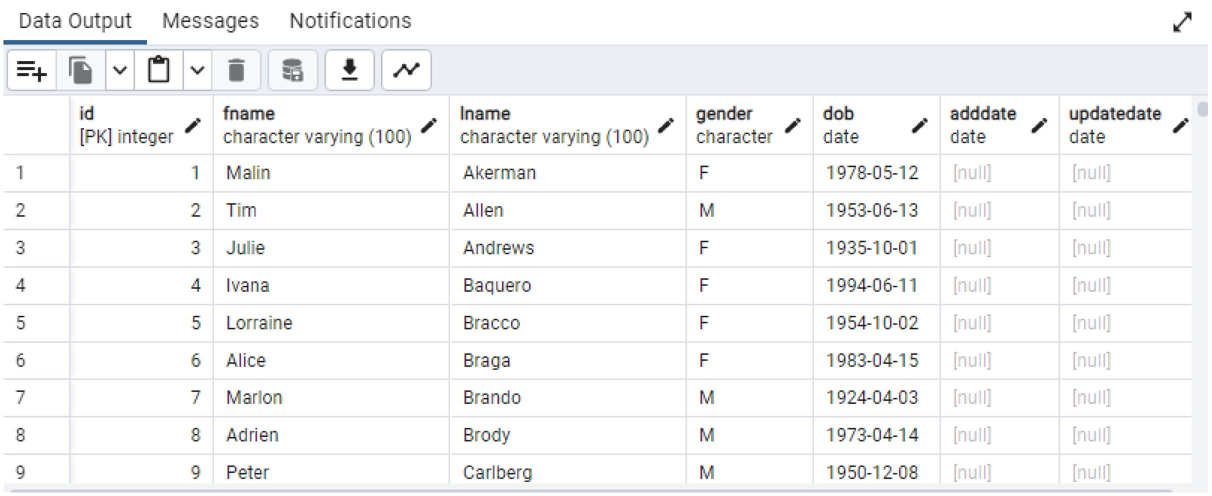
To create multiple table with constraints

```
--- To create movie table

CREATE TABLE actors (
id SERIAL PRIMARY KEY,
	fname VARCHAR(100),
	lname VARCHAR(100) NOT NULL,
	gender CHAR(1),
	dob DATE,
	adddate DATE,
	updatedate DATE
);


--- To create Director table

CREATE TABLE director(

	d_id SERIAL PRIMARY KEY,
	fname VARCHAR(100),
	lname VARCHAR(100),
	nationality VARCHAR(30),
	dob DATE,
	adddate DATE,
	updatedate DATE

);


--- To create Movie table with foreign key

CREATE TABLE movie(

	movid SERIAL PRIMARY KEY,
	movname VARCHAR(50),
	movlgth INT,
	movlang VARCHAR(20),
	ageinfo VARCHAR(20),
	releasedate DATE,
	d_id INT REFERENCES director(d_id)

);


--- To create movie revenue table

CREATE TABLE movierev (
        rid SERIAL PRIMARY KEY,
        movid INT REFERENCES movie(movid),
        revenue_dom NUMERIC(10,2),
        revenue_int NUMERIC(10,2)

);
```

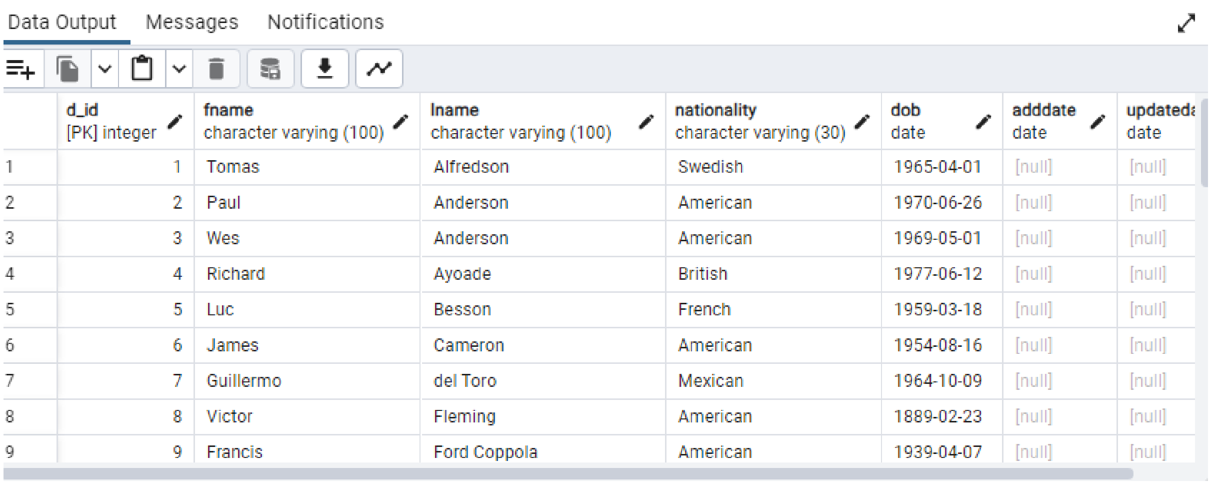
By Creating all this table inserting all values from the source and the tables as follows

Actors Table



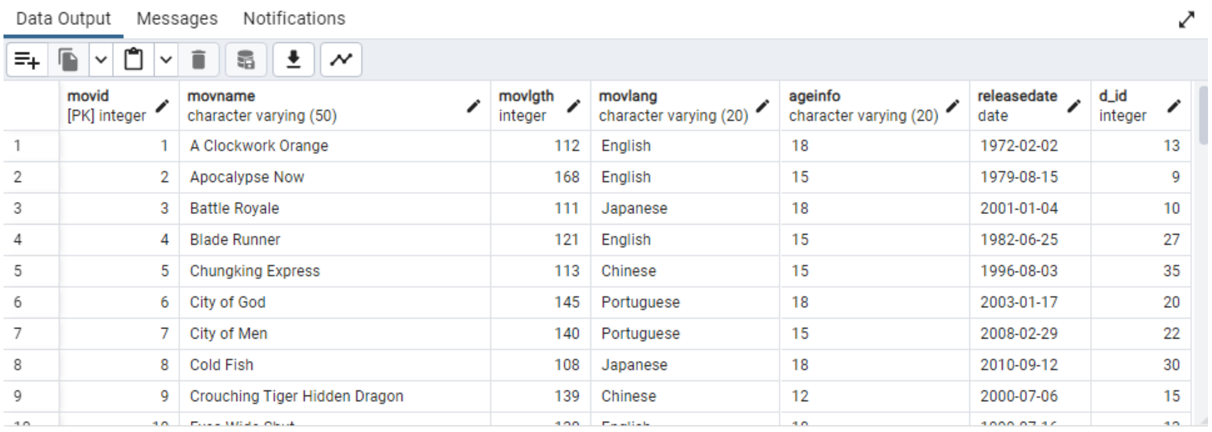
	id [PK] integer	fname character varying (100)	lname character varying (100)	gender character	dob date	adddate date	updatedate date
1	1	Malin	Akerman	F	1978-05-12	[null]	[null]
2	2	Tim	Allen	M	1953-06-13	[null]	[null]
3	3	Julie	Andrews	F	1935-10-01	[null]	[null]
4	4	Ivana	Baquero	F	1994-06-11	[null]	[null]
5	5	Lorraine	Bracco	F	1954-10-02	[null]	[null]
6	6	Alice	Braga	F	1983-04-15	[null]	[null]
7	7	Marlon	Brando	M	1924-04-03	[null]	[null]
8	8	Adrien	Brody	M	1973-04-14	[null]	[null]
9	9	Peter	Carlberg	M	1950-12-08	[null]	[null]

