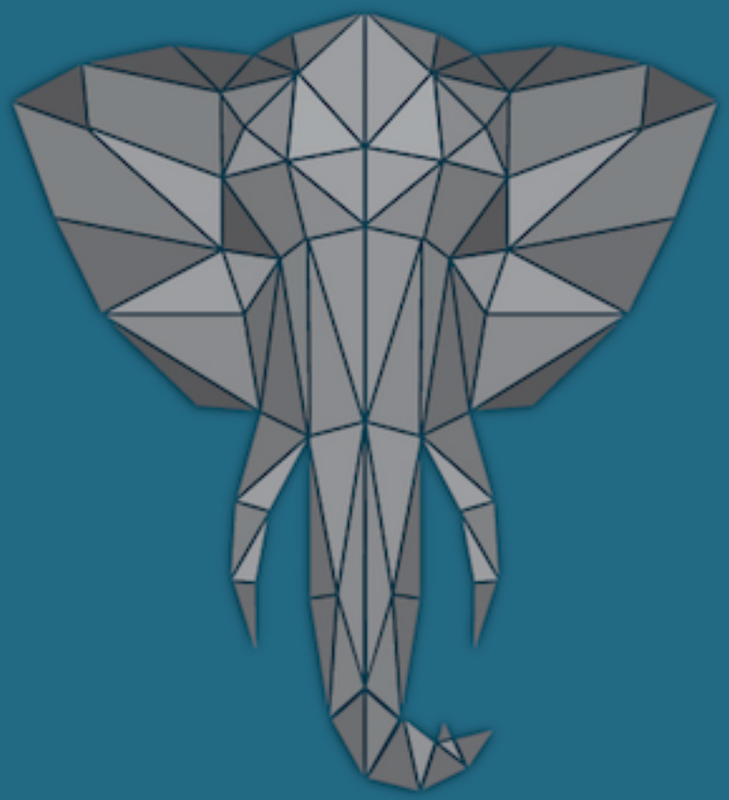
The 10 mightiest SQL queries in the world

