



VYSOKÉ UČENÍ FAKULTA ELEKTROTECHNIKY
TECHNICKÉ A KOMUNIKAČNÍCH
V BRNĚ TECHNOLOGIÍ

PROJEKT 2

Autor: Veronika Vojáčková

Obor: Informační bezpečnost

Předmět: Bezpečnost databázových systémů

Obsah

Úvod.....	4
1. Dotazy.....	4
1.1. Dotaz, který načte pouze vybrané sloupce z vybrané tabulky.....	4
1.2. Dotaz, který načte člověka dle emailu	4
1.3. Dotaz s použitím UPDATE	6
1.4. Dotaz s použitím INSERT.....	6
1.5. Dotaz s použitím DELETE.....	7
1.6. Dotaz s použitím ALTER TABLE	9
1.7. Dotaz s použitím WHERE	10
1.8. Řada dotazů s použitím LIKE, NOT LIKE	10
1.9. Řada dotazů s použitím SUBSTRING, TRIM.....	11
1.10. Řada dotazů s použitím COUNT, SUM, MIN, MAX, AVG	12
1.11. Řada dotazů s použitím GROUP BY, GROUP BY a HAVING, GROUP BY, HAVING a WHERE	15
1.12. Dotaz s použitím UNION ALL	17
1.13. Dotaz s použitím DISTINCT	17
1.14. Řada dotazů s použitím LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN.....	18
1.15. Dotaz s využitím LEFT JOIN, GROUP BY, HAVING, ORDER BY a AVG	21
1.16. Dotaz, který vrátí data z tabulky za poslední 1 den a 12 hodin.....	21
1.17. Dotaz, který vrátí data z minulého měsíce	22
1.18. Dotaz, který odstraní diakritiku.....	22
1.19. Dotaz pro paging s pomocí LIMIT a OFFSET	23
1.20. Dotaz s poddotazem ve FROM.....	24
1.21. Dotaz s poddotazem v podmínce WHERE.....	25
1.22. Dotaz používající agregační funkci a GROUP BY s HAVING.....	25
1.23. Dotaz, který spojí alespoň 5 tabulek	26
1.24. Dotaz, který spojí alespoň 3 tabulky a použije GROUP BY, HAVING a WHERE	26
2. Úprava databáze z prvního zadání pro lepší integritu	27
3. Vytvoření indexu databáze	28
4. Procedura databáze	29
5. Trigger databáze	32
6. Pohled databáze	33
7. Materializovaný pohled na databázi.....	34
8. Role teacher a student	35

Závěr.....	36
Seznam obrázků	37

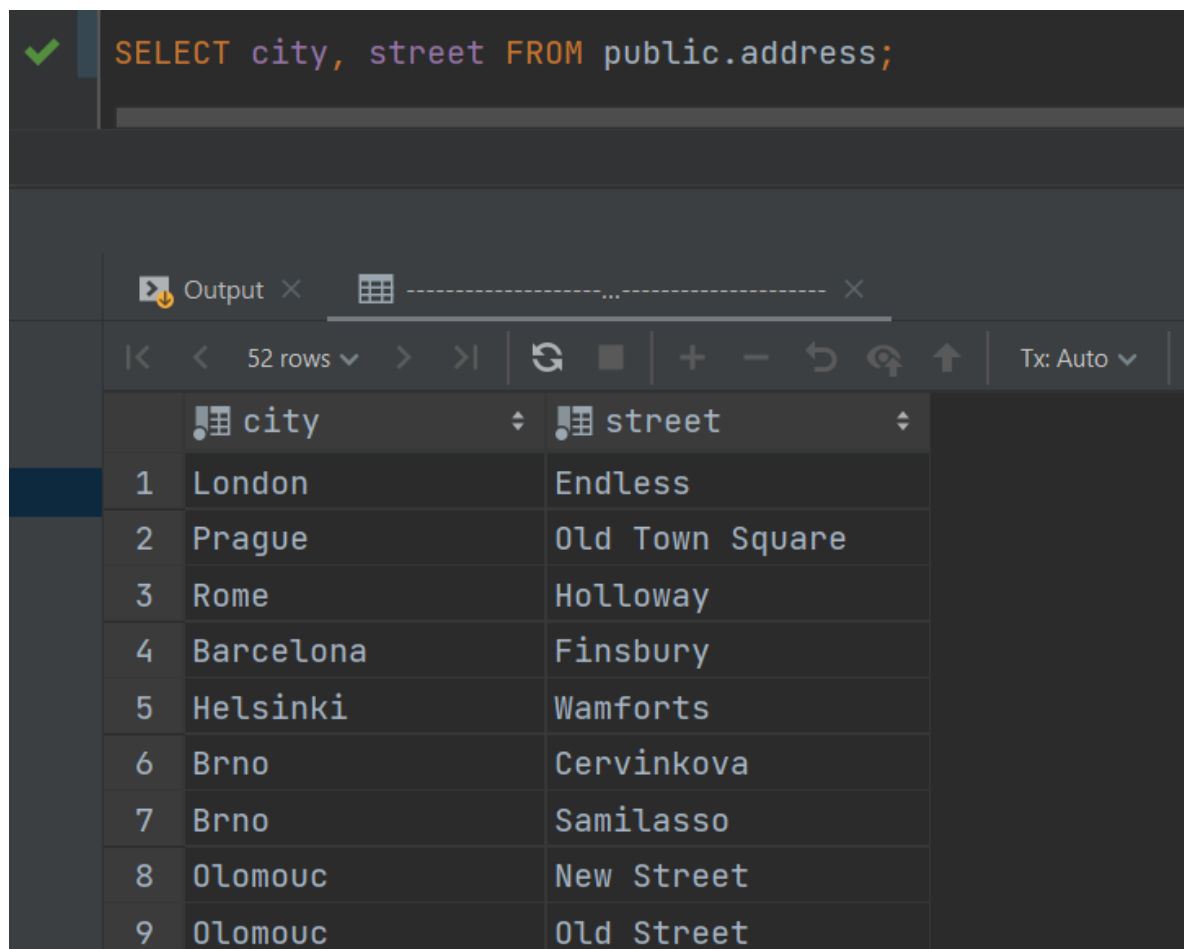
Úvod

Tato práce dokumentuje druhý projekt z předmětu Bezpečnost databázových systémů.

1. Dotazy

1.1. Dotaz, který načte pouze vybrané sloupce z vybrané tabulky

Dotaz pro načtení jména města a ulice z tabulky address. Záznamů je vypsáno více, jedná se pouze o ukázkou úspěšného vypsání.



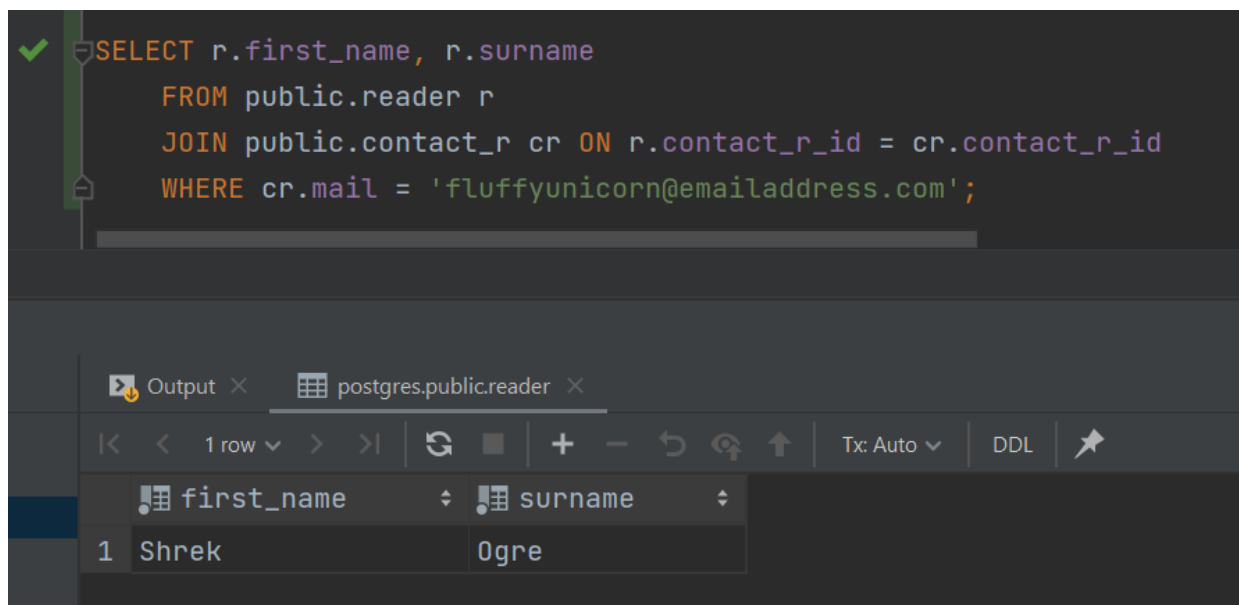
The screenshot shows a database query interface. At the top, a green checkmark indicates a successful query execution. The query text is `SELECT city, street FROM public.address;`. Below the query, there is a toolbar with various icons for navigating and editing the results. The results are displayed in a table with two columns: 'city' and 'street'. The table contains 9 rows of data, with the first row highlighted in blue. The 'city' column has a dropdown arrow, and the 'street' column has a dropdown arrow. The 'Tx: Auto' dropdown is visible on the right side of the toolbar.

	city	street
1	London	Endless
2	Prague	Old Town Square
3	Rome	Holloway
4	Barcelona	Finsbury
5	Helsinki	Wamforts
6	Brno	Cervinkova
7	Brno	Samilasso
8	Olomouc	New Street
9	Olomouc	Old Street

Obrázek č. 1 Vypsání vybraných sloupců z tabulky address

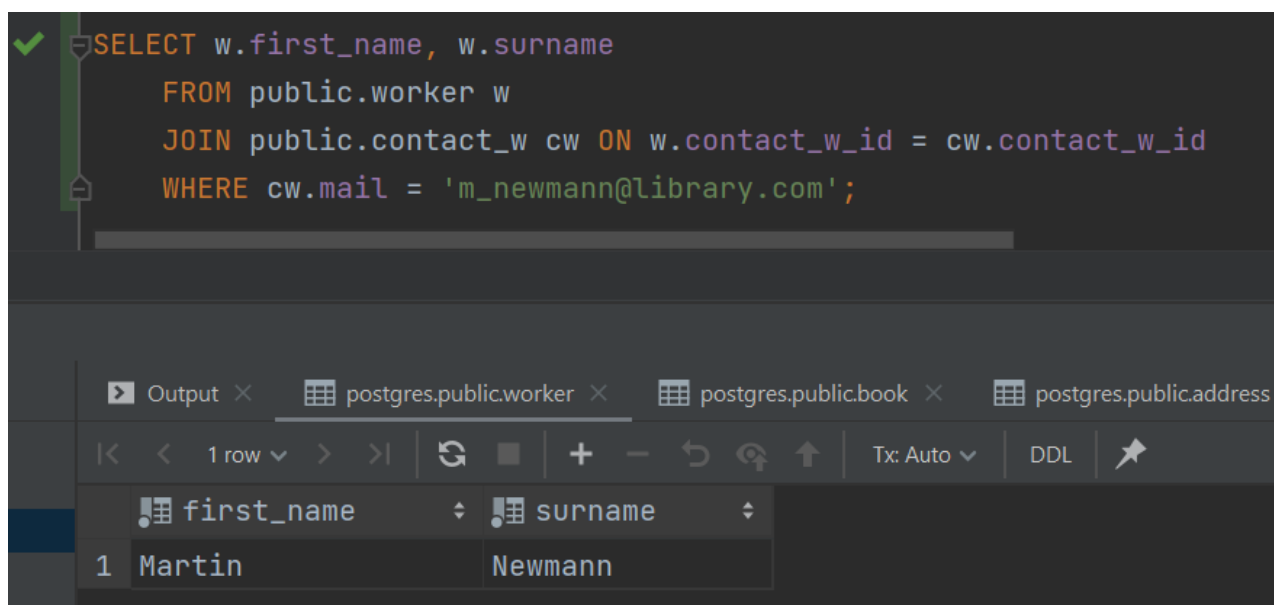
1.2. Dotaz, který načte člověka dle emailu

Dotaz k nalezení jména a příjmení čtenáře z tabulky reader dle emailové adresy z tabulky contact_r.



Obrázek č. 2 Výsledek hledání dle emailu v tabulce reader

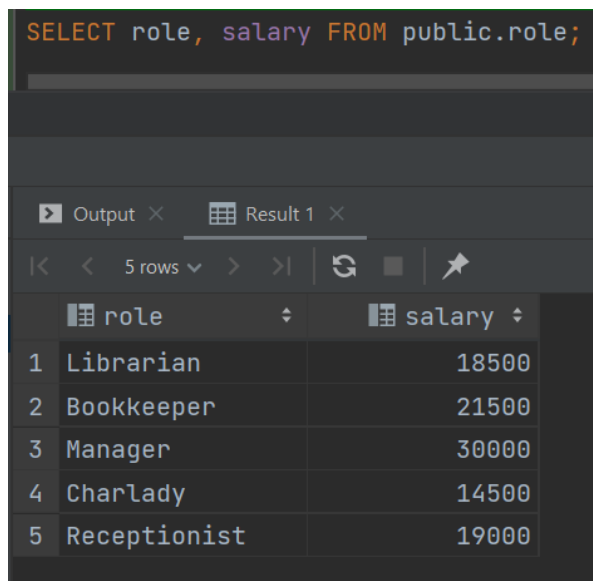
Dotaz k nalezení jména a příjmení pracovníka z tabulky worker dle emailové adresy z tabulky contact_w.



Obrázek č. 3 Výsledek hledání dle emailu v tabulce worker

1.3. Dotaz s použitím UPDATE

Dotaz update využijeme ke zvýšení platů o 1750.

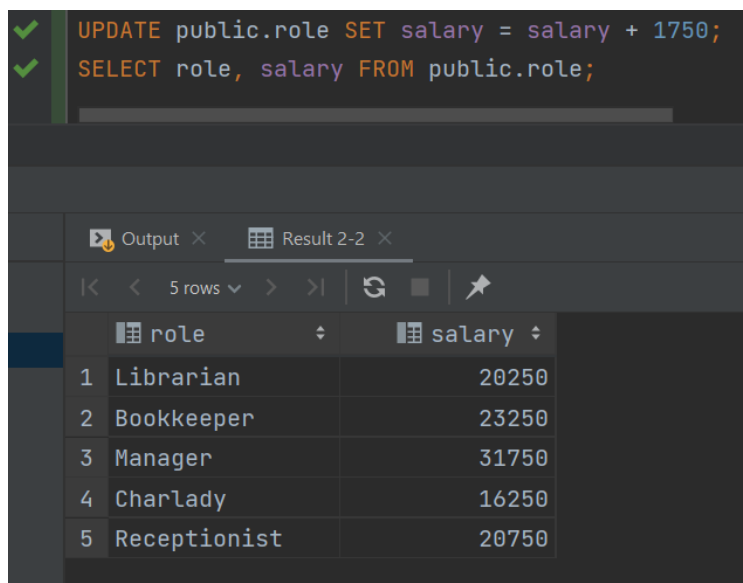


```
SELECT role, salary FROM public.role;
```

	role	salary
1	Librarian	18500
2	Bookkeeper	21500
3	Manager	30000
4	Charlady	14500
5	Receptionist	19000

Obrázek č. 4 Stav před dotazem UPDATE

Po použití dotazu UPDATE můžeme vidět zvýšení veškerých platů o 1750.



