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;
```

Data Output Messages Notifications

	pname character varying (20)
1	mango
2	banana
3	apple
4	tomato
5	beans
6	onion
7	carrot

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;
```

Data Output Messages Notifications

	pname character varying (20)
1	carrot

```
select pname from prod1
except
select pname from prod2;
```

Data Output Messages Notifications

	pname character varying (20)
1	onion
2	beans
3	tomato

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
```

Data Output Messages Notifications

	ArrEqual boolean	ArrEqual boolean	ArrNotEqual boolean	GreaterArr boolean	LesserArr boolean
1	true	false	true	false	true

```
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";
```

Data Output Messages Notifications

	ContainsRange boolean	ConatinsDate boolean	ContainsNumber boolean
1	true	true	true

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";
```

Data Output Messages Notifications

	Contains boolean	Contained by boolean	Overlap boolean
1	false	true	true

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 Messages Notifications

	ArrDims text
1	[1:2][1:3]

```
select
        array_length(array[1,2,5],1) as "ArrLength";
```

Data Output Messages Notifications

	ArrLength integer
1	3

```
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"
```

Data Output Messages Notifications

	SearchArr integer
1	4

```
--- Array Modification functions

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

Data Output Messages Notifications

	ArrRemove integer[]	ArrReplace integer[]
1	{10,20}	{100,250,300}

```
--- 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";
```

Data Output Messages Notifications

	InArr boolean	NotInArr boolean	AllArr boolean	AnyArr boolean	AnyArr boolean
1	true	true	false	true	true

--- 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
```

Data Output Messages Notifications

	arrid [PK] integer	arr text[]
1	1	{Hi,mahesh!}
2	2	{Hey,Nija!}

— Query arr element

```
select arr[1] from testarr
```

Data Output Messages Notifications

	arr text
1	Hi
2	Hey

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

Data Output Messages Notification

	arrid [PK] integer	arr text[]
1	1	{Hi,mahesh!}

--- modify array data

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

select * from testarr
```

Data Output Messages Notifications

	arrid [PK] integer	arr text[]
1	2	{Hey,Nija!}
2	1	{Hi,Mahesh}

--- display arr element

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

Data Output Messages Notifications

	ArrDisp integer
1	10
2	20
3	34
4	44

--- multidimeniion arr

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

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

select * from testarr2
```

Data Output Messages Notifications

	arrid [PK] integer	arrno integer[]
1	1	{7,49}
2	2	{10,100}

select arrno[2] from testarr2

Data Output Messages Notifications

	arrno integer
1	49
2	100

JSON in postgreSQL:

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

--- json representation

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

Data Output Messages Notifications

	JsonFormat text
1	{"booktitle":"Igikai"}

--- jsonb create table and insert values

```
create table books(
        bid serial primary key,
        binfo jsonb
);

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
```

Data Output Messages Notifications

	bid [PK] integer	binfo jsonb
1	1	{"title": "Igikai", "author": "Hector Gracia"}
2	2	{"title": "The Wise Man", "author": "William Dut"}
3	3	{"title": "The Night View", "author": "Sujatha"}

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

Data Output Messages Notifications

	TITLE jsonb
1	"Igikai"
2	"The Wise Man"
3	"The Night View"

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

Data Output Messages Notifications

	TITLE text
1	Igikai
2	The Wise Man
3	The Night View

--- 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

Data Output Messages Notifications

	bid [PK] integer	binfo jsonb
1	1	{"title": "Igikai", "author": "Hector Gracia"}
2	2	{"title": "The Wise Man", "author": "William Dut"}
3	3	{"title": "The Night View", "author": "John OSlen"}

--- update a record with boolean

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

select * from books

Data Output Messages Notifications

	bid [PK] integer	binfo jsonb
1	2	{"title": "The Wise Man", "author": "William Dut"}
2	3	{"title": "The Night View", "author": "John OSlen"}
3	1	{"title": "Igikai", "author": "Hector Gracia", "BestAuthor": true}

select row_to_json(director) from director

Data Output Messages Notifications

	row_to_json json
1	{"d_id":1,"fname":"Tomas","lname":"Alfredson","nationality":"Swedish","dob":"1965-04-01","adddate":null,"updatedate":null}
2	{"d_id":2,"fname":"Paul","lname":"Anderson","nationality":"American","dob":"1970-06-26","adddate":null,"updatedate":null}
3	{"d_id":3,"fname":"Wes","lname":"Anderson","nationality":"American","dob":"1969-05-01","adddate":null,"updatedate":null}
4	{"d_id":4,"fname":"Richard","lname":"Ayoade","nationality":"British","dob":"1977-06-12","adddate":null,"updatedate":null}
5	{"d_id":5,"fname":"Luc","lname":"Besson","nationality":"French","dob":"1959-03-18","adddate":null,"updatedate":null}
6	{"d_id":6,"fname":"James","lname":"Cameron","nationality":"American","dob":"1954-08-16","adddate":null,"updatedate":null}
7	{"d_id":7,"fname":"Guillermo","lname":"del Toro","nationality":"Mexican","dob":"1964-10-09","adddate":null,"updatedate":null}

