

SQL

pro každý den

Jak si užít SQL
a nezabloudit
v kódu

Lubomír Husar



SQL pro každý den

Lubomír Husar

Revize 1.0

Copyright

Lubomír Husar

SQL pro každý den

Jak si užít SQL a nezabloudit v kódu.

Revize: 1.0

Elektronické knihy LovelyData lze zakoupit pro vzdělávací, obchodní nebo propagační účely. Pro více informací nás kontaktujte emailem na kurzy@lovelydata.cz nebo telefonicky na +420 228 224 689.

Grafický návrh obálky © 2022 grafikli.cz

Grafická úprava a sazba © 2022 grafikli.cz

© 2022 LovelyData.cz

Knihy byla zakoupena na www.lovelydata.cz.

Knihy je určena pouze pro potřeby kupujícího. Žádná část této publikace nesmí být kopírována a rozmnožována za účelem rozšiřování v jakékoli formě či jakýmkoli způsobem bez písemného souhlasu vydavatele.

Vydavatel ani autor nenesou žádnou zodpovědnost za případné škody vzniklé použitím informací popisovaných v této publikaci. Pokud jsou ukázky kódu nebo jiné technologie, které toto dílo obsahuje nebo popisuje, předmětem licencí open source nebo práv duševního vlastnictví jiných osob, je vaší odpovědností zajistit, aby jejich použití bylo v souladu s těmito licencemi a/nebo právy.

Názvy produktů, firem, apod. použité v knize mohou být ochrannými známkami nebo registrovanými ochrannými známkami příslušných vlastníků.

LovelyData je obchodní značkou společnosti Colorbee, s.r.o.



Nikdo se nenarodil učený.

— Ptahhotep, 2400 př. n. l.

Obsah

Copyright	3
Citát	4
Úvod	6
1. SQL v prohlížeči a bez instalace	7
2. SQLite — snadný způsob jak začít s SQL	9
3. SQL — BETWEEN	18
4. SQL — CASE	24
5. SQL — LIKE	29
6. Množinové operátory	34
7. Generátor náhodných dat	40
8. Analýza zákazníků	44
9. Analýza cen benzínu (Common Table Expressions a Subqueries)	48
10. Analýza pražských pronájmů Airbnb	53
11. Pět fontů, které se skvěle hodí pro SQL	69
12. SQL nebo Python? Návod, jak si správně vybrat	75
13. V čem psát kód	81
14. Mohlo by vás zajímat	84
Rejstřík	85

Úvod

SQL je standardním jazykem všech datových analytiků. A nejen jich. Data uložená v databázích dnes využívají manažeři, obchodníci, marketingoví specialisté, účetní, personalisté, studenti - vlastně všichni, kteří pracují s daty.



Data jsou dnes cennější než ropa. Stačí se jen podívat na absolventy našich kurzů. Ať už pracují ve velkých korporacích nebo jsou z malých firem, jedno je spojuje: bez dat se v práci prostě neobejdou.

Proto jsem pro tuto knihu dal dohromady kolekci těch nejlepších návodů z Lovely blogu. Některé návody jsem aktualizoval. Také jsem přidal něco navíc. A nezapomněl jsem samozřejmě ani na příklady včetně kódu.

Tato kniha se vám bude hodit, protože úkoly, které v SQL řešíte, už před vámi určitě řešil někdo jiný. Nemusíte tak vynalézat znovu kolo, ale můžete se příklady inspirovat nebo je klidně okopírovat. Ušetříte spoustu času.

Všechny články jsou prakticky zaměřené, abyste je mohli použít při práci. Velkou výhodou je to, že jsou všechny v češtině. A protože je SQL univerzální jazyk, využijete příklady pro většinu databází. Ať už používáte Oracle, MySQL, SQL Server, PostgreSQL, SQLite nebo jiný RDBMS.

Pokud vás baví data a datová analýza, je tato kniha přesně pro vás. Získáte díky ní jistotu při psaní databázových dotazů a budete snadněji analyzovat data s pomocí jazyka SQL.

P.S. A nezapomeňte, že zaměstnavatelé stále hledají nové kolegy, kteří si s SQL rozumí. 😊

Lubomír Husar

zakladatel LovelyData.cz

Ovlivněte obsah této knihy a napište mi náměty na další články a příklady. Nejvíc mě zajímají ty, které denně řešíte v práci. Pokud budete mít ke knize nějaké připomínky, nápady nebo postřehy - budu rád, když mi je napíšete na lubomir@lovelydata.cz.

1. SQL v prohlížeči a bez instalace

1.1. SQL on-line

Možná právě s SQL začínáte a nechce se vám instalovat databázový server lokálně na počítač. Nebo máte firemní laptop a administrátor instalaci zakázal. Nebo jste prostě jen *běžný* uživatel, který se zrovna nevyžívá v instalacích nového softwaru. Ve všech případech se vám určitě budou hodit následující nástroje, které jsou on-line, zdarma a k jejich používání si vystačíte s internetovým prohlížečem.

Všechny tyto nástroje umožňují práci v několika databázích, jako např. PostgreSQL, MySQL, Oracle, SQL Server, SQLite a dalších.

Snadno si tak můžete procvičit SQL bez otravné instalace.

1.2. Odkazy

- [SQL OnLine](#)
- [SQL Fiddle](#)
- [Replit](#) (pouze SQLite)

1.3. Snadné vytváření tabulek

Jazyk SQL není sám o sobě složitý a vytváření tabulek je díky jednoduché syntaxi snadné. Pokud se vám ale nechce psát příkaz `CREATE TABLE` nebo si nejste jisti správnou syntaxí, velmi dobře poslouží jednoduchý nástroj *SQL Table Builder*.

Ten umožňuje *vizuální* tvorbu tabulek a sloupců pouhým kliknutím myši. Můžete také snadno měnit pořadí sloupců a nebo převádět SQL syntaxi mezi několika různými databázemi, které nástroj podporuje. Výsledné SQL si potom snadno zkopírujete pomocí tlačítka **[Copy]**.

Vyzkoušejte nástroj na sqltools.beekeeperstudio.io/build.

1.4. Snadné formátování

Každý kód - a to platí i pro SQL - se více čte, než píše. Je to logické. Mnohem více času budete trávit čtením SQL, které jste vy, nebo někdo jiný, už napsali. Kód budete opravovat a upravovat. Proto je důležité, aby se SQL četlo dobře.

Způsobů, jak kód formátovat, je více. Každý analytik může mít vlastní názor na to, zda-li důsledně psát klíčová slova velkými písmeny, kde začínat nový řádek nebo jestli odsazovat pomocí mezer či tabelátoru. Pokud si nejste jisti, jak správně SQL naformátovat, aby bylo dobře čitelné, vyzkoušejte on-line pomocníka na sqltools.beekeeperstudio.io/format,

2. SQLite — snadný způsob jak začít s SQL

2.1. SQL pro každého

Návod, jak dostat SQL na svoje PC snadno a rychle.

V našich [SQL kurzech](#) standardně používáme PostgreSQL, ale existuje skvělá alternativa, která umožní využívat jazyk SQL každému a navíc snadno a rychle.

Pokud nechcete (nebo nemůžete) na svůj PC instalovat PostgreSQL a související aplikace, dají se kurzy absolvovat i pomocí šikovné aplikace [DB Browser for SQLite](#) (DB4S), která v sobě kombinuje databázi a SQL editor v jednom.

2.2. Ideální pro uživatele Excelu

Má jednoduché rozhraní ve kterém se uživatelé Excelu budou rychle cítit jako doma. Stáhnout a používat ji můžete úplně zdarma. Velkou výhodou je také snadný import textových souborů (ve formátu CSV), což je užitečné, pokud potřebujete nahrávat data z Excelu nebo jiných aplikací.

Mnoho uživatelů také oceňuje fakt, že pracuje na lokálním PC. Není tedy nutné se připojovat na vzdálený server nebo na internet.

2.3. Stáhněte si instalaci

Pokud si chcete aplikaci nainstalovat na svoje PC, vyberte soubor *DB Browser for SQLite - Standard installer for 64-bit Windows* na stránce sqlitebrowser.org/dl/.

POZNÁMKA

Tento návod ukazuje použití aplikace na Windows, ale DB Browser for SQLite funguje i pro operační systémy macOS a Linux.

Windows

Our latest release (3.12.2) for Windows:

- [DB Browser for SQLite - Standard installer for 32-bit Windows](#)
- [DB Browser for SQLite - .zip \(no installer\) for 32-bit Windows](#)
- [DB Browser for SQLite - Standard installer for 64-bit Windows](#)
- [DB Browser for SQLite - .zip \(no installer\) for 64-bit Windows](#)

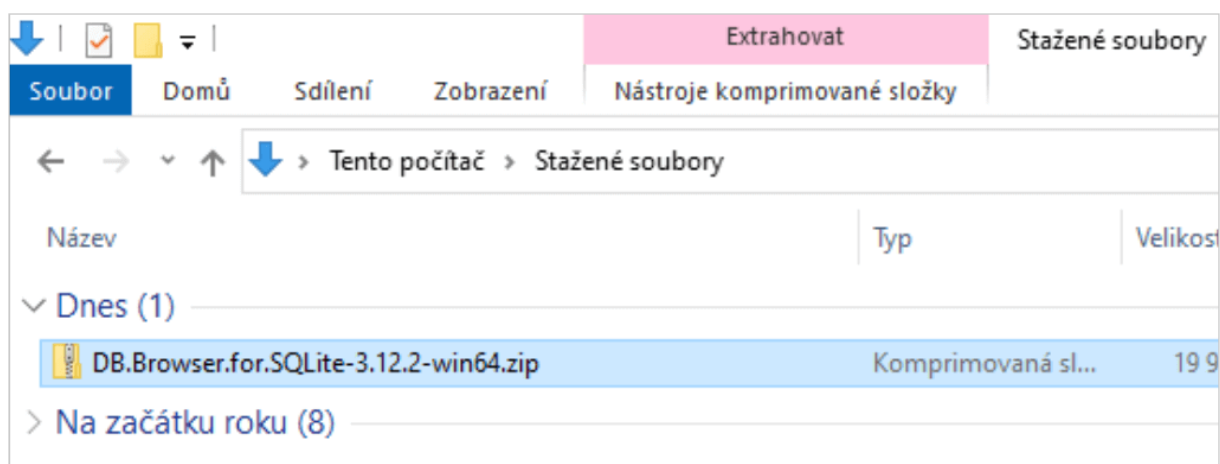
Windows PortableApp

- [DB Browser for SQLite - PortableApp](#)










Stažení aplikace

2.4. Jde to i bez instalace

V případě, že nemáte administrátorská práva (nebo nechcete procházet procesem instalace), můžete si stáhnout soubor *DB Browser for SQLite - .zip (no installer) for 64-bit Windows*, který umožní pracovat s aplikací bez nutnosti instalace.

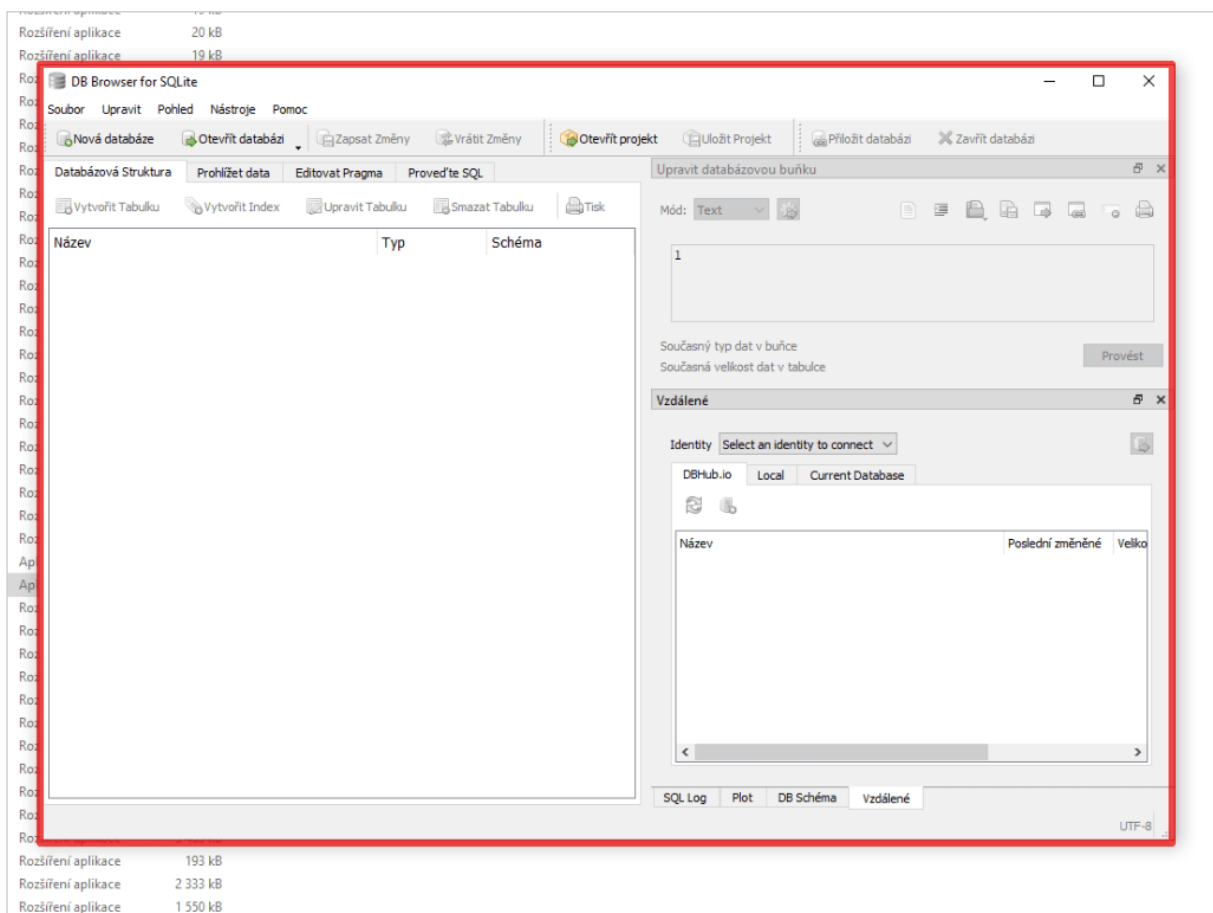


ZIP soubor rozbalte a přejděte do složky, kam jste obsah rozbalili.