Director Table



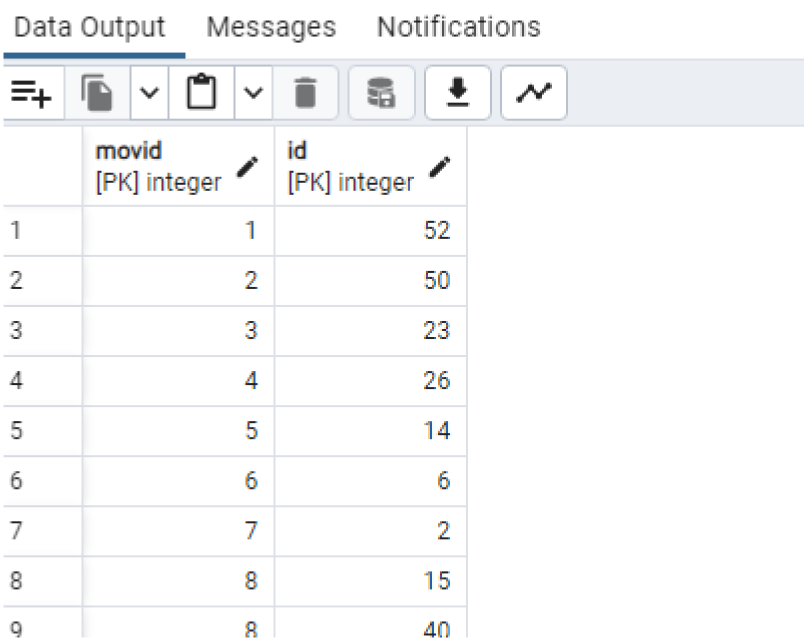
	d_id [PK] integer	fname character varying (100)	lname character varying (100)	nationality character varying (30)	dob date	adddate date	updateda date
1	1	Tomas	Alfredson	Swedish	1965-04-01	[null]	[null]
2	2	Paul	Anderson	American	1970-06-26	[null]	[null]
3	3	Wes	Anderson	American	1969-05-01	[null]	[null]
4	4	Richard	Ayoade	British	1977-06-12	[null]	[null]
5	5	Luc	Besson	French	1959-03-18	[null]	[null]
6	6	James	Cameron	American	1954-08-16	[null]	[null]
7	7	Guillermo	del Toro	Mexican	1964-10-09	[null]	[null]
8	8	Victor	Fleming	American	1889-02-23	[null]	[null]
9	9	Francis	Ford Coppola	American	1939-04-07	[null]	[null]

Movie Table



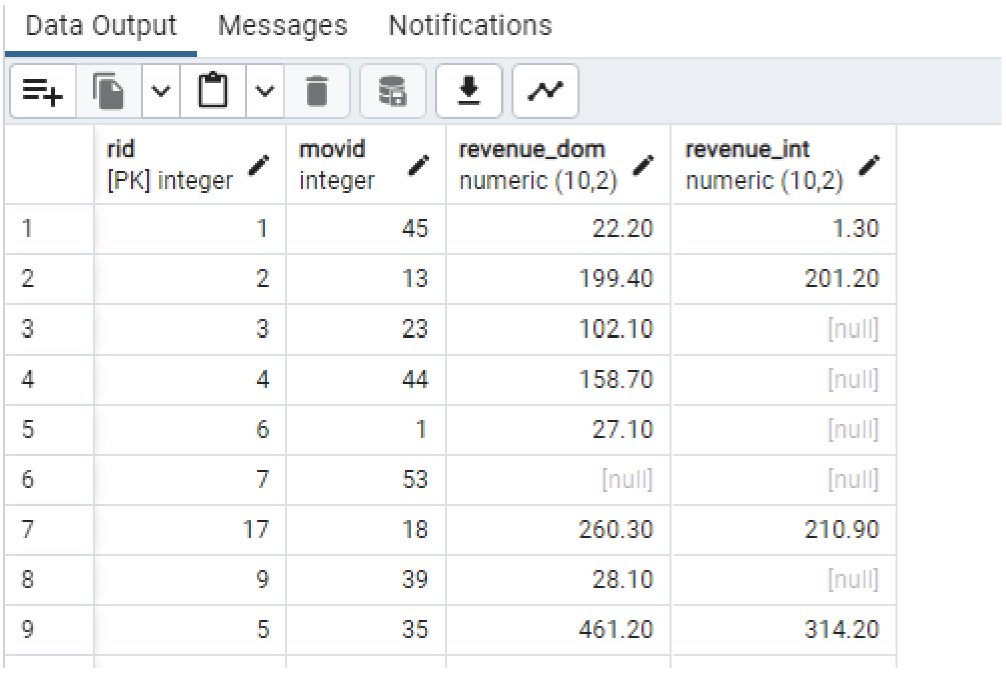
	movid [PK] integer	movname character varying (50)	movlgth integer	movlang character varying (20)	ageinfo character varying (20)	releasedate date	d_id integer
1	1	A Clockwork Orange	112	English	18	1972-02-02	13
2	2	Apocalypse Now	168	English	15	1979-08-15	9
3	3	Battle Royale	111	Japanese	18	2001-01-04	10
4	4	Blade Runner	121	English	15	1982-06-25	27
5	5	Chungking Express	113	Chinese	15	1996-08-03	35
6	6	City of God	145	Portuguese	18	2003-01-17	20
7	7	City of Men	140	Portuguese	15	2008-02-29	22
8	8	Cold Fish	108	Japanese	18	2010-09-12	30
9	9	Crouching Tiger Hidden Dragon	139	Chinese	12	2000-07-06	15

Movie Actor table



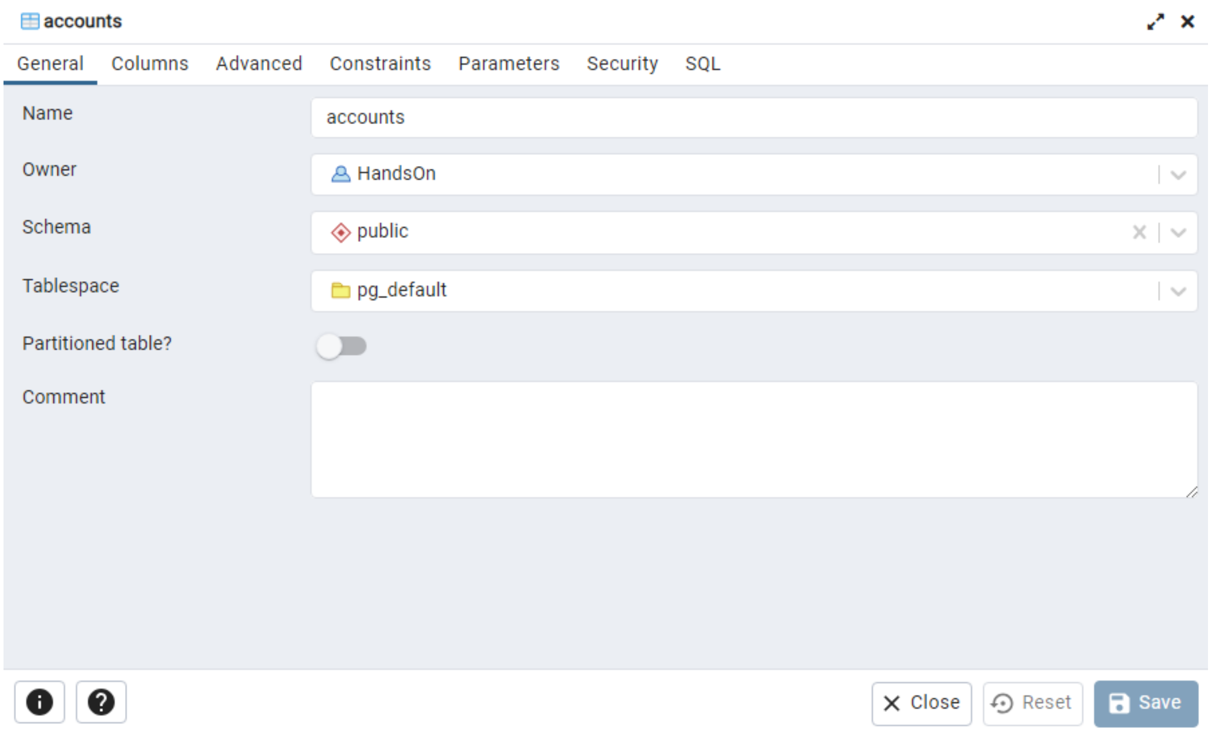
	movid [PK] integer	id [PK] integer
1	1	52
2	2	50
3	3	23
4	4	26
5	5	14
6	6	6
7	7	2
8	8	15
9	8	40

