After Join

Another look at lateral join

- Problem: show 5 items for each group
- Very simple table
  - first column has 9 distinct values. This will be the groups
  -

```
gtowell=# select *
from lt limit 20;
 acto | acti
------+------
 F    | A
 K    | B
 J    | 6
 J    | I
 L    | U
 J    | 7
 G    | J
 S    | 4
 S    | W
 L    | 7
 J    | 9
 J    | 2
 J    | I
 H    | Z
 K    | E
 G    | Z
 G    | R
 F    | B
 J    | 3
 D    | Q
(20 rows)
```

```
-- 5 acti where acto=F
select * from lt where acto='F' limit 5;

acto | acti
------+------
 F    | A
 F    | B
 F    | D
 F    | 8
 F    | C
(5 rows)
```

Postgres specific group aggregating function. Takes the field and creates an array

```
-- rather than a 5 row table, one row with an array of 5
select acto, array_agg(acti) from lt where acto='F' group by acto limit 5;
F    |
{A,B,D,8,C,I,M,9,2,A,8,K,W,6,6,9,O,4,N,9,W,Z,Y,Q,E,M,X,H,O,J,F,7,T,1,D,4,M
,G,7,W,B,3,G,3,5,S,9,A,K,D,3,F,H,A,5,F,G,L,Y,P,N,F,C,7,T,F,X,R,H,C,4,X,4,C
,7,A,O,F,N,F,K,4,7,E,C,0,9,3,O,S,O,U,G,E,K,H,8}
```

Problem

```
with aaa as (select * from lt where acto='F' limit 5)
select acto, array_agg(acti)
from aaa
group by acto;

 acto |  array_agg
------+-------------
 F    | {A,B,D,8,C}
```

Keep Going Laterally

```
select *
from (select distinct acto as dact from lt) as dlt
     join lateral (select acto, acti from lt where acto=dlt.dact limit 5) as elt
          on elt.acto=dact
order by acto, acti;
```

```
 dact | acto | acti
------+------+------
 A    | A    | 7
 A    | A    | M
 A    | A    | Q
 A    | A    | V
 A    | A    | Z
 D    | D    | 8
 D    | D    | D
 D    | D    | I
```

Point of lateral join is allow this subquery which uses a field in the thing being joined to

Lateral join is otherwise an inner join so it needs an "on" condition (or having is natural)

```
select dact, array_agg(acti)
from (select distinct acto as dact from lt) as dlt
     join lateral (select acto, acti from lt where acto=dlt.dact  limit 5) as elt
          on elt.acto=dact
group by dact
order by dact;
```

```
 dact |  array_agg
------+-------------
 A    | {M,Z,V,7,Q}
 D    | {Q,8,P,I,D}
 F    | {A,B,D,8,C}
 G    | {Z,X,H,R,J}
 H    | {T,Z,X,L,X}
 J    | {6,2,9,7,I}
 K    | {B,V,W,I,E}
 L    | {U,7,4,Q,9}
 S    | {7,W,C,4,B}
```

Postgres specific. Selects part of array.
Arrays are 1 indexed in postgres

Postgres specific.The length of array in dimension 1

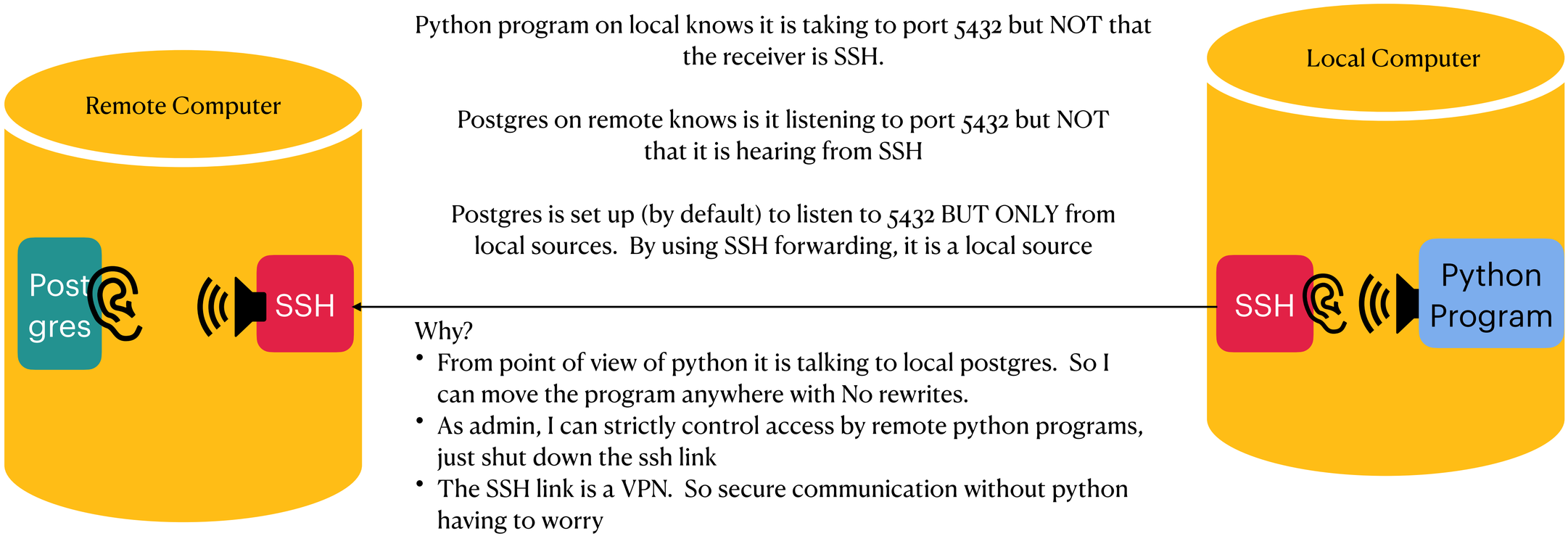
```
with aaa as (select acto, array_agg(acti) as arr from lt group by acto)
select acto, arr[1:5], array_length(arr, 1)
from aaa
order by acto;
```