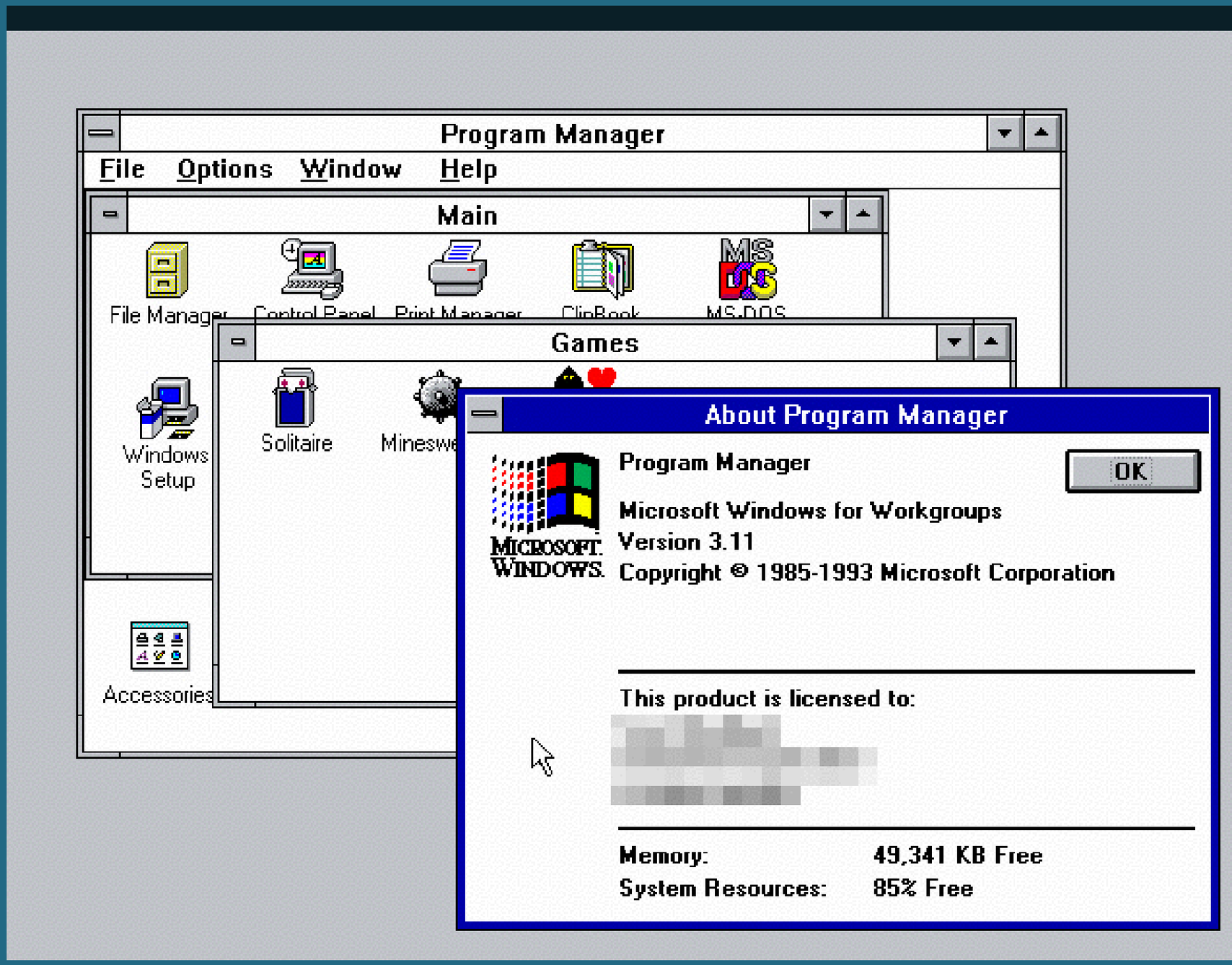


ONGRES

Pablo Gonzalez Doval

ONGRES


SONY
AUTO REVERSE
20 PRESETS
AVLS
ANTI-ROLLING MECHANISM
TUNING
1
2
3
4
5
WALKMAN
WM-FX421
FM/AM


Program Manager
File Options Window Help
Main
File Manager
Control Panel
Print Manager
ClipBook
MS-DOS
Games
Solitaire
Windows Setup
Accessories
About Program Manager
Program Manager
Microsoft Windows for Workgroups
Version 3.11
Copyright © 1985-1993 Microsoft Corporation
OK
This product is licensed to:
Memory: 49,341 KB Free
System Resources: 85% Free

PABLO GONZALEZ

DEV

Working @ OnGres (www.ongres.com)
Software Developer
PostgreSQL support

@dovaleac

pgdoval

Post-92

1. generate_series()
2. WITH RECURSIVE
3. LATERAL
4. Window functions
5. Grouping sets
6. Moving data atomically
7. Run WHATEVER you want

[EX1] Premium users

[EX1] Premium users

scoring

user	position
3	1
4	2
2	3
1	4

premium

user	position
4	1
3	3

user	position
4	1
2	2
3	3
1	4

1.

generate_series()

1. generate_series()

```
SELECT * FROM generate_series(1,8);
```

generate_series
1
2
3
4
5
6
7
8

1. generate_series()