Movie Revenue Table



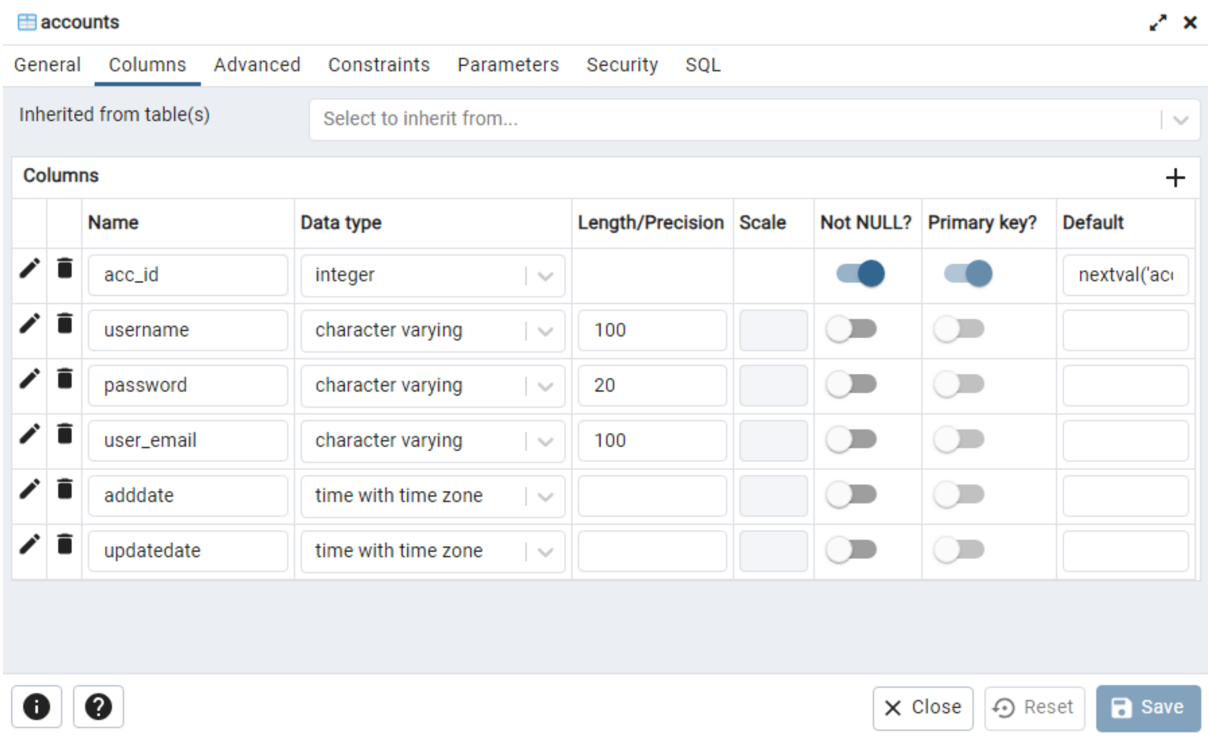
	rid [PK] integer	movid integer	revenue_dom numeric (10,2)	revenue_int numeric (10,2)
1	1	45	22.20	1.30
2	2	13	199.40	201.20
3	3	23	102.10	[null]
4	4	44	158.70	[null]
5	6	1	27.10	[null]
6	7	53	[null]	[null]
7	17	18	260.30	210.90
8	9	39	28.10	[null]
9	5	35	461.20	314.20

To Create a table using pgadmin GUI



Name	accounts
Owner	HandsOn
Schema	public
Tablespace	pg_default
Partitioned table?	
Comment	

To set the properties for the table



accounts

		Name	Data type	Length/Precision	Scale	Not NULL?	Primary key?	Default
		acc_id	integer			✓	✓	nextval('ac
		username	character varying	100				
		password	character varying	20				
		user_email	character varying	100				
		adddate	time with time zone					
		updatedate	time with time zone					

The Rename , Add Delete Column are used by options.

While inserting values into the table RETURNING keyword is used to return the value after inserting an value into the table

INSERT INTO customers(fname) values('Mahesh') RETURNING *;
It will return whole record

INSERT INTO customers(fname) values('Mahesh') RETURNING custid;
It will return the particular record.

UPSERT is used that to reduce redundancy in inserting values the ex is Below.

INSERT INTO customer(fname,city) values ('Mahesh','Chennai')
CONFLICT ON (name) DO NOTHING;

Queries in PostgreSQL:

Select Queries and its Aliases

SELECT * FROM employees; (Conventional Select Statement)

SELECT first_name,last_name FROM employees; (Selecting columns from table)

SELECT first_name AS fname,last_name AS lname from employees; (Aliases of col name)

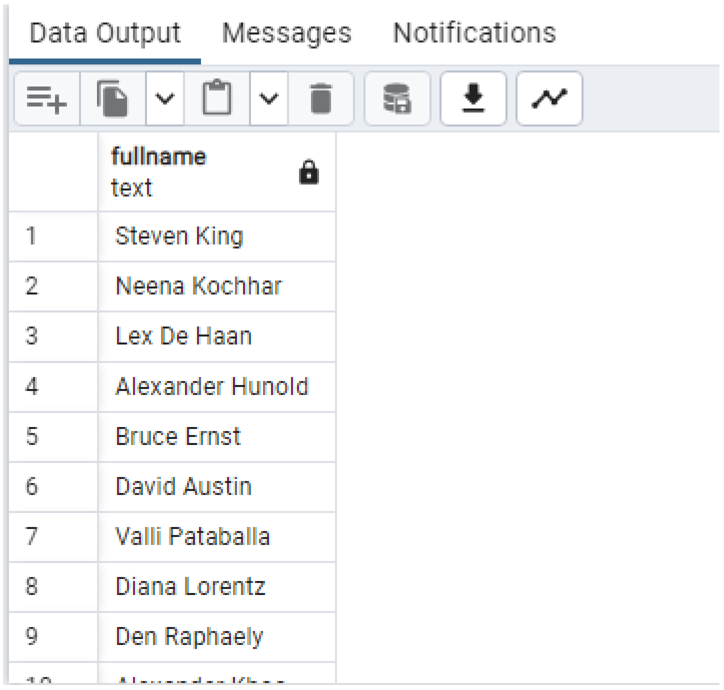
SELECT first_name "FirstName" FROM employees; (another method for aliases)

It will show output accordingly for the above Queries.

Select Statement with Expressions:

To combine firstname and last name with as name

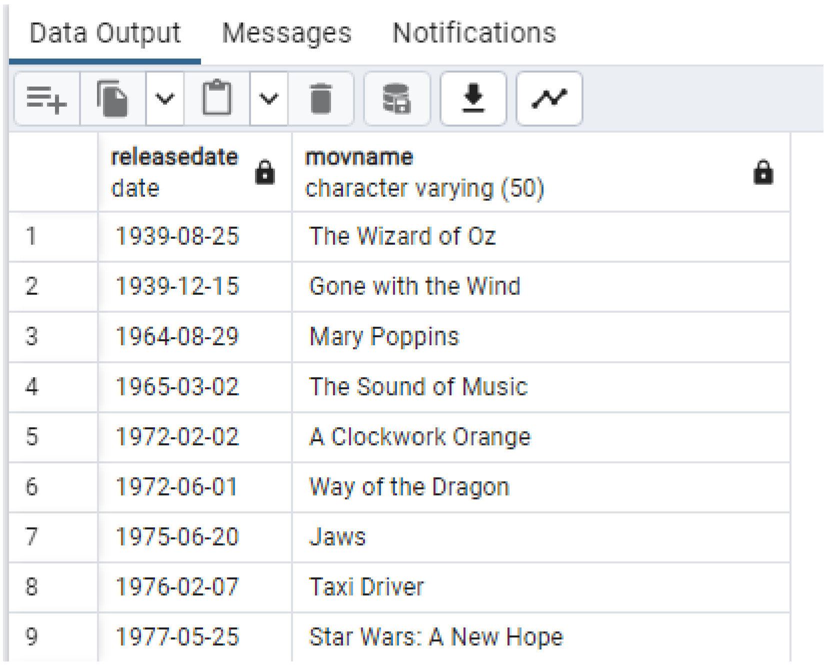
SELECT first_name||' '||last_name AS FullName FROM employees;



	fullname text
1	Steven King
2	Neena Kochhar
3	Lex De Haan
4	Alexander Hunold
5	Bruce Ernst
6	David Austin
7	Valli Pataballa
8	Diana Lorentz
9	Den Raphaely

ORDER BY Clause:
In this clause ASC ascending and DESC descending.