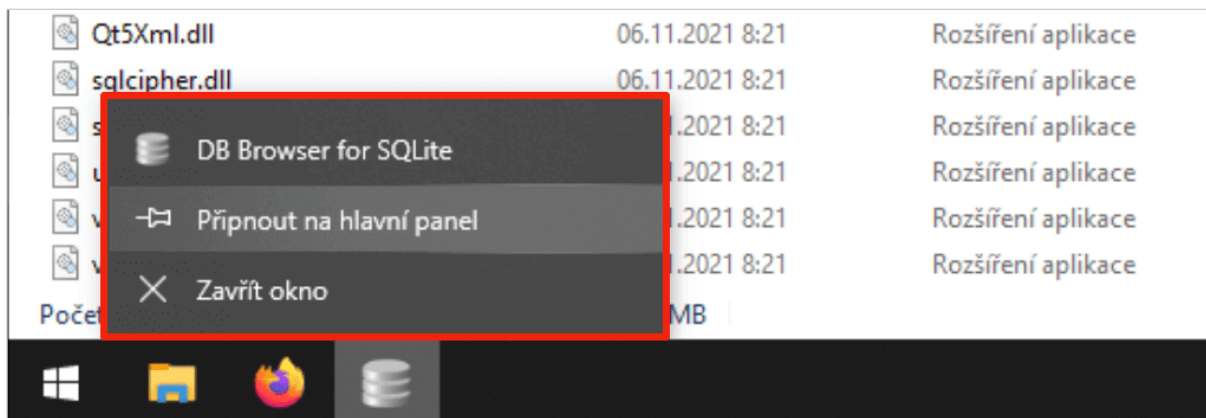
	api-ms-win-crt-utility-l1-1-0.dll	06.11.2021 8:21	Rozšíření aplikace
	concr140.dll	06.11.2021 8:21	Rozšíření aplikace
	DB Browser for SQLite.exe	06.11.2021 8:21	Aplikace
	DB Browser for SQLite.exe	06.11.2021 8:21	Aplikace
	libcrypto-1_1-x64.dll	06.11.2021 8:21	Rozšíření aplikace
	libssl-1_1-x64.dll	06.11.2021 8:21	Rozšíření aplikace
	msvc140.dll	06.11.2021 8:21	Rozšíření aplikace
	msvc140_1.dll	06.11.2021 8:21	Rozšíření aplikace
	msvc140_2.dll	06.11.2021 8:21	Rozšíření aplikace

Popis souboru: DB Browser for SQLite
 Společnost: DB Browser for SQLite Team
 Verze souboru: 3.12.2.0
 Datum vytvoření: 16.05.2021 22:56
 Velikost: 5,31 MB

Aplikaci spustíte pomocí souboru DB Browser fo SQLite.exe.



Hlavní okno aplikace



Pro další spouštění můžete aplikaci připnout na hlavní panel.

2.5. Umí CSV formát

Výhodou aplikace DB4S je, že tabulky můžete snadno vytvářet z obyčejných textových CSV souborů. Tento formát je jedním ze standardů pro přenos dat, který funguje napříč aplikacemi i operačními systémy.

Kromě Excelu umí CSV import a export mnoho dalších programů. A navíc jeho obsah zobrazíte i v obyčejném textovém editoru, protože data jsou uložena v řádcích a jednotlivé sloupce jsou odděleny čárkou. Odtud pochází i význam zkratky CSV = Comma Separated Values.

2.6. Stáhněte si data

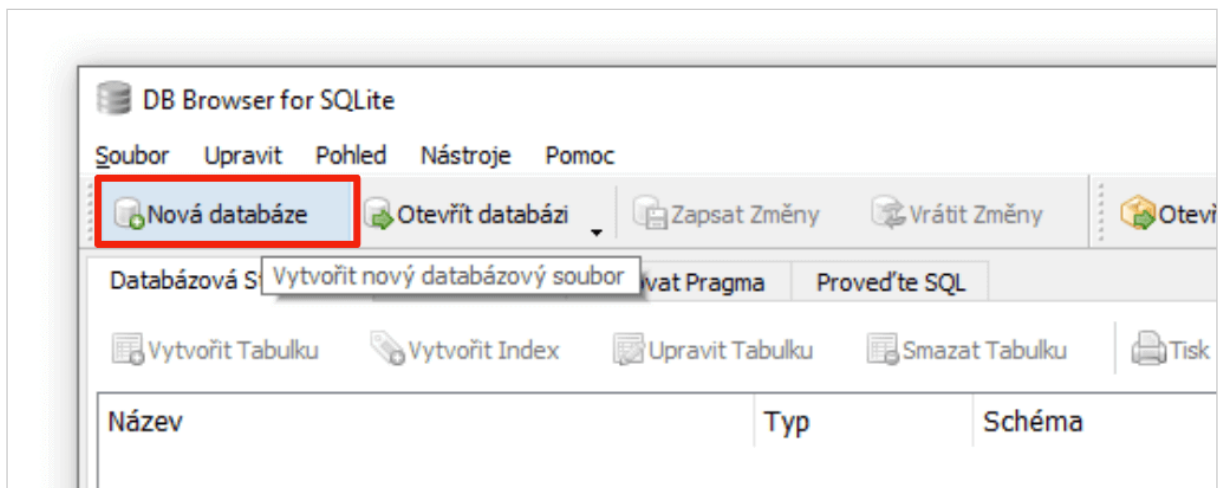
Cvičná data pocházejí ze stránky [Our World in Data](https://ourworldindata.org/), kde najdete spoustu volně dostupných dat a vizualizací. My použijeme data o stále diskutovaném očkování.

Stáhnout lovelydata.cz/media/files/ockovani.csv

2.7. Vytvořte si databázi

Vytvoření databáze je v aplikaci DB4S snadné:

- Klikněte na ikonu **[Nová databáze]**.
- Vyberte adresář, kde databázi vytvoříte (např. *Dokumenty*).
- Zadejte název právě vytvářené databáze (např. *ockovani.db*).
- Klikněte na tlačítko **[Uložit]**.



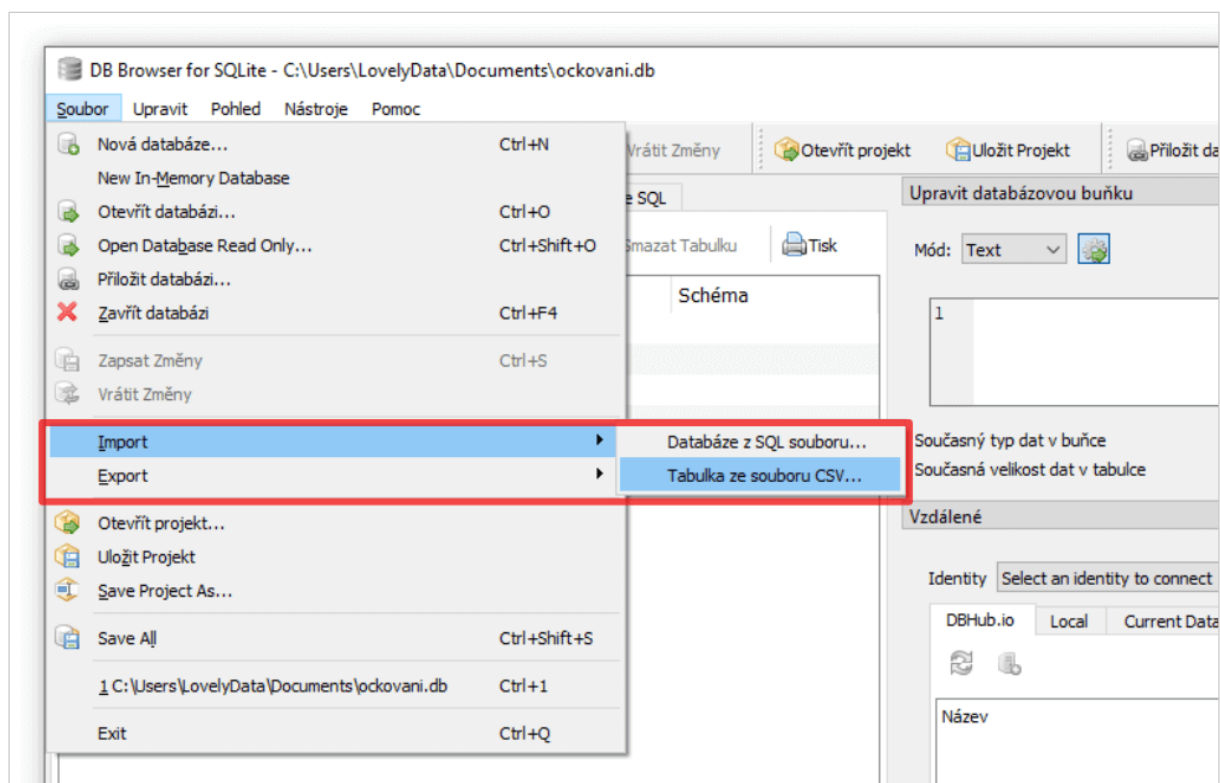
Vytvoření databáze

2.8. Nahrajte data z CSV

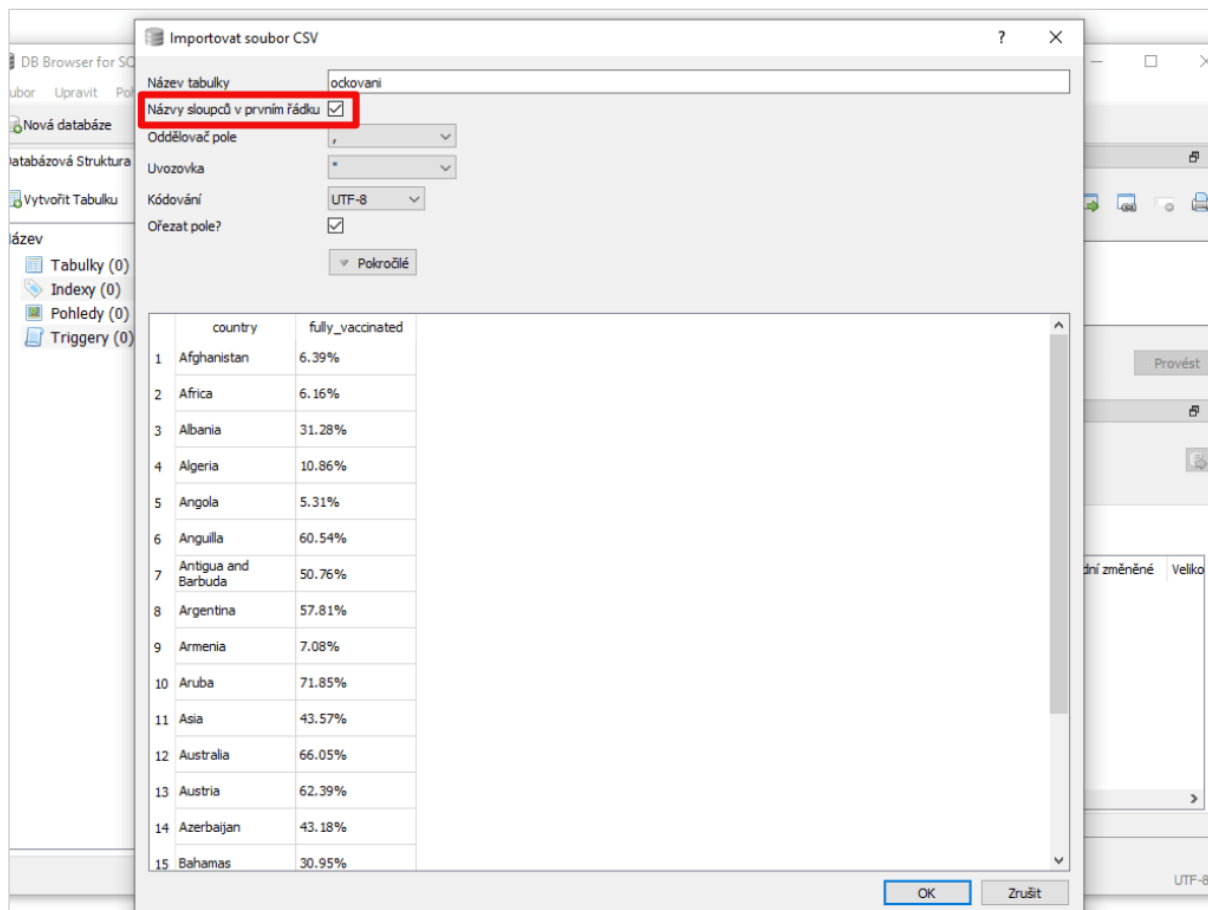
Ihned po uložení databáze se otevře okno pro vytvoření tabulky. Protože data budete nahrávat ze souboru, klikněte nyní na tlačítko **[Zrušit]**.