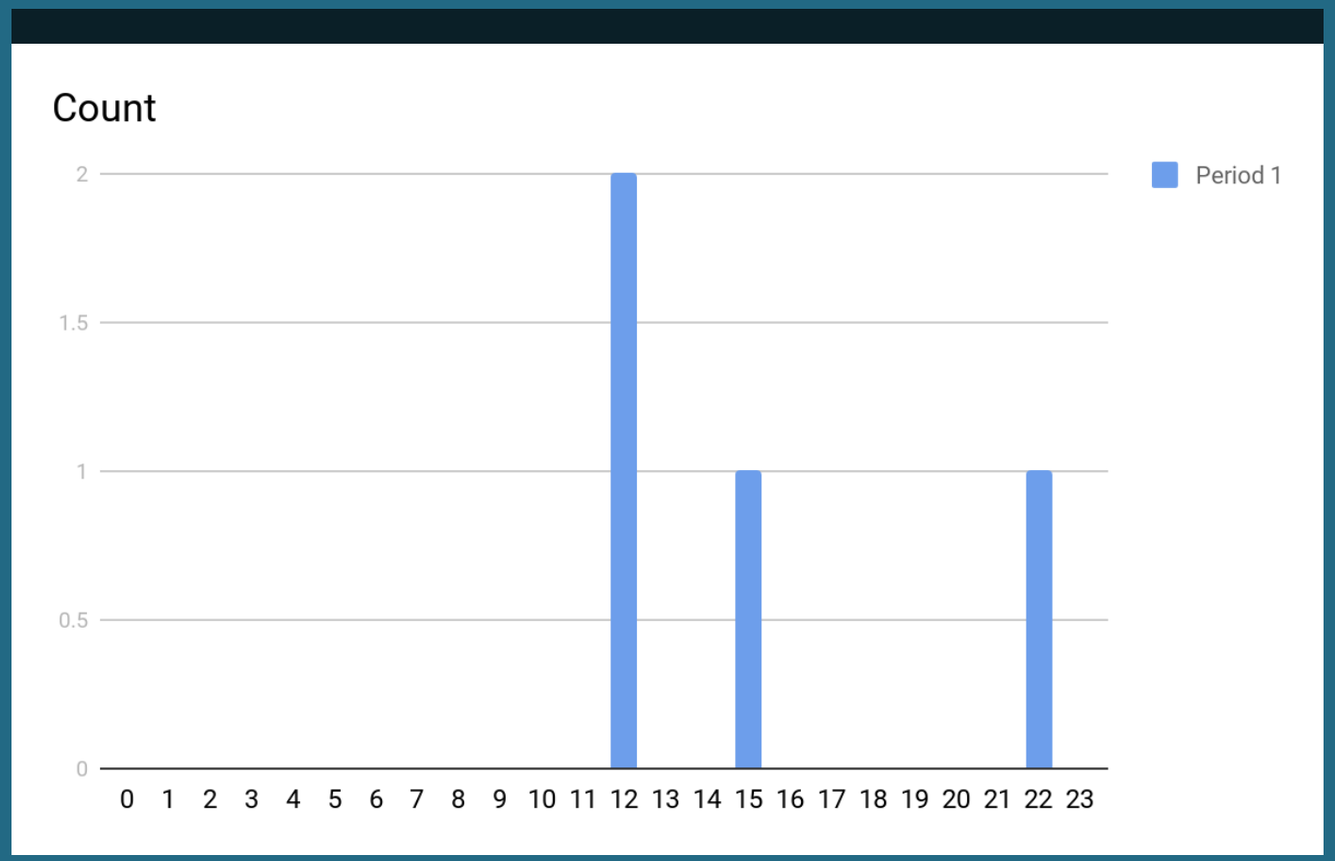
```
SELECT * FROM generate_series(1, 5, 2);
```

generate_series
1
3
5

1. generate_series()

Generate count of events per hour

id	t
1	12:23
2	12:55
3	15:06
4	22:47



1. generate_series()

```
WITH hours AS (
    SELECT extract(HOUR FROM t) as hour
    FROM events
)
SELECT hour, count(*)
FROM hours
GROUP BY hour;
```

