

Inheritance in PostgreSQL:

```
create table parent(pid serial primary key,pname text);
```

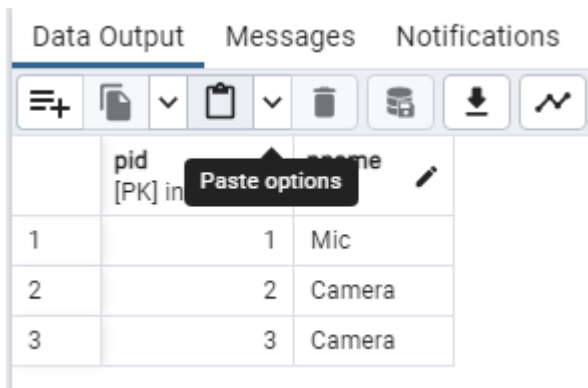
```
create table child() inherits(parent);
```

```
alter table child add constraint childpk primary key(pid);
```

```
insert into parent(pname) values('Mic'),('Camera');
```

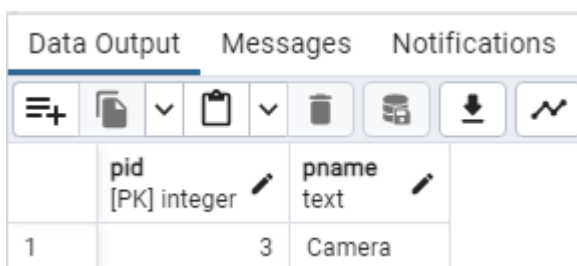
```
insert into child(pname) values('Camera');
```

```
select * from parent
```



	pid [PK] integer	pname text
1	1	Mic
2	2	Camera
3	3	Camera

```
select * from child
```



	pid [PK] integer	pname text
1	3	Camera

Partition Types in PostgreSQL:

Range,List and Hash.

Partition by Range:

```
Create table tname(var dt,...) partition by range(var);
```

Partition of:

```
Create table tname partition of prevtname for values () to ();
```

Thus is common for further more partition in the table

Partition By List:

Create table tname (var dt) partition by list(var)

Partition by Hash:

Create table tname (var dt) partition by hash(var)

<https://www.postgresql.org/docs/current/ddl-partitioning.html>

Refer the above link for partitioning examples;

SQL Functions:

create function myfunc(int,int) returns int as

,

select 1+2

language sql

select myfunc(1,2);

Data Output		Messages	Notifications
	myfunc integer		
1	3		

Dollar quote in function:

create function myfuncmul(int,int) returns int as

\$\$

select 1*2

\$\$

language sql

Data Output		Messages	Notifications
	myfuncmul integer		
1	2		

Function returning no values

create function custlname() returns void as

\$\$

update customers set last_name='Gracia'

where first_name='Hannah'

\$\$

language sql

select custlname()

To run this function

Data Output Messages Notifications			
	customer_id [PK] integer	first_name character varying (100)	last_name character varying (255)
1	1	John	Doe
2	2	Jeff	Smith
3	3	Mike	Steel
4	4	Mark	Benjamin
5	5	Hannah	Gracia

Last_name changed to given.

Function with no return occurs when the operations done like insert, update and delete.

Functions returns a single value:

create or replace function prodminprice() returns real as

\$\$

select min(unit_price) from products

\$\$

language sql

select prodminprice();

Data Output Message	
	prodminprice real
1	20

--- total revenue by each movie id

create function totalrev() returns real as

\$\$

```
select sum(revenue_dom+revenue_int) as "TotalRevenue" from movierev
```

\$\$

language sql

```
select totalrev()
```

Data Output		Messages	Notifications
	totalrev real		
1	12284.5		

To drop a function:

Drop function fname;

--- total movies by languages

create function langfunc(p_lang varchar) returns character varying as

\$\$

```
select movname from movie where movlang = p_lang
```

\$\$ language sql

```
select langfunc('Japanese')
```

Here we can give any language as input to run and get the result

Data Output		Messages	Notifications
	langfunc character varying		
1	Battle Royale		

--- to return the director name by nationality

create function dir_nation(p_nationality varchar) returns varchar as

\$\$

```
select (fname||' '||lname) as "Dir_name" from director  
where nationality = p_nationality
```

\$\$

language sql

```
select dir_nation('American');
```

Data Output		Messages	Notifications
	dir_nation character varying		
1	Paul Anderson		

```
select dir_nation('French');
```

Data Output		Messages	Notifications
	dir_nation character varying		
1	Luc Besson		

Functions also be used to return an table.

PL/SQL and Pg/PISQL:

--- to return max collection at domestic

```
create or replace function fn_man_revdom() returns int8 as
$$
begin
    return max(revenue_dom) as "MaxRevDom" from movierev;
end;
$$
language plpgsql
```

```
select fn_man_revdom()
```

Data Output		Messages	Notifications
	fn_man_revdom bigint		
1	659		

By Declaring variables

```
do
$$
    declare
    myname text:='Mahesh';
    myid int:=101;
    hdate date:='2023-12-12';

    begin
    raise notice
    'My Info % % %',
    myname,
    myid,
    hdate;
    end;
$$
```

Data Output	Messages	Notifications
NOTICE: My Info Mahesh 101 2023-12-12 DO		
Query returned successfully in 54 msec.		

This query is used to declare the variable that rename to the existing variable declared.