Tabulku vytvořte následovně:

- Zvolte v menu **Soubor** > **Import** > **Tabulka ze souboru CSV...**
- Vyhledejte soubor *ockovani.csv*, který jste stáhli v předchozím kroku.
- Ujistěte se, že je zaškrtnutá volba **[Názvy sloupců v prvním řádku]**.
- Klikněte na tlačítko **[OK]**.



Import dat z CSV



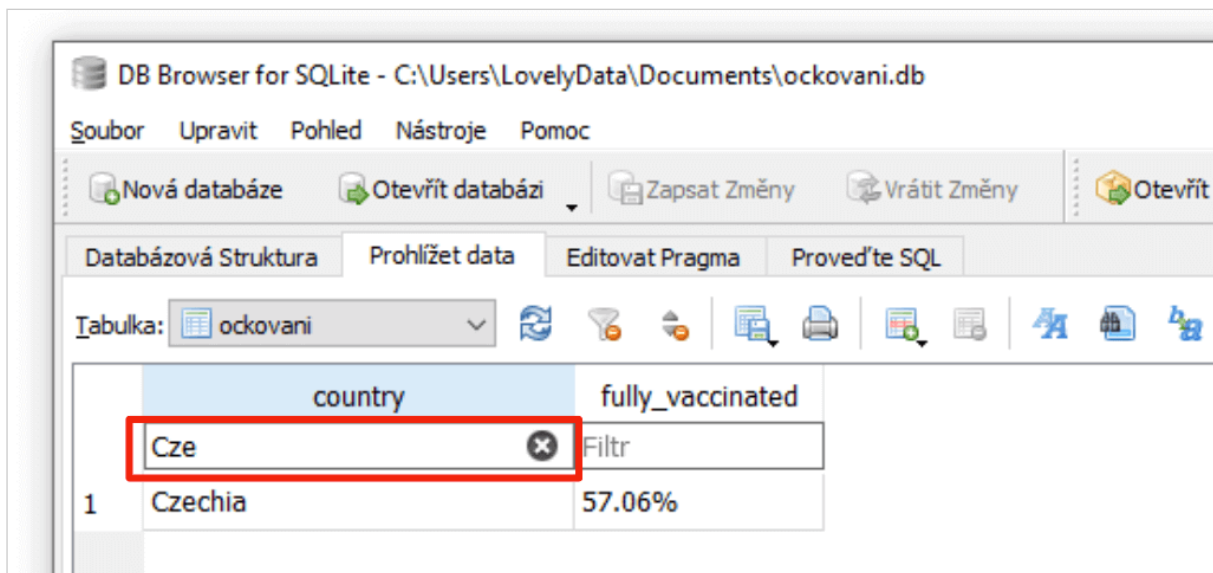
Nastavení importu

2.9. Uložte změny

Nezapomeňte provedené změny v databázi uložit. V menu zvolte možnost **Soubor** > **Zapsat změny** (nebo klávesovou zkratku **Ctrl + S** / **Cmd + S**).

2.10. Rychlé filtrování dat

Pro rychlé hledání a filtrování dat ani nepotřebujete znát jazyk SQL. Stačí přejít na záložku **Prohlížet data** a začít vyhledávat.



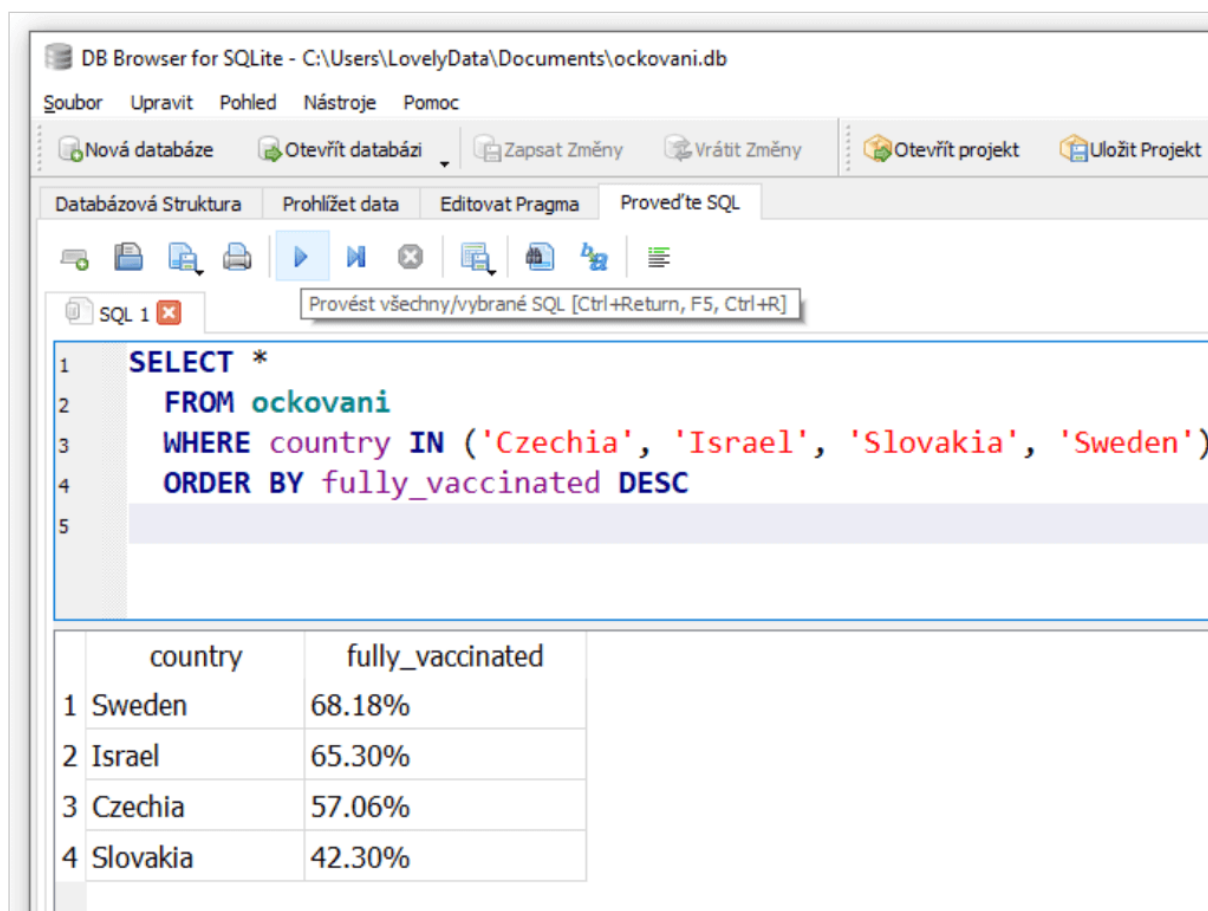
Filtrování dat

2.11. Začněte s SQL

Pokud chcete využít široké možnosti jazyka SQL, můžete rovnou přejít na záložku Provést SQL. Příkazy zadávejte do horního okna. Výsledná data se pak zobrazí v okně pod zadaným příkazem.

```
1 SELECT *  
2 FROM ockovani  
3 WHERE country IN ('Czechia', 'Israel', 'Slovakia', 'Sweden')  
4 ORDER BY fully_vaccinated DESC  
5 ;
```

SQL příkaz spusíte kliknutím na tlačítko **[Provést SQL]** (modrý trojúhelník) nebo využijte jednu z klávesových zkratk **F5**, **Ctrl + ENTER**, **Ctrl + R**, **Cmd + ENTER** nebo **Cmd + R**.



Spuštění SQL příkazu

2.12. Závěrem

Aplikace DB Browser for SQLite toho umí ještě mnohem víc - např. upravovat data, formátovat je (včetně podmíněného formátování), vizualizovat je s pomocí grafů, atd. Pro další informace se můžete podívat na dokumentaci.

Pokud se chcete naučit využívat SQL při práci, doporučujeme absolvování kurzu [SQL pro analytiky](#), který vám dá potřebné základy. Pokud složíte závěrečný test, obdržíte certifikát, kterým se pak můžete pochlubit třeba v práci.

3. SQL — BETWEEN

3.1. Úvod

SQL operátor BETWEEN porovnává, zda-li je výraz v rozmezí hodnot low a high.

POZNÁMKA

Pokud nemáte na svém počítači nainstalovaný žádný SQL server - jako např. PostgreSQL, MySQL, MS SQL Server, apod. - můžete využít např. [SQL Online](#), které je on-line a zdarma.

3.2. Vytvoření tabulky

Pro vytvoření dat použijte vzorek z datové sady *titanic*.

```
1 CREATE TABLE "titanic" (  
2     "pclass" INTEGER,  
3     "class" VARCHAR(10),  
4     "who" VARCHAR(10),  
5     "age" DECIMAL(5,2)  
6 );  
7  
8 INSERT INTO "titanic" VALUES(2,'Second','man',36.5);  
9 INSERT INTO "titanic" VALUES(3,'Third','child',9.0);  
10 INSERT INTO "titanic" VALUES(2,'Second','man',51.0);  
11 INSERT INTO "titanic" VALUES(2,'Second','woman',30.0);  
12 INSERT INTO "titanic" VALUES(1,'First','man',33.0);  
13 INSERT INTO "titanic" VALUES(1,'First','man',58.0);  
14 INSERT INTO "titanic" VALUES(2,'Second','woman',30.0);  
15 INSERT INTO "titanic" VALUES(1,'First','man',22.0);  
16 INSERT INTO "titanic" VALUES(3,'Third','child',11.0);  
17 INSERT INTO "titanic" VALUES(3,'Third','child',0.42);  
18 INSERT INTO "titanic" VALUES(3,'Third','child',1.0);  
19 INSERT INTO "titanic" VALUES(3,'Third','man',24.0);  
20 INSERT INTO "titanic" VALUES(3,'Third','man',NULL);  
21 INSERT INTO "titanic" VALUES(1,'First','man',70.0);  
22 INSERT INTO "titanic" VALUES(3,'Third','man',34.0);
```

```

23 INSERT INTO "titanic" VALUES(1,'First','man',30.0);
24 INSERT INTO "titanic" VALUES(3,'Third','child',11.0);
25 INSERT INTO "titanic" VALUES(2,'Second','man',32.0);
26 INSERT INTO "titanic" VALUES(3,'Third','man',NULL);
27 INSERT INTO "titanic" VALUES(3,'Third','woman',NULL);

```

3.3. Věk je mezi 30 - 34

Hledáme pasažéry, kterým bylo mezi 30 až 34 lety, včetně okrajových hodnot.

```

1 SELECT *
2     FROM titanic
3     WHERE age BETWEEN 30 AND 34
4 ;

```

Výsledek

pclass	class	who	age
2	Second	woman	30
1	First	man	33
2	Second	woman	30
3	Third	man	34
1	First	man	30
2	Second	man	32

3.4. Stejný příklad bez použití BETWEEN

Výše uvedený SQL příkaz by se dal i přepsat následovně. Nevýhodou je trochu horší čitelnost ve srovnání s BETWEEN.

```

1 SELECT *
2     FROM titanic
3     WHERE age >= 30
4           AND age <= 34
5 ;

```

Výsledek

pclass	class	who	age
2	Second	woman	30
1	First	man	33
2	Second	woman	30
3	Third	man	34
1	First	man	30
2	Second	man	32