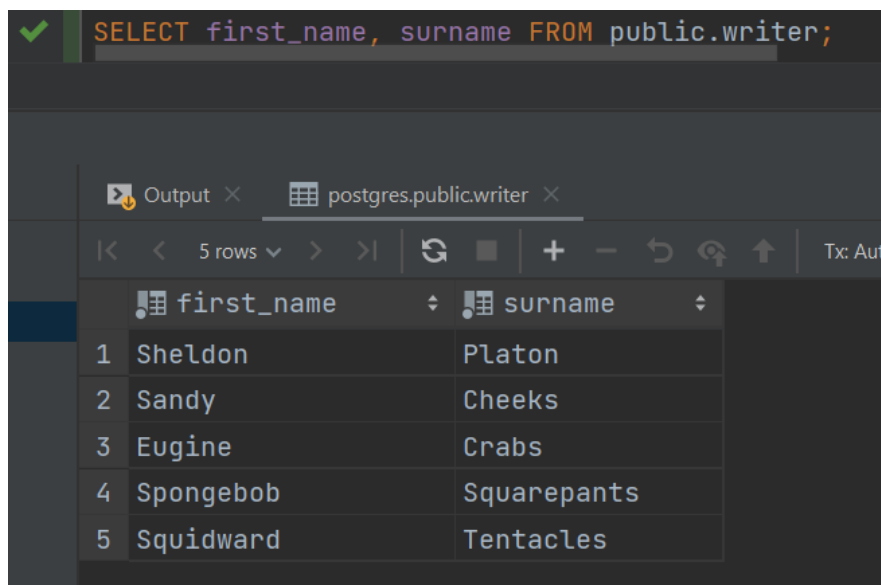
```
UPDATE public.role SET salary = salary + 1750;  
SELECT role, salary FROM public.role;
```

	role	salary
1	Librarian	20250
2	Bookkeeper	23250
3	Manager	31750
4	Charlady	16250
5	Receptionist	20750

Obrázek č. 5 Zvýšení platů po použití dotazu UPDATE

1.4. Dotaz s použitím INSERT

Jako první vypíšeme jména a příjmení všech spisovatelů z tabulky writer.

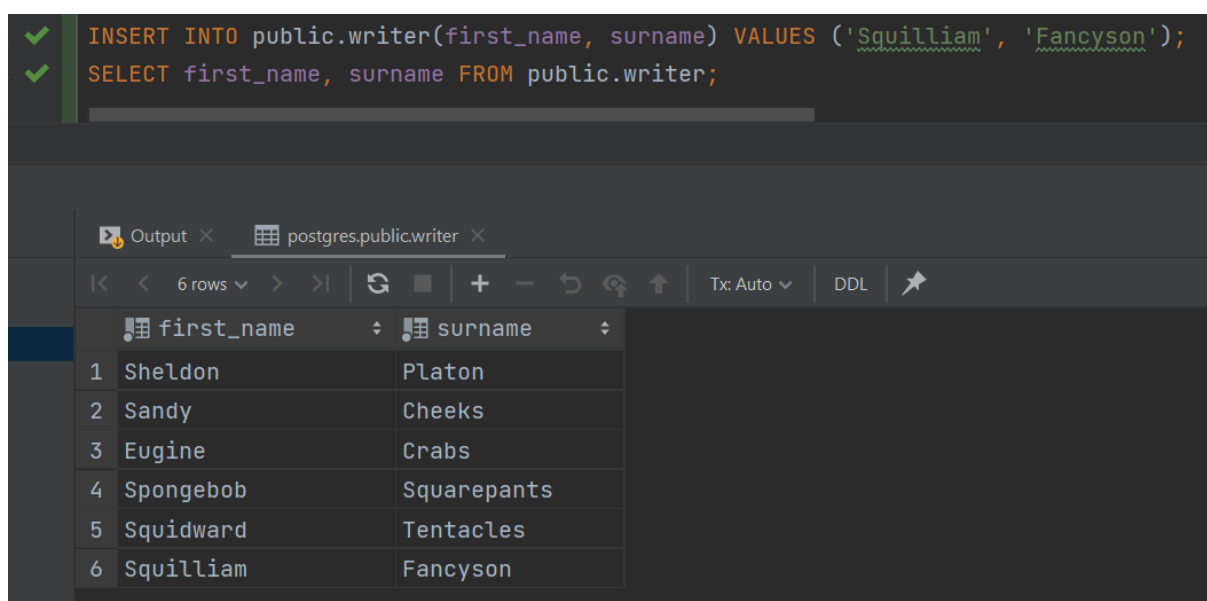


The screenshot shows a PostgreSQL query editor with a green checkmark indicating a successful query. The query is `SELECT first_name, surname FROM public.writer;`. Below the query, the results are displayed in a table with two columns: `first_name` and `surname`. The table contains five rows of data.

	first_name	surname
1	Sheldon	Platon
2	Sandy	Cheeks
3	Eugene	Crabs
4	Spongebob	Squarepants
5	Squidward	Tentacles

Obrázek č. 6 Kontrolní výpis před INSERT

Nyní použijeme dotaz INSERT a vložíme jméno 'Squilliam' a příjmení 'Fancyson' do tabulky writer. Znovu vypíšeme pro ujistění, že dotaz opravdu vložil nový záznam.



The screenshot shows a PostgreSQL query editor with two green checkmarks indicating successful queries. The first query is `INSERT INTO public.writer(first_name, surname) VALUES ('Squilliam', 'Fancyson');`. The second query is `SELECT first_name, surname FROM public.writer;`. Below the queries, the results are displayed in a table with two columns: `first_name` and `surname`. The table now contains six rows of data.

	first_name	surname
1	Sheldon	Platon
2	Sandy	Cheeks
3	Eugene	Crabs
4	Spongebob	Squarepants
5	Squidward	Tentacles
6	Squilliam	Fancyson

Obrázek č. 7 Kontrolní výpis po INSERT

1.5. Dotaz s použitím DELETE

Pro tento dotaz využijeme předchozí INSERT a smažeme jej. Využíváme mazání za pomoci id kdyby byla shoda ve jménech, abychom nesmazali data o která přijít. Opět si jako první vypíšeme aktuální stav záznamů.

```
✓ SELECT writer_id, first_name, surname FROM public.writer;
```

Output	postgres.public.writer
6 rows	<div> <div>writer_id</div> <div>first_name</div> <div>surname</div> </div>
1	1 Sheldon Platon
2	2 Sandy Cheeks
3	3 Eugene Crabs
4	4 Spongebob Squarepants
5	5 Squidward Tentacles
6	6 Squilliam Fancyson

Obrázek č. 8 Stav před použitím DELETE

Použitím dotazu DELETE smažeme chtěné údaje a následným SELECT zkontrolujeme, že byl vybraný záznam opravdu smazán.

```
7 ✓ DELETE FROM public.writer WHERE writer_id = 6;
```

```
8 ✓ SELECT writer_id, first_name, surname FROM public.writer;
```

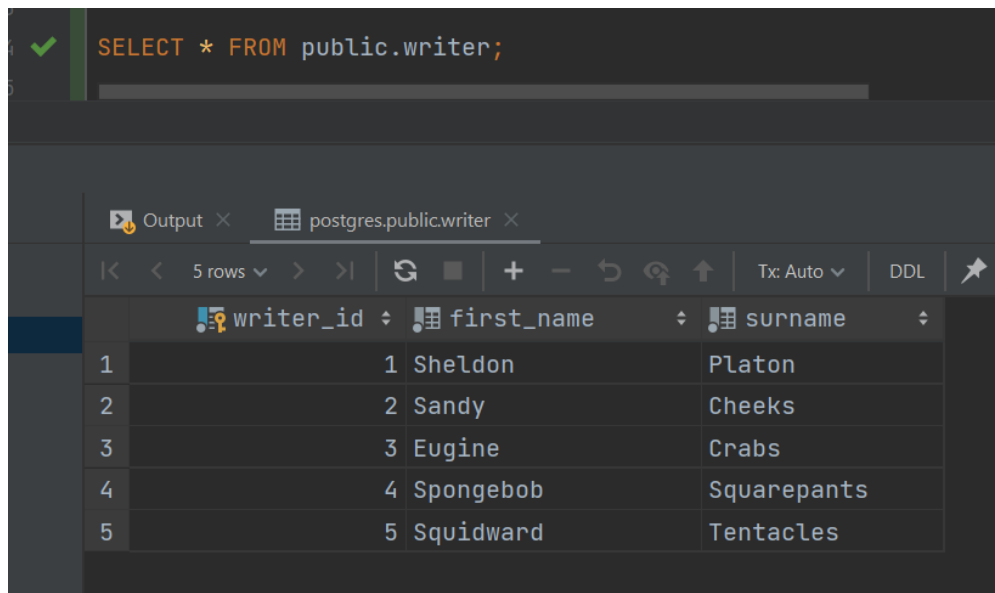
Output	postgres.public.writer
5 rows	<div> <div>writer_id</div> <div>first_name</div> <div>surname</div> </div>
1	1 Sheldon Platon
2	2 Sandy Cheeks
3	3 Eugene Crabs
4	4 Spongebob Squarepants
5	5 Squidward Tentacles

Obrázek č. 9 Stav po použití DELETE

1.6. Dotaz s použitím ALTER TABLE

Pomocí ALTER TABLE doplníme do tabulky writer nový sloupec s názvem birth_year a typem SMALLINT. Tento nový sloupec bude sloužit pro uložení roku narození autorů.

Jako první si vypíšeme všechny sloupce databáze.

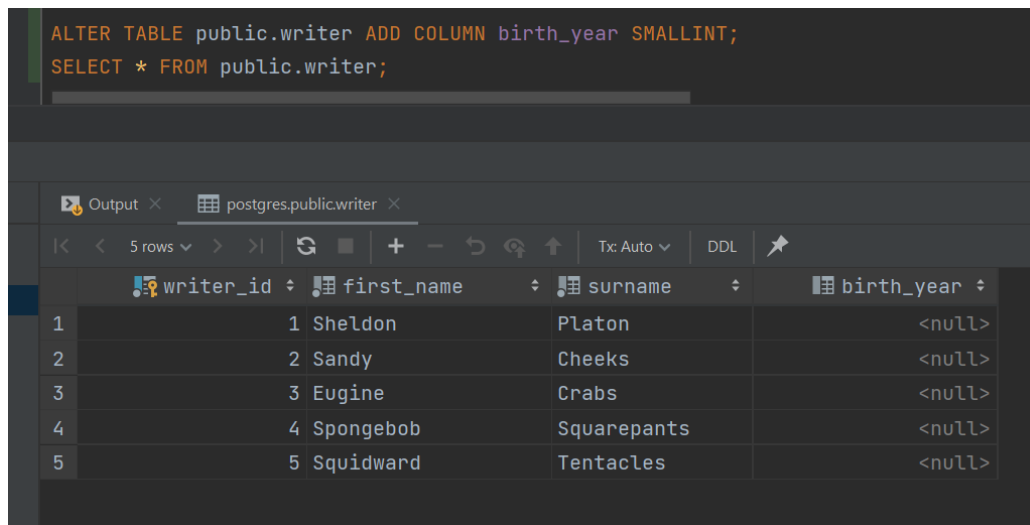


```
SELECT * FROM public.writer;
```

	writer_id	first_name	surname
1	1	Sheldon	Platon
2	2	Sandy	Cheeks
3	3	Eugene	Crabs
4	4	Spongebob	Squarepants
5	5	Squidward	Tentacles

Obrázek č. 10 Stav tabulky před použitím ALTER TABLE

Zavoláme dotaz s ALTER TABLE a opět kontrolně vypíšeme veškeré sloupce databáze.



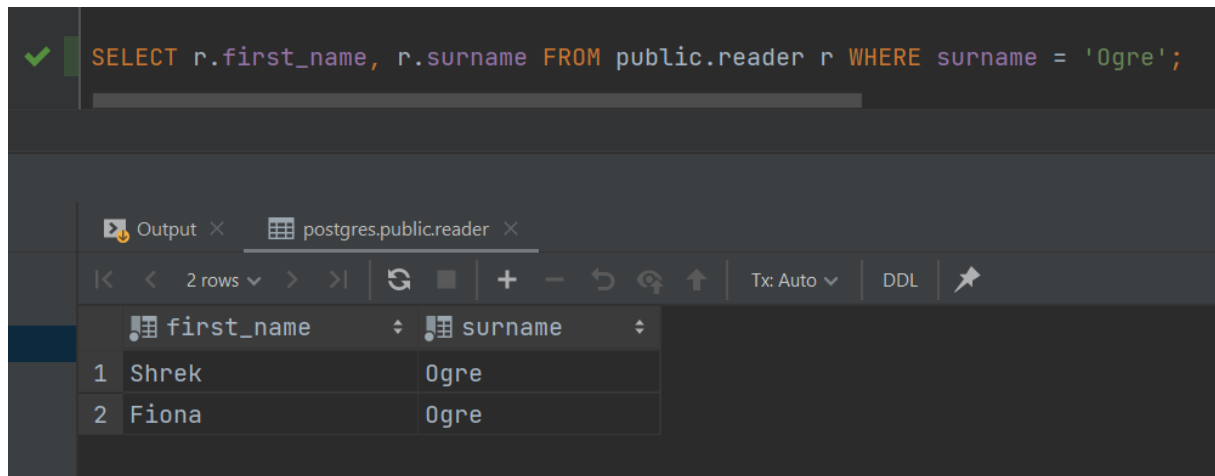
```
ALTER TABLE public.writer ADD COLUMN birth_year SMALLINT;  
SELECT * FROM public.writer;
```

	writer_id	first_name	surname	birth_year
1	1	Sheldon	Platon	<null>
2	2	Sandy	Cheeks	<null>
3	3	Eugene	Crabs	<null>
4	4	Spongebob	Squarepants	<null>
5	5	Squidward	Tentacles	<null>