PostgreSQL

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;
```

Data Output Messages Notifications

	row_to_json json
1	{"d_id":1,"fname":"Tomas","lname":"Alfredson","nationality":"Swedish"}
2	{"d_id":2,"fname":"Paul","lname":"Anderson","nationality":"American"}
3	{"d_id":3,"fname":"Wes","lname":"Anderson","nationality":"American"}
4	{"d_id":4,"fname":"Richard","lname":"Ayoade","nationality":"British"}
5	{"d_id":5,"fname":"Luc","lname":"Besson","nationality":"French"}

```
--- 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
```

Data Output Messages Notifications

	d_id [PK] integer	fname character varying (100)	lname character varying (100)	nationality character varying (30)	dob date	adddate date	updatedate date	allmovies json
1	1	Tomas	Alfredson	Swedish	1965-04-01	[null]	[null]	[{"movname":"L
2	2	Paul	Anderson	American	1970-06-26	[null]	[null]	[{"movname":"T
3	3	Wes	Anderson	American	1969-05-01	[null]	[null]	[{"movname":"C
4	4	Richard	Ayoade	British	1977-06-12	[null]	[null]	[{"movname":"S
5	5	Luc	Besson	French	1959-03-18	[null]	[null]	[{"movname":"L

```
--- json array

select json_build_array(10,20,34,44) as "BuildArrJson",
                    json_build_object(10,20,34,44) as "BuildObjectJson",
                    json_build_array('name','mahesh','email','mahesh@gmail.com') as
"BuildObjectJson"
```

Data Output Messages Notifications

	BuildArrJson json	BuildObjectJson json	BuildObjectJson json
1	[10, 20, 34, 44]	{"10" : 20, "34" : 44}	["name", "mahesh", "email", "mahesh@gmail.com"]

```
select row_to_json(a) from(
select d_id,fname,lname,nationality,
(
        select json_agg(s) as allmovies from
        (
                select movname from movie where d_id = director.d_id

        ) as s

) from director) as a;
```

Data Output Messages Notifications

	row_to_json json
1	{"d_id":1,"fname":"Tomas","lname":"Alfredson","nationality":"Swedish","allmovies":[{"movname":"Let the Right One In"}]}
2	{"d_id":2,"fname":"Paul","lname":"Anderson","nationality":"American","allmovies":[{"movname":"There Will Be Blood"}]}
3	{"d_id":3,"fname":"Wes","lname":"Anderson","nationality":"American","allmovies":[{"movname":"Grand Budapest Hotel"},
4	{"d_id":4,"fname":"Richard","lname":"Ayoade","nationality":"British","allmovies":[{"movname":"Submarine"}]}
5	{"d_id":5,"fname":"Luc","lname":"Besson","nationality":"French","allmovies":[{"movname":"Leon"},

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
```

Data Output Messages Notifications

	schemaname name	tablename name	indexname name	tablespace name	indexdef text
1	public	director	idxdir	[null]	CREATE INDEX idxdir ON public.director USING btree (d_id, fname, lname, nationali...
2	public	director	director_pkey	[null]	CREATE UNIQUE INDEX director_pkey ON public.director USING btree (d_id)
3	public	movie	idxmov	[null]	CREATE UNIQUE INDEX idxmov ON public.movie USING btree (movid, d_id, movna...
4	public	movie	movie_pkey	[null]	CREATE UNIQUE INDEX movie_pkey ON public.movie USING btree (movid)

select * from pg_stat_all_indexes

Data Output Messages Notifications

	relid oid	indexrelid oid	schemaname name	relname name	indexrelname name	idx_scan bigint	last_idx_s timestam
1	16888	16892	public	actors	actors_pkey	0	[null]
2	16895	16899	public	director	director_pkey	0	[null]
3	16902	16906	public	movie	movie_pkey	0	[null]
4	25626	25627	pg_toast	pg_toast_25622	pg_toast_25622_index	0	[null]

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

Data Output Messages Notifications

```
DROP INDEX

Query returned successfully in 114 msec.
```

explain select * from director

Data Output Messages Notifications

	QUERY PLAN text
1	Seq Scan on director (cost=0.00..1.37 rows=37 width=53...