3.5. Alternativní zápis podmínky

Nebo také takto, ale BETWEEN stále vyhrává, protože je nejčitelnější.

```
1 SELECT *
2     FROM titanic
3     WHERE 30 <= age
4           AND age <= 34
5 ;
```

Výsledek

pclass	class	who	age
2	Second	woman	30
1	First	man	33
2	Second	woman	30
3	Third	man	34
1	First	man	30
2	Second	man	32

3.6. Věk není mezi 30 - 34

Podmínku můžeme snadno negovat pomocí NOT. Všimněte si, že výsledek neobsahuje záznamy, které mají ve sloupci age hodnotu NULL.

```
1 SELECT *
2     FROM titanic
3     WHERE age NOT BETWEEN 30 AND 34
4 ;
```

Výsledek

pclass	class	who	age
2	Second	man	36.5
3	Third	child	9
2	Second	man	51
1	First	man	58
1	First	man	22
3	Third	child	11
3	Third	child	0.42
3	Third	child	1
3	Third	man	24
1	First	man	70
3	Third	child	11

3.7. Ověření vrácených hodnot pomocí DISTINCT

Pomocí DISTINCT můžeme ověřit, že se ve vrácených hodnotách nenachází 30, 31, 32, 33 a 34. Hodnoty si seřadíme pomocí ORDER BY, pro lepší čitelnost.

```
1 SELECT DISTINCT AGE
2     FROM titanic
3     WHERE age NOT BETWEEN 30 AND 34
4     ORDER BY age
5 ;
```

Výsledek

```
age
----
0.42
1
9
11
22
24
36.5
51
58
70
```

3.8. Příklady

3.8.1. Příklad 1

Kolik bylo pasažérům, kteří splňovali následující podmínky:

- Věk mezi 50 - 70 lety
- a cestovali v první třídě?

```
1 SELECT COUNT(*) AS "počet"
2     FROM titanic
3     WHERE age BETWEEN 50 AND 70
4           AND pclass = 1
5 ;
```

Výsledek

```
počet
-----
2
```

3.9. Příklad 2

Titanic se potopil v roce 1912. Kterým pasažérům by v roce 2020 bylo mezi 144,5 - 160 lety?

```
1 SELECT *
2     FROM titanic
3     WHERE age + (2020 - 1912) BETWEEN 144.5 AND 160
4 ;
```

Výsledek

```
pclass  class  who age
-----
2       Second man 36.5
2       Second man 51
```

3.10. Použité funkce a příkazy SQL

V tomto článku jsme použili následující funkce a příkazy SQL:

BETWEEN	Porovnává, zda-li je výraz v rozmezí hodnot — dokumentace SQL
NOT	Negace výrazu — dokumentace SQL .
DISTINCT	Unikátní hodnoty (odstraní duplicity) — dokumentace SQL .
ORDER BY	Seřadí výsledek — dokumentace SQL .

4. SQL — CASE

Potřebujete použít SQL příkaz CASE a nejste si úplně jistí, jak funguje?

Jednoduše se dá vysvětlit jako *IF-THEN-ELSE* pro SQL. Umožňuje vyhodnocovat podmínky a podle výsledku se rozhodnout, co dál.

Základní syntaxe CASE není složitá.

4.1. Vytvořte si tabulku

Zkopírujte následující kód a spusťte ho. Vytvoří se tabulka s testovacími daty.

```
1 CREATE TABLE kurzy
2 (
3     id INT,
4     nazev VARCHAR(50),
5     cena DECIMAL(7,2),
6     v_prodeji BOOLEAN,
7     prodano_licenci INT
8 );
9
10
11 INSERT INTO kurzy VALUES (1, 'SQL základy', 3900, 1, 120);
12 INSERT INTO kurzy VALUES (2, 'Python úvod', 4100, 1, 140);
13 INSERT INTO kurzy VALUES (3, 'Excel KT', 3100, 1, 330);
14 INSERT INTO kurzy VALUES (4, 'Power BI', 3900, 1, 210);
15 INSERT INTO kurzy VALUES (5, 'Tableau', 4300, 1, 190);
16 INSERT INTO kurzy VALUES (6, 'Android', 3100, 1, 40);
17 INSERT INTO kurzy VALUES (7, 'RPA', 3100, 1, 50);
18 INSERT INTO kurzy VALUES (8, 'Strojové učení', 4800, 1, 30);
19 INSERT INTO kurzy VALUES (9, 'Základy jógy', 2900, 0, 0);
```

4.2. Začínáme

Nejlépe si vyzkoušíte použití CASE podmínky na příkladech.

Váš kolega chce vědět, jaké kurzy jsou aktuálně v prodeji. Pokud kurz není v prodeji, označíte ho *Nelze zakoupit*, v opačném případě jako *Lze zakoupit*.

Tento příklad ukazuje nejjednodušší syntaxi CASE.

```
1 SELECT nazev, v_prodeji,  
2     CASE v_prodeji  
3         WHEN 0  
4         THEN 'Nelze zakoupit'  
5         ELSE 'Lze zakoupit'  
6     END AS "V prodeji"  
7 FROM kurzy  
8 ;
```

Výsledek

nazev	v_prodeji	V prodeji
SQL základy	1	Lze zakoupit
Python úvod	1	Lze zakoupit
Excel KT	1	Lze zakoupit
Power BI	1	Lze zakoupit
Tableau	1	Lze zakoupit
Android	1	Lze zakoupit
RPA	1	Lze zakoupit
Strojové učení	1	Lze zakoupit
Základy jógy	0	Nelze zakoupit

4.3. Porovnání ceny

Pokud kurz stojí méně než 3500 Kč, označíte ho jako *Do 3500 Kč*, pokud více, tak jako *Nad 3500 Kč*. Nakonec kurzy seřadíte podle relativní ceny a názvu.

Všimněte si drobné změny v syntaxi, kdy cenu porovnáváte hned za klíčovým slovem WHEN.

```
1 SELECT nazev, cena,  
2     CASE WHEN cena < 3500  
3         THEN 'Do 3500 Kč'  
4         ELSE 'Nad 3500 Kč'  
5     END AS "Relativní cena"
```

```

6 FROM kurzy
7 ORDER BY "Relativní cena",
8         nazev
9 ;

```

Výsledek

nazev	cena	Relativní cena
-----	-----	-----
Android	3100	Do 3500 Kč
Excel KT	3100	Do 3500 Kč
RPA	3100	Do 3500 Kč
Základy jógy	2900	Do 3500 Kč
Power BI	3900	Nad 3500 Kč
Python úvod	4100	Nad 3500 Kč
SQL základy	3900	Nad 3500 Kč
Strojové učení	4800	Nad 3500 Kč
Tableau	4300	Nad 3500 Kč

4.4. Kombinace podmínek

Obě výše uvedené podmínky můžete zkombinovat do jednoho SQL příkazu a získat tak všechny informace najednou. Díky tomu nemusíte psát víc příkazů pod sebe.

```

1 SELECT nazev,
2        CASE v_prodeji
3          WHEN 0
4            THEN 'Nelze zakoupit'
5            ELSE 'Lze zakoupit'
6          END AS "V prodeji",
7        CASE WHEN cena < 3500
8              THEN 'Do 3500 Kč'
9              ELSE 'Nad 3500 Kč'
10         END AS "Relativní cena"
11 FROM kurzy
12 ;

```

nazev	V prodeji	Relativní cena
-----	-----	-----
SQL základy	Lze zakoupit	Nad 3500 Kč
Python úvod	Lze zakoupit	Nad 3500 Kč
Excel KT	Lze zakoupit	Do 3500 Kč
Power BI	Lze zakoupit	Nad 3500 Kč
Tableau	Lze zakoupit	Nad 3500 Kč
Android	Lze zakoupit	Do 3500 Kč
RPA	Lze zakoupit	Do 3500 Kč
Strojové učení	Lze zakoupit	Nad 3500 Kč
Základy jógy	Nelze zakoupit	Do 3500 Kč

4.5. Složitější podmínky

Praxe samozřejmě přináší komplikovanější případy, kdy s jednoduchými podmínkami nevystačíte. Naštěstí je SQL CASE dobře připraveno i na tyto výzvy.

Kolegyně chce přehledně zobrazit prodejnost kurzů. Pokud se jich prodalo více než 200, zobrazíme hvězdičky. Pokud byl prodej více než 100, dostanou palce nahoru. Méně než 100 znamená obličej bez výrazu. Všechno ostatní je propadák.

```

1 SELECT nazev, prodano_licenci,
2     CASE
3         WHEN prodano_licenci > 200 THEN '☆☆☆'
4         WHEN prodano_licenci > 100 THEN '👍👍'
5         WHEN prodano_licenci BETWEEN 1 AND 100 THEN '😐'
6         ELSE '💣'
7     END AS "Prodejnost"
8 FROM kurzy
9 ORDER BY prodano_licenci DESC
10 ;

```

Výsledek

nazev	prodano_licenci	Prodejnost
-----	-----	-----
Excel KT	330	☆☆☆
Power BI	210	☆☆☆
Tableau	190	👍👍
Python úvod	140	👍👍
SQL základy	120	👍👍
RPA	50	😐
Android	40	😐
Strojové učení	30	😐
Základy jógy	0	💣

4.6. Další kroky

Pokud se chcete naučit využívat SQL, doporučujeme absolvování kurzu [SQL pro analytiky](#), který vám dá potřebné základy.

Pokud složíte závěrečný test, obdržíte certifikát, kterým se pak můžete pochlubit třeba v práci.

5. SQL — LIKE

5.1. Úvod

Znáte všechny taje, které SQL operátor LIKE nabízí? Vsadíme se, že o některých možná ani nevíte! 😊

5.2. Vytvořte si tabulku s daty

Zkopírujte následující kód a vytvořte tabulku s několika testovacími řádky.

```
1 CREATE TABLE students
2 (
3     id          INT,
4     first_name  VARCHAR(50),
5     last_name   VARCHAR(50)
6 );
7
8 INSERT INTO students VALUES (1, 'Jana', 'Dvořáková');
9 INSERT INTO students VALUES (2, 'Steve', 'Jobs');
10 INSERT INTO students VALUES (3, 'Jan', 'Novák');
11 INSERT INTO students VALUES (4, 'John', 'Lennon');
12 INSERT INTO students VALUES (5, 'Bill', 'Gates');
13 INSERT INTO students VALUES (6, 'Paul', 'McCartney');
14 INSERT INTO students VALUES (7, 'JAN_ADAM', 'V. ');
15 INSERT INTO students VALUES (8, 'XXX', 'yk%m*6%_a');
```

5.3. Syntaxe

Syntaxe operátoru LIKE není složitá.

%	Procento nahrazuje více znaků nebo také žádný znak.
_	Podtržítko nahrazuje jeden libovolný znak.
\	Zpětné lomítko je tzv. Escape znak. Díky němu můžeme vyhledávat v řetězci znaky % a _.

5.4. Příklady

5.4.1. Křestní jména začínající na "J"

```
1 SELECT * FROM students
2     WHERE first_name LIKE 'J%';
```

Výsledek

id	first_name	last_name
1	Jana	Dvořáková
3	Jan	Novák
4	John	Lennon
7	JAN_ADAM	V.

5.4.2. Příjmení, která obsahují "a"

```
1 SELECT * FROM students
2     WHERE last_name LIKE '%a%';
```

Výsledek

id	first_name	last_name
5	Bill	Gates
6	Paul	McCartney
8	XXX	yk%*6%_a

5.4.3. Příjmení, která neobsahují "a"

```
1 SELECT * FROM students
2     WHERE last_name NOT LIKE '%a%';
```

Výsledek

id	first_name	last_name
1	Jana	Dvořáková
2	Steve	Jobs
3	Jan	Novák
4	John	Lennon
7	JAN_ADAM	V.

5.4.4. Křestní jména, které mají jako 2. písmeno "a"

```
1 SELECT * FROM students
2   WHERE first_name LIKE '_a%';
```

Výsledek

id	first_name	last_name
1	Jana	Dvořáková
3	Jan	Novák
6	Paul	McCartney

5.4.5. Příjmení, která mají délku 5 znaků

```
1 SELECT * FROM students
2 -- Znak podtržítka je napsán 5x
3 WHERE last_name LIKE '_____';
```

Výsledek

id	first_name	last_name
3	Jan	Novák
5	Bill	Gates

5.4.6. Křestní jména nebo příjmení, která obsahují podtržítko

```
1 SELECT * FROM students
2     WHERE first_name LIKE '%\_%'
3     OR last_name LIKE '%\_%';
```

Výsledek

id	first_name	last_name
7	JAN_ADAM	V.
8	XXX	yk%*6%_a

5.4.7. Příjmení, která obsahují procento

```
1 SELECT * FROM students
2     WHERE last_name LIKE '%\%%';
```

Výsledek

id	first_name	last_name
8	XXX	yk%*6%_a

POZNÁMKA

U některých databází, např. (SQL Server) nebo (Oracle), je nutné uvést klíčové slovo (ESCAPE).

```
1 SELECT * FROM students
2     WHERE last_name LIKE '%\%%' ESCAPE '\';
```


INVESTUJTE DO SEBE

KURZY

Python, SQL, Tableau, Strojové učení, Power BI, Excel, R, RPA, Android, Projektový management, Kariéra v datech

Získejte nové dovednosti v kurzech, kterým věří největší společnosti a státní instituce.

