Combine Queries using Set operation - Union and Intersection:

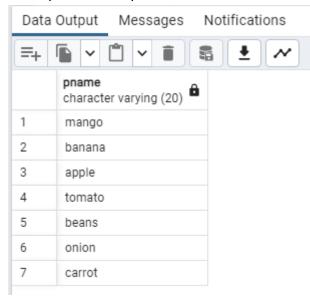
create table prod1(pid serial primary key,pname varchar(20));

create table prod2(pid serial primary key,pname varchar(20));

insert into prod2(pname) values('apple'),('mango'),('carrot'),('banana');

select pname from prod1 union

select pname from prod2;



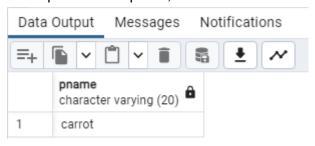
It will select all the products which are common and full data into the table.

Intersection function
It only returns the common data from both the table

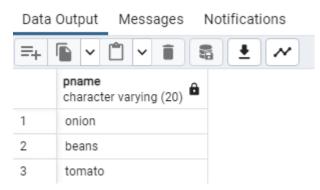
select pname from prod1

intersect

select pname from prod2;



select pname from prod1 except select pname from prod2;



Returns the value with the products in table 1 except table 2.

PostgreSQL Schemas:

To create Schema

Create schema schemaname;

create schema demo

To rename the schema

Alter schema schemaname rename to newname alter schema demo rename to test

To drop schema

Drop schema schemaname;

To create Schema we have pg gui to access and set the permission for Schema props

How to move a table to the schema Alter schema schemaname

Array Functions in postgresql:

Initialise the array as follows:

select

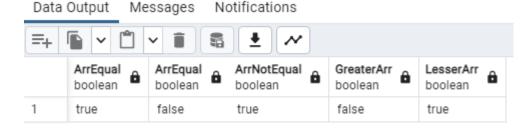
```
array[1,2,3] as "intArr",
array[2.662::float] as "floatArr",
array[current_date,current_date+10] as "dateArr";
```

Comparison operators in array:

select

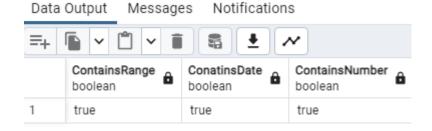
array[1,2,3]=array[1,2,3] as "ArrEqual", array[1,2]=array[2,3] as "ArrEqual", array[1,2,3]<>array[4,5,6] as "ArrNotEqual", array[1,2,3]>=array[2,3,4] as "GreaterArr", array[1,2,3,4]<=array[2,3,4,5] as "LesserArr";

--- it returns the true or false based on the condition



select

int4range(1,4) @> int4range(2,4) as "ContainsRange", daterange(current_date,current_date+30) @> current_date+15 as "ConatinsDate", numrange(1.6,6.6) && numrange(0,6) as "ContainsNumber";

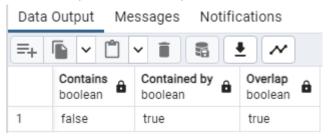


Inclusion Operator:

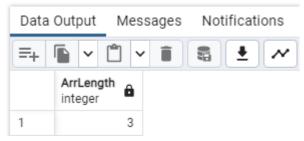
--- using inclusion operator

select

array[2,3,4] @> array[3,4,5] as "Contains", array['a','b'] <@ array['a','b'] as "Contained by", array[1,2,3] && array[2,3,4] as "Overlap";



```
Array Constructions:
--- array conact or combine
select
       array[2,3,4] || array[3,4,2] as "CombinedArr";
select
       array_cat(array[1,2,4],array[5,10]) as "ArrayCat";
select
       10 || array[1,2,4] as "AddedeleArr",
       array[10,20] || 30 as "AddeleArr";
select
       array_prepend(4,array[1,4,5]) as "ArrPrepend",
       array_append(array[1,4,5],4) as "Arrappend";
--- Array n Dimensions
select
       array_ndims(array[[1,2,5],[4,8,9]]) as "ArrDims"; returns how many dimensions.
select
       array_dims(array[[1,2,5],[4,8,9]]) as "ArrDims";
 Data Output
                              Notifications
                 Messages
 =+
        ArrDims
        text
 1
        [1:2][1:3]
select
       array_length(array[1,2,5],1) as "ArrLength";
```



select

array_lower(array[1,2,5],1) as "ArrLower";