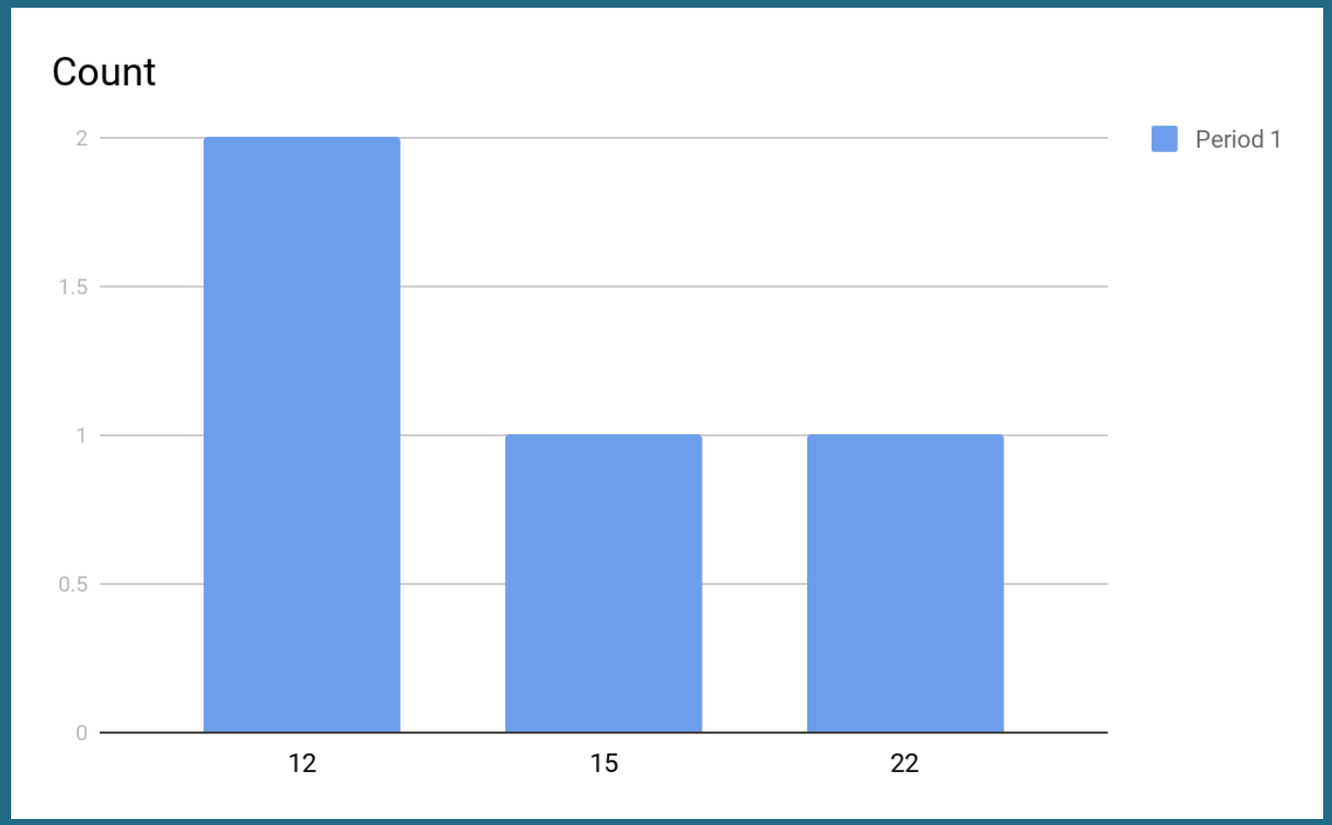

1. generate_series()

Does it work?

1. generate_series()

Generate event count per hour

id	t
1	12:23
2	12:55
3	15:06
4	22:47



1. generate_series()

```sql
WITH hours AS (
    SELECT extract(HOUR FROM t) as hour
    FROM events
),
hour_count AS(
    SELECT hour, count(*) as t
    FROM hours
    GROUP BY hour
)
SELECT e.h, coalesce(hc.t, 0)
FROM hour_count hc
RIGHT JOIN (SELECT generate_series(0,23) as h) e
ON hc.hour = e.h;
```


2. WITH RECURSIVE

2. WITH RECURSIVE

```
WITH RECURSIVE t(n) AS (
VALUES (1)
UNION ALL
SELECT n+1 FROM t WHERE n < 100
) SELECT sum(n) FROM t;
```

sum
5050

2. WITH RECURSIVE

id	name	parent_id
1	good1	-
2	wrong1	-
3	good2	1
4	good3	2
5	wrong2	1
6	good4	3
7	good5	4

id	name	parent_id
1	good1	-
3	good2	1
6	good4	3

name NOT LIKE ‘wrong%’