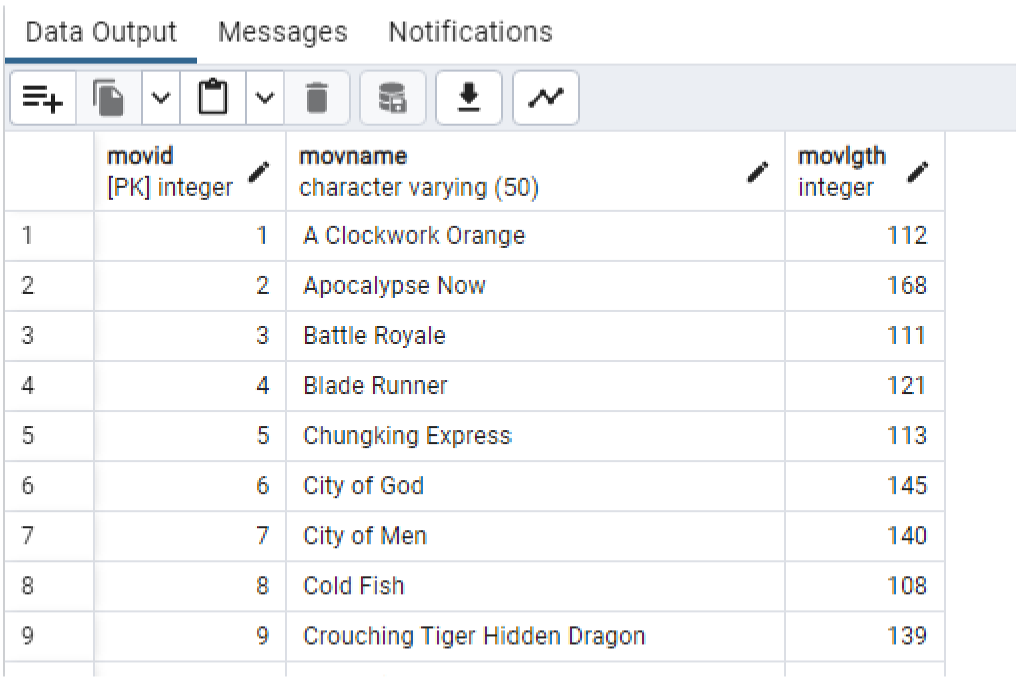
select releasedate,movname from movie order by releasedate asc,movname desc;



	releasedate date	movname character varying (50)
1	1939-08-25	The Wizard of Oz
2	1939-12-15	Gone with the Wind
3	1964-08-29	Mary Poppins
4	1965-03-02	The Sound of Music
5	1972-02-02	A Clockwork Orange
6	1972-06-01	Way of the Dragon
7	1975-06-20	Jaws
8	1976-02-07	Taxi Driver
9	1977-05-25	Star Wars: A New Hope

Order by clause with column name with number

select movid,movname,movlgth from movie order by 1 asc,3 desc;



	movid [PK] integer	movname character varying (50)	movlgth integer
1	1	A Clockwork Orange	112
2	2	Apocalypse Now	168
3	3	Battle Royale	111
4	4	Blade Runner	121
5	5	Chungking Express	113
6	6	City of God	145
7	7	City of Men	140
8	8	Cold Fish	108
9	9	Crouching Tiger Hidden Dragon	139

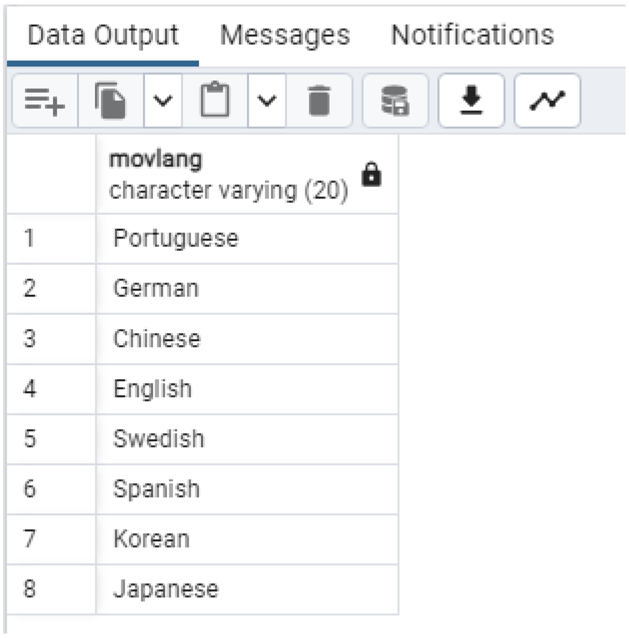
Here the movid as 1 , movname as 2 and movlgth as 3 so it can easily map the column name by number.

OrderBy clause with Null values
Syntax:

```
select movid,movname,movlgth from movie order by 1 asc,3 desc NULLS LAST;

select movid,movname,movlgth from movie order by 1 asc,3 desc NULLS FIRST;
```

Distinct in Postgresql:

```
select distinct movlang from movie;
```



	movlang character varying (20)
1	Portuguese
2	German
3	Chinese
4	English
5	Swedish
6	Spanish
7	Korean
8	Japanese

It returns the unique value from the column selected.

Here we can use order by clause with the props.

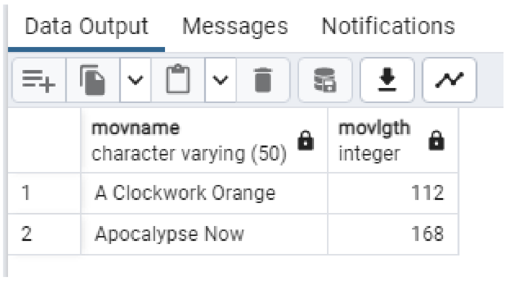
FILTERING DATA in PostgreSQL:

Using conditions and operators we can filter the data.

Where keyword is used to filter the condition of the data;

All Operators Combined and the following below

```
select movname,movlgth from movie where movname like 'A%' or movname like '%E' and
movlgth
between 120 and 140
order by movname asc,movlgth desc;
```

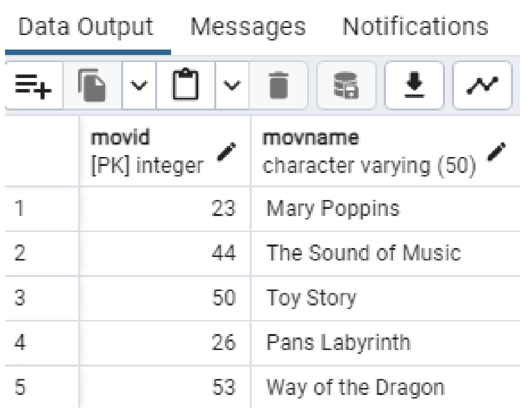


	movname character varying (50)	movlgth integer
1	A Clockwork Orange	112
2	Apocalypse Now	168

All the filter concepts and operators together with the query.

LIMIT and OFFSET in PostgreSQL:

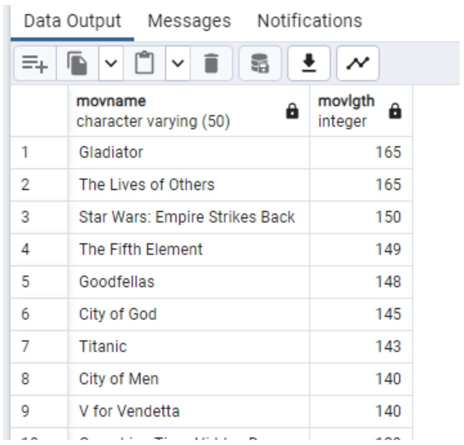
select movid,movname from movie order by movlgth asc limit 5;



	movid [PK] integer	movname character varying (50)
1	23	Mary Poppins
2	44	The Sound of Music
3	50	Toy Story
4	26	Pans Labyrinth
5	53	Way of the Dragon

Limit keyword is used to retrieve the data at the limit number.
Offset keyword is used to start the row data and to the limit to retrieve the data.