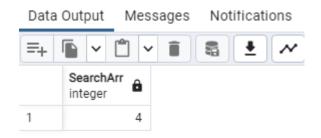
select

array_upper(array[1,2,5],1) as "ArrUpper";

--- Array search functions

select

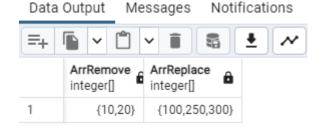
array_position(array[10,30,25,45,205],45) as "SearchArr"



--- Array Modification functions

select

array_remove(array[10,20,30],30) as "ArrRemove", array_replace(array[100,200,300],200,250) as "ArrReplace";



--- Array comparison with IN ALL ANY and SOME select

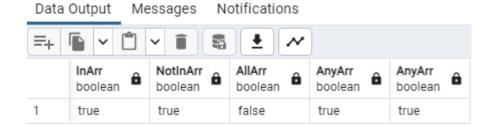
12 in (22,12,33) as "InArr",

20 not in (12,33,10) as "NotInArr",

34 = all(array[12,44,34,5]) as "AllArr", --- to check whether all number is 34

34 = any(array[34,22,24]) as "AnyArr",

34 = some(array[34,22,24]) as "AnyArr";

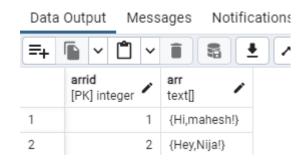


--- Array table

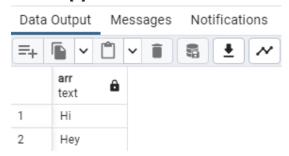
create table testarr(arrid serial primary key,arr text array);

insert into testarr(arr) values(array['Hi', 'mahesh!']),(array['Hey', 'Nija!']);

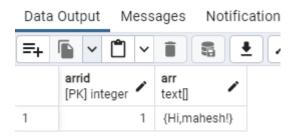
select * from testarr



— Query arr element select arr[1] from testarr



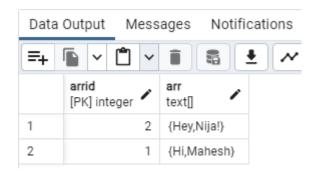
select * from testarr where arr[2] = 'mahesh!'



--- modify array data

update testarr set arr[2] = 'Mahesh' where arrid=1

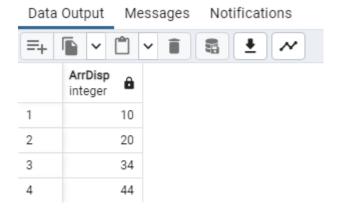
select * from testarr



--- display arr element

select

unnest(array[10,20,34,44]) as "ArrDisp"

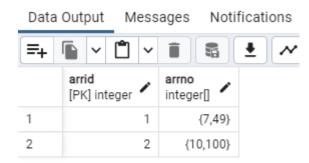


--- multidimeniion arr

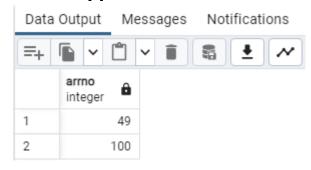
create table testarr2(arrid serial primary key,arrno int[][]);

insert into testarr2(arrno) values('{7,49}'),('{10,100}');

select * from testarr2



select arrno[2] from testarr2

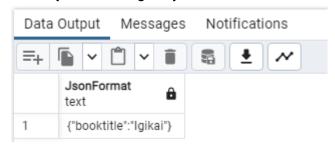


JSON in postgreSQL:

Json Syntax: {key:value,key:value}

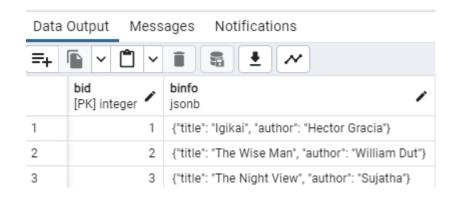
--- json representation

select '{"booktitle":"Igikai"}' as "JsonFormat"

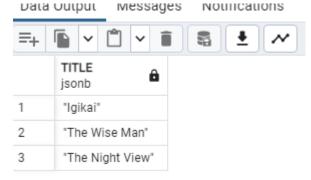


--- jsonb create table and insert values

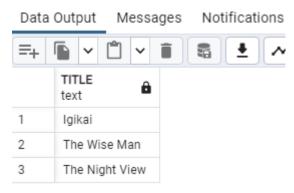
insert into books(binfo) values('{"title":"Igikai","author":"Hector Gracia"}'); insert into books(binfo) values('{"title":"The Wise Man","author":"William Dut"}') insert into books(binfo) values('{"title":"The Night View","author":"Sujatha"}'); select * from books



select binfo->'title' as "TITLE" from books



select binfo->>'title' as "TITLE" from books



- --- json update delete
- --- update a record
- --- using || to add value

update books set binfo = binfo || '{"author":"John OSlen"}' where binfo->>'author' = 'Sujatha'

select * from books



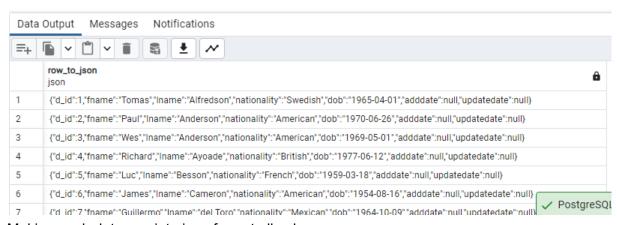
--- update a record with boolean

update books set binfo = binfo || '{"BestAuthor":true}' where binfo->>'author' = 'Hector Gracia'

select * from books



select row_to_json(director) from director



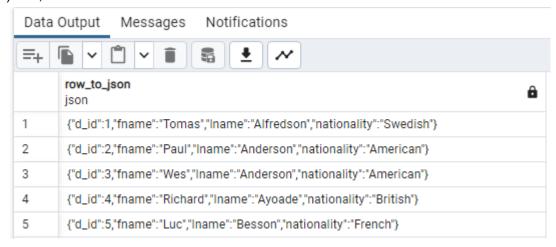
Making each data row into json format all columns

--- create json table

select * from director

select row_to_json(t) from(
 select d_id,fname,lname,nationality from director

) as t;



--- json aggregate

select *,(

select json_agg(x) as allmovies from (select movname from movie where d_id=director.d_id) as x

) from director limit 5

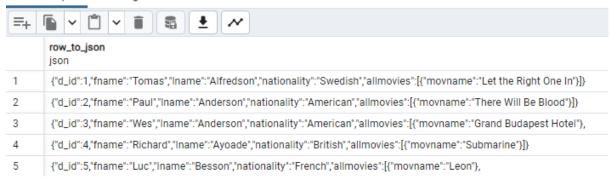


--- json array

Data Output Messages Notifications



Data Output Messages Notifications



Index Operations:

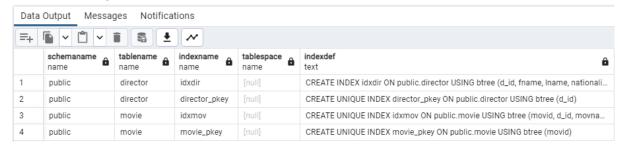
- --- index operations in postgresql
- --- to create an index and unique index

create index idxdir on director(d_id,fname,lname,nationality)