Obrázek č. 11 Stav tabulky po použití ALTER TABLE

1.7. Dotaz s použitím WHERE

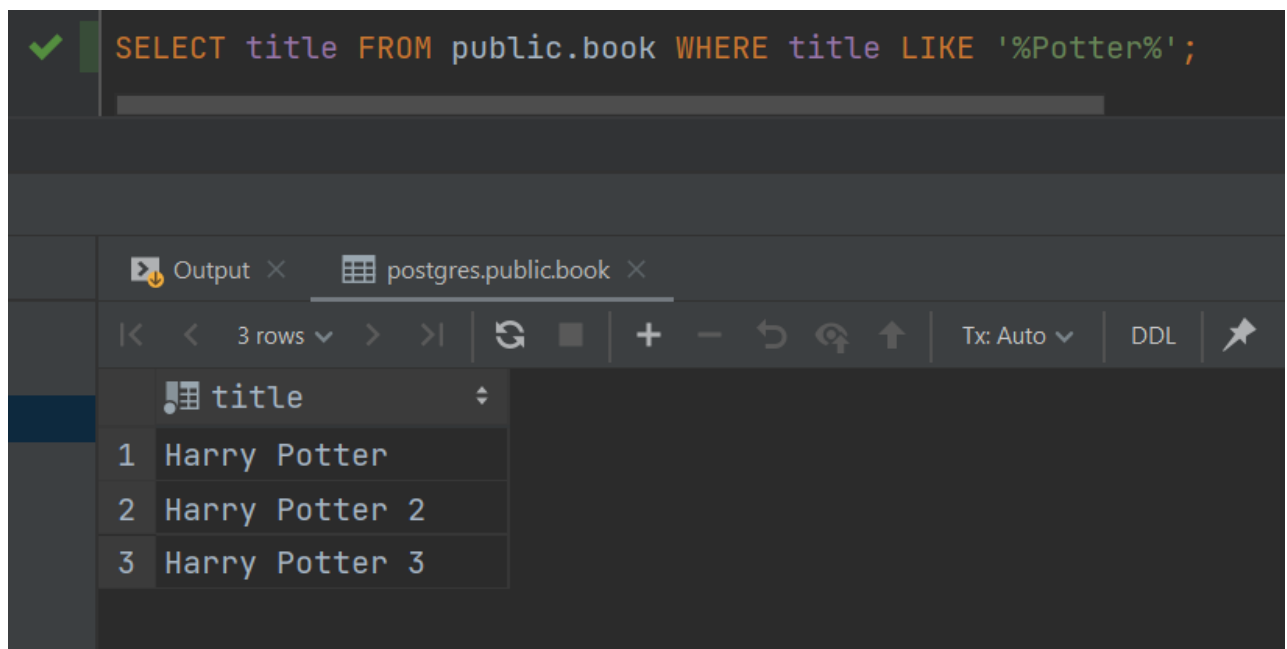
Využitím příkazu WHERE zjistíme čtenáře se stejným příjmením.



Obrázek č. 12 Vypsání čtenářů se shodným příjmením za pomoci WHERE

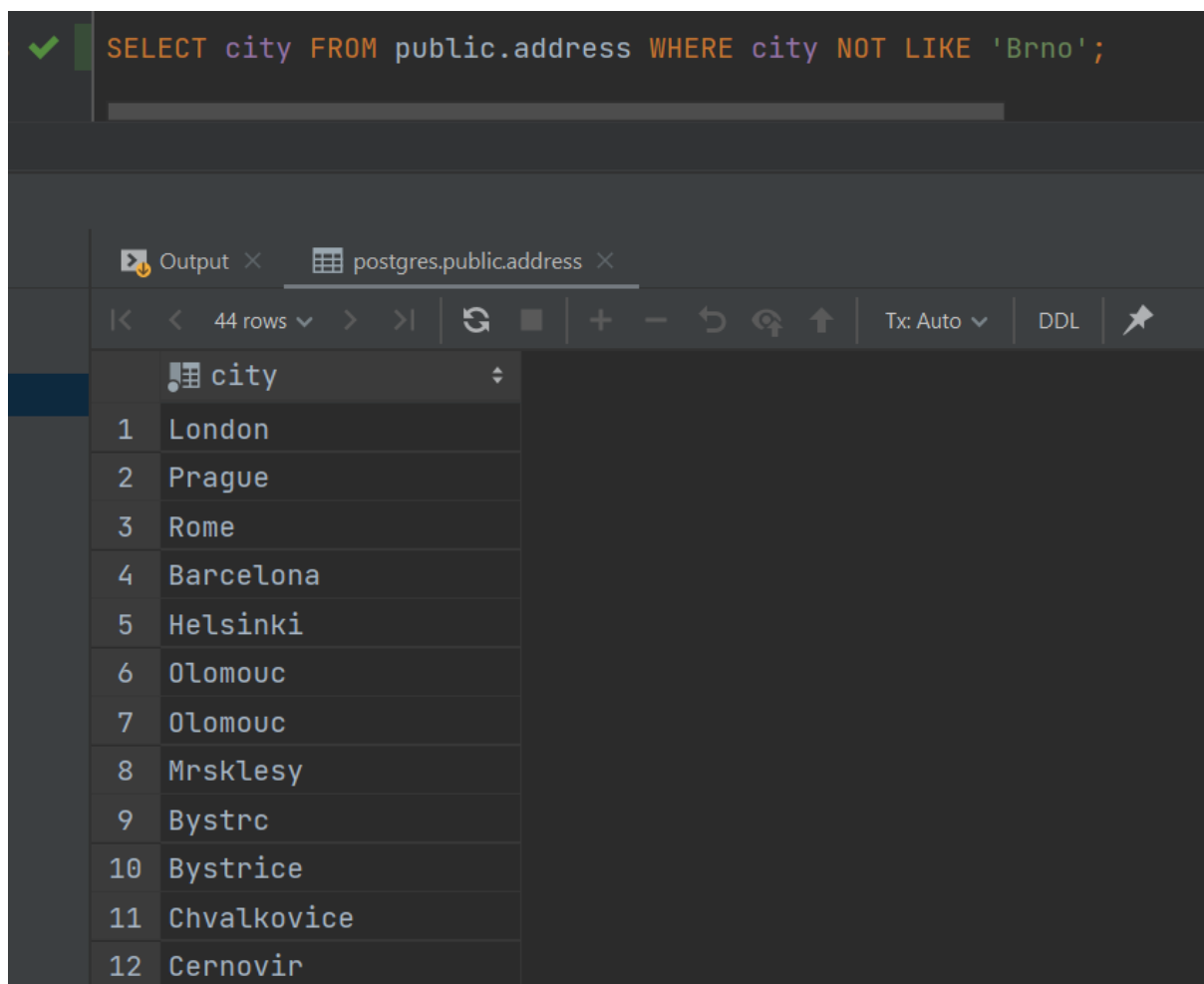
1.8. Řada dotazů s použitím LIKE, NOT LIKE

Použitím dotazu s LIKE vypíšeme všechny knihy které v sobě obsahují slovo „Potter“.



Obrázek č. 13 Vypsání všech knih, které mají v názvu Potter

Použitím dotazu s NOT LIKE vypíšeme všechny adresy, které nejsou Brno.



```
SELECT city FROM public.address WHERE city NOT LIKE 'Brno';
```

Output × postgres.public.address ×

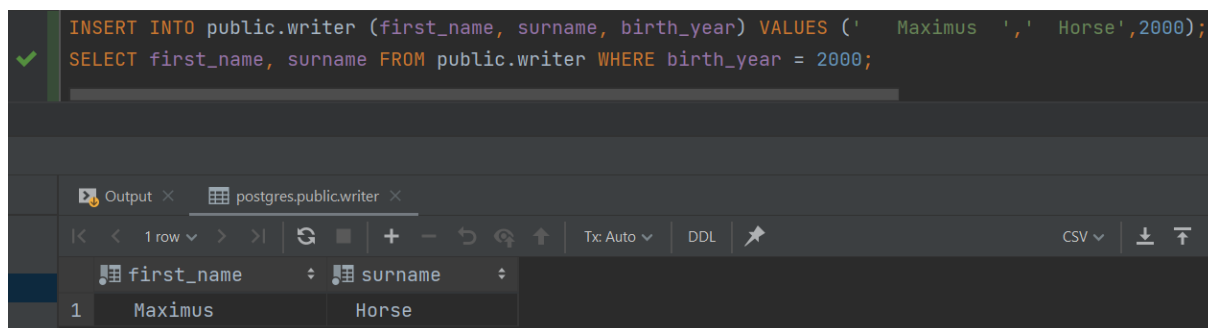
44 rows

	city
1	London
2	Prague
3	Rome
4	Barcelona
5	Helsinki
6	Olomouc
7	Olomouc
8	Mrsklesy
9	Bystrc
10	Bystrice
11	Chvalkovice
12	Cernovir

Obrázek č. 14 Vypsání všech adres kromě Brna

1.9. Řada dotazů s použitím SUBSTRING, TRIM

Pro využití TRIM nejprve přidáme nový záznam, ve kterém jsou použity bílé znaky. Těch se právě pomocí funkce TRIM zbavíme.



```
INSERT INTO public.writer (first_name, surname, birth_year) VALUES (' Maximus ', ' Horse', 2000);  
SELECT first_name, surname FROM public.writer WHERE birth_year = 2000;
```

Output × postgres.public.writer ×

1 row

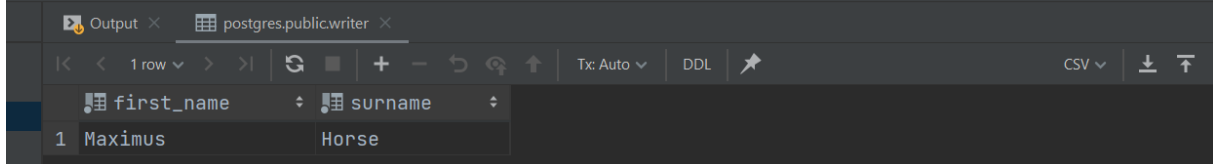
	first_name	surname
1	Maximus	Horse

Obrázek č. 15 Vypsání záznamu s bílými znaky

```

INSERT INTO public.writer (first_name, surname, birth_year) VALUES ('  Maximus  ',' Horse',2000);
UPDATE public.writer
SET first_name = TRIM (first_name), surname = TRIM (surname);
SELECT first_name, surname FROM public.writer WHERE birth_year = 2000;

```



first_name	surname
Maximus	Horse

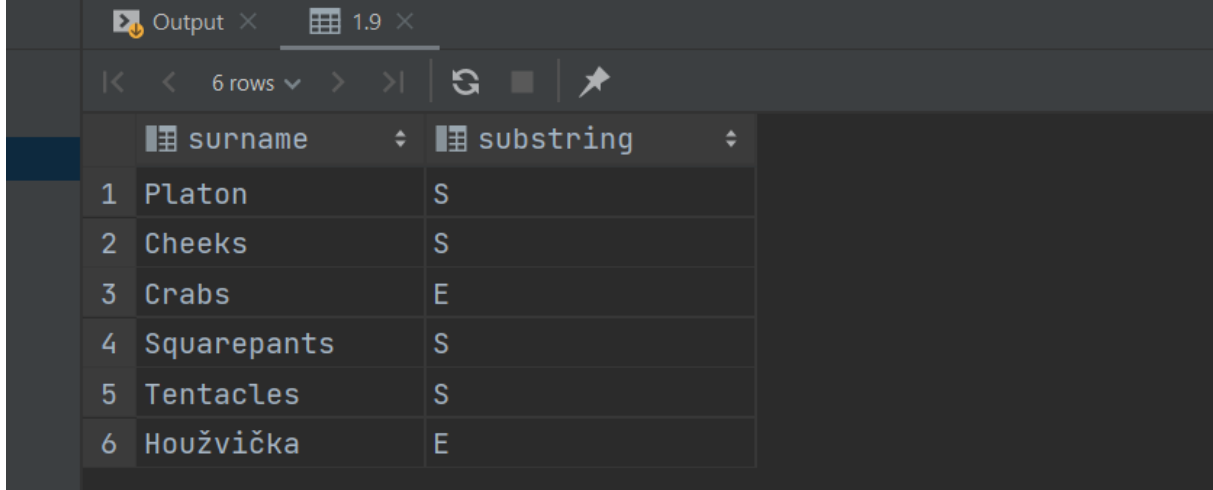
Obrázek č. 16 Opětovné vypsání záznamu, tentokrát už zbaveného bílých znaků

Funkci SUBSTRING použijeme pro vytáhnutí prvního znaku ze jména autora pro získání iniciálu.

```

SELECT surname, SUBSTRING(first_name, 1, 1) FROM public.writer;

```



surname	substring
Platon	S
Cheeks	S
Crabs	E
Squarepants	S
Tentacles	S
Houžvička	E

Obrázek č. 17 Vypsání příjmení a iniciálu jména autorů

1.10. Řada dotazů s použitím COUNT, SUM, MIN, MAX, AVG

Dotaz COUNT je použit pro získání počtu knih vydaných v každém roce. Záznamů je vypsáno více, jedná se pouze o ukázkou úspěšného vypsání.

```
SELECT COUNT(b.title), b.public_year
FROM public.book b
GROUP BY b.public_year
ORDER BY b.public_year;
```

	count	public_year
15	7	1992
16	1	1993
17	1	1994
18	2	1995
19	1	1996
20	2	1997

Obrázek č. 18 Dotaz s použitím COUNT

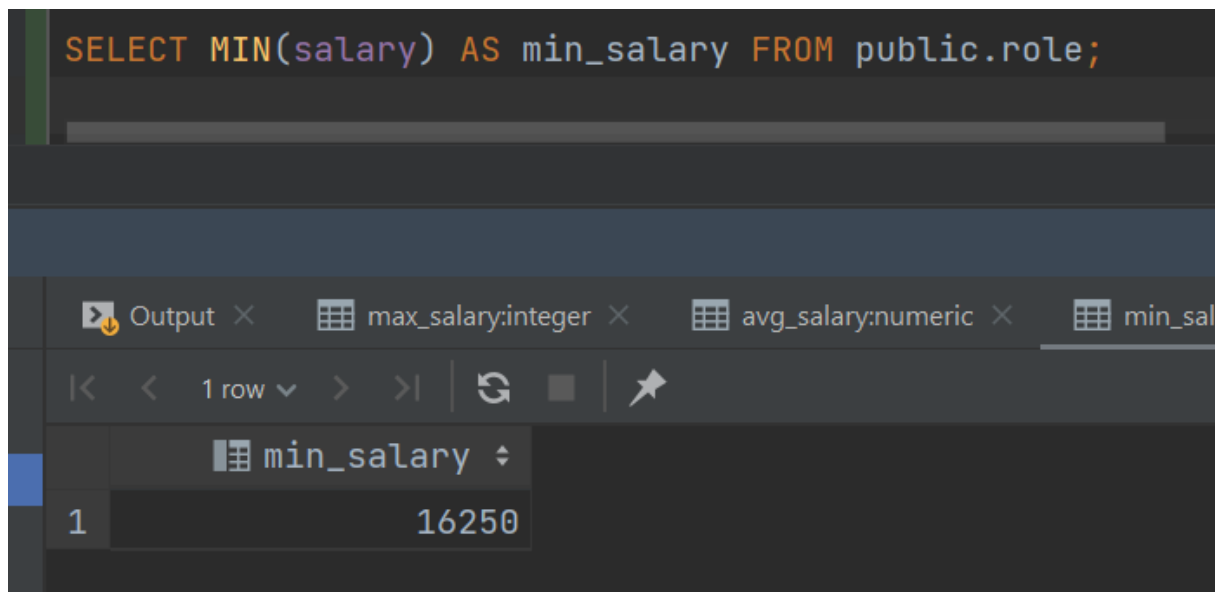
Dotazem SUM vypočítáme množství peněz potřebných na funkci každé knihovny, kde máme nějaké zaměstnance.

```
SELECT w.library_id, SUM(r.salary)
FROM public.role r
JOIN public.worker w ON r.role_id = w.role_id
GROUP BY w.library_id;
```