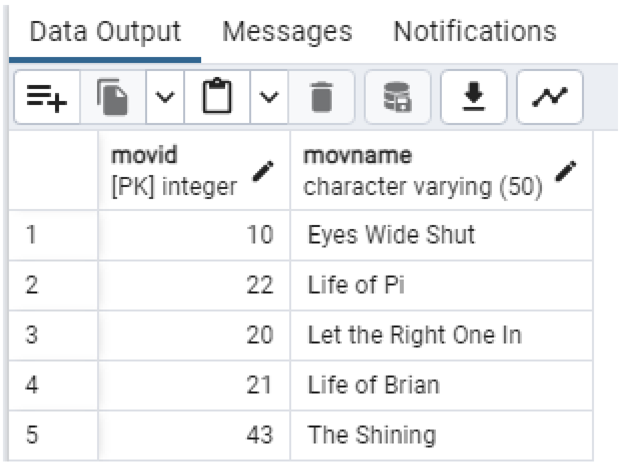
select movname,movlgth from movie order by movlgth desc limit 10 offset 2;



	movname character varying (50)	movlgth integer
1	Gladiator	165
2	The Lives of Others	165
3	Star Wars: Empire Strikes Back	150
4	The Fifth Element	149
5	Goodfellas	148
6	City of God	145
7	Titanic	143
8	City of Men	140
9	V for Vendetta	140

Fetch keyword is used to fetch the row data with the condition.

select movid,movname from movie order by movlgth desc fetch first 5 row only offset 20;

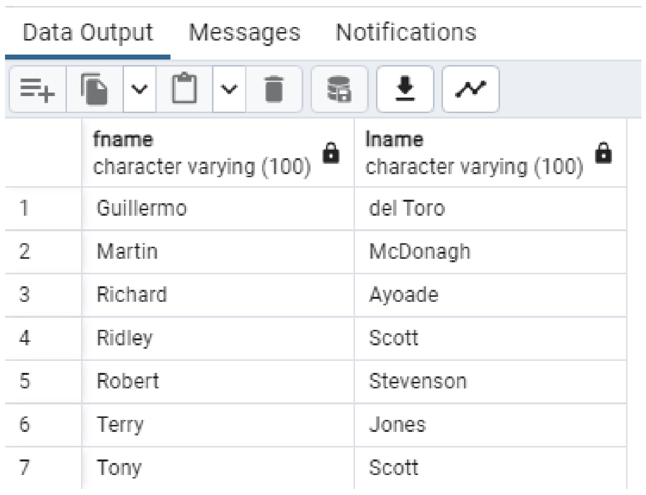


	movid [PK] integer	movname character varying (50)
1	10	Eyes Wide Shut
2	22	Life of Pi
3	20	Let the Right One In
4	21	Life of Brian
5	43	The Shining

Retrieving data from 20 row and retrivning 5 records from the condition.

IN and NOT IN in PostgreSQL:

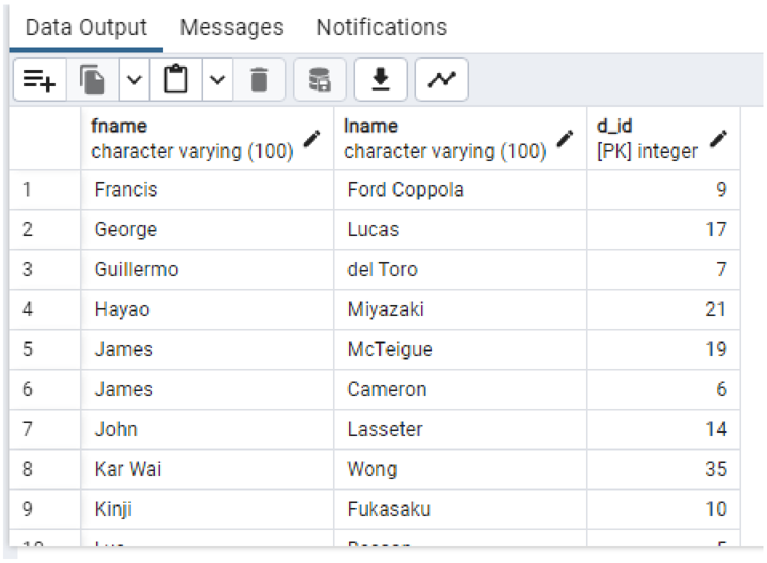
select fname,lname from director where nationality IN('British','Mexican') order by fname;



	fname character varying (100)	lname character varying (100)
1	Guillermo	del Toro
2	Martin	McDonagh
3	Richard	Ayoade
4	Ridley	Scott
5	Robert	Stevenson
6	Terry	Jones
7	Tony	Scott

This query checks whether the director in the nationality or not

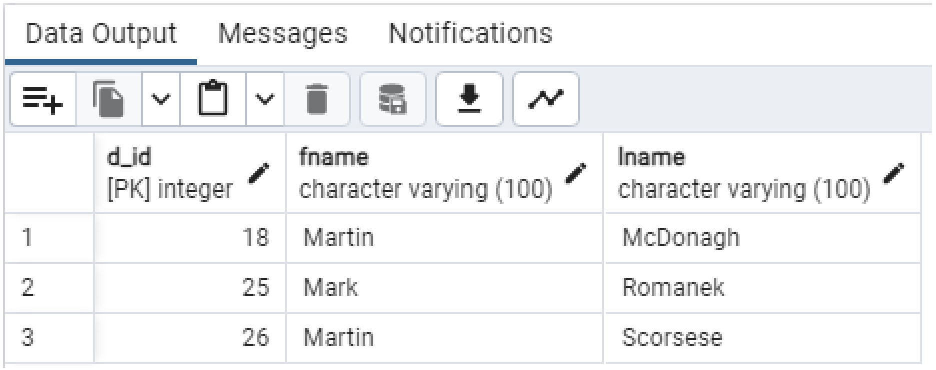
select fname,lname,d_id from director where nationality NOT IN('British') order by fname LIMIT 10 OFFSET 5;



	fname character varying (100)	lname character varying (100)	d_id [PK] integer
1	Francis	Ford Coppola	9
2	George	Lucas	17
3	Guillermo	del Toro	7
4	Hayao	Miyazaki	21
5	James	McTeigue	19
6	James	Cameron	6
7	John	Lasseter	14
8	Kar Wai	Wong	35
9	Kinji	Fukasaku	10

Like and ILike in Postgresql:

select d_id,fname,lname from director where fname like 'M%' or fname ilike '%A';



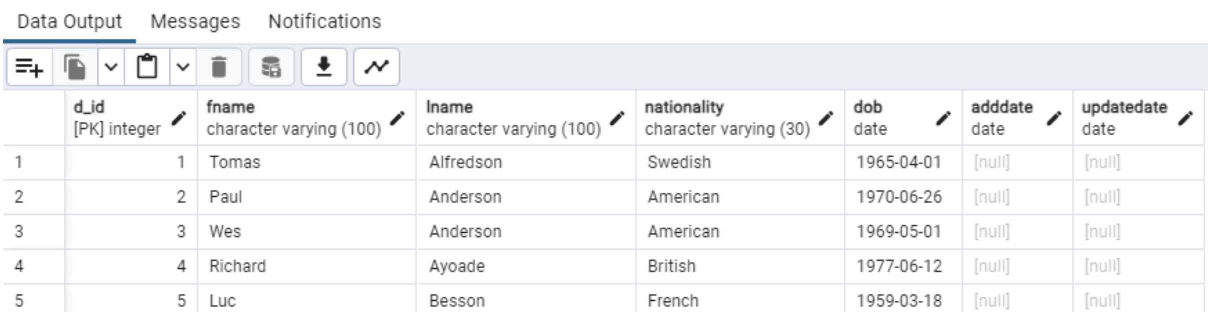
	d_id [PK] integer	fname character varying (100)	lname character varying (100)
1	18	Martin	McDonagh
2	25	Mark	Romanek
3	26	Martin	Scorsese

Here the starting string starts with M and not end with A.

With all orderby limit offset fetch properties we can write a query according to the condition.

IS NULL and IS NOT NULL:

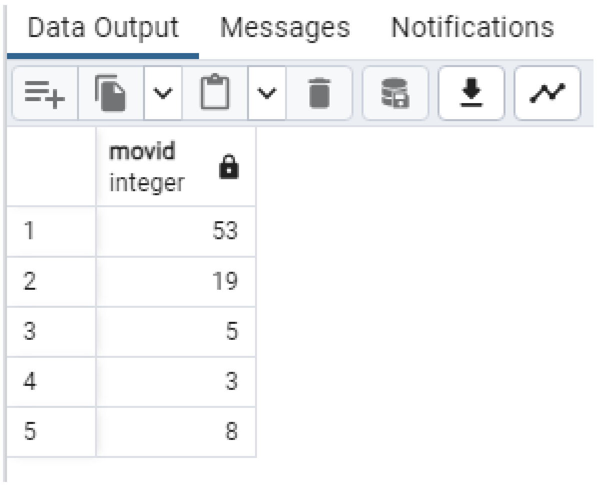
select * from director where dob IS NOT NULL LIMIT 5;



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

The is null and not null is used to check whether the data is null or not.
If the col is null it will return the data with null value and not.

```
select movid from movierev
where revenue_dom is null
and revenue_int is null;
```

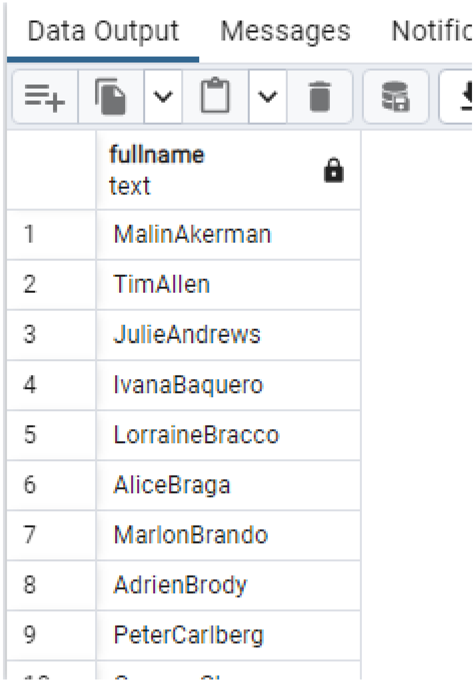


	movid integer
1	53
2	19
3	5
4	3
5	8

Here the movie revenue which the data from movie revenue from domestic and international are null abd these syntax is also to check if the data is not null.