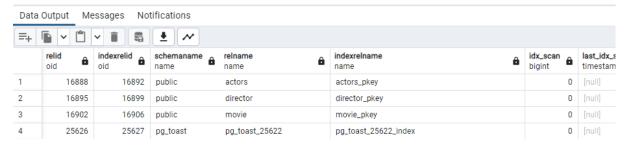
create unique index idxmov on movie(movid,d_id,movname)

--- to view all index

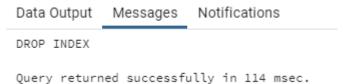
select * from pg_indexes limit 4



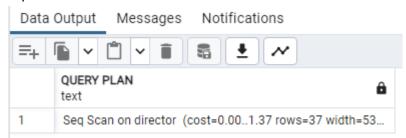
select * from pg_stat_all_indexes



--- to drop a index drop index if exists idxmov



explain select * from director



It will explain all the type and info about the statement and detailed explanation about the data and type of data operations done

explain analyze select * from director



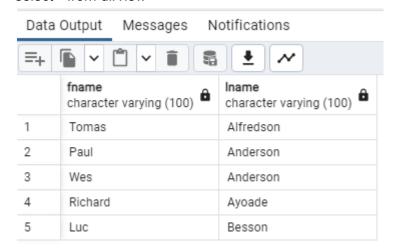
Same as the above but the time includes with each row and analyze the information.

Views in PostgreSQL:

--- create views in postgresql

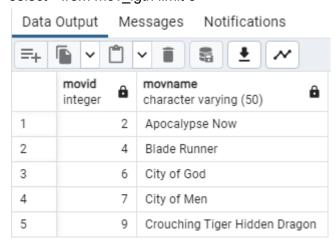
create view dirview as select fname, Iname from director;

select * from dirview



--- we can do filter, and implement clause create or replace view mov_lgth as select movid,movname from movie where movlgth>120

select * from mov_lgth limit 5



--- we can join multiple table with the view

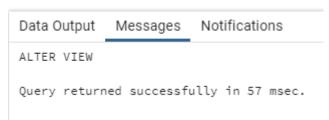
create or replace view director_movie as select mv.movid,mv.movname,d.fname,d.lname from movie mv inner join director d using (d_id) where mv.movlgth>120

select * from director_movie

Data	Data Output Messages Notifications				
=+					
	movid integer	movname character varying (50)	fname character varying (100)	Iname character varying (100)	
1	2	Apocalypse Now	Francis	Ford Coppola	
2	4	Blade Runner	Ridley	Scott	
3	6	City of God	Fernando	Meirelles	
4	7	City of Men	Paulo	Morelli	
5	9	Crouching Tiger Hidden Dragon	Ang	Lee	

In view we can do all the operations like in the sql

How to rename the view: alter view director_movie rename to dir_mov



To delete a view

drop view mov_lgth



To implement with union create view alldirsactors as select fname, Iname from actors union all select fname, Iname from director

select * from alldirsactors limit 10

Data Output Messages Notifications				
	fname character varying (100)	Iname character varying (100)		
1	Malin	Akerman		
2	Tim	Allen		
3	Julie	Andrews		
4	Ivana	Baquero		
5	Lorraine	Bracco		
6	Alice	Braga		
7	Marion	Brando		
8	Adrien	Brody		
9	Peter	Carlberg		
10	Gemma	Chan		

To implement with intersect and except

create view alldirsactorsint as select fname,Iname from actors intersect select fname,Iname from director

select * from alldirsactorsint limit 10

Data	Output Messages N	otifications		
	fname character varying (100)	Iname character varying (100)		
1	Bruce	Lee		
2	Terry	Jones		

create view alldirsactorsxp as select fname, Iname from actors except select fname, Iname from director

select * from alldirsactorsxp limit 10

Data Output Messages Notifications			
	fname character varying (100)	Iname character varying (100)	
1	Uma	Thurman	
2	Dandan	Song	
3	Andy	Lau	
4	Alexandre	Rodrigues	
5	Eric	Idle	
6	Paul	Dano	
7	Darlan	Cunha	
8	John	Travolta	
9	Lina	Leandersson	
10	Rafe	Spall	