2. WITH RECURSIVE

```sql
WITH RECURSIVE flattened(id) AS (
    SELECT id
    FROM t
    WHERE parent_id IS NULL
    AND name NOT LIKE 'wrong%'

    UNION ALL

    SELECT t.id
    FROM t
    JOIN flattened
    ON flattened.id = t.parent_id
    AND parent_id IS NOT NULL
    WHERE name NOT LIKE 'wrong%'
)
SELECT t.*
FROM t
JOIN flattened
ON flattened.id = t.id;
```


[EX1] Premium users

[EX1] Premium users

scoring

user	position
3	1
4	2
2	3
1	4

premium

user	position
4	1
3	3

user	position
4	1
2	2
3	3
1	4

[EX1] Premium users

```
WITH premium AS
(
    select *
    from pusers
), total_users AS
(
    select count(*)
    from tusers
)
```


[EX1] Premium users

```
free AS
(
    select ROW_NUMBER() OVER(
    ORDER BY id ASC) as target_place, name
    from tusers
    WHERE name not in (select name from pusers)
)
```


[EX1] Premium users

```
series AS
(
    select s.*, ROW_NUMBER() OVER(
    ORDER BY n ASC) as target_place
    from
    (
        select generate_series as n
        from generate_series(1, (select count(*) from tusers))
    ) s
    LEFT JOIN premium
    ON premium.target_place = s.n
    where premium.name is null
)
```


[EX1] Premium users

```sql
SELECT *
FROM
(
    SELECT *
    FROM premium

    UNION ALL

    (
        SELECT
            s.n,
            f.name
        FROM series s

        JOIN free f
        ON s.target_place = f.target_place
    )
) u
ORDER BY 1 ASC;
```


[EX2] User sessions

[EX2] User sessions

events

id	user_id	time
1	1	12:40
2	1	13:05
3	2	13:40
4	1	13:45
5	2	14:02
6	1	18:23

sessions

user_id	events
1	2
1	1
1	1
2	2

*Two consecutive events by the same user belong to the same session iff t2-t1<= 30 minutes

3. LATERAL

3. LATERAL

```sql
SELECT t.*
FROM team t
JOIN stadium s
ON t.id = s.team_id
WHERE s.name = 'Riazor';
```

3. LATERAL

```sql
SELECT t.*
FROM team t
WHERE id = (
    SELECT team_id
    FROM stadium
    WHERE s.name = 'Riazor'
)
```

3. LATERAL

teams

id	name
1	Deportivo
2	Barcelona
3	Real Madrid
4	Atlético Madrid

players

id	team_id	name	value	country
1	2	Messi	150	ARG
2	2	Iniesta	100	ESP
3	2	Umtiti	70	FRA
4	3	Modric	100	CRO
5	3	Cristiano Ronaldo	95	POR
6	4	Griezmann	120	FRA

3. LATERAL

```sql
SELECT t.*, value
FROM teams t
JOIN (
    SELECT
        team_id AS id,
        sum(value) as value
    FROM players
    GROUP BY team_id
    ) v
ON t.id = v.id;
```

id	name	value
1	Deportivo	80
2	Barcelona	1000
3	Real Madrid	1000
4	Atlético Madrid	700

3. LATERAL

team	best_player	value	country
Deportivo	Emre Çolak	15	TUR
Barcelona	Messi	150	ARG
Real Madrid	Modric	100	CRO
Atlético	Griezmann	120	FRA

3. LATERAL

```sql
SELECT t.name, p.name, p.value, p.country
FROM teams t
JOIN players p
ON t.id = p.team_id
AND NOT EXISTS (
    SELECT id
    FROM players p2
    WHERE p2.team_id = t.id
    AND p2.value > p.value
);
```



LATERAL
!