	library_id	sum
1	3	80000
2	4	20750

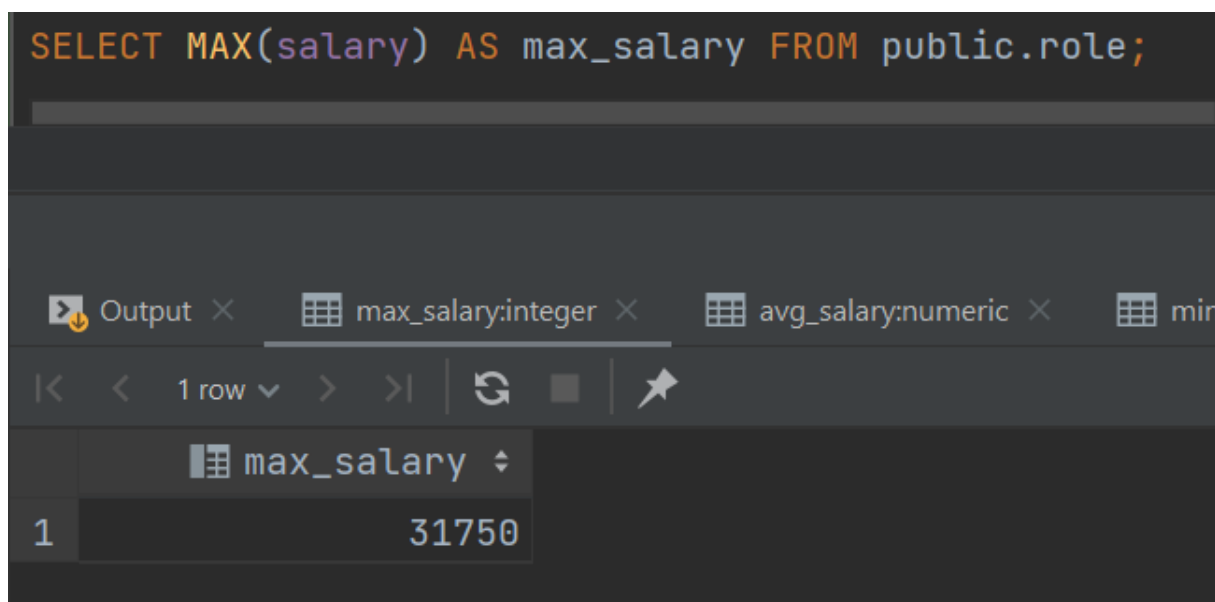
Obrázek č. 19 Dotaz s použitím SUM

Dotaz MIN využít pro zjištění nejmenšího platu.



Obrázek č. 20 Dotaz s použitím MIN

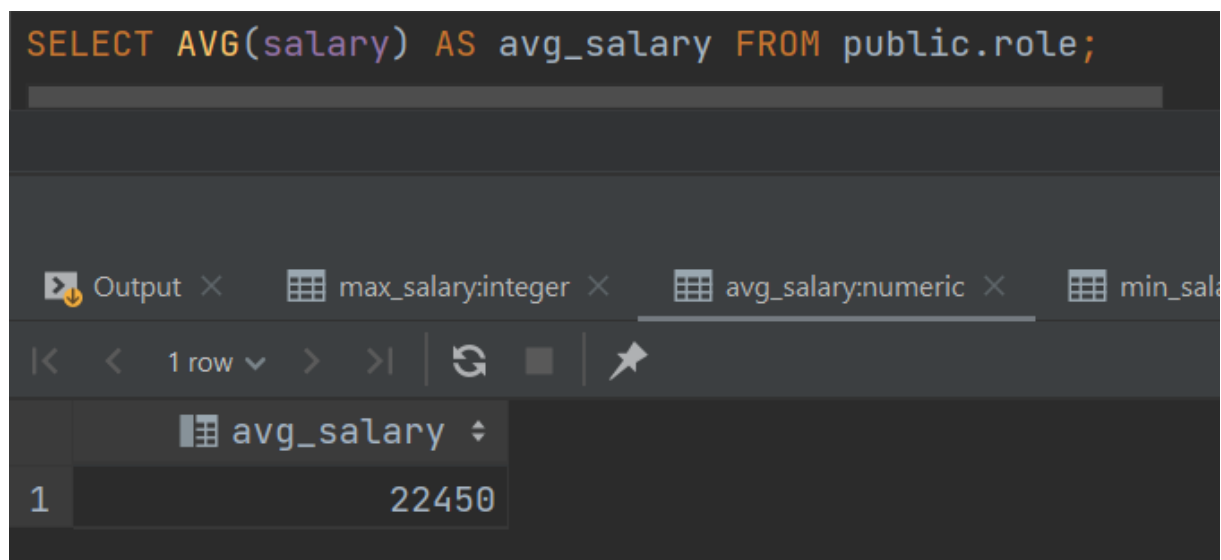
Dotaz MAX využít pro zjištění nejvyššího platu.



Obrázek č. 21 Dotaz s použitím MAX

Dotaz AVG využít pro zjištění průměrného platu.

```
SELECT AVG(salary) AS avg_salary FROM public.role;
```



The screenshot shows a database interface with a query editor at the top containing the SQL statement: `SELECT AVG(salary) AS avg_salary FROM public.role;`. Below the editor, there are tabs for 'Output', 'max_salary:integer', 'avg_salary:numeric', and 'min_salary:integer'. The 'avg_salary:numeric' tab is selected, displaying a table with one row and one column labeled 'avg_salary'. The value in the row is 22450.

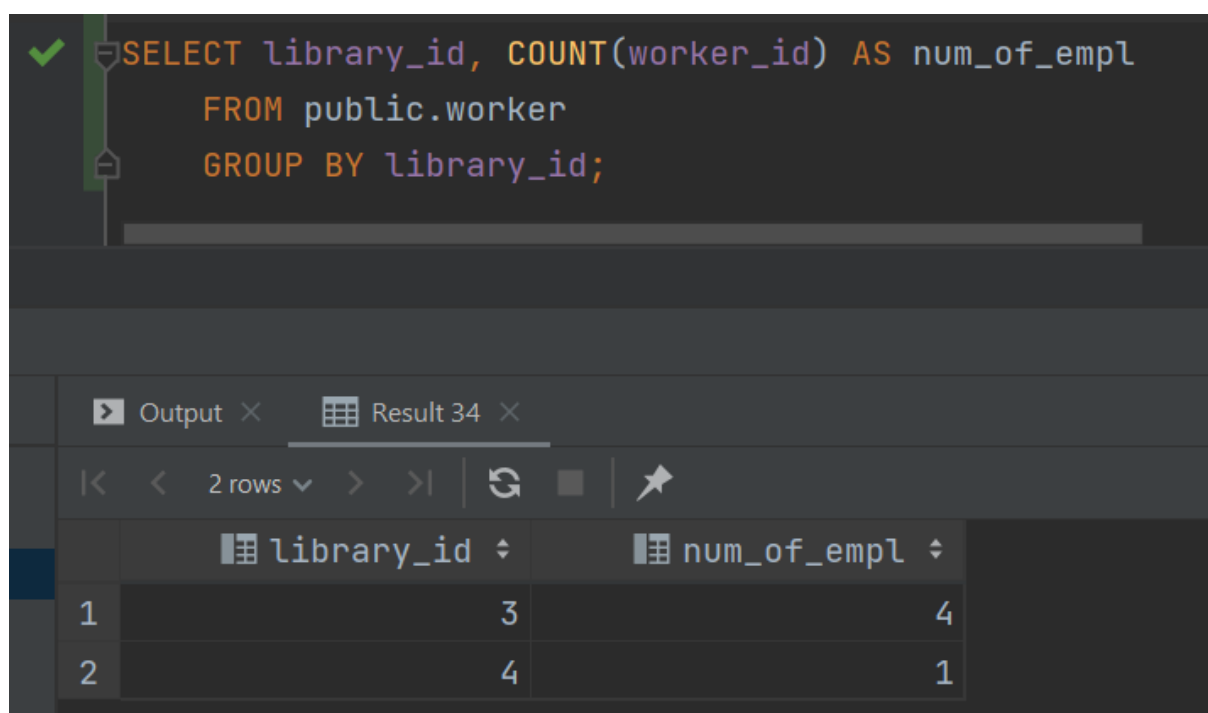
	avg_salary
1	22450

Obrázek č. 22 Dotaz s použitím AVG

1.11. Řada dotazů s použitím GROUP BY, GROUP BY a HAVING, GROUP BY, HAVING a WHERE

Využití dotazu s GROUP BY pro zjištění počtu zaměstnanců na jednotlivých pobočkách.

```
SELECT library_id, COUNT(worker_id) AS num_of_empl  
FROM public.worker  
GROUP BY library_id;
```

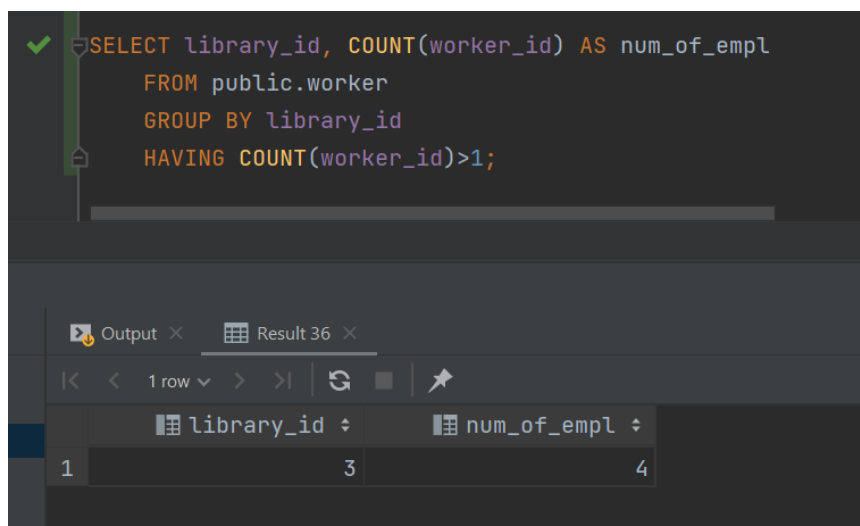


The screenshot shows a database interface with a query editor at the top containing the SQL statement: `SELECT library_id, COUNT(worker_id) AS num_of_empl FROM public.worker GROUP BY library_id;`. Below the editor, there are tabs for 'Output' and 'Result 34'. The 'Result 34' tab is selected, displaying a table with two columns: 'library_id' and 'num_of_empl'. There are two rows of data.

	library_id	num_of_empl
1	3	4
2	4	1

Obrázek č. 23 Dotaz s použitím GROUP BY

S pomocí GROUP BY a HAVING zobrazíme pouze pobočky, které mají více než 1 zaměstnanec.

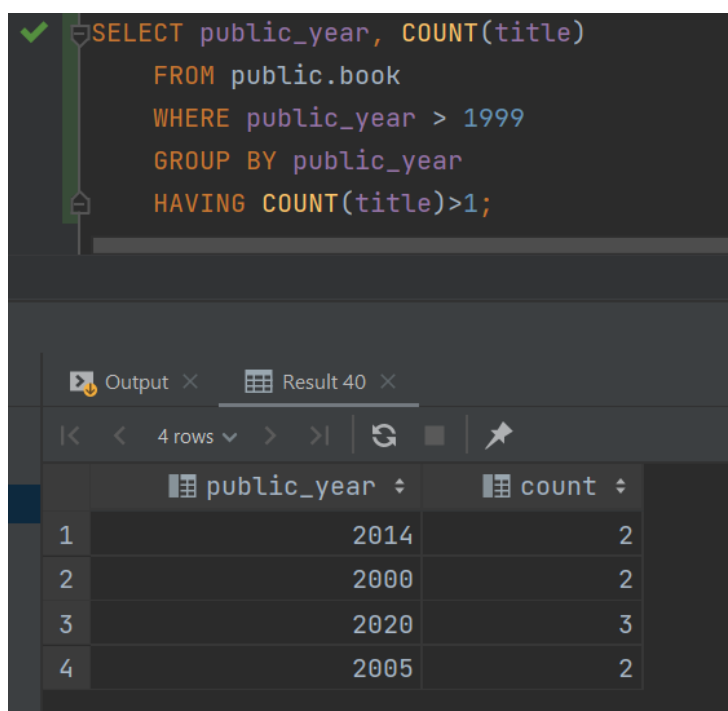


```
SELECT library_id, COUNT(worker_id) AS num_of_empl
FROM public.worker
GROUP BY library_id
HAVING COUNT(worker_id)>1;
```

library_id	num_of_empl
1	3

Obrázek č. 24 Dotaz s použitím GROUP BY a HAVING

Za pomocí GROUP BY, HAVING A WHERE vypisujeme počty knih vydaných po roce 1999, kde máme více než 1 vydanou knihu.