Newname alias for oldname

```
do
$$
    declare
        moviename movie.movname%TYPE;
    begin
        select
        movname
        from movie
        into moviename
        where movid=10;
        raise notice 'Movie Id 10 is % ',moviename;

    end;
$$
```

Data Output	Messages	Notifications
NOTICE: Movie Id 10 is Eyes Wide Shut DO		
Query returned successfully in 52 msec.		

--- in out without returns

```
create function fn_sum(in x int,in y int,out z int) as
$$
begin
z=x*y;
end;
$$
language plpgsql
```

```
select fn_sum(10,25)
```

Data Output	Messages	Notifications
<div> <div>≡+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> </div>		
	fn_sum integer	🔒
1	250	

--- return query results

```
create function fn_name() returns setof director as
$$
begin
return query
select (fname||' '||lname) as "Director_name" from director;
end;
$$
language plpgsql;
```

```
select * from fn_name()
```

Data Output	Messages	Notifications
<div> <div>≡+</div> <div>📄</div> <div>▼</div> <div>📋</div> <div>▼</div> <div>🗑️</div> <div>🗄️</div> <div>⬇️</div> <div>📈</div> </div>		
	Director_name text	🔒
1	Tomas Alfredson	
2	Paul Anderson	
3	Wes Anderson	
4	Richard Ayoade	
5	Luc Besson	

Control Structures :

Condition statement

Loop statement

Exception statement

Conditinal statement: if ,if else, nested if.

--- if statement to check length of fname and lname

```
create function fn_check(p_fname varchar,p_lname varchar) returns text as
$$
begin
    if char_length(p_fname)>char_length(p_lname) then
        return 'fname is Greater';

    elsif char_length(p_fname)=char_length(p_lname) then
        return 'Equal Length';

    else
        return 'lname is Greater';
    end if;
end;
$$
language plpgsql;
```

select fn_check('mahesh','vaithi')

Data Output			Messages	Notifications
	fn_check	text		
1	Equal Length			

select fn_check('mahesh','v')

Data Output			Messages	Notif
	fn_check	text		
1	fname is Greater			


```
select fn_check('m','vaithi')
```

Data Output			Messages	Notifications
	fn_check	text		
1	Iname is Greater			

--- if statement in movierevenue table

```
create function fn_coll(revdom real) returns text as  
$$
```

```
begin
```

```
    if revdom>120 then  
        return 'High Collection';  
    elsif revdom = 120 then  
        return 'Average Collection';  
    else  
        return 'Low Collection';  
    end if;
```

```
end;
```

```
$$
```

```
language plpgsql;
```

```
select fn_coll(revenue_dom) from movierev
```

Data Output			Messages	Notifications
	fn_coll	text		
1	Low Collection			
2	High Collection			
3	Low Collection			
4	High Collection			
5	Low Collection			

Loop Statements in PostgreSQL:

```
Loop
```

```
Exit;
```

```
Exit when
```

```
Endloop;
```

--- Loops in PostgreSQL

```

do
$$
begin
    for counter in 1..100
    loop
        raise notice 'counter %',counter;
    end loop;

end;
$$

```

--- foreach array

```

do
$$
declare
    arr1 int[]:=array[10,20,3,100];
    ar int;
begin
    foreach ar in array arr1
    loop
        raise info 'array ele %',ar;
    end loop;
end;
$$ language plpgsql;

```

Data Output	Messages	Notifications
INFO: array ele 10		
INFO: array ele 20		
INFO: array ele 3		
INFO: array ele 100		
DO		
Query returned successfully in 58 msec.		

--- while loop

```

do
$$
declare
counter int:=1;
endcounter int:=100;

begin

    while counter<endcounter

```

end;

Data Output	Messages	Notifications
INFO: increment 2		
INFO: increment 3		
INFO: increment 4		
INFO: increment 5		
INFO: increment 6		
INFO: increment 7		
INFO: increment 8		
INFO: increment 9		
INFO: increment 10		
INFO: increment 11		
INFO: increment 12		

```
--- exception handling
```

do

```
--- stored procedures
```

```
create procedure dirname(firstname text,lastname text,dir_id int)
```

```

as
$$
begin
    select (fname||' '||lname) as "DirName",d_id from director;
end;
$$ language plpgsql;

```

```
select dirname(firstname,lastname,dir_id);
```

Data Output		Messages	Notifications
<div><div><div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div><div><div></div><div></div></div></div></div></div>			
	<div><div>DirName</div><div>text</div><div></div></div>	<div><div></div><div></div></div>	<div><div>d_id</div><div>[PK] integer</div><div></div></div>
1	Tomas Alfredson		1
2	Paul Anderson		2
3	Wes Anderson		3
4	Richard Ayoade		4
5	Luc Besson		5

To drop procedure

```
Drop procedure procname;
```

Triggers in PostgreSQL;

A trigger can be associated with Table,View or Foreign Table.

Types of Triggers:

RowLevel , StatementLevel

RowLevel- used to do operations on the multiple row level

StatementLevel - used to do operations on each statement level.

Trigger Table:

Before - Do all Operations insert , update , delete

After - same as above

Instead of - Truncate.

Pros and Cons of Triggers:

--- triggers in postgresql

syntax:

```
create trigger trigger_name()  
{before|after} {event}  
on table_name  
[for [each] {row|statement}]  
execute procedure trigger_name
```