In the updatable view we cant use the inbuilt functions on the query

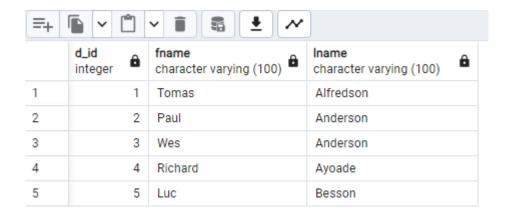
We can done CRUD operations on view table.

We can do constraints on view table.

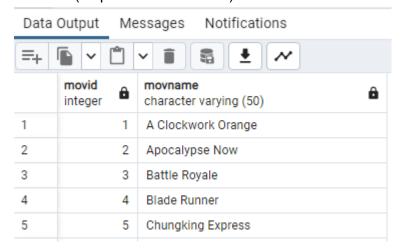
Materialized View:

--- materialized view dirmov as select d_id,fname,Iname from director

select * from dirmov



create materialized view materialmov as select movid,movname from movie with data (output shows with data)



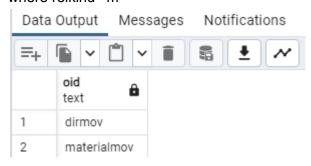
We can do all the operations done in the view also.

To drop the materialized view Drop materializedview viewname;

--- to list all materialized view

select oid::regclass::text from pg_class

where relkind='m'



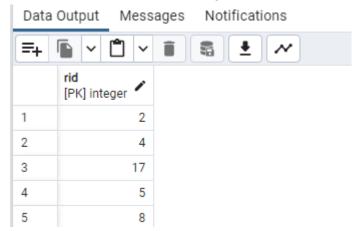
SubQueries :

Queries inside Queries

To write the subquery in logical way.

select rid from movierev

where revenue_dom>(select avg(revenue_dom) from movierev)



Like wise according to the condition the subquery should be written.

Common Table expressions:

```
with dirmov1 as
(
select mv.movname,d.fname,d.lname from movie mv
join director d using (d_id) where mv.d_id=1
)
select * from dirmov1
```



We can do filter, joins, and views with the cte.

Using Returning * we can recursively done.

What is Summarization? Subtotals and groupsets

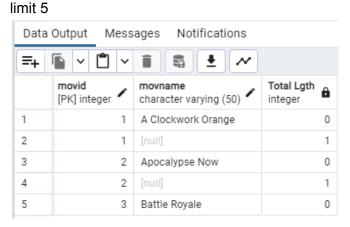
select movid,movname,sum(movlgth) as "Total Lgth" from movie group by movid,rollup(movname) order by movid,movname limit 5

Data Output Messages Notifications			
=+	<u> </u>		
	movid [PK] integer	movname character varying (50)	Total Lgth bigint
1	1	A Clockwork Orange	112
2	1	[null]	112
3	2	Apocalypse Now	168
4	2	[null]	168
5	3	Battle Royale	111

select movid,movname,sum(movlgth) as "Total Lgth" from movie group by movid,rollup(movid,movname) order by movid,movname limit 5

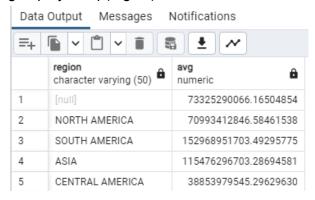
Data	Data Output Messages Notifications			
	movid [PK] integer	movname character varying (50)	Total Lgth bigint	
1	1	A Clockwork Orange	112	
2	1	[null]	112	
3	1	[null]	112	
4	2	Apocalypse Now	168	
5	2	[null]	168	

select movid,movname,grouping(movname) as "Total Lgth" from movie group by movid,rollup(movname) order by movid,movname