UČÍ SE
S NÁMI

Air Bank, Akademie múzických umění v Praze, Atlas Copco Services, Banka CREDITAS, Bosch, Broker Consulting, Centrální depozitář cenných papírů, ČEPS, Česká pojišťovna, Česká spořitelna, Českomoravská stavební spořitelna, Československá obchodní banka, Český hydrometeorologický ústav, ČEZ Distribuce, Dopravní podnik hl. m. Prahy, Dr. Max, Ecco Slovakia, Emco, Fakultní nemocnice Královské Vinohrady, GUMOTEX, Hypoteční banka, innogy, Invia.cz, J&T Services, Komerční pojišťovna, KPMG Česká republika, Lindab, Masarykova univerzita, Ministerstvo vnitra ČR, MND, Národní knihovna České republiky, Nejvyšší kontrolní úřad, O2, Porsche Engineering, Pražská energetika, RAIFFEISENBANK, Sharp Business Systems, Siemens, Sony DADC Czech Republic, Státní pokladna, Statutární město Jihlava, T-Mobile Czech Republic, UniCredit Bank, Univerzita Karlova, Fakulta sociálních věd, Univerzita Hradec Králové, Ústav zemědělské ekonomiky a informací, Vysoká škola ekonomická v Praze, Výzkumný ústav meliorací a ochrany půdy, Wunderman, YVES ROCHER, Zeměměřický úřad a mnoho dalších

6. Množinové operátory

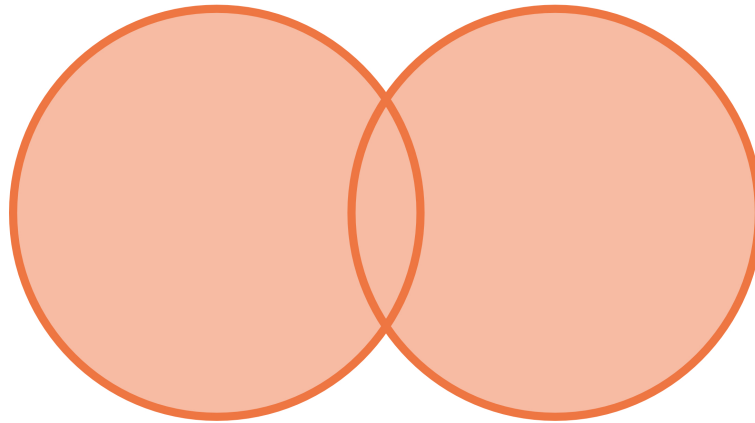
Operace s množinami (angl. SET operations) se používají ke spojení dvou nebo více dotazů do jednoho výsledku. Kombinované dotazy musí vracet **stejný počet sloupců**, které musí mít **kompatibilní datové typy**. Názvy sloupců stejné být nemusí.

6.1. Vytvoření tabulky

Použijte následující SQL a vytvořte dvě jednoduché tabulky na kterých si množinové operace vyzkoušíte. Všimněte si, že Maďarsko se vyskytuje v obou tabulkách.

```
1 CREATE TABLE country_1
2 (
3     name VARCHAR(20),
4     code CHAR(3)
5 );
6
7 CREATE TABLE country_2
8 (
9     name VARCHAR(20),
10    code CHAR(3)
11 );
12
13 INSERT INTO country_1 VALUES ('Bulgaria', 'BGN');
14 INSERT INTO country_1 VALUES ('Croatia', 'HRK');
15 INSERT INTO country_1 VALUES ('Czechia', 'CZK');
16 INSERT INTO country_1 VALUES ('Denmark', 'DKK');
17 INSERT INTO country_1 VALUES ('Hungary', 'HUF');
18
19 INSERT INTO country_2 VALUES ('Hungary', 'HUF');
20 INSERT INTO country_2 VALUES ('Poland', 'PLN');
21 INSERT INTO country_2 VALUES ('Romania', 'RON');
22 INSERT INTO country_2 VALUES ('Sweden', 'SEK');
```

6.2. UNION



Příkaz UNION (sjednocení) se používá ke spojení výsledků dvou nebo více příkazů SELECT. Z výsledkové sady však odstraní duplicitní řádky. V případě sjednocení musí být počet sloupců a datový typ v obou tabulkách, na které se operace UNION aplikuje, stejný.

Protože máme v obou tabulkách stejné sloupce, můžeme použít oblíbenou *. Jinak bychom museli jednotlivé sloupce vyjmenovat.

```
1 SELECT * FROM country_1
2 UNION
3 SELECT * FROM country_2
4 ;
```

Výsledek

name	code
Bulgaria	BGN
Croatia	HRK
Czechia	CZK
Denmark	DKK
Hungary	HUF
Poland	PLN

Romania	RON
Sweden	SEK

6.3. UNION ALL

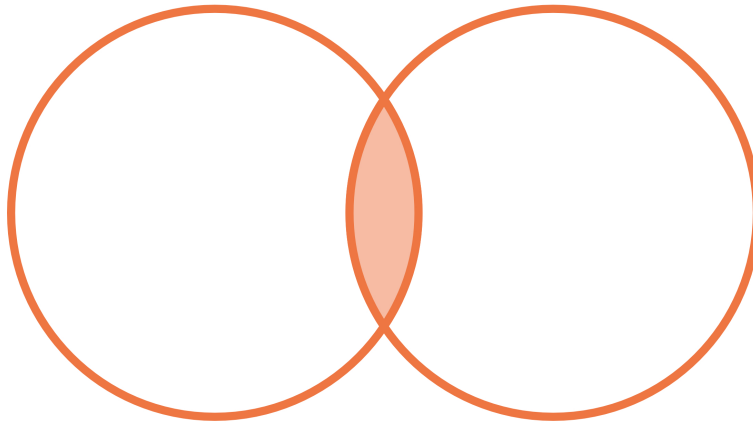
Tato operace funguje stejně jako UNION. S jediným rozdílem - UNION ALL zobrazí také duplicitní řádky.

```
1 SELECT * FROM country_1
2 UNION ALL
3 SELECT * FROM country_2
4 ;
```

Výsledek

name	code
-----	-----
Bulgaria	BGN
Croatia	HRK
Czechia	CZK
Denmark	DKK
Hungary	HUF
Hungary	HUF
Poland	PLN
Romania	RON
Sweden	SEK

6.4. INTERSECT



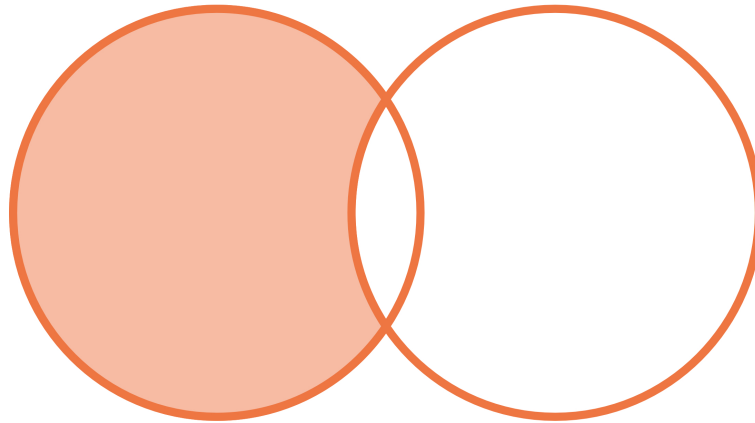
Operace INTERSECT (průnik) vrátí pouze záznamy, které jsou společné pro oba příkazy SELECT. V obou tabulkách se opakuje jen Maďarsko a proto bude výsledkem jen jediný záznam.

```
1 SELECT * FROM country_1
2 INTERSECT
3 SELECT * FROM country_2
4 ;
```

Výsledek

name	code
-----	-----
Hungary	HUF

6.5. EXCEPT/MINUS



Operace EXCEPT/MINUS sloučí výsledky dvou dotazů a vrátí pouze ty, které se vyskytují v prvním dotazu.

```
1 SELECT * FROM country_1
2 EXCEPT
3 SELECT * FROM country_2
4 ;
```

Výsledek

name	code
-----	-----
Bulgaria	BGN
Croatia	HRK
Czechia	CZK
Denmark	DKK

Pokud pořadí dotazů *obrátime*, dostaneme pochopitelně jiný výsledek. Je to tím, že při operaci EXCEPT/MINUS **záleží na pořadí**, na rozdíl od ostatních množinových operací. V obou případech ale ve výsledku bude chybět Maďarsko.

```
1 SELECT * FROM country_2
2 EXCEPT
3 SELECT * FROM country_1
4 ;
```

Výsledek

name	code
Poland	PLN
Romania	RON
Sweden	SEK

POZNÁMKA

Zde se projevuje mírný rozdíl mezi jednotlivými SQL *dialekty*. Většina RDBMS používá klíčové slovo EXCEPT, u Oracle DB se můžete setkat s MINUS, které ale funguje stejně.

6.6. Dokumentace

Detailní informace najdete v dokumentacích pro běžně používané RDBMS:

- [SQLite](#)
- [PostgreSQL](#)
- [SQL Server - EXCEPT a INTERSECT](#)
- [SQL Server - UNION](#)
- [Oracle](#)

7. Generátor náhodných dat

7.1. Úvod

Vyzkoušejte si **SQL hacky**, díky kterým můžete snadno generovat testovací data.

Následující SQL dotaz vrátí 100 řádků. Každý řádek bude mít 3 sloupce - číslo (integer, tedy celé číslo), řetězec (libovolné znaky velkými písmeny nebo číslice) a datum (náhodné datum v minulosti, bez času).

Ukážeme si, jak taková data vygenerovat v různých databázích. Konkrétně PostgreSQL, MySQL, Microsoft SQL Server a SQLite.

V PostgreSQL využijeme šikovnou funkci `generate_series`, díky které je generátor triviální záležitostí na pár řádků. Pro ostatní databáze použijeme *rekurzivní* Common Table Expressions (CTE).

7.2. PostgreSQL

```
1 SELECT
2   CAST(RANDOM()*1000+1 AS INT) AS "číslo",
3   UPPER(LEFT(MD5(RANDOM()::text),10)) AS "řetězec",
4   CURRENT_DATE - CAST(RANDOM()*1000+1 AS INT) AS "datum"
5 FROM generate_series(1,100);
```

Výsledek

číslo	řetězec	datum
911	3F1B6A3F84	2019-03-16
939	8DBDED88C7	2020-10-05
683	FE48DE864E	2018-03-16
---	-----	-----

7.3. MySQL

```
1 WITH RECURSIVE generate_series(value) AS (  
2   SELECT 1  
3   UNION ALL  
4   SELECT value + 1 FROM generate_series  
5     WHERE value + 1 <= 100  
6 )  
7 SELECT  
8   FLOOR(RAND()*1000+1) AS "číslo",  
9   UPPER(LEFT(REPLACE(UUID(), '-', ''), 10)) AS "řetězec",  
10  DATE_ADD(CURRENT_DATE, INTERVAL FLOOR(RAND()*-1000+1) DAY )  
    AS "datum"  
11 FROM generate_series;
```

Výsledek

číslo	řetězec	datum

376	EFD3AE5B1E	2019-02-01
85	EFD3AEAC1E	2019-06-29
221	EFD3AEB91E	2019-02-27
---	-----	-----

7.4. SQL Server

```
1 WITH generate_series(value) AS (  
2   SELECT 1  
3   UNION ALL  
4   SELECT value + 1 FROM generate_series  
5     WHERE value + 1 <= 100  
6 )  
7 SELECT  
8   CAST(CRYPT_GEN_RANDOM(1) As INT) AS "číslo",  
9   LEFT(REPLACE(NEWID(), '-', ''), 10) AS "řetězec",  
10  DATEADD(day, CAST(CRYPT_GEN_RANDOM(1) As INT)*-1, CONVERT  
    (DATE, CURRENT_TIMESTAMP)) AS "datum"
```

```
11 FROM generate_series;
```

Výsledek

číslo	řetězec	datum
182	5881FAF0FE	2020-09-23
113	F4034A7305	2020-06-21
179	723F9BAADD	2020-07-27
---	-----	-----

7.5. SQLite

```
1 WITH RECURSIVE generate_series(value) AS (  
2   SELECT 1  
3   UNION ALL  
4   SELECT value + 1 FROM generate_series  
5     WHERE value + 1 <= 100  
6 )  
7 SELECT  
8   ABS(RANDOM()%1000) AS "číslo",  
9   HEX(RANDOMBLOB(5)) AS "řetězec",  
10  DATE(DATE(), '- ' || ABS(RANDOM()%1000) || ' day') AS "datum"  
11 FROM generate_series;
```

Výsledek

číslo	řetězec	datum
573	D59E7BA651	2019-08-19
81	298EAEAE83	2018-07-16
215	1D585B476F	2020-04-22
---	-----	-----

7.6. Závěr

Současný SQL standard a moderní RDBMS dokazují, že SQL je programovacím jazykem. I když si stále zachovalo svoji přístupnost pro začátečníky, disponuje mocnými nástroji, které snesou srovnání s jinými programovacími jazyky.