This solution only works because the original solution put things into an array!! If the goal had been the table at top, then no. (can unnest(arr[1:5]))

Local Forward in SSH

For example using python and postgres

Recall: LocalForward 5432 localhost:5432



Python program on local knows it is taking to port 5432 but NOT that the receiver is SSH.

Postgres on remote knows is it listening to port 5432 but NOT that it is hearing from SSH

Postgres is set up (by default) to listen to 5432 BUT ONLY from local sources. By using SSH forwarding, it is a local source

Why?

- From point of view of python it is talking to local postgres. So I can move the program anywhere with No rewrites.
- As admin, I can strictly control access by remote python programs, just shut down the ssh link
- The SSH link is a VPN. So secure communication without python having to worry
- Run compute heavy python NOT on DB server

use pg8000 rather than psycopg2

Listening Ports on 165.106.10.170

Table 1

[gtowell@loin	~]>	ss		-tulpn |	grep		
tcp LISTEN	0			244	POSTGRE	127.0.0.1:5432	0.0.0.0:*
tcp LISTEN	0			511		127.0.0.1:45913	0.0.0.0:*
tcp LISTEN	0			244		127.0.0.1:5433	0.0.0.0:*
tcp LISTEN	0			128		127.0.0.1:6011	0.0.0.0:*
tcp LISTEN	0			511		127.0.0.1:45787	0.0.0.0:*
tcp LISTEN	0			511		127.0.0.1:43291	0.0.0.0:*
tcp LISTEN	0			511		127.0.0.1:45531	0.0.0.0:*
tcp LISTEN	0			511		127.0.0.1:43739	0.0.0.0:*
tcp LISTEN	0			511		127.0.0.1:44637	0.0.0.0:*
tcp LISTEN	0			100		127.0.0.1:49152	0.0.0.0:*
tcp LISTEN	0			70		127.0.0.1:33060	0.0.0.0:*
tcp LISTEN	0			4096		127.0.0.1:27017	0.0.0.0:*
tcp LISTEN	0			151		127.0.0.1:3306	0.0.0.0:*
tcp LISTEN	0			4096		0.0.0.0:111	0.0.0.0:*
tcp LISTEN	0			511		127.0.0.1:43699	0.0.0.0:*
tcp LISTEN	0			511		127.0.0.1:40563	0.0.0.0:*
tcp LISTEN	0			511		127.0.0.1:41813	0.0.0.0:*
tcp LISTEN	0			4096	DNS	0.0.53%lo:53	0.0.0.0:*
tcp LISTEN	0			128	ssh	0.0.0.0:22	0.0.0.0:*
tcp LISTEN	0			5	ipps	127.0.0.1:631	0.0.0.0:*
tcp LISTEN	0			128	X window	[::1]:6011	[::]:*
tcp LISTEN	0			4096	sun RPC	[::]:111	[::]:*
tcp LISTEN	0			511	http	*:80	*:*
tcp LISTEN	0			128	ssh	[::]:22	[::]:*
tcp LISTEN	0			5	ipps	[::1]:631	[::]:*

SQL Problem

- From each department, find the names of the people earning the 2 highest salaries

- As a starter, just find the 2 highest salaries in each department

id	name	dept_name	salary
31955	Moreira	Accounting	71351.42
79081	Ullman	Accounting	47307.10
43779	Romero	Astronomy	79070.08
63287	Jaekel	Athletics	103146.87
16807	Yazdi	Athletics	98333.65
81991	Valtchev	Biology	77036.18
80759	Queiroz	Biology	45538.32
34175	Bondi	Comp. Sci.	115469.11
3335	Bourrier	Comp. Sci.	80797.83
90376	Bietzk	Cybernetics	117836.50
63395	McKinnon	Cybernetics	94333.99

- From each department, find the names of the people earning the 2 highest salaries

```
with aaa as (select distinct dept_name from instructor),
     bbb as (select aaa.dept_name as deptt, salary as sall
                    from aaa join lateral (select dept_name, salary from instructor sii
                                           where sii.dept_name=aaa.dept_name
                                           order by salary desc
                                           limit 2) as lsi
                                 on lsi.dept_name=aaa.dept_name
                    order by aaa.dept_name asc, lsi.salary desc)

select id, name, dept_name, salary
from instructor as i1, bbb
where i1.dept_name=bbb.deptt and i1.salary=bbb.sall
order by dept_name asc, salary desc;
```