3. LATERAL

```sql
SELECT t.name, p.name, p.value, p.country
FROM teams t
JOIN LATERAL (
    SELECT *
    FROM players p
    WHERE p.team_id = t.id
    ORDER BY p.value DESC
    LIMIT 1
) p
ON t.id = p.team_id;
```


4. Window functions

4. Window functions

```sql
SELECT p.team_id, p.name, p.value
FROM players p
JOIN (
    SELECT team_id, max(value) as value
    FROM players p
    GROUP BY team_id
) best
ON p.team_id = best.team_id
AND p.value = best.value;
```

team_id	name	value
1	Emre Çolak	15
2	Messi	150
3	Modric	100
4	Griezmann	120

4. Window functions

id	team_id	name	value	country
1	2	Messi	150	ARG
2	2	Iniesta	100	ESP
3	2	Umtiti	70	FRA
4	3	Modric	100	CRO
5	3	Cristiano Ronaldo	95	POR
6	4	Griezmann	120	FRA

4. Window functions

position	id	team_id	name	value	country
1	1	2	Messi	150	ARG
2	2	2	Iniesta	100	ESP
3	3	2	Umtiti	70	FRA
1	4	3	Modric	100	CRO
2	5	3	Cristiano Ronaldo	95	POR
1	6	4	Griezmann	120	FRA

4. Window functions

```
SELECT *, rank()
    OVER (
        PARTITION BY team_id
        ORDER BY value DESC
    ) AS position
FROM players
```


4. Window functions

```sql
SELECT t.name, p.name, p.value, p.country
FROM teams t
JOIN (
    SELECT *, rank()
        OVER (
            PARTITION BY team_id
            ORDER BY value DESC
        ) AS position
    FROM players
) p
ON t.id = p.team_id
WHERE p.position = 1;
```


4. Window functions

row_number()
rank()
dense_rank()
percent_rank()
cume_dist()
ntile(num_buckets integer)

lag(value anyelement [, offset integer [,default anyelement]])
lead(value anyelement [, offset integer [,default anyelement]])
first_value(value any)
last_value(value any)
nth_value(value any, nth integer)

[EX2] User sessions

[EX2] User sessions

events

id	user_id	time
1	1	12:40
2	1	13:05
3	2	13:40
4	1	13:45
5	2	14:02
6	1	18:23

sessions

user_id	events
1	2
1	1
1	1
2	2

*Two consecutive events by the same user belong to the same session iff t2-t1<= 30 minutes

[EX2] User sessions

```
WITH presessions (user_id, t, s) AS (
SELECT user_id, t, CASE WHEN t - lag(t, 1) OVER (PARTITION BY user_id
ORDER BY user_id, t) > '30 minutes'::interval THEN row_number() OVER
(PARTITION BY user_id ORDER BY user_id, t) ELSE 0 END
FROM events
)
```


[EX2] User sessions

```
sessions (user_id, t, s) AS (
SELECT user_id, t, max(s) OVER (PARTITION BY user_id ORDER BY user_id, t)
FROM presessions
)
```


[EX2] User sessions

```
SELECT DISTINCT user_id, s, count(s) OVER (PARTITION BY user_id, s ORDER
BY
user_id, s)
FROM sessions ORDER BY user_id ASC, s ASC;
```


[EX3] Palindromes

[EX3] Palindromes

words

pneumonoultramicroscopicsilicovolc
anoconiosis
pseudopseudohypoparathyroidism
floccinaucinihilipilification
antidisestablishmentarianism
supercalifragilisticexpialidocious
incomprehensibilities
honorificabilitudinitatibus
tattarrattat

palindromes

user_id	events
racard	r ACA rd
racard	RACAR d
ifisi	IFI si
ifisi	If ISI

5. Grouping sets

5. Grouping sets

```
SELECT country, team_id, avg(value)
FROM players
GROUP BY GROUPING SETS(
    country,
    team_id,
    ()
);
```

country	team_id	avg
ARG		150
CRO		100
ESP		100
FRA		95
POR		95
	1	15
	2	106.6667
	3	97.5
	4	120
		105.8333

5. Grouping sets

```
GROUP BY CUBE(country, team_id)
```

```
GROUP BY GROUPING SETS(
    (country, team_id),
    country,
    team_id,
    ()
);
```


5. Grouping sets

```
GROUP BY ROLLUP(country, region, city);
```

```
GROUP BY GROUPING SETS(
    (country, region, city),
    (country, region),
    (country),
    ()
);
```