8. Analýza zákazníků

8.1. Úvod

Na jednoduchých příkladech si ukážeme, že analýza dat pomocí SQL je rychlá a snadná. Ukážeme si využití agregačních funkcí MIN, MAX, LIMIT/TOP a ROW_NUMBER. Další výhodou těchto příkladů je, že se podobná zadání objevují u pohovorů. 😊

Pokud nemáte databázový server nainstalovaný lokálně, můžete využít SQL v cloudu. Např. [sqliteonline.com/sqliteonline.com](https://sqliteonline.com/) nebo [sqlfiddle.com/sqlfiddle.com](https://sqlfiddle.com/).

8.2. Vytvoření databáze

Použijte následující SQL pro vytvoření databáze spolu s několika řádky.

```
1 CREATE TABLE prodej
2 (
3     id INT,
4     zakaznik VARCHAR(100),
5     celkem DECIMAL(8,2)
6 );
7
8 INSERT INTO prodej VALUES (1, 'Dovoz potravin, s.r.o.',
9                             150000);
9 INSERT INTO prodej VALUES (2, 'Prodej textilu, a.s.', 20000);
10 INSERT INTO prodej VALUES (3, 'Zážitková agentura, s.r.o.',
11                              350000);
11 INSERT INTO prodej VALUES (4, 'Restaurace a stravování, spol s
12     r.o.', 1000);
12 INSERT INTO prodej VALUES (5, 'Průmysl, a.s.', 900000);
```

8.3. Zobrazte zákazníka s nejvyšším obratem

Použijeme poddotaz (subquery) a funkci MAX, která vrátí nejvyšší hodnotu.

```
1 SELECT *
2   FROM prodej
3  WHERE celkem = (
4      SELECT MAX(celkem)
5      FROM prodej
6  );
```

Výsledek

id	zakaznik	celkem
5	Průmysl, a.s.	900000.00

8.4. Zobrazte zákazníka s nejnižším obratem

Budeme postupovat stejně, jako v předchozím příkladě, jen použijeme funkci MIN, která vrátí nejnižší hodnotu.

```
1 SELECT *
2   FROM prodej
3  WHERE celkem = (
4      SELECT MIN(celkem)
5      FROM prodej
6  );
```

Výsledek

id	zakaznik	celkem
4	Restaurace a stravování, spol s r.o.	1000.00

8.5. Zobrazte první tři zákazníky podle obratu

Seřadíme si data pomocí ORDER BY a pak použijeme funkci LIMIT/TOP.

```
1 -- PostgreSQL
2 SELECT *
3   FROM prodej
4   ORDER BY celkem DESC
5   LIMIT 3;
6
7 -- SQL Server
8 SELECT TOP 3 *
9   FROM prodej
10  ORDER BY celkem DESC
```

Výsledek

id	zakaznik	celkem
5	Průmysl, a.s.	900000.00
3	Zážitková agentura, s.r.o.	350000.00
1	Dovoz potravin, s.r.o.	150000.00

8.6. Vyberte zákazníka, který má druhý nejvyšší obrat

Zde využijeme CTE (Common Table Expressions) a funkci ROW_NUMBER, která nám očísluje vrácené řádky.

Výhodou tohoto postupu je, že snadno můžeme vybírat zákazníky podle obratu pouhou změnou čísla.

```
1 WITH zakaznici AS (
2   SELECT ROW_NUMBER() OVER (ORDER BY celkem DESC) AS rn, *
3   FROM prodej
4 )
5 SELECT * FROM zakaznici
```

```
6 WHERE rn = 2; -- Druhý nejvyšší obrat
```

Výsledek

rn	id	zakaznik	celkem
2	3	Zážitková agentura, s.r.o.	350000.00

8.7. Závěr

Syntaxe SQL agregačních funkcí není složitá. Napsání dotazu, který agregační funkce používá, je často rychlejší než *naklikání* kontingenční tabulky v Excelu. A navíc, v databázi můžete agregovat klidně stovky milionů řádků. To v tabulkovém procesoru půjde jen těžko. 😞

9. Analýza cen benzínu (Common Table Expressions a Subqueries)

9.1. Úvod

Po úspěšně zvládnutých základech SQL se před vámi otevřou složitější - a pro někoho zajímavější - témata tohoto jazyka. Mezi tato témata patří dva pojmy, které spolu úzce souvisí:

- CTE (Common Table Expressions)
- Subqueries (Poddotazy)

V mnoha případech si můžete vybrat. Stejného výsledku často dosáhnete jak s CTE, tak i s poddotazy. Výběr metody je proto spíš na vás, než na nějakých *best practices*.

9.2. Common Table Expressions

CTE je novější funkcionalita jazyka SQL než starší poddotazy. CTE je vlastně poddotaz, který ale definujete *na začátku SQL* pomocí klauzule `WITH`.

Mnoho uživatelů tvrdí, že je taková syntaxe čitelnější. To je věc názoru a záleží na tom, jak složité SQL musíte napsat. U kratších a jednodušších dotazů ten rozdíl moc nepoznáte. U těch složitějších, ale bude dotaz vytvořený pomocí CTE přehlednější.

CTE také umí něco, co žádný poddotaz nesvede. A to, že dokáže odkazovat na sebe sama, neboli je rekurzivní. To znamená, že pokud rekurzivní CTE napíšete špatně, můžete způsobit nekonečnou smyčku. A databázový administrátor nebude mít radost. 😞

9.3. Subquery

Poddotaz se někdy též označuje jako *Nested query* (vnořený dotaz) nebo *Subselect*. Jedná se vlastně o dotaz vnořený v dalším dotazu. Počet úrovní vnoření je v RDBMS limitován. SQL Server má například limit 32 úrovní.

9.4. Vytvoření tabulky

Protože jeden příklad vydá za tisíc slov, vyzkoušíte si obě metody s využitím stejných dat.

Použijte následující SQL a vytvořte tabulky spolu s daty.

```
1 CREATE TABLE gasoline
2 (
3     country VARCHAR(20),
4     price_eur DECIMAL(3,2)
5 );
6
7 INSERT INTO gasoline VALUES ('Austria', 1.96);
8 INSERT INTO gasoline VALUES ('Belgium', 1.85);
9 INSERT INTO gasoline VALUES ('Bulgaria', 1.64);
10 INSERT INTO gasoline VALUES ('Croatia', 1.64);
11 INSERT INTO gasoline VALUES ('Czech Republic', 1.82);
12 INSERT INTO gasoline VALUES ('Denmark', 2.22);
13 INSERT INTO gasoline VALUES ('Finland', 2.19);
14 INSERT INTO gasoline VALUES ('France', 1.84);
15 INSERT INTO gasoline VALUES ('Germany', 1.81);
16 INSERT INTO gasoline VALUES ('Greece', 2.15);
17 INSERT INTO gasoline VALUES ('Hungary', 1.29);
18 INSERT INTO gasoline VALUES ('Italy', 1.88);
19 INSERT INTO gasoline VALUES ('Netherlands', 2.13);
20 INSERT INTO gasoline VALUES ('Poland', 1.53);
21 INSERT INTO gasoline VALUES ('Romania', 1.68);
22 INSERT INTO gasoline VALUES ('Slovakia', 1.83);
23 INSERT INTO gasoline VALUES ('Spain', 1.90);
```

9.5. Průměrné ceny

Jaká je průměrná cena benzínu?

```
1 SELECT ROUND(AVG(price_eur),2) AS "Průměrná cena"
2 FROM gasoline
3 ;
```

Výsledek

Průměrná cena

1.84

9.6. Subquery - vyšší než průměr

S využitím předchozího dotazu snadno zjistíte v kterých zemích zaplatíte za benzín víc než průměrnou cenu. Všimněte si, že poddotaz nepojmenováváte. Na rozdíl od Common Table Expressions, kde je pojmenování vyžadováno.

```
1 SELECT *
2 FROM gasoline
3 WHERE price_eur >
4     (SELECT ROUND(AVG(price_eur),2) AS "Průměrná cena"
5      FROM gasoline)
6 ;
```

Výsledek

country	price_eur
-----	-----
Austria	1.96
Belgium	1.85
Denmark	2.22
Finland	2.19
Greece	2.15
Italy	1.88
Netherlands	2.13
Spain	1.9

9.7. CTE - nižší než průměr

V kterých zemích se dá koupit benzín za nižší než průměrnou cenu?

```
1 WITH avg_price AS (
```

```

2     SELECT ROUND(AVG(price_eur),2) AS avg_price_rounded
3     FROM gasoline
4 )
5 SELECT *
6 FROM gasoline g CROSS JOIN avg_price ap
7 WHERE g.price_eur < ap.avg_price_rounded
8 ;

```

Výsledek

country	price_eur	avg_price_rounded
-----	-----	-----
Bulgaria	1.64	1.84
Croatia	1.64	1.84
Czech Republic	1.82	1.84
Germany	1.81	1.84
Hungary	1.29	1.84
Poland	1.53	1.84
Romania	1.68	1.84
Slovakia	1.83	1.84

9.8. Kde platí nejméně a nejvíce

Common Table Expressions a poddotazy můžete klidně kombinovat. Například, když se chcete dozvědět v jaké zemi platí za benzín nejmíň a v jaké naopak nejvíc.

```

1 WITH min_max AS (
2     SELECT MIN(price_eur) AS price_eur FROM gasoline
3     UNION
4     SELECT MAX(price_eur) FROM gasoline
5 )
6 SELECT *
7 FROM gasoline g
8 WHERE EXISTS (
9     SELECT *
10    FROM min_max
11   WHERE price_eur = g.price_eur

```

```
12 )  
13 ORDER BY g.price_eur  
14 ;
```

Výsledek

country	price_eur
-----	-----
Hungary	1.29
Denmark	2.22

POZNÁMKA

Common Table Expressions i poddotazy lze využít nejen na dotazování (SELECT), ale i pro úpravu dat - INSERT, UPDATE, DELETE.

10. Analýza pražských pronájmů Airbnb

10.1. Úvod

Dnešní doba má pro datové analytiky spoustu výhod. Jednou z nich je i to, že velké firmy běžně sdílejí svoje data s veřejností. Nesdílejí samozřejmě úplně vše, ale to, co dávají k dispozici, bývá zajímavým zdrojem informací.

10.2. Data z Airbnb

Airbnb poskytuje několik datových sad, které si [můžete stáhnout](#) zcela volně a bez registrace.

Data, která pro naši dnešní datovou analýzu použijeme, jsou z března 2022. Airbnb nabízí i několik starších, archivních datových sad. To pro případ, že bychom potřebovali srovnat, jak se situace vyvíjela v průběhu času (např. během epidemie, po různých regulacích, apod.).

Pro tuto analýzu datová analýza použijeme soubor data.insideairbnb.com/czech-republic/prague/prague/2022-03-21/visualisations/listings.csv a uložíme ho pod názvem *airbnb-prague-listings.csv*.

Tato sada obsahuje souhrnné informace a metriky pro pronájmy a ubytování v Praze, které jsou dle autorů vhodné pro vizualizace.

10.3. Struktura dat

Soubor (tabulka) obsahuje následující sloupce:

- id
- name
- host_id
- host_name
- neighbourhood_group
- neighbourhood
- latitude
- longitude

- room_type
- price
- minimum_nights
- number_of_reviews
- last_review
- reviews_per_month
- calculated_host_listings_count
- availability_365
- number_of_reviews_ltm
- license

I když jsou názvy většiny sloupců popisné a zřejmé, je vždy dobré mít po ruce [Data Dictionary](#), které nám význam jednotlivých položek vysvětlí.

10.4. Lokální datová analytika

Další z výhod, které si dnes datoví analytici užívají, je výběr z obrovského množství nástrojů. Mnoho z nich je zdarma a často fungují na lokálním PC.

POZNÁMKA

Věděli jste, že databáze [SQLite](#), kterou pro tuto analýzu používáme má limit velikosti databáze neuvěřitelných 281 terabyte!

Kdo by si ještě před pár lety představil, že bude bez problémů analyzovat na svém PC desítky nebo i stovky milionů řádků?