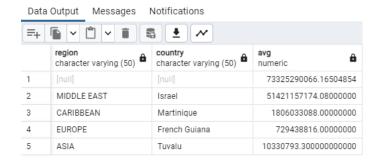


WINDOW FUNCTIONS in PostgreSQL:

Groupby rollup select region,avg(imports) from trades group by rollup(region)



select region,country,avg(imports) from trades group by rollup(region,country) limit 5

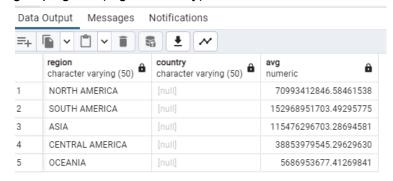


GroupBy Cube:

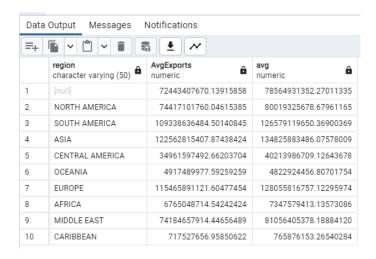
select region,country,avg(imports) from trades group by cube(region,country) limit



select region,country,avg(imports) from trades group by grouping sets(region,country) limit 5



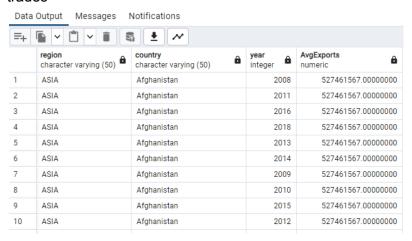
select region,avg(exports) as "AvgExports",avg(exports) filter(where year>=1998) from trades group by rollup(region) limit 10



select region, country, year, avg(exports) over() as "AvgExports" from trades limit 5

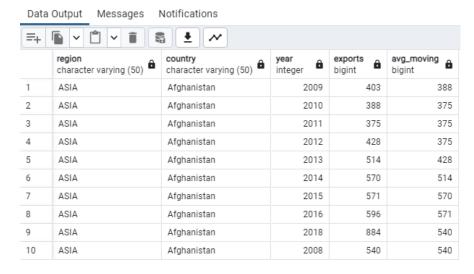


select region, country, year, avg(exports) over(partition by country) as "AvgExports" from trades

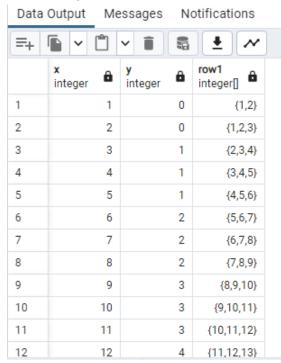


--- sliding dynamic window

select region,country,year,exports,min(exports) over(partition by country rows between 1 preceding and 1 following) as "avg_moving" from trades limit 10



select *,x/3 as y,array_agg(x) over(order by x rows between 1 preceding and 1 following) as row1 from generate_series(1,25) as x;



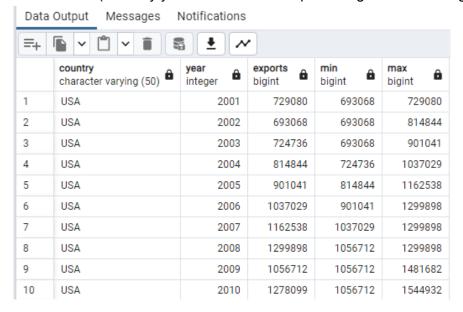
— window function:

select country,year,exports,min(exports) over w,max(exports) over w from trades

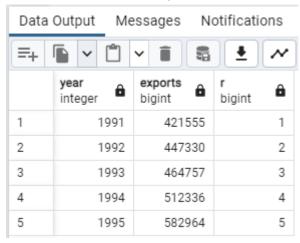
where

country='USA' and year>2000

window w as (order by year rows between 1 preceding and 2 following)



select year,exports,rank() over(order by exports) as r from trades where country='USA' limit 5