Concat and Concat_WS :

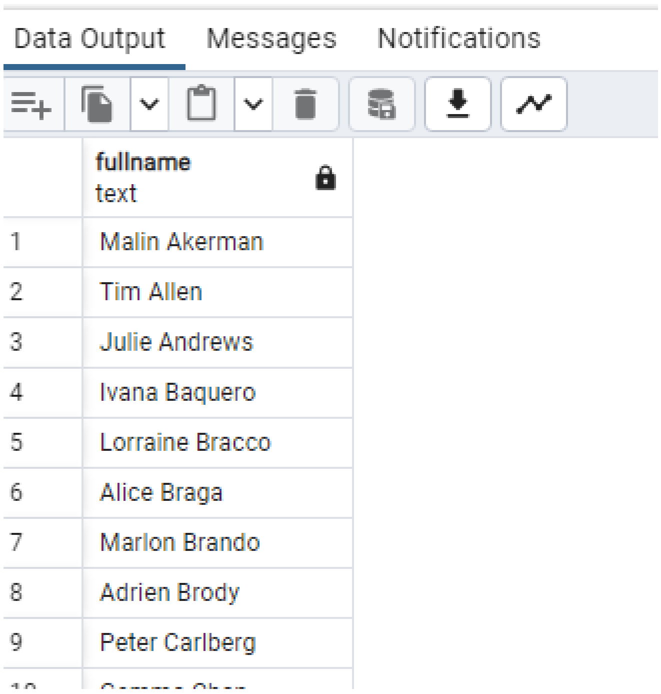
```
select concat(fname,lname) as FullName from actors;
```



	fullname text
1	MalinAkerman
2	TimAllen
3	JulieAndrews
4	IvanaBaquero
5	LorraineBracco
6	AliceBraga
7	MarlonBrando
8	AdrienBrody
9	PeterCarlberg

Without separator

select concat_ws(' ',fname,lname) as FullName from actors;



	fullname text
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

With separator .

We can also use pipe operator to concat the string as example above.

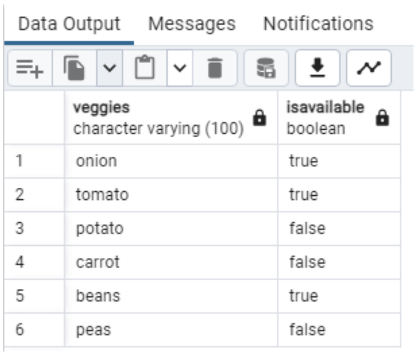
DataTypes in PostgreSQL:

Boolean:

create table veggieslist(veggies varchar(100),isavailable boolean not null);
Created table and assigned isavailable as boolean

insert into veggieslist values('peas',FALSE);
Inserted values into the table into respective value

select * from veggieslist;

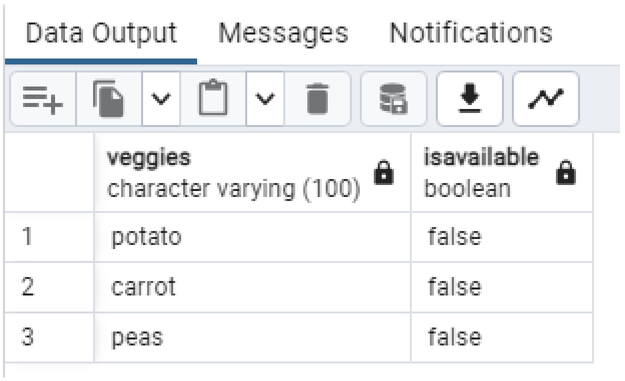


	veggies character varying (100)	isavailable boolean
1	onion	true
2	tomato	true
3	potato	false
4	carrot	false
5	beans	true
6	peas	false

Returns all the record

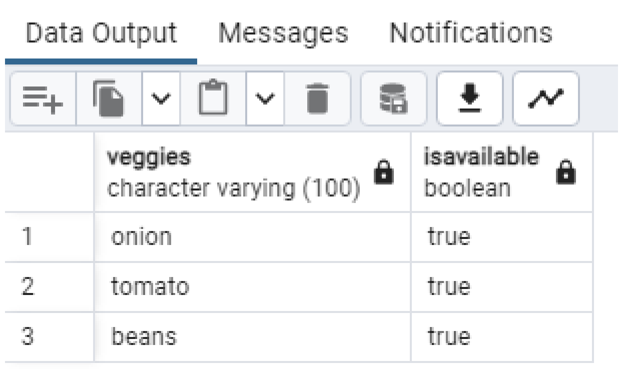
Here ‘0’ represents false and ‘1’ represents true

select * from veggieslist where isavailable='0';



	veggies character varying (100)	isavailable boolean
1	potato	false
2	carrot	false
3	peas	false

select * from veggieslist where isavailable;



	veggies character varying (100)	isavailable boolean
1	onion	true
2	tomato	true
3	beans	true

Here the default the isavailable is returns true.

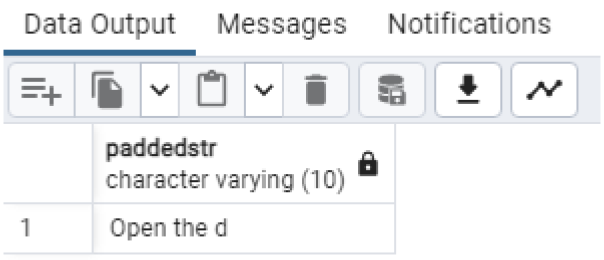
CHAR,VARHAR,TEXT datatypes:

CHAR - is series of character.
VARCHAR - is mixed of alphanumerical
CHARACTER VARYING - variable length dynamic
TEXT - same as the CHAR but it is string.

The above syntax are used in the queries with the size.

To reduce the padding size

select 'Open the door'::varchar(10) as paddedstr;



	paddedstr character varying (10)
1	Open the d

This can be used to reduce the padded string with the given str fixed length.

Numeric datatype:

Types: smallint , int , bigint
2bytes , 4bytes and 8bytes each.

And Serial is one of the numeric datatype ANSI SQL identify the column.

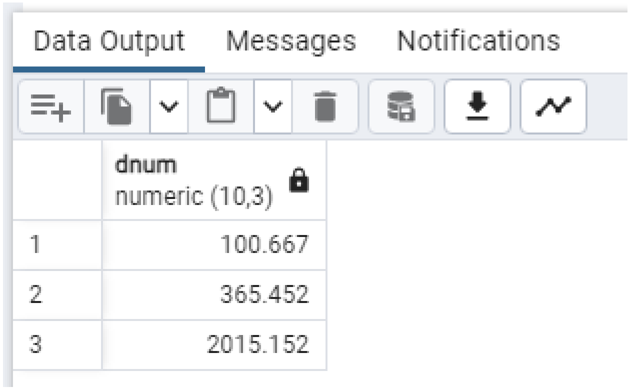
The examples are done at the previous table creation.

Decimal Datatypes:

create table decimalval (dnum numeric(10,3));
Created table with the condition

insert into decimalval values(2015.15164);
Inserted values with the values

select * from decimalval;



	dnum numeric (10,3)
1	100.667
2	365.452
3	2015.152

It retrieves the values with the 3 decimal as given.

