lots!

Operator	Description	Example	Result
+	addition	$2 + 3$	5
-	subtraction	$2 - 3$	-1
*	multiplication	$2 * 3$	6
/	division (integer division truncates the result)	$4 / 2$	2
%	modulo (remainder)	$5 \% 4$	1
^	exponentiation	$2.0 ^ 3.0$	8
/	square root	/ 25.0	5
/	cube root	/ 27.0	3
!	factorial	5 !	120
!!	factorial (prefix operator)	!! 5	120
@	absolute value	@ -5.0	5
&	bitwise AND	91 & 15	11
	bitwise OR	32 3	35
#	bitwise XOR	17 # 5	20
~	bitwise NOT	~1	-2
<<	bitwise shift left	1 << 4	16
>>	bitwise shift right	8 >> 2	2

string operations

Function	Return Type	Description	Example	Result
<code>string string</code>	text	String concatenation	<code>'Post' 'greSQL'</code>	PostgreSQL
<code>string non-string</code> or <code>non-string string</code>	text	String concatenation with one non-string input	<code>'Value: ' 42</code>	Value: 42
<code>bit_length(string)</code>	int	Number of bits in string	<code>bit_length('jose')</code>	32
<code>char_length(string)</code> or <code>character_length(string)</code>	int	Number of characters in string	<code>char_length('jose')</code>	4
<code>lower(string)</code>	text	Convert string to lower case	<code>lower('TOM')</code>	tom
<code>octet_length(string)</code>	int	Number of bytes in string	<code>octet_length('jose')</code>	4
<code>overlay(string placing string from int [for int])</code>	text	Replace substring	<code>overlay('Txxxxas' placing 'hom' from 2 for 4)</code>	Thomas
<code>position(substring in string)</code>	int	Location of specified substring	<code>position('om' in 'Thomas')</code>	3
<code>substring(string [from int] [for int])</code>	text	Extract substring	<code>substring('Thomas' from 2 for 3)</code>	hom
<code>substring(string from pattern)</code>	text	Extract substring matching POSIX regular expression. See Section 9.7 for more information on pattern matching.	<code>substring('Thomas' from '...\$')</code>	mas
<code>substring(string from pattern for escape)</code>	text	Extract substring matching SQL regular expression. See Section 9.7 for more information on pattern matching.	<code>substring('Thomas' from '%#"o_a#"_' for '#')</code>	oma
<code>trim([leading trailing both] [characters] from string)</code>	text	Remove the longest string containing only the characters (a space by default) from the start/end/both ends of the string	<code>trim(both 'x' from 'xTomxx')</code>	Tom
<code>upper(string)</code>	text	Convert string to upper case	<code>upper('tom')</code>	TOM

Base Functions

Function	Return Type	Description	Example	Result
<code>abs(x)</code>	(same as input)	absolute value	<code>abs(-17.4)</code>	17.4
<code>cbrt(dp)</code>	dp	cube root	<code>cbrt(27.0)</code>	3
<code>ceil(dp or numeric)</code>	(same as input)	smallest integer not less than argument	<code>ceil(-42.8)</code>	-42
<code>ceiling(dp or numeric)</code>	(same as input)	smallest integer not less than argument (alias for <code>ceil</code>)	<code>ceiling(-95.3)</code>	-95
<code>degrees(dp)</code>	dp	radians to degrees	<code>degrees(0.5)</code>	28.6478897565412
<code>div(y numeric, x numeric)</code>	numeric	integer quotient of y/x	<code>div(9,4)</code>	2
<code>exp(dp or numeric)</code>	(same as input)	exponential	<code>exp(1.0)</code>	2.71828182845905
<code>floor(dp or numeric)</code>	(same as input)	largest integer not greater than argument	<code>floor(-42.8)</code>	-43
<code>ln(dp or numeric)</code>	(same as input)	natural logarithm	<code>ln(2.0)</code>	0.693147180559945
<code>log(dp or numeric)</code>	(same as input)	base 10 logarithm	<code>log(100.0)</code>	2
<code>log(b numeric, x numeric)</code>	numeric	logarithm to base b	<code>log(2.0, 64.0)</code>	6.0000000000
<code>mod(y, x)</code>	(same as argument types)	remainder of y/x	<code>mod(9,4)</code>	1
<code>pi()</code>	dp	" π " constant	<code>pi()</code>	3.14159265358979
<code>power(a dp, b dp)</code>	dp	a raised to the power of b	<code>power(9.0, 3.0)</code>	729
<code>power(a numeric, b numeric)</code>	numeric	a raised to the power of b	<code>power(9.0, 3.0)</code>	729
<code>radians(dp)</code>	dp	degrees to radians	<code>radians(45.0)</code>	0.785398163397448
<code>round(dp or numeric)</code>	(same as input)	round to nearest integer	<code>round(42.4)</code>	42
<code>round(v numeric, s int)</code>	numeric	round to s decimal places	<code>round(42.4382, 2)</code>	42.44
<code>sign(dp or numeric)</code>	(same as input)	sign of the argument (-1, 0, +1)	<code>sign(-8.4)</code>	-1
<code>sqrt(dp or numeric)</code>	(same as input)	square root	<code>sqrt(2.0)</code>	1.4142135623731
<code>trunc(dp or numeric)</code>	(same as input)	truncate toward zero	<code>trunc(42.8)</code>	42
<code>trunc(v numeric, s int)</code>	numeric	truncate to s decimal places	<code>trunc(42.4382, 2)</code>	42.43

Function	Return Type	Description	Example	Result
<code>width_bucket(numeric, b1 numeric, b2 numeric, count int)</code>	int	return the bucket to which operand would be assigned in an equidepth histogram with count buckets, in the range b1 to b2	<code>width_bucket(5.35, 0.024, 10.06, 5)</code>	3
<code>width_bucket(op dp, b1 dp, b2 dp, count int)</code>	int	return the bucket to which operand would be assigned in an equidepth histogram with count buckets, in the range b1 to b2	<code>width_bucket(5.35, 0.024, 10.06, 5)</code>	3

How About the square root operator!

File: 07.sql

```
select |/25.0;
```

and a factorial operator!

```
select 5!;
```

and

```
select !! 5;
```

That is *FUN* ! Not 1 but 2 factorial operators.