

# Od tabulkového editoru k analýze dat pomocí SQL

Vojta Filipec  
13.-14.5.2020



# whoami

 jaderná fyzika

 profesní zkušenosti:

- banky: vývoj a validace modelů pro řízení kreditního rizika, monitoring, automatizace...
- fintech, farmaceutický průmysl, FMCG: “analytik - konzultant”

 data scientist



# obsah kurzu

1. představení účastníků
2. představení SQL
3. zpracování dat z jedné tabulky: 2x 40-50 min + 10 min cvičení
  - a. existující sloupce
  - b. vytváření nových sloupců
  - c. podmínky a funkce v nich
  - d. shlukování
4. studium na doma
5. propojování tabulek: 2x 40-50 min + 10 min cvičení
  - a. joiny
  - b. sjednocení
6. práce na vašich tématech

dotazy: k tématu se ptejte, obecné nápady do MS Teams



# představení účastníků

- Vaše jméno
- co s daty děláte
- co byste se chtěl(a) naučit
- největší “bolest” při zpracování dat v MS Excel



# proč SQL místo tabulkového editoru

Okamžité benefity:

- snadné pro uživatele MS Excel
- objem dat
- výsledky na pár řádků
- přístup k datům

Dlouhodobé benefity:

- objemy, rychlost výpočtu, replikovatelnost (debugging, verzování)
- SQL = praktický vstup do světa programování
- SQL (získání dat) + tabulkový editor (prezentace výsledku) = ❤️

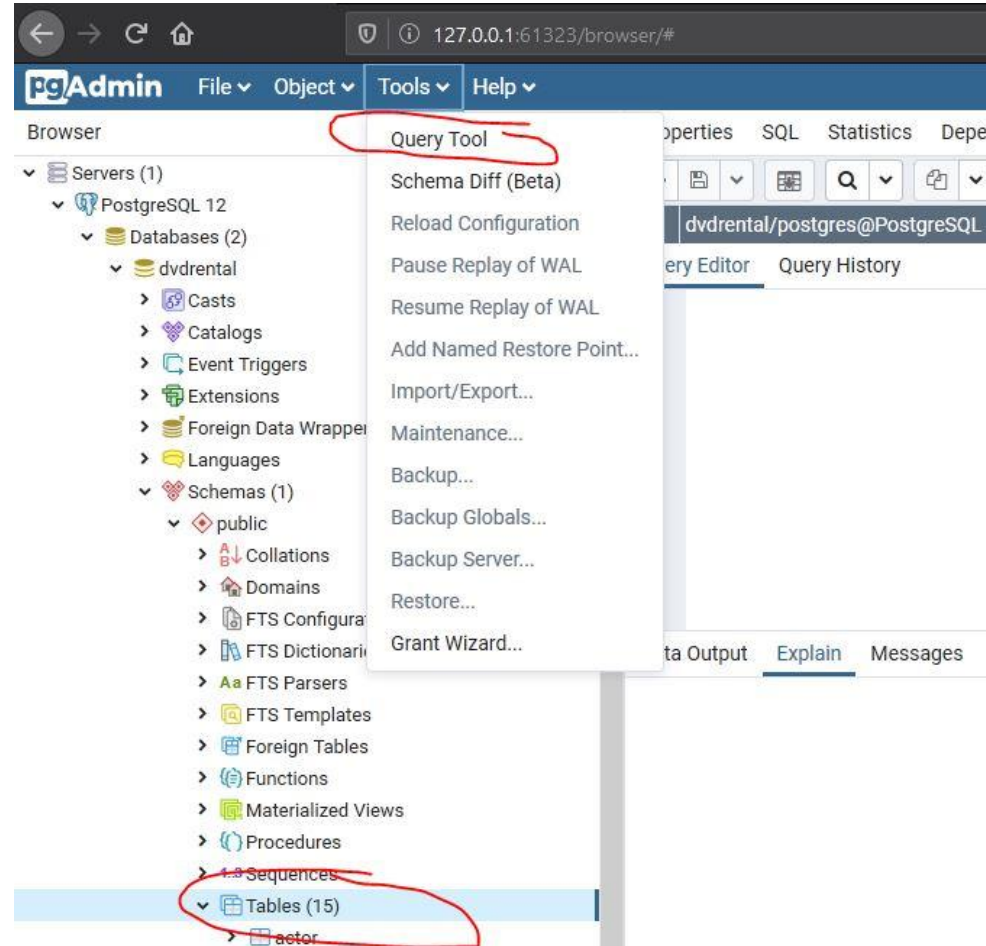
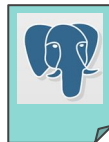


# začínáme s SQL

- SQL = Structured Query Language
  - “structured” = dotaz s klíčovými slovy
  - “query” = skriptování
- 
- tabulky

# začínáme s SQL

- SQL = Structured Query Language
- “structured” = dotaz s klíčovými slovy
- “query” = skriptování
- tabulky
- PgAdmin -> Tools -> Query Tool
- otevření programu ‘scripts.sql’
- SQL není case-sensitive:
  - SELECT = select = Select
  - actor = Actor



# **Práce s daty z jedné tabulky**

select, from, where, order by, group by





# **SELECT a FROM: výběr sloupců z tabulky**



Jak to udělat v Excelu?