10.5. DB Browser for SQLite

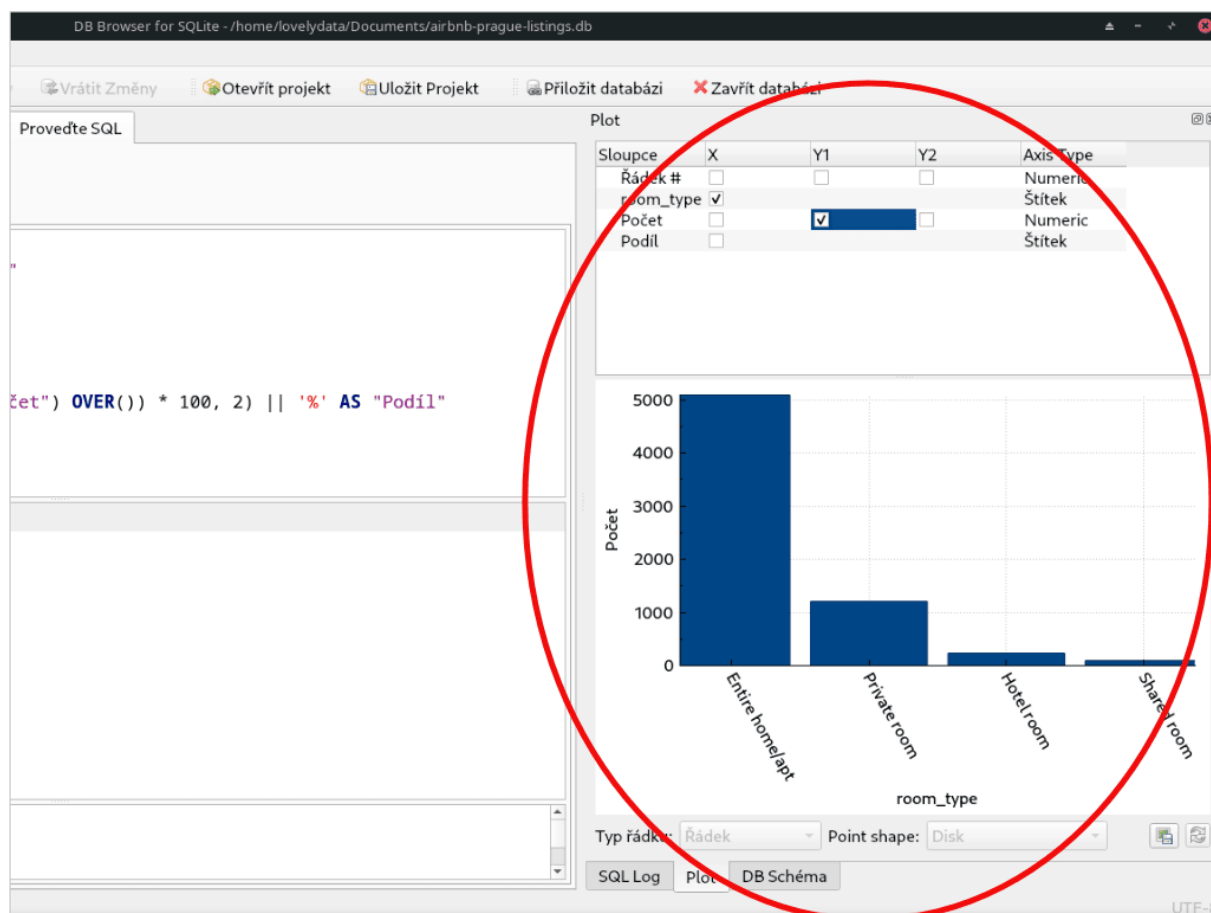
Než vůbec začneme data analyzovat, převedeme zdrojový soubor CSV do SQL tabulky. SQL je ideální jazyk, který nám pomůže snadno a rychle zkoumat data.

Jako SQL databázi využijeme SQLite, která je součástí nástroje s mírně krkolomným názvem DB Browser for SQLite. Ten umí spoustu věcí, je zdarma

a návod na jeho instalaci jsme si už popsali.

10.6. Rychlé grafy

Skvělou funkcionalitou DB Browseru je možnost jednoduchého vytváření vizualizací. Stačí pár kliknutí a zobrazí se graf, který je propojen s daty v tabulce. Barvy můžete měnit a graf také snadno uložíte do souboru.



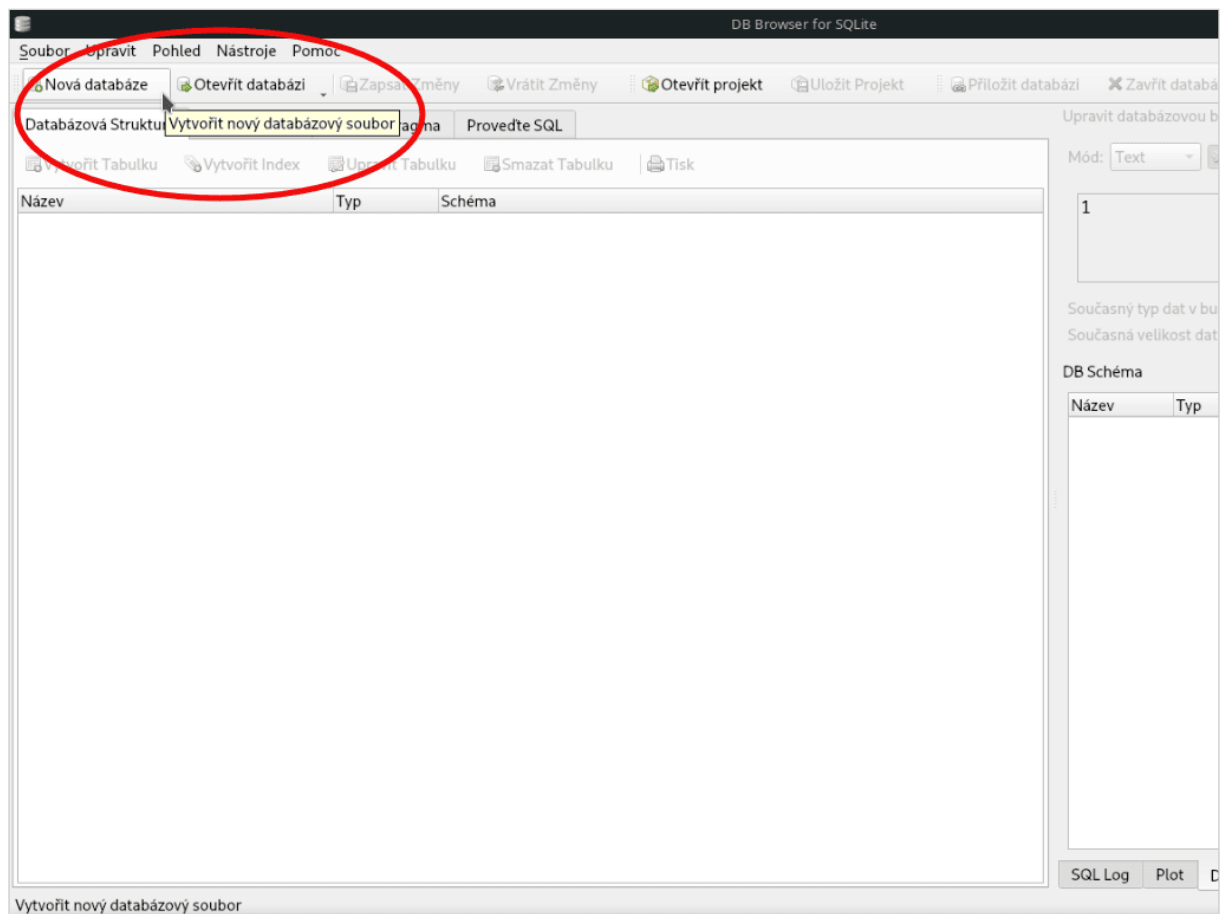
Grafy

Pokud se vám možnost Plot nezobrazuje, zapněte jí v menu **Pohled > Plot**.

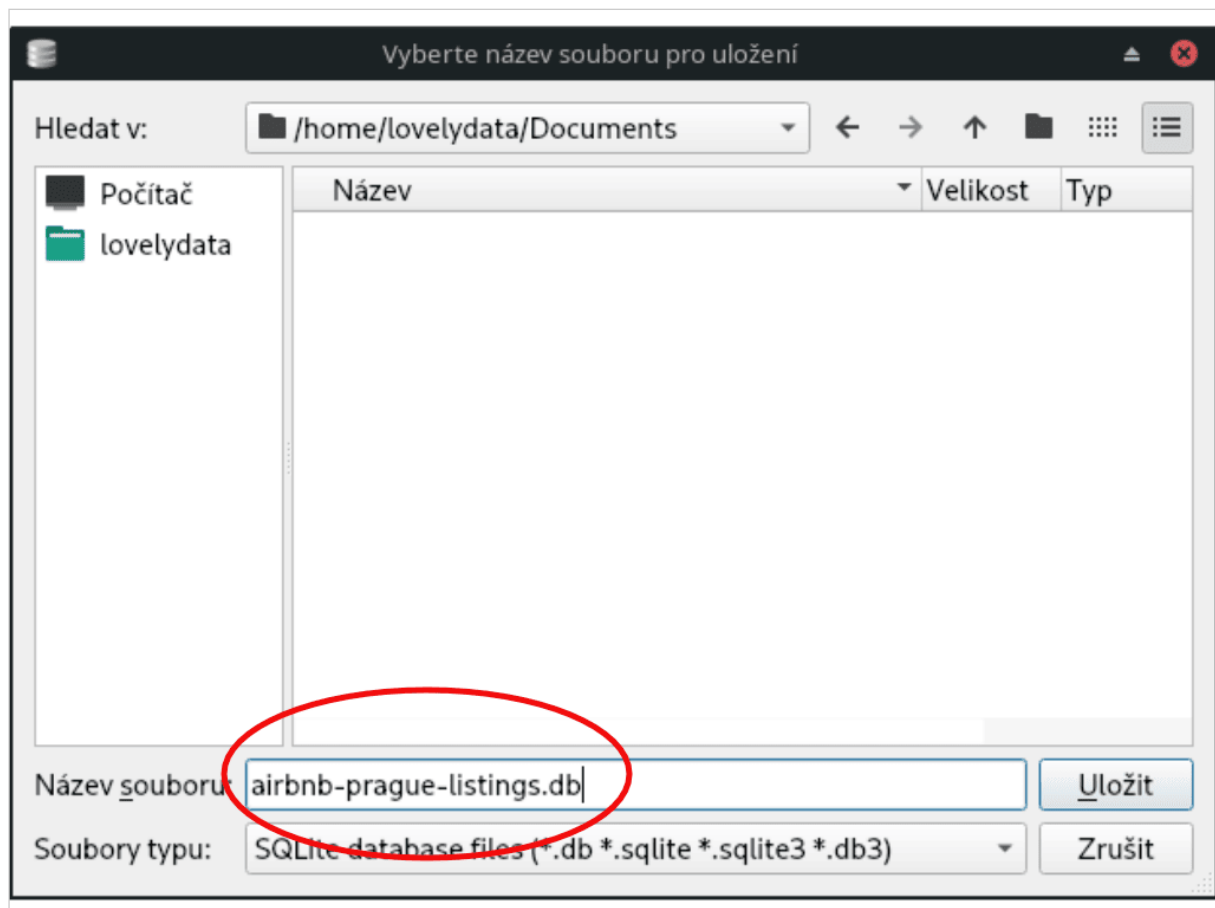
10.7. Vytvoření nové databáze

Nejdřív si vytvoříme novou, prázdnou databázi.