Date/time Datatypes:
Date , Time , Timestamp and Timestampz

Date - yyyy-mm-dd

Time - HH:MM , HH:MM:SS , HHMMSS

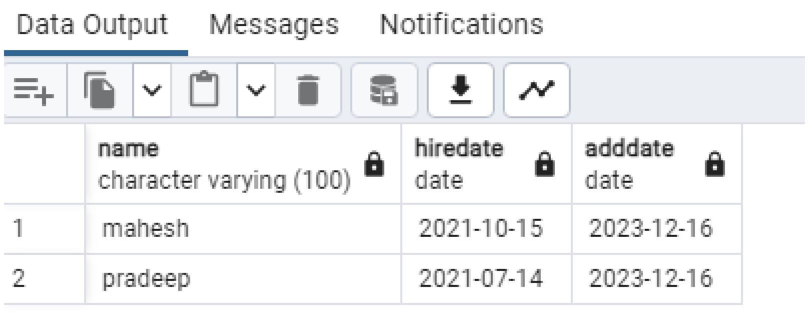
Timestamp - as format

Timestampz - as format

```
create table testdate(name varchar(100),
                                        hiredate date not null,
                                        adddate date default current_date);
```
Created table with parameters

```
insert into testdate(name,hiredate) values('mahesh','2021-10-15'),('pradeep','2021-07-14');
```
Inserted data

```
select * from testdate;
```



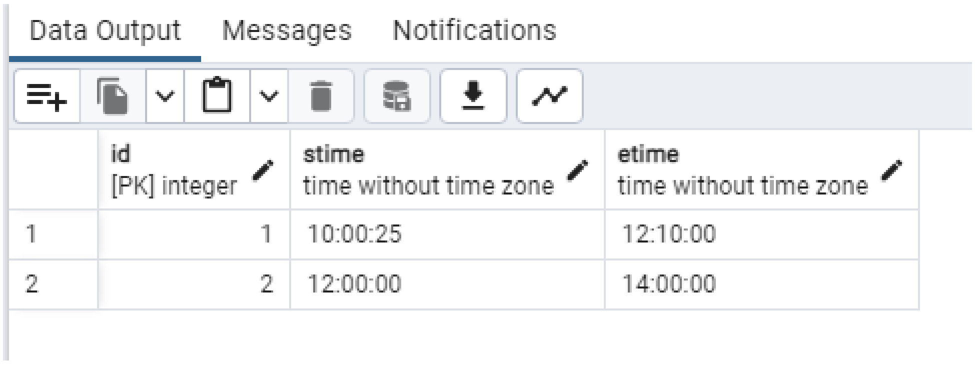
	name character varying (100)	hiredate date	adddate date
1	mahesh	2021-10-15	2023-12-16
2	pradeep	2021-07-14	2023-12-16

Retrieving the data.

```
create table testtime(id serial primary key,
                                        stime time not null,
                                        etime time not null);
```
Created table to test time

```
insert into testtime values(1,'10:00:25','12:10:00'),(2,'12:00:00','14:00:00');
```
Inserting values to table

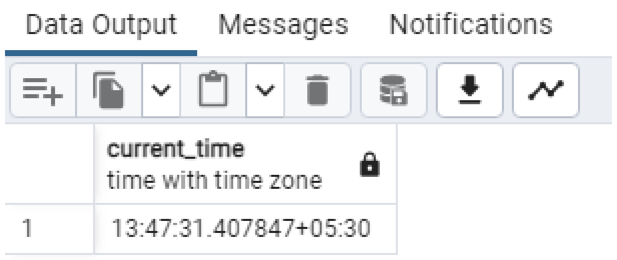
select * from testtime;


Data Output | Messages | Notifications

	id [PK] integer	stime time without time zone	etime time without time zone
1	1	10:00:25	12:10:00
2	2	12:00:00	14:00:00

Retrieving data from the table

select current_time;


Data Output | Messages | Notifications

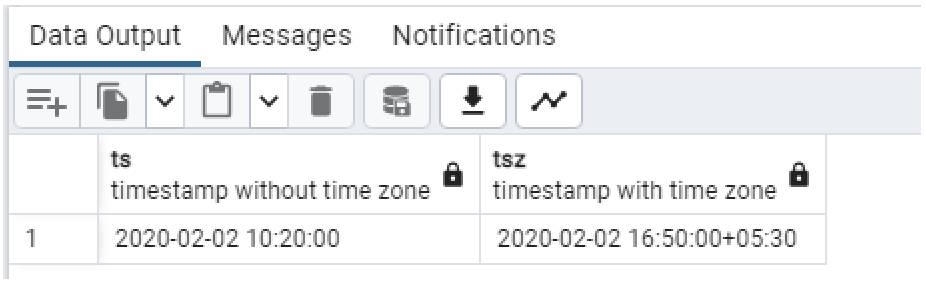
	current_time time with time zone
1	13:47:31.407847+05:30

Some time operations are done in arithmetic operators also.

create table testts (ts TIMESTAMP,tsz TIMESTAMPTZ);
Created table for timestamp and timestamtz

insert into testts values('2020-02-02 10:20:00-01','2020-02-02 10:20:00-01');
Inserting values

select * from testts;


Data Output | Messages | Notifications

	ts timestamp without time zone	tsz timestamp with time zone
1	2020-02-02 10:20:00	2020-02-02 16:50:00+05:30

Retrieving data.

UUID in PostgreSQL:

create extension if not exists "uuid-ossp";
Create extension for uuid.

select uuid_generate_v1();

select uuid_generate_v4();

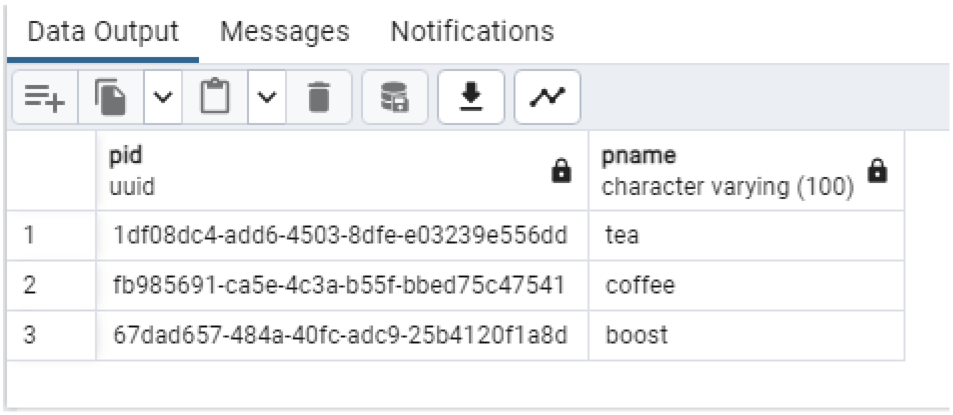
It will return the unique id

```
create table testuuid(

            pid uuid default uuid_generate_v4(),
            pname varchar(100)
);
```

Created table with input of uuid

insert into testuuid(pname) values('tea'),('coffee'),('boost');
Inserting values

select * from testuuid;



Data Output   Messages   Notifications

	pid uuid	pname character varying (100)
1	1df08dc4-add6-4503-8dfe-e03239e556dd	tea
2	fb985691-ca5e-4c3a-b55f-bbed75c47541	coffee
3	67dad657-484a-40fc-adc9-25b4120f1a8d	boost

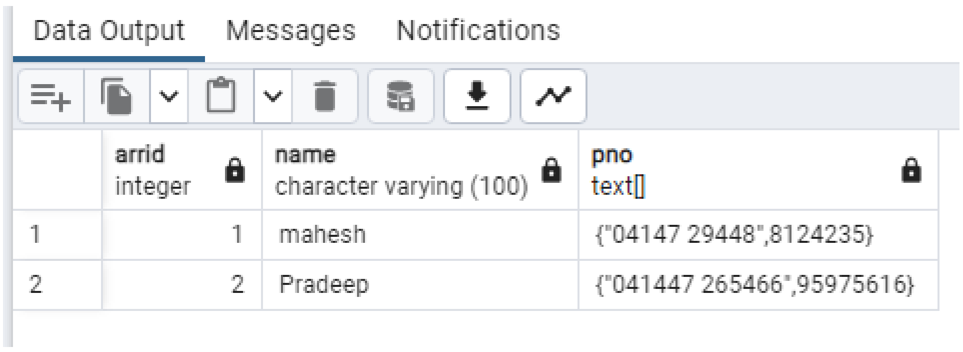
All data with unique uuid.

ARRAYS in PostgreSQL;

create table testarr (arrid serial,name varchar(100),pno text[]);
Created table with arr

insert into testarr(name,pno) values ('mahesh',ARRAY['04147 29448','8124235']),('Pradeep',ARRAY['041447 265466','95975616']);
Inserted values

select * from testarr;

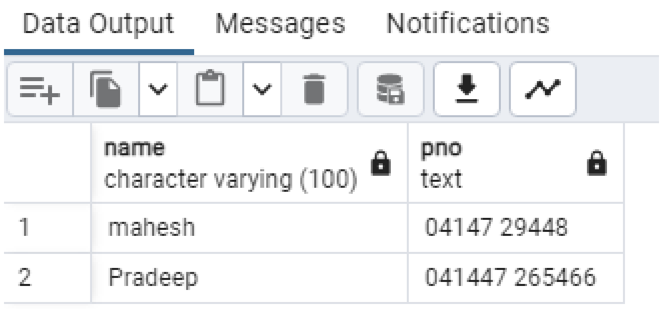


	arrid integer	name character varying (100)	pno text[]
1	1	mahesh	{"04147 29448",8124235}
2	2	Pradeep	{"041447 265466",95975616}

All data into the array.