5. Grouping sets

```
SELECT
GROUPING(country, team_id),
country, team_id, avg(value)
FROM players
GROUP BY GROUPING SETS(
    country,
    team_id,
    ()
);
```

grouping	country	team_id	avg
1	ARG		150
1	CRO		100
1	ESP		100
1	FRA		95
1	POR		95
2		1	15
2		2	106.6667
2		3	97.5
2		4	120
3			105.8333

6. Moving data atomically

6. Moving data atomically

t1

stay	id
t	1
t	2
f	3
f	4

t2

id
5

t1

stay	id
t	1
t	2

t2

id
3
4
5

6. Moving data atomically

```
WITH moved_rows AS (
    DELETE FROM t1
    WHERE stay = false
    RETURNING *
)
INSERT INTO t2
SELECT id FROM moved_rows;
```


7. Run WHATEVER you want

7. Run WHATEVER you want

DO allows you to run whatever you want:

- No parameters
- No result
- You can run any PostgreSQL compatible procedure
  - pl/pgsql
  - perl
  - python
- The procedure is not stored anywhere

[EX3] Palindromes

[EX3] Palindromes

words

pneumonoultramicroscopicsilicovolc
anoconiosis
pseudopseudohypoparathyroidism
floccinaucinihilipilification
antidisestablishmentarianism
supercalifragilisticexpialidocious
incomprehensibilities
honorificabilitudinitatibus
tattarrattat

palindromes

user_id	events
racard	r ACA rd
racard	RACAR d
ifisi	IFI si
ifisi	If ISI

https://blog.jooq.org/2017/08/22/finding-all-palindromes-contained-in-strings-with-sql/

By **@lukaseder**

[EX3] Palindromes

```
WITH RECURSIVE
  words (word) AS (
    VALUES
      ('pneumonoultramicroscopicsilicovolcanoconiosis'),
      ('pseudopseudohypoparathyroidism'),
      ('floccinaucinihilipilification'),
      ('antidisestablishmentarianism'),
      ('supercalifragilisticexpialidocious'),
      ('incomprehensibilities'),
      ('honorificabilitudinitatibus'),
      ('tattarrattat')
  )
```


[EX3] Palindromes

```
starts (word, start) AS (
  SELECT word, 1 FROM words
  UNION ALL
  SELECT word, start + 1 FROM starts WHERE start < length(word)
)
```


[EX3] Palindromes

```
palindromes (word, palindrome, start, length) AS (
  SELECT word, substring(word, start, x), start, x
  FROM starts CROSS JOIN (VALUES(0), (1)) t(x)
  UNION ALL
  SELECT word, palindrome, start, length + 2
  FROM (
    SELECT
      word,
      substring(word, start - length / 2, length) AS palindrome,
      start, length
    FROM palindromes
  ) AS p
  WHERE start - length / 2 > 0
  AND start + (length - 1) / 2 <= length(word)
  AND substring(palindrome, 1, 1) =
      substring(palindrome, length(palindrome), 1)
)
```