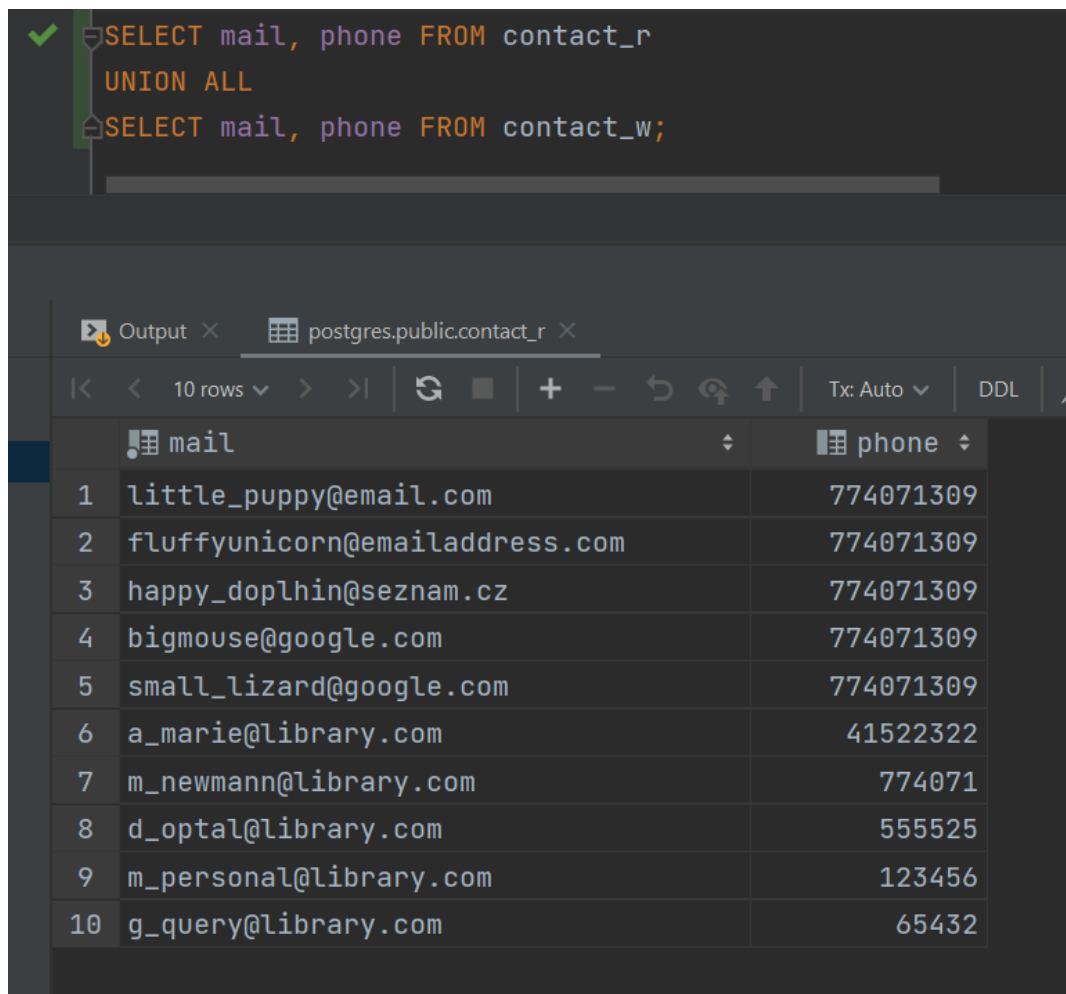
```
SELECT public_year, COUNT(title)
FROM public.book
WHERE public_year > 1999
GROUP BY public_year
HAVING COUNT(title)>1;
```

public_year	count
2014	2
2000	2
2020	3
2005	2

Obrázek č. 25 Dotaz s použitím GROUP BY, HAVING a WHERE

1.12. Dotaz s použitím UNION ALL

Dotaz s UNION ALL použijeme pro vypsání všech mailů a telefonů zaměstnanců a čtenářů.



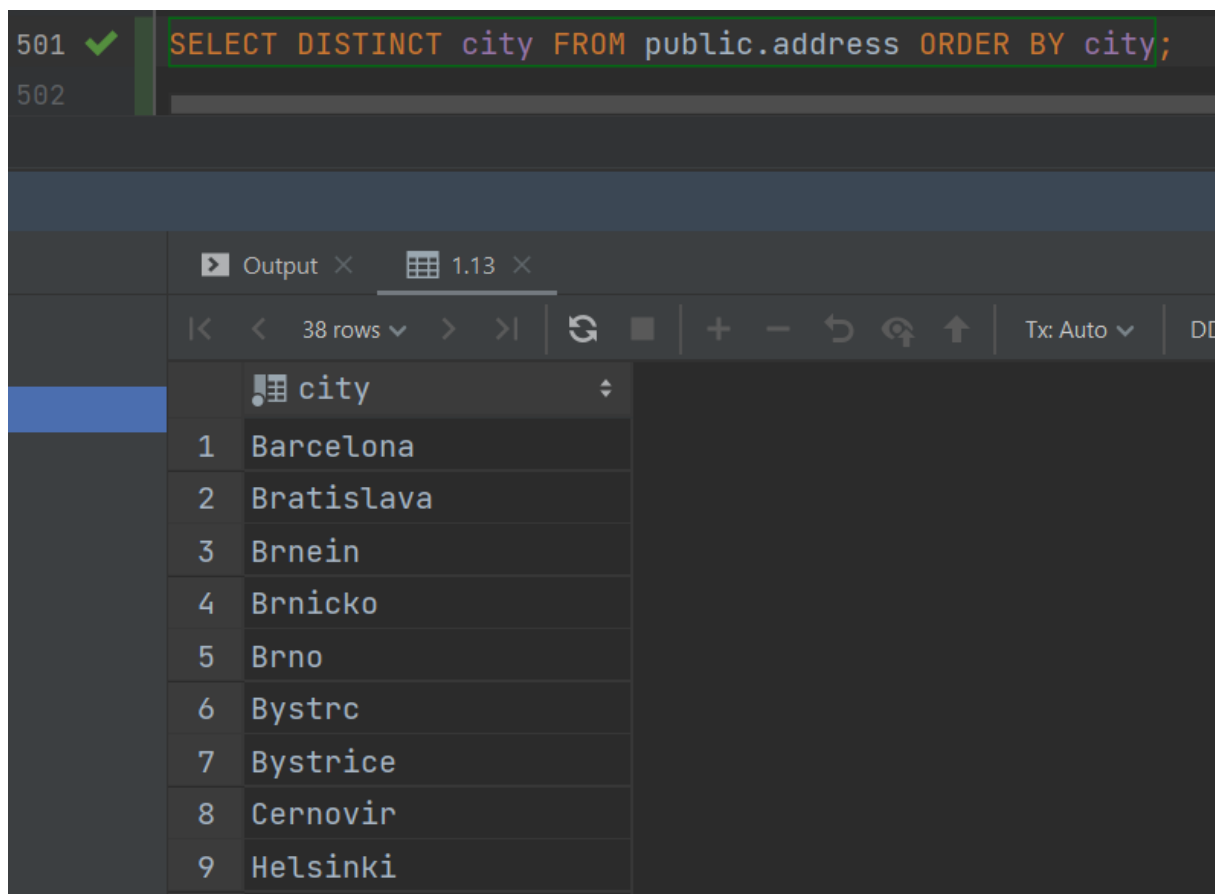
The screenshot shows a PostgreSQL query editor with a dark theme. At the top, a SQL query is entered: `SELECT mail, phone FROM contact_r UNION ALL SELECT mail, phone FROM contact_w;`. Below the query, the results are displayed in a table. The table has two columns: 'mail' and 'phone'. There are 10 rows of data. The first five rows have phone numbers starting with 774071309, and the last five rows have phone numbers starting with 41522322, 774071, 555525, 123456, and 65432.

	mail	phone
1	little_puppy@email.com	774071309
2	fluffyunicorn@emailaddress.com	774071309
3	happy_dolphin@seznam.cz	774071309
4	bigmouse@google.com	774071309
5	small_lizard@google.com	774071309
6	a_marie@library.com	41522322
7	m_newmann@library.com	774071
8	d_optal@library.com	555525
9	m_personal@library.com	123456
10	g_query@library.com	65432

Obrázek č. 26 Dotaz s použitím UNION ALL

1.13. Dotaz s použitím DISTINCT

Jelikož příkaz DISTINCT slouží k zobrazení jedinečných záznamů v tabulce využijeme jej pro získání všech měst. Výsledek dotazu je vypsán v alfabetském pořadí pro lehčí ověření, že se provedl správně. Záznamů bylo vypsáno více, jedná se pouze o ilustrační ukázkou vypsání.



Obrázek č. 27 Dotaz s použitím *DISTINCT*

1.14. Řada dotazů s použitím LEFT JOIN, RIGHT JOIN, FULL OUTER JOIN

Dotaz s LEFT JOIN a RIGHT JOIN použijeme dvakrát k zobrazení stejných dat, aby vynikl rozdíl. V obou příkazech zobrazujeme jméno a příjmení čtenáře spolu s jeho emailem a telefonním číslem.

V případě LEFT JOIN můžeme vidět, že spojujeme tabulku reader s tabulkou contact_r dle tabulky reader, čímž se pomocí SELECT zobrazí všichni čtenáři a kontakty, které k nim máme.

V případě RIGHT JOIN vidíme, že tabulky spojujeme na základě tabulky contact_r, proto se zobrazí i emaily a čísla, která nemají přiřazený žádný kontakt.

The screenshot shows a SQL query editor with a query that uses a LEFT JOIN to combine data from a 'reader' table and a 'contact_r' table. The query is highlighted with a green box. Below the query, the results are displayed in a table with 5 rows.

```

SELECT r.first_name, r.surname, cr.mail, cr.phone
FROM public.reader r
LEFT JOIN public.contact_r cr ON r.contact_r_id = cr.contact_r_id;

```

	first_name	surname	mail	phone
1	Kevin	Minion	little_puppy@email.com	774071309
2	Shrek	Ogre	fluffyunicorn@emailaddress.com	774071309
3	Fiona	Ogre	happy_dolphin@seznam.cz	774071309
4	Doris	Fish	bigmouse@google.com	774071309
5	Bob Robert	Minion	small_lizard@google.com	774071309

Obrázek č. 28 Dotaz s použitím LEFT JOIN

The screenshot shows a SQL query editor with a query that uses a RIGHT JOIN to combine data from a 'reader' table and a 'contact_r' table. The query is highlighted with a green box. Below the query, the results are displayed in a table with 7 rows, including two rows with null values.

```

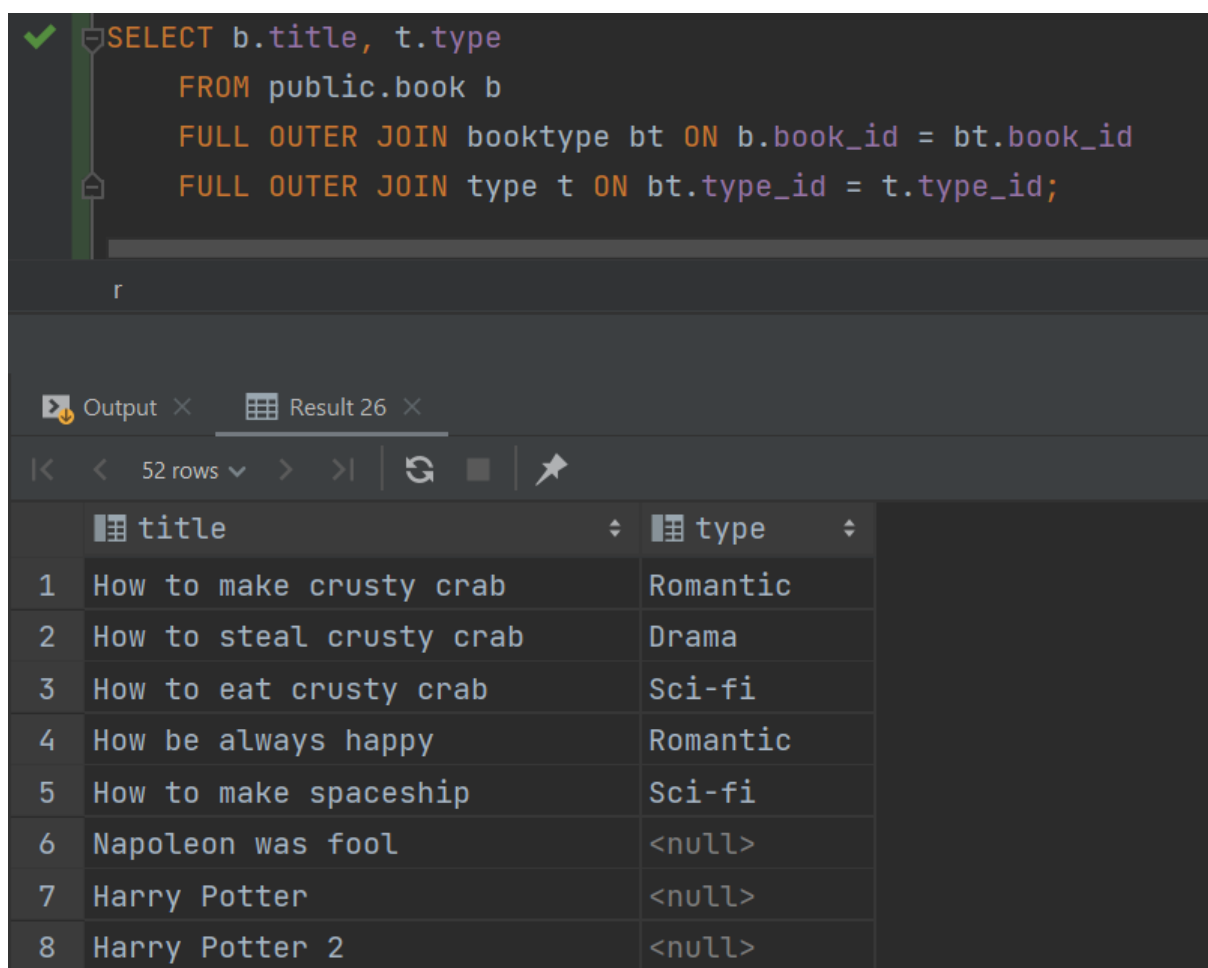
SELECT r.first_name, r.surname, cr.mail, cr.phone
FROM public.reader r
RIGHT JOIN public.contact_r cr ON r.contact_r_id = cr.contact_r_id;

```

	first_name	surname	mail	phone
1	Kevin	Minion	little_puppy@email.com	774071309
2	Shrek	Ogre	fluffyunicorn@emailaddress.com	774071309
3	Fiona	Ogre	happy_dolphin@seznam.cz	774071309
4	Doris	Fish	bigmouse@google.com	774071309
5	Bob Robert	Minion	small_lizard@google.com	774071309
6	<null>	<null>	m_personal@library.com	123456
7	<null>	<null>	g_query@library.com	65432

Obrázek č. 29 Dotaz s použitím RIGHT JOIN

Pro dotaz s FULL OUTER JOIN využijeme spojení tabulek book a type pro vypsání všech knih a žánrů. Můžeme vidět, že se vypisuje vše z obou tabulek, protože ne každá kniha má žánr, a ne každý žánr má přiřazenou knihu.