# SELECT a FROM: výběr sloupců z tabulky

```
SELECT [seznam sloupců]

FROM [název tabulky]

;
```

seznam sloupců:

- sloupce oddělujte čárkou
- \* ... všechny sloupce

středník: konec query

# SELECT a FROM prakticky

1. sem píšeme query - označíme
2. zde ji spustíme, nebo F5
3. výsledek
4. počet řádků ve výsledku



cvičení

1

3

2

4

The screenshot shows a PostgreSQL query editor interface. The query entered is:

```
select film_id, title, rental_rate  
from film ;
```

The results are displayed in a table with the following columns: film\_id (integer), title (character varying (255)), and rental\_rate (numeric (4,2)).

	film_id integer	title character varying (255)	rental_rate numeric (4,2)
1	133	Chamber Italian	4.99
2	384	Grosse Wonderful	4.99
3	8	Airport Pollock	4.99
4	98	Bright Encounters	4.99
5	1	Academy Dinosaur	0.99
6	2	Ace Goldfinger	4.99
7	3	Adaptation Holes	2.99
8	4	Affair Prejudice	2.99
9	5	African Egg	2.99
10	6	Agent Truman	2.99
11	7	Airplane Sierra	
12	9	Alabama Devi	

At the bottom, a green status bar indicates: "Successfully run. Total query runtime: 333 msec. 1000 rows affected."

# DISTINCT, LIMIT, ORDER BY

```
SELECT DISTINCT [sloupce]

FROM [název tabulky]

ORDER BY [sloupce pro řazení]

LIMIT [kolik řádků zobrazit]

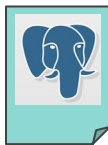
;
```

DISTINCT: zobrazí každou kombinaci hodnot uvedených sloupců jen jednou (🕒)

ORDER BY:

- řazení dle hodnot sloupců,
- “DESC” za jménem sloupce: sestupné řazení
- bez tohoto není pořadí garantováno

LIMIT: omezení počtu řádků, pro rychlý přehled (🕒)



# WHERE: výběr řádků

```
SELECT [seznam sloupců]



FROM [název tabulky]

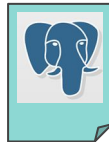
WHERE [podmínka]

;
```



podmínka:

- jedna, nebo více: AND, OR, (), NOT
- <, >, <=, >=, =, <>
- IN (hodnota1, hodnota2, hodnota3) 
- LIKE 'Ann%' ... hodnota začíná na Ann a pokračuje libovolně 
- vyhodnocuje se na začátku



# funkce pro složitější podmínky

```
WHERE length(title) < 10
```

```
WHERE activebool is False
```

jméno sloupce

```
WHERE upper(first_name)= 'ANN'
```

```
SELECT length(country) AS delka_jmena
```

- LENGTH: délka textu
- is True, is False
- first\_name || ' ' || last\_name
- UPPER

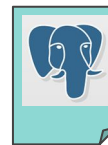
Funkce lze použít také v SELECT (i jinde)!



# CASE-WHEN: podmínky

```
SELECT rating,  
  
       CASE WHEN rating = 'G' THEN 1  
  
            WHEN rating = 'PG-13' THEN 2  
  
            ELSE 3  
  
       END as rating_group  
  
FROM film;
```

- obdoba IF() z MS Excel
- hledá se první splněná podmínka



# CREATE: vytváření vlastních tabulek

```
CREATE TABLE [jméno tabulky] AS
```

```
SELECT [seznam sloupců]
```

```
FROM [...]
```

```
WHERE [...]
```

```
ORDER BY [...]
```

```
;
```

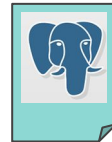
```
DROP TABLE IF EXISTS [jméno tab.]
```



jméno tabulky:

- vhodné: sample\_with\_values, SampleWithValues
- nevhodné: mytable23, "Sample w. values"

seznam sloupců: unikátní jména







# GROUP BY: agregace řádků

Kolik plateb provedli zákazníci 318 a 281?

Kolik každý z nich zaplatil?



payment		
payment_id	customer_id	amount
19682	281	2.99
19683	281	2.99
20001	318	2.99
25294	281	0.99
25295	281	6.99
...	...	...