To extract the data from array

select name,pno[1] from testarr;



	name character varying (100)	pno text
1	mahesh	04147 29448
2	Pradeep	041447 265466

It will extract the data from the index.

Hstore in postgresql:
create extension if not exists hstore;

create table tesths (bid serial primary key,title varchar(200) not null, info hstore);

insert into tesths(title,info) values('Igikai',

'

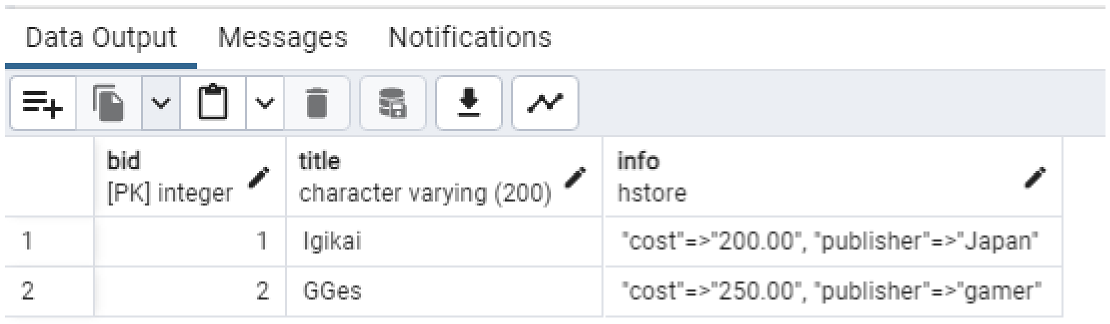
"publisher"=>"Japan","cost"=>"200.00" '),('GGes','

"publisher"=>"gamer",

"cost"=>"250.00"

');

select * from tesths;

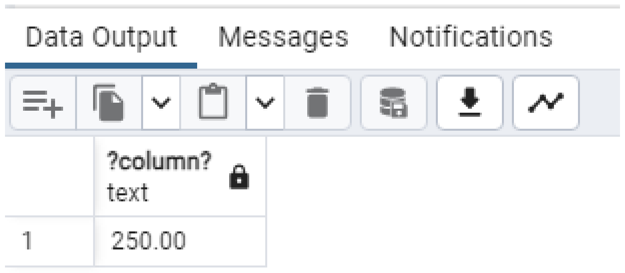


Data Output Messages Notifications

	bid [PK] integer	title character varying (200)	info hstore
1	1	Igikai	"cost"=>"200.00", "publisher"=>"Japan"
2	2	GGes	"cost"=>"250.00", "publisher"=>"gamer"

Basically this hstore is used to store the address of the column data

select info->'cost' from tesths where bid=2;

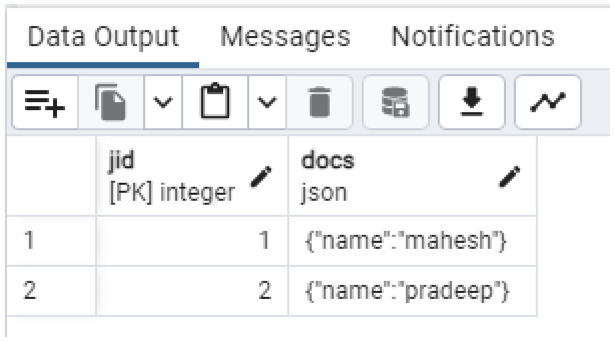


Data Output Messages Notifications

	?column? text
1	250.00

JSON in postgreSQL:

create table testjson (jid serial primary key,docs json);
insert into testjson(docs) values('{"name":"mahesh"}'),('{"name":"pradeep"}');
select * from testjson;



Data Output Messages Notifications

	jid [PK] integer	docs json
1	1	{"name":"mahesh"}
2	2	{"name":"pradeep"}

alter table testjson alter column docs type jsonb;

select * from testjson where docs @>'1';

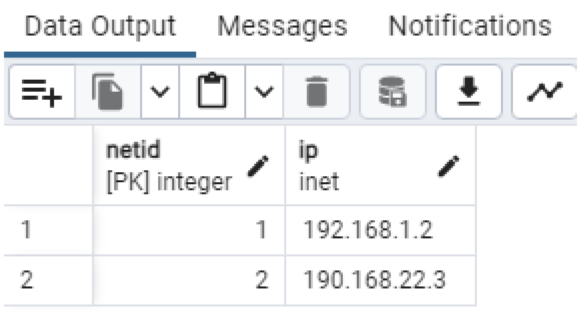
Network Addresses in postgresql:

Network Address Types:
Cidr , inet , macaddr , macaddr8

create table testadd(netid serial primary key,ip inet);

insert into testadd(ip) values('192.168.1.2'),('190.168.22.3');

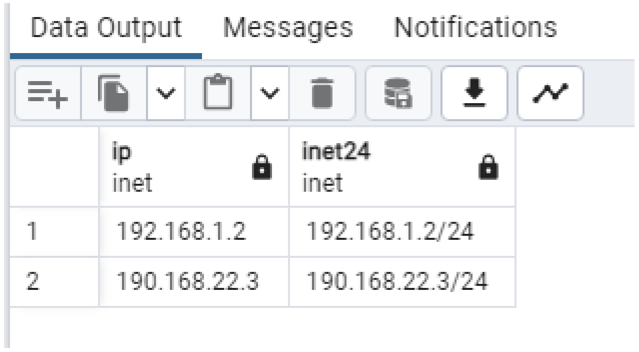
select * from testadd;



	netid [PK] integer	ip inet
1	1	192.168.1.2
2	2	190.168.22.3

To change it into 24bit

select ip,set_masklen(ip,24) as inet24 from testadd;



	ip inet	inet24 inet
1	192.168.1.2	192.168.1.2/24
2	190.168.22.3	190.168.22.3/24

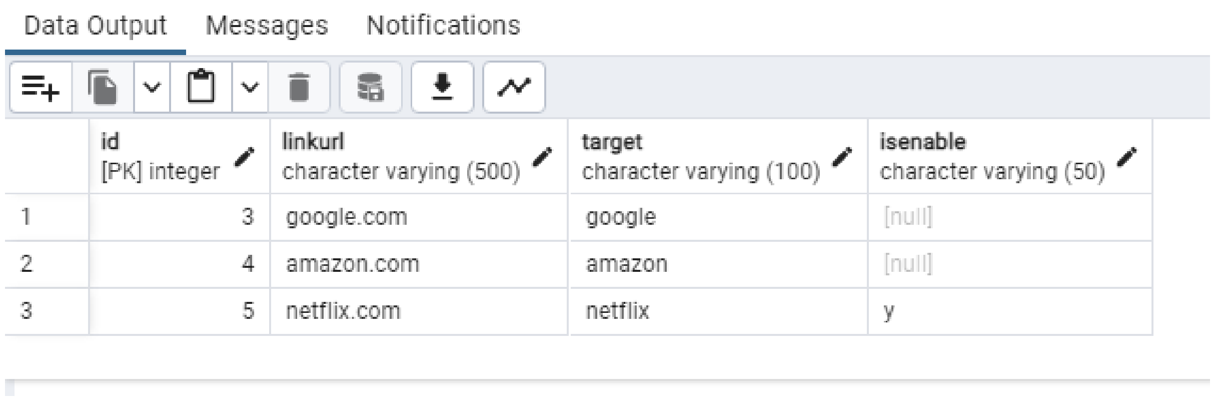
For all the type of address set_masklen(addtype,bit)

Modify and alter table values and constraints
create table wlink(id serial primary key,linkurl varchar(500),target varchar(100));

insert into wlink(linkurl,target) values('google.com','google'),('amazon.com','amazon');

alter table wlink add constraint linkaddr unique(linkurl);

```
select * from wlink;

alter table wlink add column isenable varchar(50);

insert into wlink(linkurl,target,isenable) values('netflix.com','netflix','y');
```



	id [PK] integer	linkurl character varying (500)	target character varying (100)	isenable character varying (50)
1	3	google.com	google	[null]
2	4	amazon.com	amazon	[null]
3	5	netflix.com	netflix	y