The screenshot shows a SQL query in a dark-themed editor. The query is a FULL OUTER JOIN between a table named 'book' (aliased as 'b') and a table named 'booktype' (aliased as 'bt'). The join is performed on 'book_id' and 'type_id' respectively. The result set is displayed in a table with two columns: 'title' and 'type'. The first 8 rows are visible, showing various book titles and their corresponding genres. Rows 6, 7, and 8 have a null value in the 'type' column.

```

SELECT b.title, t.type
FROM public.book b
FULL OUTER JOIN booktype bt ON b.book_id = bt.book_id
FULL OUTER JOIN type t ON bt.type_id = t.type_id;

```

	title	type
1	How to make crusty crab	Romantic
2	How to steal crusty crab	Drama
3	How to eat crusty crab	Sci-fi
4	How be always happy	Romantic
5	How to make spaceship	Sci-fi
6	Napoleon was fool	<null>
7	Harry Potter	<null>
8	Harry Potter 2	<null>

Obrázek č. 30 Vypsání s použitím FULL OUTER JOIN

47	How to survive pandemic	<null>
48	Just boys things	<null>
49	Something	<null>
50	Who is my neighbor?	<null>
51	<null>	Horror
52	<null>	Fantasy

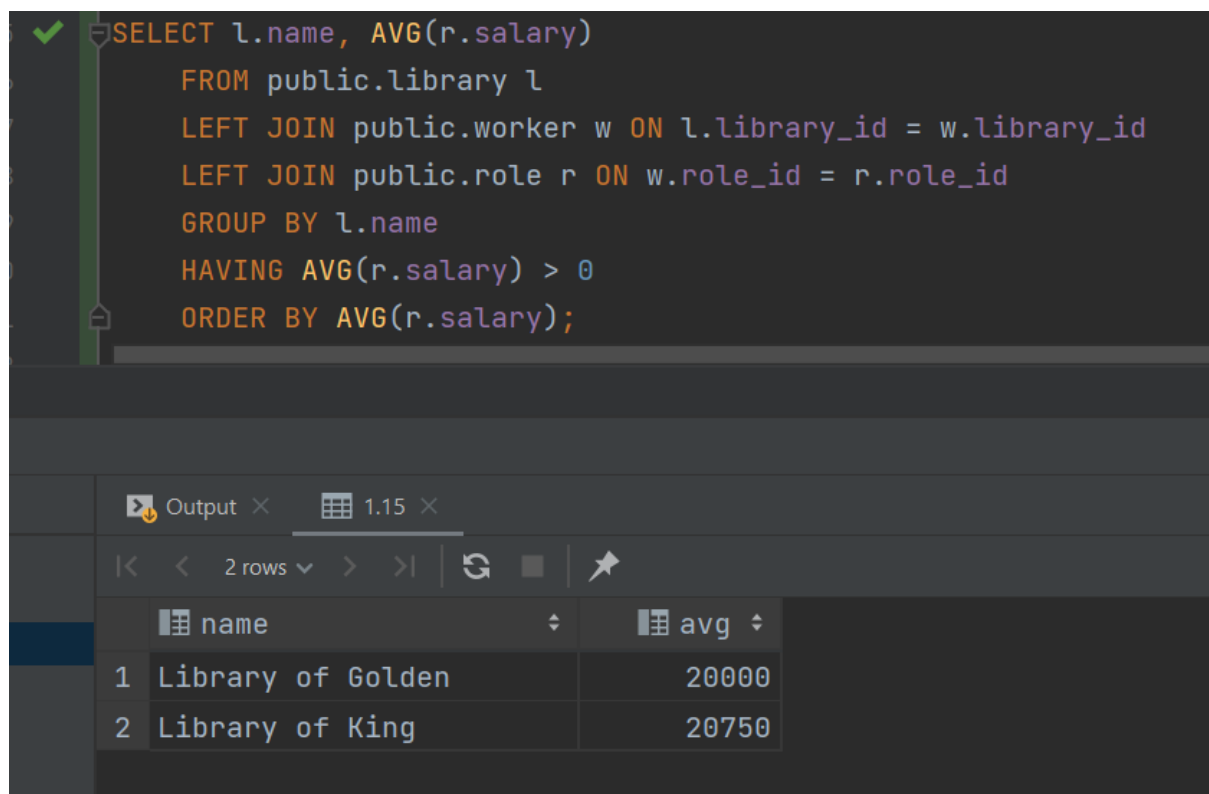
Obrázek č. 31 Ukázka vypsání dotazu s FULL OUTER JOIN

Záznamy jsou vypsány všechny, obrázky jsou pouze ilustrační pro reálnou ukázkou psané definice.

1.15. Dotaz s využitím LEFT JOIN, GROUP BY, HAVING, ORDER BY a AVG

Tento dotaz využíváme pro vypsání knihoven dle průměrného platu na jednotlivých pobočkách. Abychom vypsali pouze pobočky se zaměstnanci pomocí HAVING si vytváříme podmínku, že průměrný plat je větší než 0.

```
✓ SELECT l.name, AVG(r.salary)
   FROM public.library l
  LEFT JOIN public.worker w ON l.library_id = w.library_id
  LEFT JOIN public.role r ON w.role_id = r.role_id
  GROUP BY l.name
  HAVING AVG(r.salary) > 0
  ORDER BY AVG(r.salary);
```



	name	avg
1	Library of Golden	20000
2	Library of King	20750

Obrázek č. 32 Dotaz s použitím LEFT JOIN, GROUP BY, HAVING, ORDER BY a AVG

1.16. Dotaz, který vrátí data z tabulky za poslední 1 den a 12 hodin

Pro tento dotaz bylo potřeba přidat několik záznamů do tabulky borrow, aby byla splněna podmínka záznamu mladšího než 1 den a 12 hodin.

The screenshot shows a PostgreSQL query editor with a successful query execution. The query selects data from the `public.borrow` table for the last 36 hours. The results are displayed in a table with 3 rows.

```

SELECT borrow_date, book_id, reader_id
FROM public.borrow
WHERE borrow_date > now() - interval '36 hours';

```

	borrow_date	book_id	reader_id
1	2021-11-08 00:00:00.000000	2	3
2	2021-11-08 00:00:00.000000	1	2
3	2021-11-08 00:00:00.000000	3	1

Obrázek č. 33 Výpis z tabulky za poslední 1 den a 12 hodin

1.17. Dotaz, který vrátí data z minulého měsíce

Zde je opět potřeba vložit data do tabulky `borrow` s vyhovujícími daty z měsíce říjen.

The screenshot shows a PostgreSQL query editor with two successful queries. The first query inserts two rows into the `public.borrow` table for the month of October 2021. The second query selects data from the `public.borrow` table for the month of October 2021. The results are displayed in a table with 2 rows.

```

INSERT INTO public.borrow (borrow_date, book_id, reader_id) VALUES ('2021-10-01', 12, 2);
INSERT INTO public.borrow (borrow_date, book_id, reader_id) VALUES ('2021-10-31', 12, 2);
SELECT borrow_date, book_id, reader_id
FROM public.borrow
WHERE borrow_date >= date_trunc('month', current_date - interval '1' month)
and borrow_date < date_trunc('month', current_date);

```

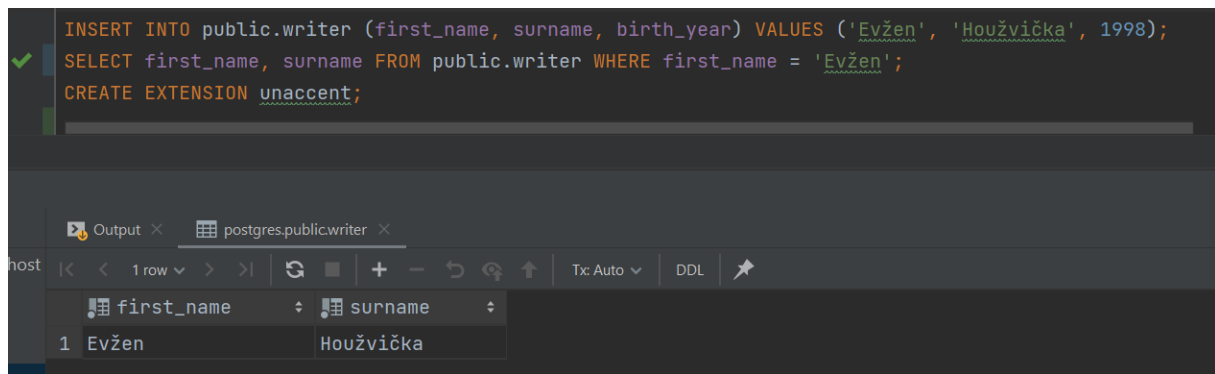
	borrow_date	book_id	reader_id
1	2021-10-01 00:00:00.000000	12	2
2	2021-10-31 00:00:00.000000	12	2

Obrázek č. 34 Výpis z tabulky za minulý měsíc

1.18. Dotaz, který odstraní diakritiku

Pro tento dotaz vložíme nový záznam do tabulky `writer` a vytvoříme nové rozšíření `unaccent`.

```
INSERT INTO public.writer (first_name, surname, birth_year) VALUES ('Evžen', 'Houžvička', 1998);
SELECT first_name, surname FROM public.writer WHERE first_name = 'Evžen';
CREATE EXTENSION unaccent;
```

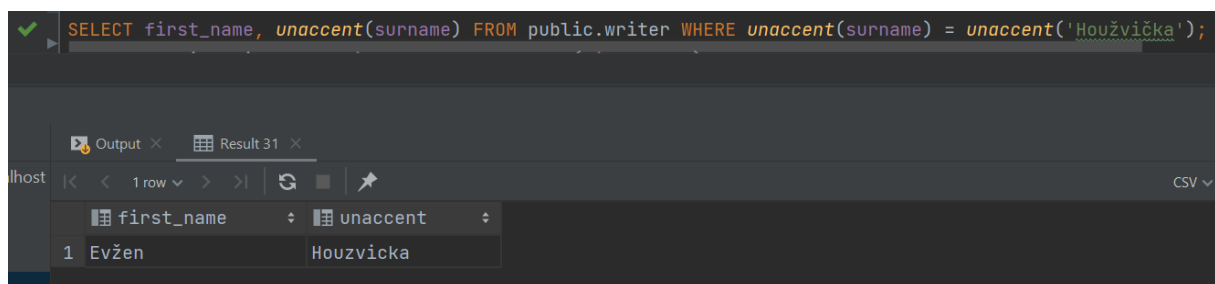


first_name	surname
1 Evžen	Houžvička

Obrázek č. 35 Tvorba nového záznamu a rozšíření

Dotazem SELECT bohužel neodstraníme diakritiku, pouze ji nezobrazíme při vypsání.

```
SELECT first_name, unaccent(surname) FROM public.writer WHERE unaccent(surname) = unaccent('Houžvička');
```

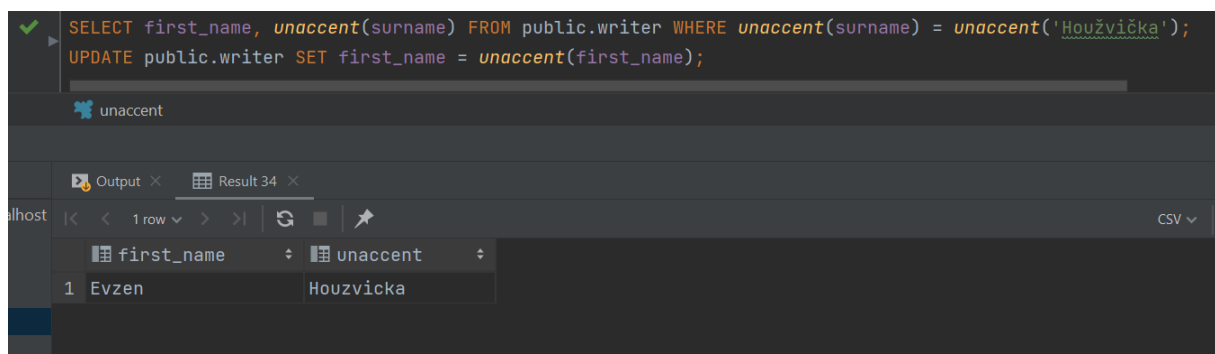


first_name	unaccent
1 Evžen	Houzvicka

Obrázek č. 36 Vypsání části záznamu bez diakritiky

Pro opravdové odstranění diakritiky musíme využít dotaz s UPDATE a poté můžeme znovu vypsát stejný SELECT pro porovnání změn. Nyní v tabulce máme namísto „Evžen“ uloženo „Evzen“, zatímco „Houžvička“ stále zůstává nezměněno, pouze je vypsáno bez diakritiky.

```
SELECT first_name, unaccent(surname) FROM public.writer WHERE unaccent(surname) = unaccent('Houžvička');
UPDATE public.writer SET first_name = unaccent(first_name);
```

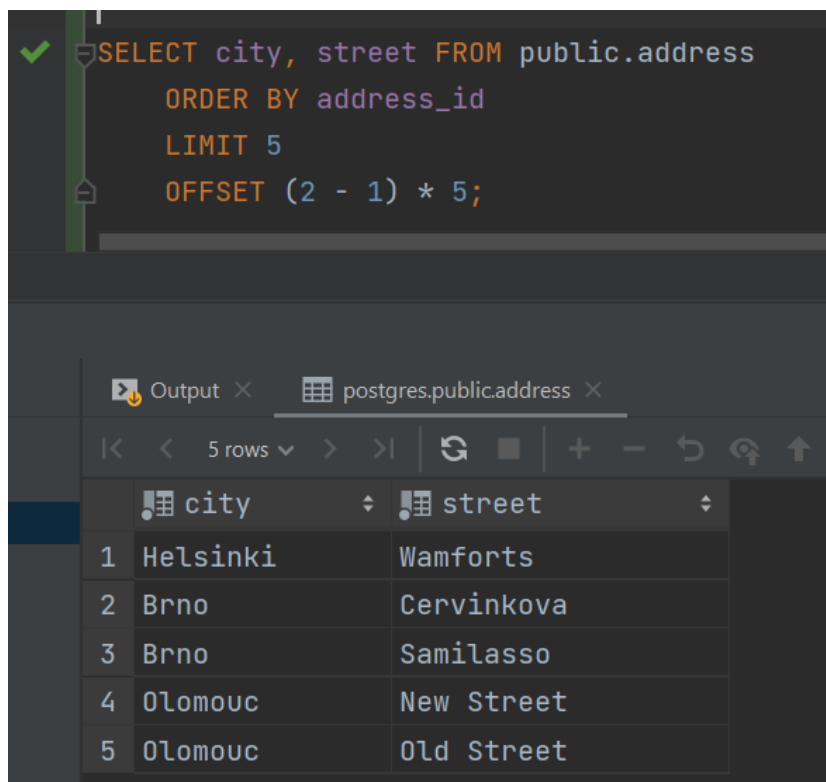


first_name	unaccent
1 Evzen	Houzvicka

Obrázek č. 37 Vypsání záznamu bez diakritiky pro srovnání

1.19. Dotaz pro paging s pomocí LIMIT a OFFSET

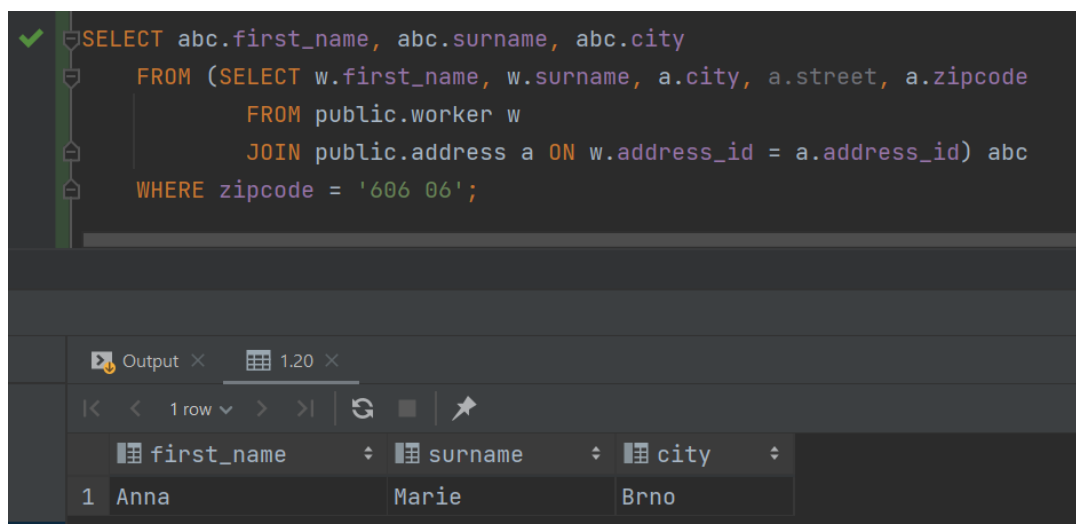
Dotaz pro paging z tabulky address, vybíráme 5 záznamů z druhé stránky.