# GROUP BY: agregace řádků

```
SELECT customer_id,  
  
       count(payment_id) as pocet_plateb,  
  
       sum(amount) as celkova_vyse  
  
FROM [...]  
  
GROUP BY customer_id  
  
;
```

agregační funkce:

- count(sloupec), count(\*)
- count(distinct sloupec)
- sum()
- avg(), min(), max()

<https://www.postgresql.org/docs/12/functions-aggregate.html>

# GROUP BY: agregace řádků

```
SELECT customer_id,  
       count(payment_id) as pocet_plateb,  
       sum(amount) as celkova_vyse  
FROM [...]  
GROUP BY customer_id  
;
```

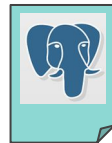
[definice skupiny]

[agregované sloupce]

agregační funkce:

- count(sloupec), count(\*)
- count(distinct sloupec)
- sum()
- avg(), min(), max()

<https://www.postgresql.org/docs/12/functions-aggregate.html>





# domácí práce: funkce pro datum a čas

- datum a čas:
  - [funkce](#)
  - [datové typy](#)
- domácí práce:
  - ve skriptu
  - Přečtěte si SQL dotazy a pomocí odkazů výše zjistěte, jak fungují funkce pro datum a čas.



# domácí práce: HAVING

HAVING funguje podobně jako WHERE, ale...

- patří až za GROUP BY
- filtruje agregované řádky ve výstupu a pouze tyto řádky
- (WHERE filtruje řádky na vstupu)

```
SELECT customer_id,  
  
       count(payment_id) as pocet_plateb,  
  
       sum(amount) as celkova_vyse  
  
FROM [...]  
  
WHERE sum(amount) > 60  
  
GROUP BY customer_id  
  
HAVING sum(amount) > 60  
  
;
```

# Získávání dat z vícero tabulek

join, union



# Kvíz na rozeehřátí

- na mobilu: [www.menti.com](https://www.menti.com)
- kód 24 69 92
- místo <?> doplnit správnou odpověď
- vždy jen 1 odpověď správná
- 30 s na odpověď




# Spojování tabulek “vedle” a “pod” sebe

film	
film_id	title
110	'Cabin Flash'
341	'Frost Head'
343	'Full Flatliners'
...	...
189	'Creatures Shakespeare'
216	'Day Unfaithful'
280	'Empire Malkovich'


inventory_store_1		
inventory_id	film_id	store_id
496	110	1
1553	341	1
1563	343	1

inventory_store_2		
inventory_id	film_id	store_id
861	189	2
971	216	2
1264	280	2

vedle sebe: spojení pomocí klíče  
(společného sloupce)

- Jak se jmenují filmy v tabulce inventory\_store\_1?
- 

pod sebe: prodloužení o další záznamy

- Kolikrát je film s id=189 v obou pobočkách?
- 



# Vedle sebe: JOIN, pod sebe: UNION

film	
film_id	title
110	'Cabin Flash'
341	'Frost Head'
343	'Full Flatliners'
...	...
189	'Creatures Shakespeare'
216	'Day Unfaithful'
280	'Empire Malkovich'



inventory_store_1		
inventory_id	film_id	store_id
496	110	1
1553	341	1
1563	343	1



inventory_store_2		
inventory_id	film_id	store_id
861	189	2
971	216	2
1264	280	2



# JOIN

spojování tabulek vedle sebe



# Propojení tabulek v MS Excel: VLOOKUP

VLOOKUP (vazebná podmínka mezi tab 1 a tab 2; ✓)

čemu říkáme tab 2; ✓

co z tab 2 chceme získat; ✓

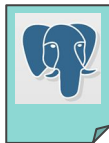
přesně vs. přibližně ✗)

inventory_store_1		
inventory_id	film_id	store_id
496	110	1
1553	341	1
1563	343	1

film	
film_id	title
110	'Cabin Flash'
341	'Frost Head'
343	'Full Flatliners'
...	...
189	'Creatures Shakespeare'
216	'Day Unfaithful'
280	'Empire Malkovich'

# JOIN: propojení tabulek

```
SELECT is1.store_id, is1.inventory_id,  
       f.title ✓  
FROM inventory_store_1 as is1  
  
JOIN film as f ✓  
  
ON is1.film_id = f.film_id ✓  
;
```



film	
film_id	title
110	'Cabin Flash'
341	'Frost Head'
343	'Full Flatliners'
...	...
189	'Creatures Shakespeare'
216	'Day Unfaithful'
280	'Empire Malkovich'

inventory_store_1		
inventory_id	film_id	store_id
496	110	1
1553	341	1
1563	343	1