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

Data Output Messages Notifications

	QUERY PLAN text
1	Seq Scan on director (cost=0.00..1.37 rows=37 width=530) (actual time=0.014..0.016 rows=37 loops...
2	Planning Time: 0.088 ms
3	Execution Time: 0.029 ms

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,lname from director;
```

```
select * from dirview
```

Data Output Messages Notifications

	fname character varying (100)	lname character varying (100)
1	Tomas	Alfredson
2	Paul	Anderson
3	Wes	Anderson
4	Richard	Ayoade
5	Luc	Besson

--- 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
```

Data Output Messages Notifications

	movid integer	movname character varying (50)
1	2	Apocalypse Now
2	4	Blade Runner
3	6	City of God
4	7	City of Men
5	9	Crouching Tiger Hidden Dragon

--- 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 Output Messages Notifications

	movid integer	movname character varying (50)	fname character varying (100)	lname 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

Data Output Messages Notifications

```
ALTER VIEW

Query returned successfully in 57 msec.
```

To delete a view

drop view mov_lgth

Data Output Messages Notifications

```
DROP VIEW

Query returned successfully in 50 msec.
```

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

select * from alldirsactors limit 10

Data Output Messages Notifications

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

To implement with intersect and except

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

select * from alldirsactorsint limit 10

Data Output Messages Notifications

	fname character varying (100)	lname character varying (100)
1	Bruce	Lee
2	Terry	Jones

```
create view alldirsactorsxp as
select fname,lname from actors
except
select fname,lname from director

select * from alldirsactorsxp limit 10
```

Data Output Messages Notifications

	fname character varying (100)	lname 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
create materialized view dirmov as
select d_id,fname,lname from director

select * from dirmov
```

	d_id integer	fname character varying (100)	lname character varying (100)
1	1	Tomas	Alfredson
2	2	Paul	Anderson
3	3	Wes	Anderson
4	4	Richard	Ayoade
5	5	Luc	Besson

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

Data Output Messages Notifications

	movid integer	movname character varying (50)
1	1	A Clockwork Orange
2	2	Apocalypse Now
3	3	Battle Royale
4	4	Blade Runner
5	5	Chungking Express

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'

Data Output Messages Notifications

	oid text
1	dirmov
2	materialmov

SubQueries :
Queries inside Queries

To write the subquery in logical way.

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

Data Output Messages Notifications

	rid [PK] integer
1	2
2	4
3	17
4	5
5	8

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

Common Table expressions:

```
with movie_lgth_min as
(
        select movname from movie where movlgth>100
)
select * from movie_lgth_min
```