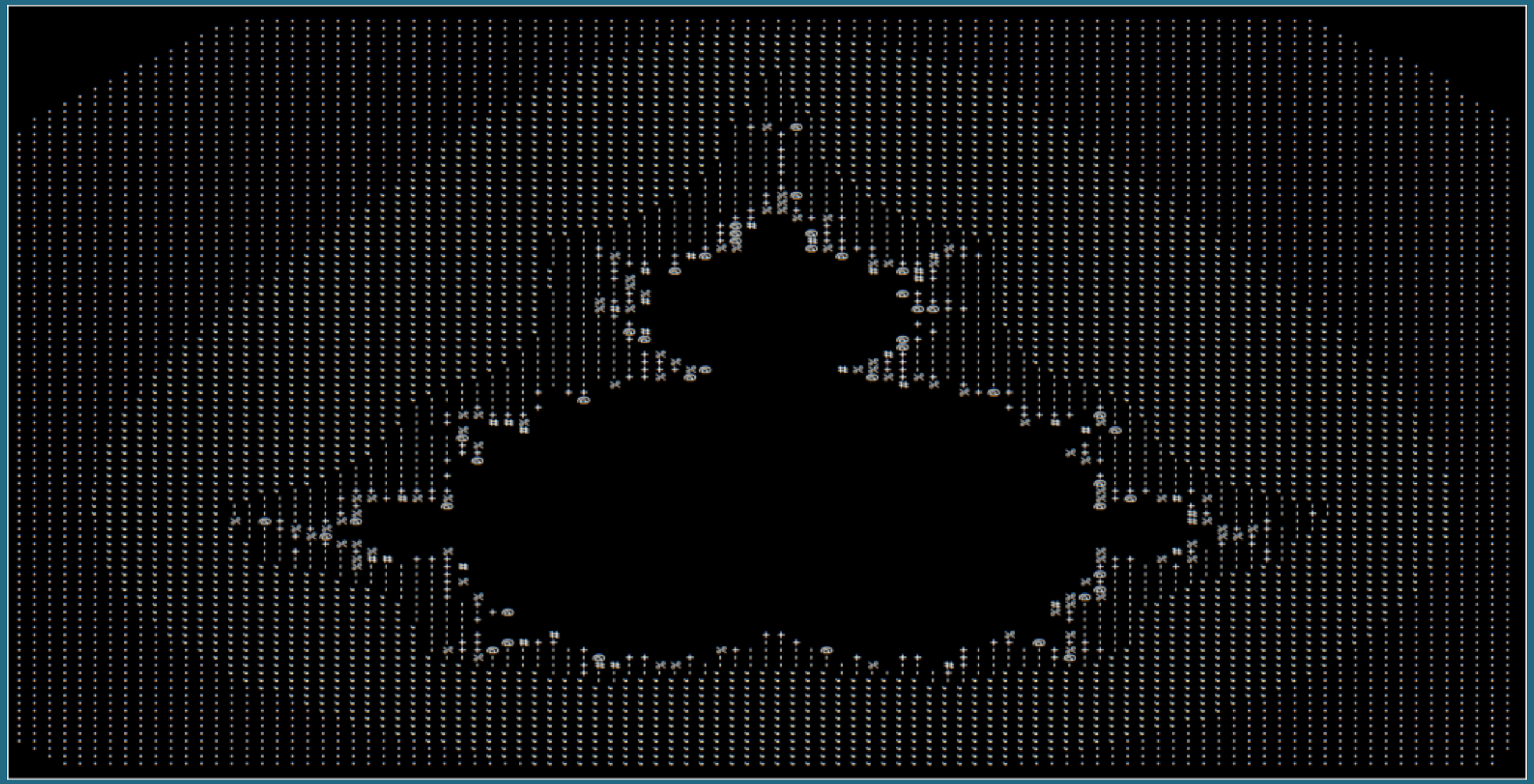

[EX3] Palindromes

```
SELECT DISTINCT
  word,
  trim(replace(word, palindrome, ' ' || upper(palindrome) || ' '))
    AS palindromes
FROM palindromes
WHERE length(palindrome) > 1
ORDER BY 2
```


Extra!

```
WITH RECURSIVE x(i) AS (
VALUES(0) UNION ALL
SELECT i + 1 FROM x WHERE i < 101
),
Z(Ix, Iy, Cx, Cy, X, Y, I) AS (
SELECT Ix, Iy, X::FLOAT, Y::FLOAT, X::FLOAT, Y::FLOAT, 0
FROM
(SELECT -2.2 + 0.031 * i, i FROM x) AS xgen(x,ix)
CROSS JOIN
(SELECT -1.5 + 0.031 * i, i FROM x) AS ygen(y,iy)
UNION ALL
SELECT Ix, Iy, Cx, Cy, X * X - Y * Y + Cx AS X, Y * X * 2 + Cy, I + 1
FROM Z
WHERE X * X + Y * Y < 16.0
AND I < 27
),
Zt (Ix, Iy, I) AS (
SELECT Ix, Iy, MAX(I) AS I
FROM Z
GROUP BY Iy, Ix
ORDER BY Iy, Ix
)
SELECT array_to_string(
array_agg(SUBSTRING(' .,,,-----++++%%%%@@@@#### ', GREATEST(I,1),1)),''
)
FROM Zt
GROUP BY Iy
ORDER BY Iy;
```


Fractals!



Thank you

Questions?