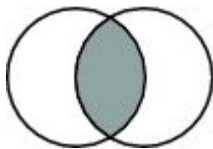
is1, f8 ... alias tabulky, libovolný

místo aliasu: plné jméno tabulky

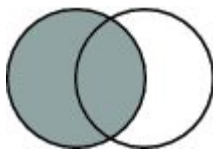
ON ... vazebná podmínka



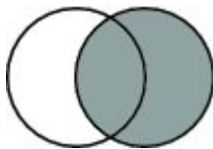
# joiny: INNER vs. OUTER



INNER JOIN: průnik řádků



OUTER JOIN:



- vše z levé tab. (LEFT OUTER JOIN) + odpovídající řádky z pravé tab.
- vše z pravé tab. (RIGHT OUTER JOIN) + odpovídající řádky z levé tab.
- záměna pořadí tabulek = “opačný” typ joinu



# joiny: INNER vs. OUTER

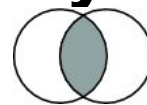
film	
film_id	title
110	'Cabin Flash'
341	'Frost Head'
343	'Full Flatliners'
...	...
189	'Creatures Shakespeare'
216	'Day Unfaithful'
280	'Empire Malkovich'

inventory_store_1		
inventory_id	film_id	store_id
496	110	1
1553	341	1
1563	343	1

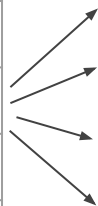
inventory_store_2		
inventory_id	film_id	store_id
861	189	2
971	216	2
1264	280	2



# JOIN: vazba typu "1:1" vs. "1:many"



left: film_3	
film_id	title
110	'Cabin Flash'
111	'Caddyshack Jedi'
343	'Full Flatliners'



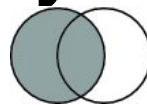
right: five_films		
inventory_id	film_id	store_id
496	110	1
493	110	1
494	110	1
495	110	1
497	111	2

INNER JOIN			
film_id	title	inventory_id	store_id
110	'Cabin Flash'	496	1
110	'Cabin Flash'	493	1
110	'Cabin Flash'	494	1
110	'Cabin Flash'	495	1
111	'Caddyshack Jedi'	497	2

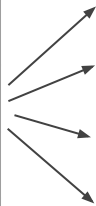
vazba 1:1 ... jako u LOOKUP (film 111)

vazba 1:many ... všechny shody ve výskytu, **nikoliv** jen první výskyt (film 110)

# JOIN: vazba typu "1:1" vs. "1:many"



left: film_3	
film_id	title
110	'Cabin Flash'
111	'Caddyshack Jedi'
343	'Full Flatliners'

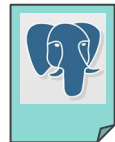


right: five_films		
inventory_id	film_id	store_id
496	110	1
493	110	1
494	110	1
495	110	1
497	111	2

LEFT OUTER JOIN			
film_id	title	inventory_id	store_id
110	'Cabin Flash'	496	1
110	'Cabin Flash'	493	1
110	'Cabin Flash'	494	1
110	'Cabin Flash'	495	1
111	'Caddyshack Jedi'	497	2
343	'Full Flatliners'	null	null

vazba 1:1 ... jako u LOOKUP (film 111)

vazba 1:many ... všechny shody ve výskytu, **nikoliv** jen první výskyt (film 110)







# NULL

chybějící hodnota, není to však prázdný text

částečný ekvivalent #N/A

ověření: `is null`

nahrazení: `coalesce(arg1, arg2)`





# UNION

spojování tabulek pod sebe





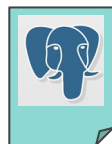
# UNION: zapisování tabulek pod sebe

inventory_store_1		
inventory_id	film_id	store_id
496	110	1
1553	341	1
1563	343	1

inventory_store_2		
inventory_id	film_id	store_id
861	189	2
971	216	2
1264	280	2

- UNION: vyloučí duplicity
- UNION ALL: ponechá všechny řádky

Tabulky musí mít stejný počet sloupců a jejich typ.



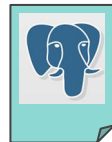


# domácí práce: sub-query a EXIST

ověření existence záznamu v jiné tabulce:

- toto se v MS Excelu hledá zdlouhavě přes VLOOKUP
- <https://www.postgresqltutorial.com/postgresql-subquery/>
- <https://www.postgresqltutorial.com/postgresql-any/>

lze vyřešit joinem, zde jen pro úplnost





# Děkuji za pozornost + prosím o feedback

 [linkedin.com/in/vojtech-filipek/](https://linkedin.com/in/vojtech-filipek/)

 [github.com/vojtech-filipek](https://github.com/vojtech-filipek)





# Zdroje

- <https://www.postgresqltutorial.com/> -> studujte postupně, nebo hledejte kapitoly
- <https://www.postgresql.org/docs/12/> -> ofic. dokumentace, velmi obsáhlá, hledejte dle klíčového slova (např. “functions”)
- <https://github.com/vojtech-filipec/sql-pro-uzivatele-excelu> -> tento kurz

# další typy joinů