```
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
```

Data Output Messages Notifications

	movname character varying (50)	fname character varying (100)	lname character varying (100)
1	Let the Right One In	Tomas	Alfredson

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

	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 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
limit 5

Data Output Messages Notifications

	movid [PK] integer	movname character varying (50)	Total Lgth integer
1	1	A Clockwork Orange	0
2	1	[null]	1
3	2	Apocalypse Now	0
4	2	[null]	1
5	3	Battle Royale	0

WINDOW FUNCTIONS in PostgreSQL:

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

Data Output Messages Notifications

	region character varying (50)	avg numeric
1	[null]	73325290066.16504854
2	NORTH AMERICA	70993412846.58461538
3	SOUTH AMERICA	152968951703.49295775
4	ASIA	115476296703.28694581
5	CENTRAL AMERICA	38853979545.29629630

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

Data Output Messages Notifications

	region character varying (50)	country character varying (50)	avg numeric
1	[null]	[null]	73325290066.16504854
2	MIDDLE EAST	Israel	51421157174.08000000
3	CARIBBEAN	Martinique	1806033088.00000000
4	EUROPE	French Guiana	729438816.00000000
5	ASIA	Tuvalu	10330793.300000000000

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

Data Output Messages Notifications

	region character varying (50)	country character varying (50)	avg numeric
1	[null]	[null]	73325290066.16504854
2	MIDDLE EAST	Israel	51421157174.08000000
3	CARIBBEAN	Martinique	1806033088.00000000
4	EUROPE	French Guiana	729438816.00000000
5	ASIA	Tuvalu	10330793.300000000000

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

Data Output Messages Notifications

	region character varying (50)	country character varying (50)	avg numeric
1	NORTH AMERICA	[null]	70993412846.58461538
2	SOUTH AMERICA	[null]	152968951703.49295775
3	ASIA	[null]	115476296703.28694581
4	CENTRAL AMERICA	[null]	38853979545.29629630
5	OCEANIA	[null]	5686953677.41269841

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

Data Output Messages Notifications

	region character varying (50)	AvgExports numeric	avg numeric
1	[null]	72443407670.13915858	78564931352.27011335
2	NORTH AMERICA	74417101760.04615385	80019325678.67961165
3	SOUTH AMERICA	109338636484.50140845	126579119650.36900369
4	ASIA	122562815407.87438424	134825883486.07578009
5	CENTRAL AMERICA	34961597492.66203704	40213986709.12643678
6	OCEANIA	4917489977.59259259	4822924456.80701754
7	EUROPE	115465891121.60477454	128055816757.12295974
8	AFRICA	6765048714.54242424	7347579413.13573086
9	MIDDLE EAST	74184657914.44656489	81056405378.18884120
10	CARIBBEAN	717527656.95850622	765876153.26540284

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

Data Output Messages Notifications

	region character varying (50)	country character varying (50)	year integer	AvgExports numeric
1	ASIA	Afghanistan	2008	72443407670.13915858
2	ASIA	Afghanistan	2009	72443407670.13915858
3	ASIA	Afghanistan	2010	72443407670.13915858
4	ASIA	Afghanistan	2011	72443407670.13915858
5	ASIA	Afghanistan	2012	72443407670.13915858

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

Data Output Messages Notifications

	region character varying (50)	country character varying (50)	year integer	AvgExports numeric
1	ASIA	Afghanistan	2008	527461567.00000000
2	ASIA	Afghanistan	2011	527461567.00000000
3	ASIA	Afghanistan	2016	527461567.00000000
4	ASIA	Afghanistan	2018	527461567.00000000
5	ASIA	Afghanistan	2013	527461567.00000000
6	ASIA	Afghanistan	2014	527461567.00000000
7	ASIA	Afghanistan	2009	527461567.00000000
8	ASIA	Afghanistan	2010	527461567.00000000
9	ASIA	Afghanistan	2015	527461567.00000000
10	ASIA	Afghanistan	2012	527461567.00000000

--- 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

Data Output Messages Notifications

	region character varying (50)	country character varying (50)	year integer	exports bigint	avg_moving bigint
1	ASIA	Afghanistan	2009	403	388
2	ASIA	Afghanistan	2010	388	375
3	ASIA	Afghanistan	2011	375	375
4	ASIA	Afghanistan	2012	428	375
5	ASIA	Afghanistan	2013	514	428
6	ASIA	Afghanistan	2014	570	514
7	ASIA	Afghanistan	2015	571	570
8	ASIA	Afghanistan	2016	596	571
9	ASIA	Afghanistan	2018	884	540
10	ASIA	Afghanistan	2008	540	540

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;

Data Output Messages Notifications

	x integer	y integer	row1 integer[]
1	1	0	{1,2}
2	2	0	{1,2,3}
3	3	1	{2,3,4}
4	4	1	{3,4,5}
5	5	1	{4,5,6}
6	6	2	{5,6,7}
7	7	2	{6,7,8}
8	8	2	{7,8,9}
9	9	3	{8,9,10}
10	10	3	{9,10,11}
11	11	3	{10,11,12}
12	12	4	{11,12,13}

— 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)

Data Output Messages Notifications

	country character varying (50)	year integer	exports bigint	min bigint	max bigint
1	USA	2001	729080	693068	729080
2	USA	2002	693068	693068	814844
3	USA	2003	724736	693068	901041
4	USA	2004	814844	724736	1037029
5	USA	2005	901041	814844	1162538
6	USA	2006	1037029	901041	1299898
7	USA	2007	1162538	1037029	1299898
8	USA	2008	1299898	1056712	1299898
9	USA	2009	1056712	1056712	1481682
10	USA	2010	1278099	1056712	1544932

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

Data Output Messages Notifications

	year integer	exports bigint	r bigint
1	1991	421555	1
2	1992	447330	2
3	1993	464757	3
4	1994	512336	4
5	1995	582964	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

	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

Data Output Messages Notifications

	year integer	imports bigint	first_value bigint	last_value bigint
1	1988	19350	19350	51807
2	1988	20920	19350	51807
3	1988	12254	19350	51807
4	1988	17889	19350	51807
5	1988	250293	19350	51807
6	1988	33025	19350	51807
7	1988	187353	19350	51807
8	1988	1666	19350	51807
9	1988	20283	19350	51807
10	1988	0	19350	51807

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

Data Output Messages Notifications

	year integer	imports bigint	first_value bigint	last_value bigint
1	1991	508944	508944	508944
2	1992	553496	508944	553496
3	1993	603153	508944	603153
4	1994	689029	508944	689029
5	1995	770821	508944	770821
6	1996	817627	508944	817627
7	1997	898025	508944	898025
8	1998	944350	508944	944350
9	1999	1059220	508944	1059220
10	2000	1217932	508944	1217932

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

Data Output Messages Notifications

	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'
```

Data Output Mes

	?column? boolean
1	true

All the operators are used to do the same operation.

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

Data Output Messages Notifications

	regexp_matches text[]
1	{op}

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

Data Output Messages Notifications

	regexp_replace text
1	mahesh is cracked

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

Data Output Messages Notifications

	regexp_split_to_table text
1	Nijan
2	Mahesh
3	Leo

--- regex split to array

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

Data Output Messages Notifications

	row record
1	("{N,i,j,a,n,"","",M,a,h,e,s,h,"","",L,e,o}",1)