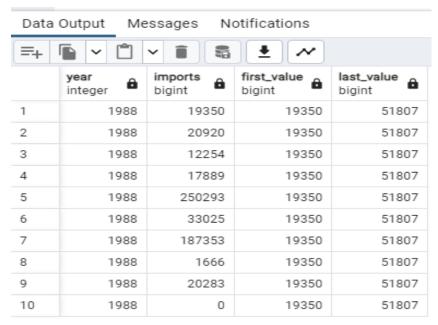


select year,exports,ntile(5) over(order by country) from trades where country in ('USA','France','Belgium') and year>2000

Data Output Messages Notifications				
=+	~ 1	v i 6	• ~	
	year integer	exports bigint	ntile integer	
1	2001	190309	1	
2	2002	215803	1	
3	2003	255553	1	
4	2004	307690	1	
5	2005	335691	1	
6	2006	366835	1	
7	2007	431743	1	
8	2008	471797	1	
9	2009	370879	1	
10	2010	407595	1	
11	2011	475957	1	
12	2012	446854	1	

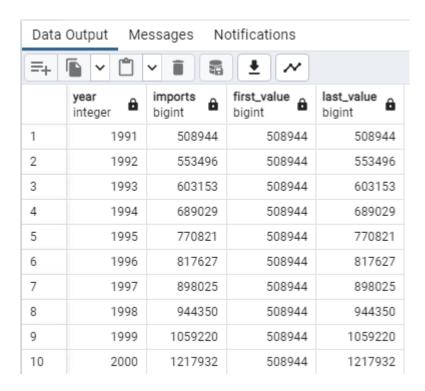
select year,imports,first_value(imports) over(order by year),last_value(imports) over(order by year)

from trades limit 10



select year,imports,first_value(imports) over(order by year),last_value(imports) over(order by year)

from trades where country='USA' limit 10



select year,imports, row_number() over(order by year) from trades where country='USA' limit 10

Data Output Messages Notifications				
=+	• •	v i s	• ~	
	year integer	imports bigint	row_number bigint	
1	1991	508944	1	
2	1992	553496	2	
3	1993	603153	3	
4	1994	689029	4	
5	1995	770821	5	
6	1996	817627	6	
7	1997	898025	7	
8	1998	944350	8	
9	1999	1059220	9	
10	2000	1217932	10	

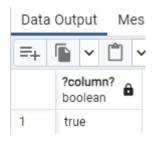
Using Regular Expressions in PostgreSQL:

Posix , similar to are types of regex in this concept

List of Syntax refer notes.

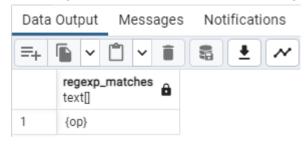
- ~ (matches regex case sensitive)
- ~*(matches regex case insensitive)
- !~(no regex case sensitive)
- !~*(no regex case insensitive)

select 'Same'~*'same'

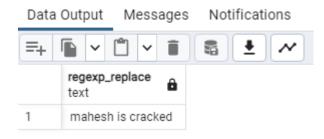


All the operators are used to do the same operation.

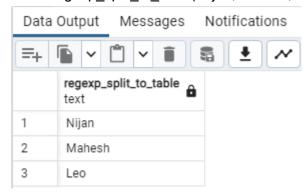
select regexp_matches('mahesh is op','op','g')



Replacing word from the source string. select regexp_replace('mahesh is op','op','cracked','g')



select regexp_split_to_table('Nijan,Mahesh,Leo',',');



--- regex split to array

select (regexp_split_to_array('Nijan,Mahesh,Leo',"),1);