Obrázek č. 38 Výpis pagging

1.20. Dotaz s poddotazem ve FROM

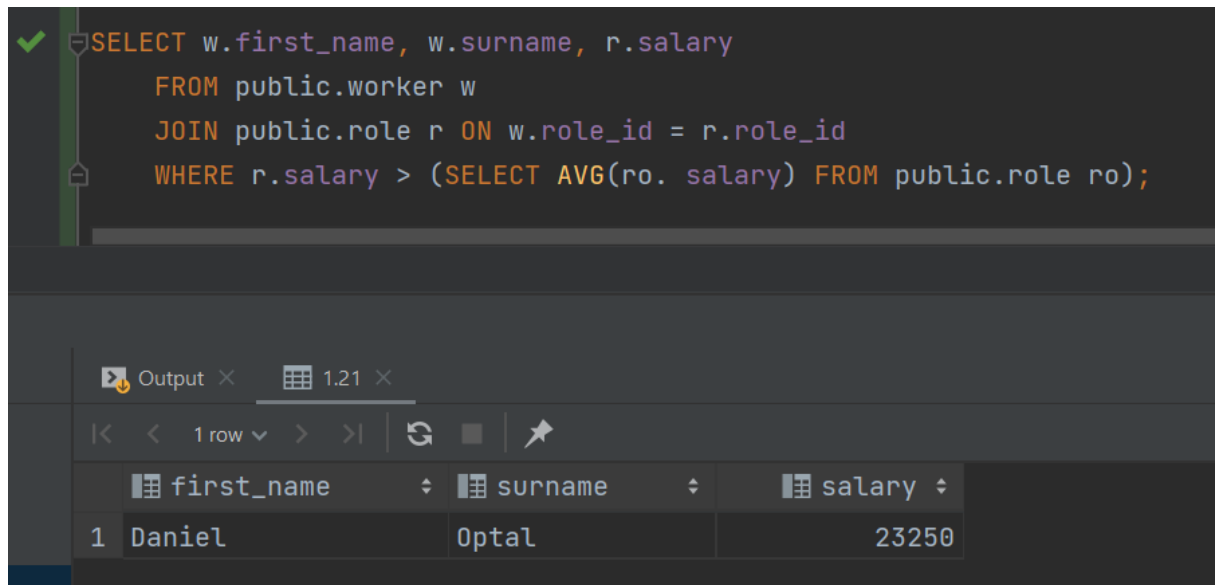
Tento dotaz s poddotazem využijeme k vypsání jména a příjmení všech zaměstnanců, kteří pochází z místa, jehož zipcode je „606 06“.



Obrázek č. 39 Dotaz s poddotazem ve FROM

1.21. Dotaz s poddotazem v podmínce WHERE

Dotaz s poddotazem v podmínce WHERE nám vypíše jméno zaměstnanců, kteří mají plat větší než průměrný plat.



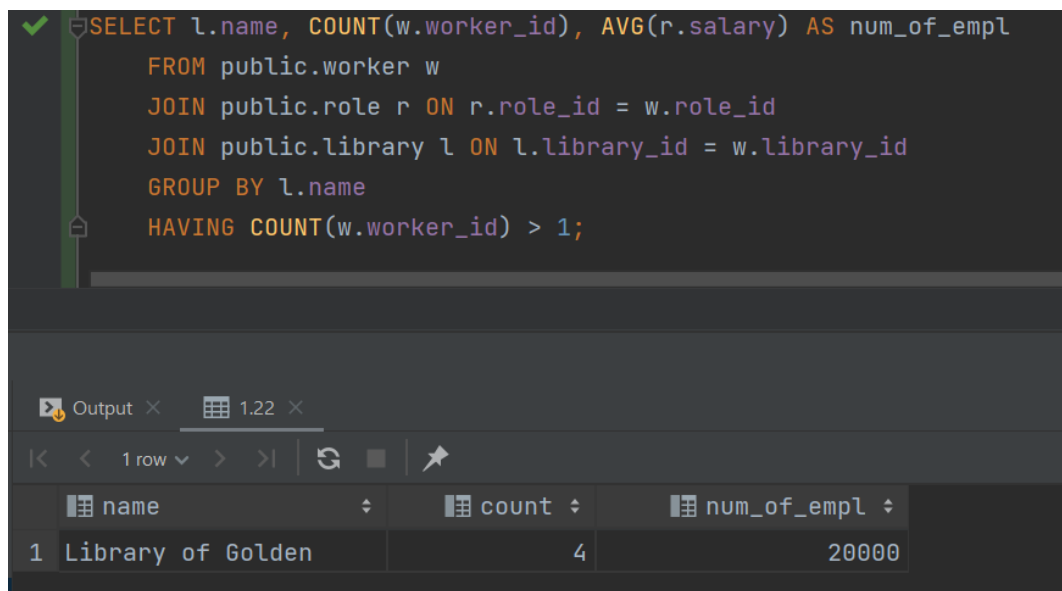
```
SELECT w.first_name, w.surname, r.salary
FROM public.worker w
JOIN public.role r ON w.role_id = r.role_id
WHERE r.salary > (SELECT AVG(ro. salary) FROM public.role ro);
```

	first_name	surname	salary
1	Daniel	Optal	23250

Obrázek č. 40 Dotaz s poddotazem ve WHERE

1.22. Dotaz používající agregační funkci a GROUP BY s HAVING

Dotazem s GROUP BY a HAVING využijeme pro vypísání názvu knihovny, počtu zaměstnanců a jejich průměrného platu všude, kde je více než 1 zaměstnanec.



```
SELECT l.name, COUNT(w.worker_id), AVG(r.salary) AS num_of_empl
FROM public.worker w
JOIN public.role r ON r.role_id = w.role_id
JOIN public.library l ON l.library_id = w.library_id
GROUP BY l.name
HAVING COUNT(w.worker_id) > 1;
```

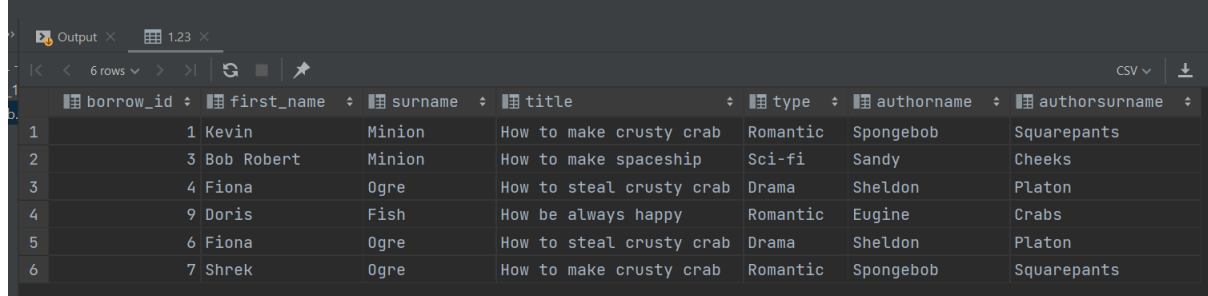
	name	count	num_of_empl
1	Library of Golden	4	20000

Obrázek č. 41 Dotaz používající agregační funkci, GROUP BY a HAVING

1.23. Dotaz, který spojí alespoň 5 tabulek

Tímto dotazem spojíme tabulky borrow, reader, book, type a writer. Dotazem zjišťujeme, jakou knihu má čtenář půjčenou a kdo je autorem oné knížky a jaký žánr tato kniha má.

```
SELECT br.borrow_id, r.first_name, r.surname, bk.title, t.type, w.first_name AS AuthorName, w.surname AS AuthorSurname
FROM public.borrow br
JOIN public.reader r ON br.reader_id = r.reader_id
JOIN public.book bk ON br.book_id = bk.book_id
JOIN public.bookwriter bw ON bk.book_id = bw.book_id
JOIN public.writer w ON bw.writer_id = w.writer_id
JOIN public.booktype bt ON bk.book_id = bt.book_id
JOIN public.type t ON bt.type_id = t.type_id;
```



	borrow_id	first_name	surname	title	type	authorname	authorsurname
1	1	Kevin	Minion	How to make crusty crab	Romantic	Spongebob	Squarepants
2	3	Bob Robert	Minion	How to make spaceship	Sci-fi	Sandy	Cheeks
3	4	Fiona	Ogre	How to steal crusty crab	Drama	Sheldon	Platon
4	9	Doris	Fish	How be always happy	Romantic	Eugine	Crabs
5	6	Fiona	Ogre	How to steal crusty crab	Drama	Sheldon	Platon
6	7	Shrek	Ogre	How to make crusty crab	Romantic	Spongebob	Squarepants

Obrázek č. 42 Dotaz, který spojuje více než 5 tabulek

1.24. Dotaz, který spojí alespoň 3 tabulky a použije GROUP BY, HAVING a WHERE

Tento dotaz použijeme pro spojení tabulek worker, library a role. Pomocí funkce WHERE určíme podmínku, že plat musí být vyšší než 15 000, pomocí GROUP BY sjednotíme záznamy na základě jména knihovny a s pomocí HAVING určíme podmínku, že počet zaměstnanců je vyšší než 1.

```
✓ SELECT l.name, COUNT(w.worker_id) AS num_of_empl
    FROM public.worker w
    JOIN public.library l ON w.library_id = l.library_id
    JOIN public.role r ON r.role_id = w.role_id
    WHERE r.salary > 15000
    GROUP BY l.name
    HAVING COUNT(w.worker_id) > 1;
```

Output x 1.24 x

1 row

	name	num_of_empl
1	Library of Golden	4

Obrázek č. 43 Dotaz, který spojuje alespoň 3 tabulky a obsahuje WHERE, GROUP BY a HAVING

2. Úprava databáze z prvního zadání pro lepší integritu

Tabulka user byla pro svou nadbytečnost odstraněna.

```
DROP TABLE public.user;
```

Obrázek č. 44 Dotaz pro odstranění nadbytečné tabulky

V tabulce reader bylo nastaveno kaskádování pro tabulku contact_r, kvůli návaznosti na kontaktní údaje. Mezi tabulkami reader a address zůstala kaskádovitost NO ACTION.

Pro vztah mezi tabulkou worker a contact_w bylo nastaveno kaskádování. Pro zbytek vztahů zůstalo kaskádování NO ACTION.

```

CONSTRAINT fk_reader_address
    FOREIGN KEY (address_id)
    REFERENCES public.address (address_id)
    ON DELETE NO ACTION
    ON UPDATE NO ACTION,
CONSTRAINT fk_reader_contact
    FOREIGN KEY (contact_r_id)
    REFERENCES public.contact_R (contact_r_id)
    ON DELETE CASCADE
    ON UPDATE CASCADE,

```

Obrázek č. 45 Ukázka kaskádování

3. Vytvoření indexu databáze

První index vytváříme u tabulky borrow nad sloupcem borrow_date pro rychlejší vyhledávání v datumech, kdy si čtenář půjčil knihu. Druhý index je vytvořen ve stejné tabulce nad sloupcem return_date, pro rychlejší vyhledávání, kdy čtenář vrátil knihu.

```

DROP INDEX IF EXISTS date_index;
CREATE INDEX IF NOT EXISTS date_index ON public.borrow USING btree(borrow_date);
CREATE INDEX IF NOT EXISTS date_index ON public.borrow(borrow_date);
SELECT b.borrow_id, b.borrow_date, b.book_id, b.reader_id FROM public.borrow b WHERE borrow_date = '2021-10-01';

```

	borrow_id	borrow_date	book_id	reader_id
1	11	2021-10-01 00:00:00.000000	12	2

Obrázek č. 46 Vytvoření indexu pro borrow_date a následné vypsání SELECTU s tímto sloupcem

```
DROP INDEX IF EXISTS return_date_index;  
CREATE INDEX IF NOT EXISTS return_date_index ON public.borrow USING btree(return_date);  
CREATE INDEX IF NOT EXISTS return_date_index ON public.borrow(return_date);
```

Obrázek č. 47 Index nad return_date v tabulce borrow

Třetí index je tvořen v tabulce book nad sloupcem title pro rychlejší vyhledávání dle názvu knihy.

```
DROP INDEX IF EXISTS book_title_index;  
CREATE INDEX IF NOT EXISTS book_title_index ON public.book USING btree(title);  
CREATE INDEX IF NOT EXISTS book_title_index ON public.book(title);
```

Obrázek č. 48 Index nad sloupcem title v tabulce book

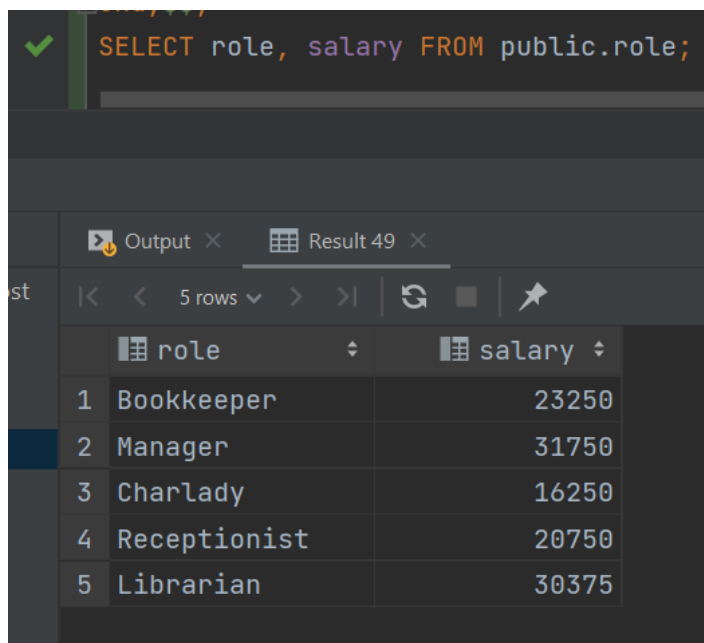
Poslední index je vytvořen v tabulce writer nad sloupcem surname pro rychlejší vyhledávání dle příjmení autora. Index nevytváříme nad sloupcem se jménem, jelikož zde hrozí velká shoda jmen, což by nám proces moc neurychlilo, vyhledáváme většinou proto dle příjmení autora.

```
DROP INDEX IF EXISTS writer_surname_index;  
CREATE INDEX IF NOT EXISTS writer_surname_index ON public.writer USING btree(surname);  
CREATE INDEX IF NOT EXISTS writer_surname_index ON public.writer(surname);
```

Obrázek č. 49 Index nad sloupcem surname v tabulce writer

4. Procedura databáze

Je vytvořena procedura pro zvýšení platů o různé procenta dle pozice.



```
SELECT role, salary FROM public.role;
```

	role	salary
1	Bookkeeper	23250
2	Manager	31750
3	Charlady	16250
4	Receptionist	20750
5	Librarian	30375

Obrázek č. 50 Stav před zavoláním procedury

```

DROP PROCEDURE IF EXISTS salary_change();
CREATE OR REPLACE PROCEDURE salary_change ()
language plpgsql
as $$
begin

    UPDATE public.role
    SET salary = salary + 0.5*salary
    WHERE role_id = 2;

    UPDATE public.role
    SET salary = salary + 0.5*salary
    WHERE role_id = 1;

    UPDATE public.role
    SET salary = salary + 0.25*salary
    WHERE role_id = 3;

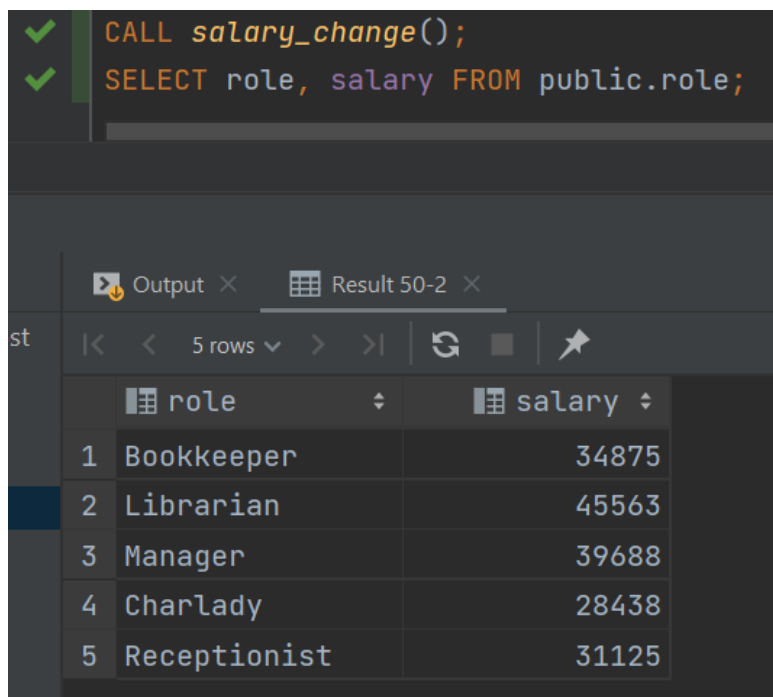
    UPDATE public.role
    SET salary = salary + 0.75*salary
    WHERE role_id = 4;

    UPDATE public.role
    SET salary = salary + 0.5*salary
    WHERE role_id = 5;

    commit;
end;$$;

```

Obrázek č. 51 Procedura pro zvýšení platu



Obrázek č. 52 Zavolání procedury s následným vypsáním pro porovnání údajů

5. Trigger databáze

Trigger vytvoříme pro hlídání, že částka platu je kladné, větší než 0.

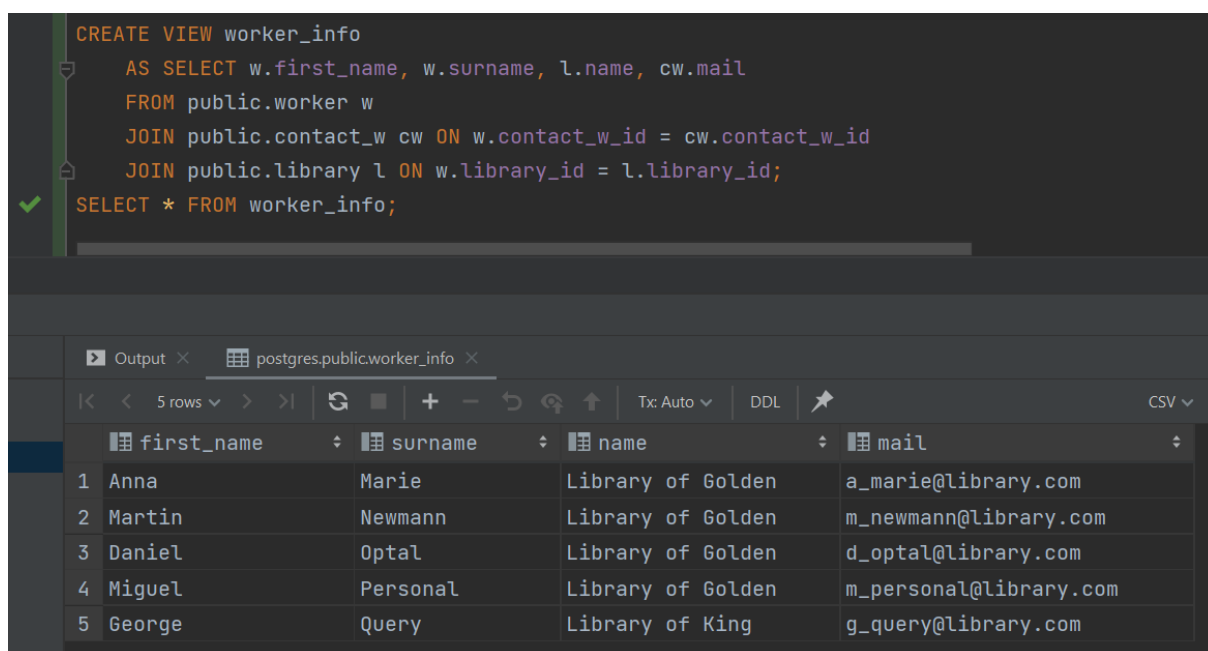

```
550 CREATE OR REPLACE FUNCTION salary_verification_func()
551 RETURNS TRIGGER
552 LANGUAGE PLPGSQL
553 AS
554 $$
555 BEGIN
556 IF NEW.salary < 0 THEN
557 RAISE EXCEPTION 'Salary must be positive.';
558 END IF;
559 RETURN NEW;
560 END;
561 $$
562
563 CREATE TRIGGER salary_verification
564 BEFORE INSERT ON public.role
565 FOR EACH ROW EXECUTE PROCEDURE salary_verification_func();
566
567 ! INSERT INTO role(salary) VALUES(-100);
568
569 |
```

[P0001] ERROR: Salary must be positive.
Kde: PL/pgSQL funkce salary_verification_func() řádek 4 na RAISE

Obrázek č. 53 Tvorba triggeru spolu s jeho následným ověřením

6. Pohled databáze

Zobrazení databáze pro získání jména a příjmení pracovníka, jeho emailu a také jména knihovny ve které pracuje.



```

CREATE VIEW worker_info
AS SELECT w.first_name, w.surname, l.name, cw.mail
FROM public.worker w
JOIN public.contact_w cw ON w.contact_w_id = cw.contact_w_id
JOIN public.library l ON w.library_id = l.library_id;
SELECT * FROM worker_info;

```

	first_name	surname	name	mail
1	Anna	Marie	Library of Golden	a_marie@library.com
2	Martin	Newmann	Library of Golden	m_newmann@library.com
3	Daniel	Optal	Library of Golden	d_optal@library.com
4	Miguel	Personal	Library of Golden	m_personal@library.com
5	George	Query	Library of King	g_query@library.com

Obrázek č. 54 Pohled databáze

7. Materializovaný pohled na databázi

Vytváříme materializovaný pohled na databázi pro zobrazení jména a příjmení autora s počtem knih v jednotlivých žánrech. Spojujeme tak tabulky writer, bookwriter, book, booktype a type. Jako agregační funkce je zde zvolena COUNT určení počtu knih. Vypisujeme pouze autory které mají zadané příjmení a záznamy sdružujeme dle jména, příjmení autora a žánru knihy. Tento pohled je velice důležitý například ve chvíli, kdy chceme zjistit počty děl od autora či počty děl v jednotlivých žánrech u autora.

```

CREATE MATERIALIZED VIEW mat_view_1 AS SELECT w.first_name, w.surname, COUNT(b.title), t.type
FROM public.writer w
JOIN public.bookwriter bw ON w.writer_id = bw.writer_id
JOIN public.book b ON bw.book_id = b.book_id
JOIN public.booktype bt ON b.book_id = bt.book_id
JOIN public.type t ON bt.type_id = t.type_id
WHERE w.surname IS NOT NULL
GROUP BY w.first_name, w.surname, t.type
HAVING COUNT(b.title) > 0;

SELECT * FROM mat_view_1;

```

	first_name	surname	count	type
1	Sheldon	Platon	1	Drama
2	Spongebob	Squarepants	1	Romantic
3	Eugene	Crabs	1	Romantic
4	Sandy	Cheeks	1	Sci-fi

Obrázek č. 55 Materializovaný pohled databáze

8. Role teacher a student

Role jsou vytvořeny dle zadání. Učitel má neomezená práva nad tabulkou worker a pak select pouze nad vybranými sloupci z tabulky role a reader. Student má možnost pouze selectu v tabulkách borrow a book.

```

DROP ROLE IF EXISTS teacher;
CREATE ROLE teacher NOSUPERUSER;
GRANT SELECT, INSERT, UPDATE, DELETE ON public.worker TO teacher;
-- REVOKE all ON public.role FROM teacher;
GRANT SELECT (role) ON public.role TO teacher;
GRANT SELECT (first_name) ON public.reader TO teacher;
GRANT SELECT (surname) ON public.reader TO teacher;

DROP ROLE IF EXISTS student;
CREATE ROLE student NOSUPERUSER;
GRANT SELECT ON public.borrow, public.book TO student;

```

Obrázek č. 56 Vytvoření rolí a přidělení práv

Závěr

V tomto projektu bylo úspěšně dokázáno, čeho všeho jde dotazováním dosáhnout. Dále zde bylo prakticky vyzkoušeno několik agregačních funkcí. V neposlední řadě díky tomuto projektu vznikla možnost vyzkoušet si práci s pohledy, triggerem a procedurami.

Seznam obrázků

Obrázek č. 1 Vypsání vybraných sloupců z tabulky address.....	4
Obrázek č. 2 Výsledek hledání dle emailu v tabulce reader.....	5
Obrázek č. 3 Výsledek hledání dle emailu v tabulce worker	5
Obrázek č. 4 Stav před dotazem UPDATE	6
Obrázek č. 5 Zvýšení platů po použití dotazu UPDATE.....	6
Obrázek č. 6 Kontrolní výpis před INSERT	7
Obrázek č. 7 Kontrolní výpis po INSERT	7
Obrázek č. 8 Stav před použitím DELETE.....	8
Obrázek č. 9 Stav po použití DELETE	8
Obrázek č. 10 Stav tabulky před použitím ALTER TABLE.....	9
Obrázek č. 11 Stav tabulky po použití ALTER TABLE	9
Obrázek č. 12 Vypsání čtenářů se shodným příjmením za pomoci WHERE	10
Obrázek č. 13 Vypsání všech knih, které mají v názvu Potter.....	10
Obrázek č. 14 Vypsání všech adres kromě Brna	11
Obrázek č. 15 Vypsání záznamu s bílými znaky.....	11
Obrázek č. 16 Opětovné vypsání záznamu, tentokrát už zbaveného bílých znaků	12
Obrázek č. 17 Vypsání příjmení a iniciálu jména autorů	12
Obrázek č. 18 Dotaz s použitím COUNT	13
Obrázek č. 19 Dotaz s použitím SUM.....	13
Obrázek č. 20 Dotaz s použitím MIN	14
Obrázek č. 21 Dotaz s použitím MAX.....	14
Obrázek č. 22 Dotaz s použitím AVG	15
Obrázek č. 23 Dotaz s použitím GROUP BY	15
Obrázek č. 24 Dotaz s použitím GROUP BY a HAVING.....	16
Obrázek č. 25 Dotaz s použitím GROUP BY, HAVING a WHERE.....	16
Obrázek č. 26 Dotaz s použitím UNION ALL	17
Obrázek č. 27 Dotaz s použitím DISTINCT	18
Obrázek č. 28 Dotaz s použitím LEFT JOIN	19
Obrázek č. 29 Dotaz s použitím RIGHT JOIN.....	19
Obrázek č. 30 Vypsání s použitím FULL OUTER JOIN	20
Obrázek č. 31 Ukázka vypsání dotazu s FULL OUTER JOIN.....	20
Obrázek č. 32 Dotaz s použitím LEFT JOIN, GROUP BY, HAVING, ORDER BY a AVG.....	21
Obrázek č. 33 Výpis z tabulky za poslední 1 den a 12 hodin	22
Obrázek č. 34 Výpis z tabulky za minulý měsíc	22
Obrázek č. 35 Tvorba nového záznamu a rozšíření	23
Obrázek č. 36 Vypsání části záznamu bez diakritiky	23
Obrázek č. 37 Vypsání záznamu bez diakritiky pro srovnání	23
Obrázek č. 38 Výpis pagging.....	24
Obrázek č. 39 Dotaz s poddotazem ve FROM.....	24
Obrázek č. 40 Dotaz s poddotazem ve WHERE	25
Obrázek č. 41 Dotaz používající agregační funkci, GROUP BY a HAVING	25
Obrázek č. 42 Dotaz, který spojuje více než 5 tabulek.....	26
Obrázek č. 43 Dotaz, který spojuje alespoň 3 tabulky a obsahuje WHERE, GROUP BY a HAVING	27
Obrázek č. 44 Dotaz pro odstranění nadbytečné tabulky.....	27

Obrázek č. 45 Ukázka kaskádování	28
Obrázek č. 46 Vytvoření indexu pro borrow_date a následné vypsání SELECTU s tímto sloupcem.....	28
Obrázek č. 47 Index nad return_date v tabulce borrow.....	29
Obrázek č. 48 Index nad sloupcem title v tabulce book	29
Obrázek č. 49 Index nad sloupcem surname v tabulce writer	29
Obrázek č. 50 Stav před zavoláním procedury	30
Obrázek č. 51 Procedura pro zvýšení platu	31
Obrázek č. 52 Zavolání procedury s následným vypsáním pro porovnání údajů.....	32
Obrázek č. 53 Tvorba triggeru spolu s jeho následným ověřením	33
Obrázek č. 54 Pohled databáze	34
Obrázek č. 55 Materializovaný pohled databáze	35
Obrázek č. 56 Vytvoření rolí a přidělení práv	35