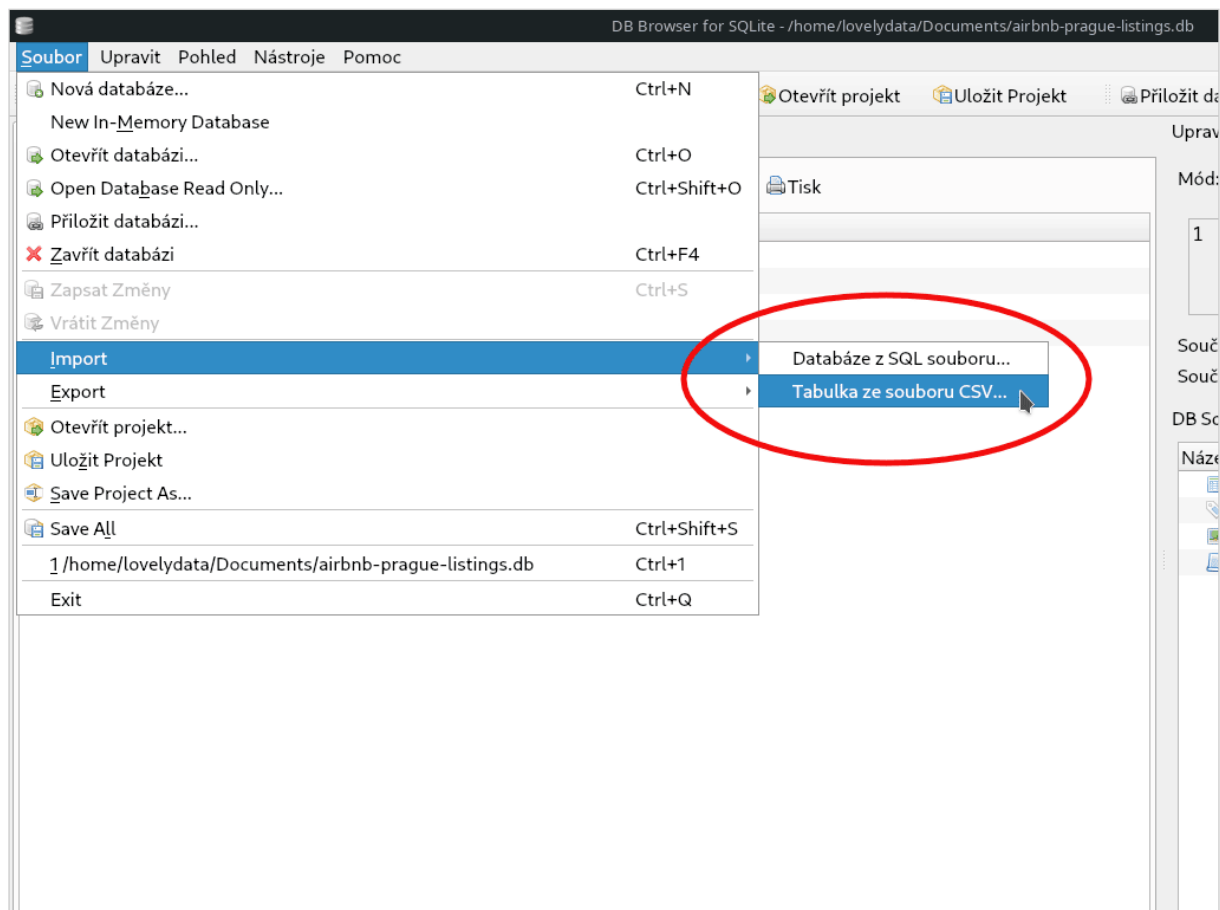
Do ní naimportujeme CSV soubor *airbnb-prague-listings.csv*, který automaticky vytvoří novou tabulku s názvem *airbnb-prague-listings*.



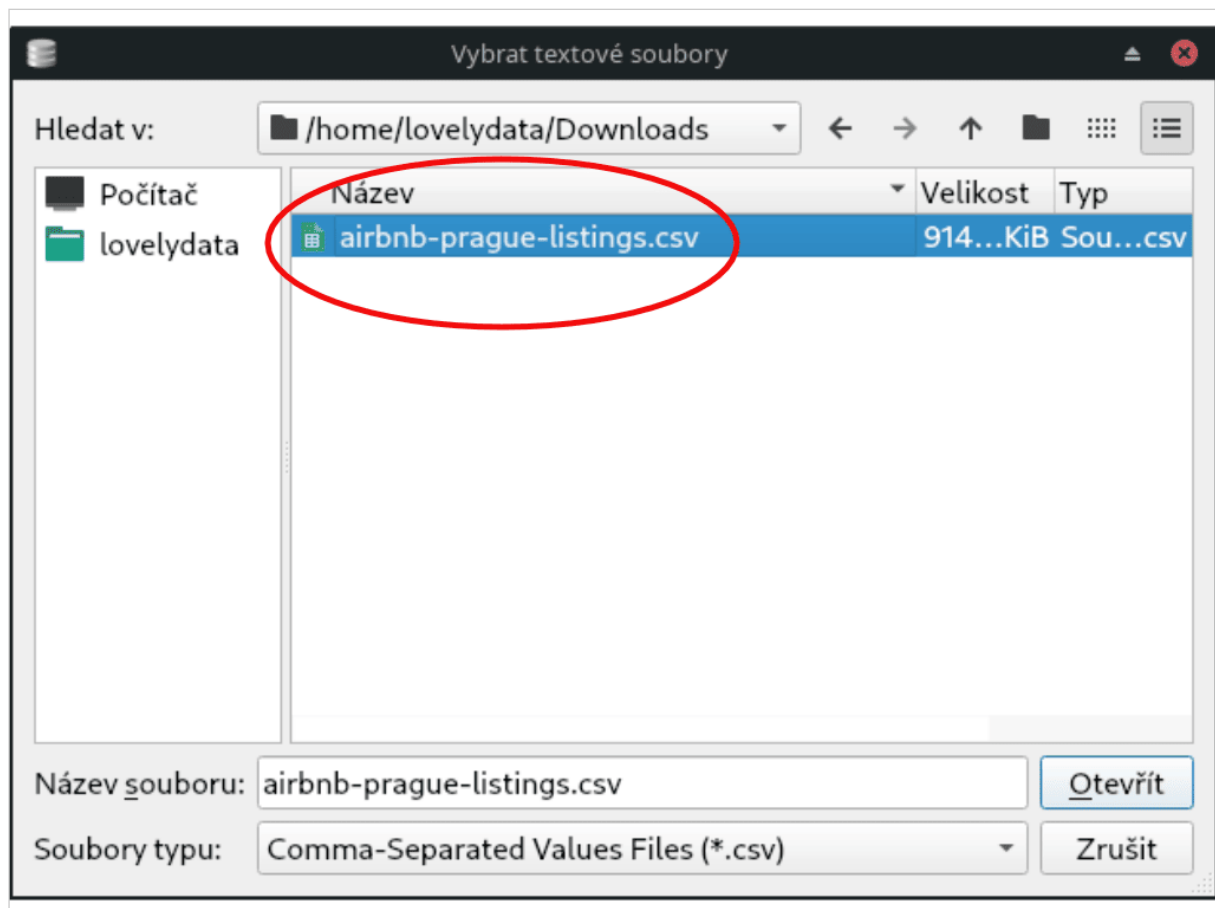
Vytvoření nové databáze



Název databáze



Tabulka ze souboru CSV



Název souboru

Importovat soubor CSV

Název tabulky:

Názvy sloupců v prvním řádku: ☒

Oddělovací pole:

Uvozovka:

Kódování:

Ořezat pole?: ☒

	id	name	host_id	host_name	hbourhood_g	neighbourhood	latitude	lo
1	3884	Enjoy/rela...	3128	Regina		Praha 1	50.08411	14.4
2	23163	Quiet 1BR ...	5282	Klara		Praha 1	50.08229	14.4
3	23169	M302-Coz...	5282	Klara		Praha 1	50.0883	14.4
4	26755	Central ...	113902	Daniel+Bea		Praha 1	50.08729	14.4
5	30762	R23-Elega...	5282	Klara		Praha 1	50.08821	14.4
6	42514	"NEWLY ...	185641	Victoria		Praha 1	50.08228	14.4
7	50967	Ruterra 2 ...	227945	Alex And ...		Praha 3	50.07939	14.4
8	55844	Renovated...	227945	Alex And ...		Praha 2	50.07795	14.4
9	55856	Renovated...	227945	Alex And ...		Praha 2	50.07617	14.4
10	70003	Yellow ...	227945	Alex And ...		Praha 3	50.08557	14.4
11	75298	PRAGUE ...	398991	Emanuele		Praha 1	50.092285	14.4
12	76254	Great & ...	406700	Rena		Praha 2	50.06996	14.4

Import CSV

Takto jednoduše získáme z textového CSV souboru SQL databázi, díky které můžeme analyzovat data lokálně. *Velmi rychlé, pohodlné a praktické.*

10.8. Praha a Airbnb

Airbnb zveřejňuje vybrané pražské statistiky na [webu](#). My se jimi budeme inspirovat, ale nebudeme je na 100 % kopírovat.

I když jsou v datové sadě ceny za noc uvedené v lokální měně (tedy v Kč), na výše uvedené stránce se zobrazují v amerických dolarech. Pouhým selským rozumem tak odhalíme, že uváděná průměrná cena ubytování \$3505/noc (asi 82000 Kč) je **nereálná**. A to i pro Prahu.

10.9. Celkový počet záznamů

Kolik je záznamů v souboru, který jsme si stáhli? Na to odpoví jednoduché SQL.

```
1 SELECT COUNT(*) AS "Celkový počet"
2 FROM "airbnb-prague-listings"
3 ;
```

Výsledek

```
Celkový počet
-----
6632
```

10.10. Typ pokoje

Hostitelé na Airbnb mohou nabízet celé domy, byty, soukromé anebo sdílené pokoje. Nověji také hotelové pokoje.

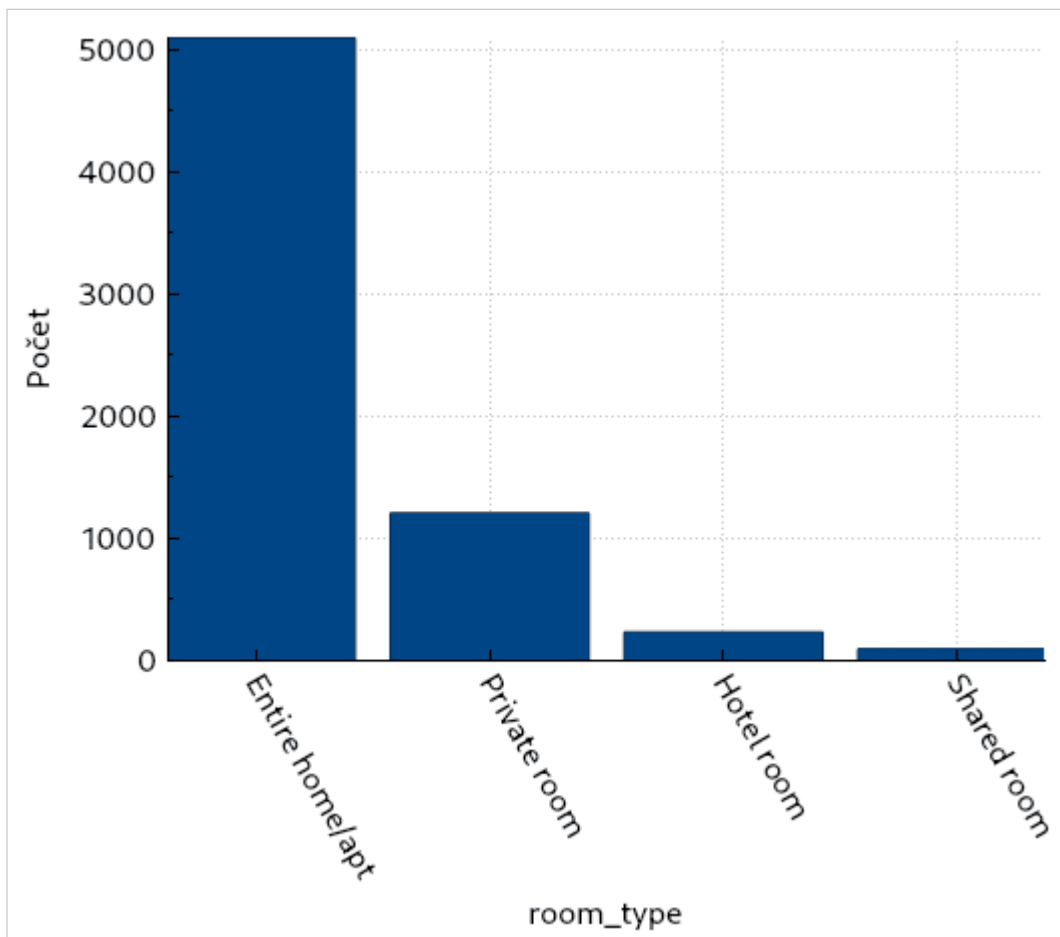
Tento SQL dotaz nám ukáže, jakých typů ubytování je v Praze nejvíce.

```
1 WITH airbnb AS (
2     SELECT room_type, COUNT(*) AS "Počet"
3     FROM "airbnb-prague-listings"
4     GROUP BY room_type
5     ORDER BY 2 DESC
6 )
7 SELECT *, ROUND(((0.0+"počet")/SUM("počet") OVER())) * 100, 2)
8     || '%' AS "Podíl"
9 FROM airbnb
10 ;
```

Výsledek

room_type	Počet	Podíl

Entire home/apt	5095	76.82%
Private room	1208	18.21%
Hotel room	234	3.53%
Shared room	95	1.43%



Typ pokoje

Všimli jste si způsobu, jakým počítáme podíl v procentech? SQLite (stejně jako další RDBMS) umí [pokročilé agregační funkce](#), které jsou nazývány *window functions* nebo také *analytical functions*. Takové funkce dokáží pracovat s hodnotami z více řádků. My jsme využili SUM, abychom dostali celkový součet jednotlivých typů pokojů. Díky tomu pak snadno spočítáme podíl v procentech.

10.11. Průměrná doba pobytu

Co myslíte, bude v nabídce pražských Airbnb pronájmů více krátkodobých nebo dlouhodobých pobytů?

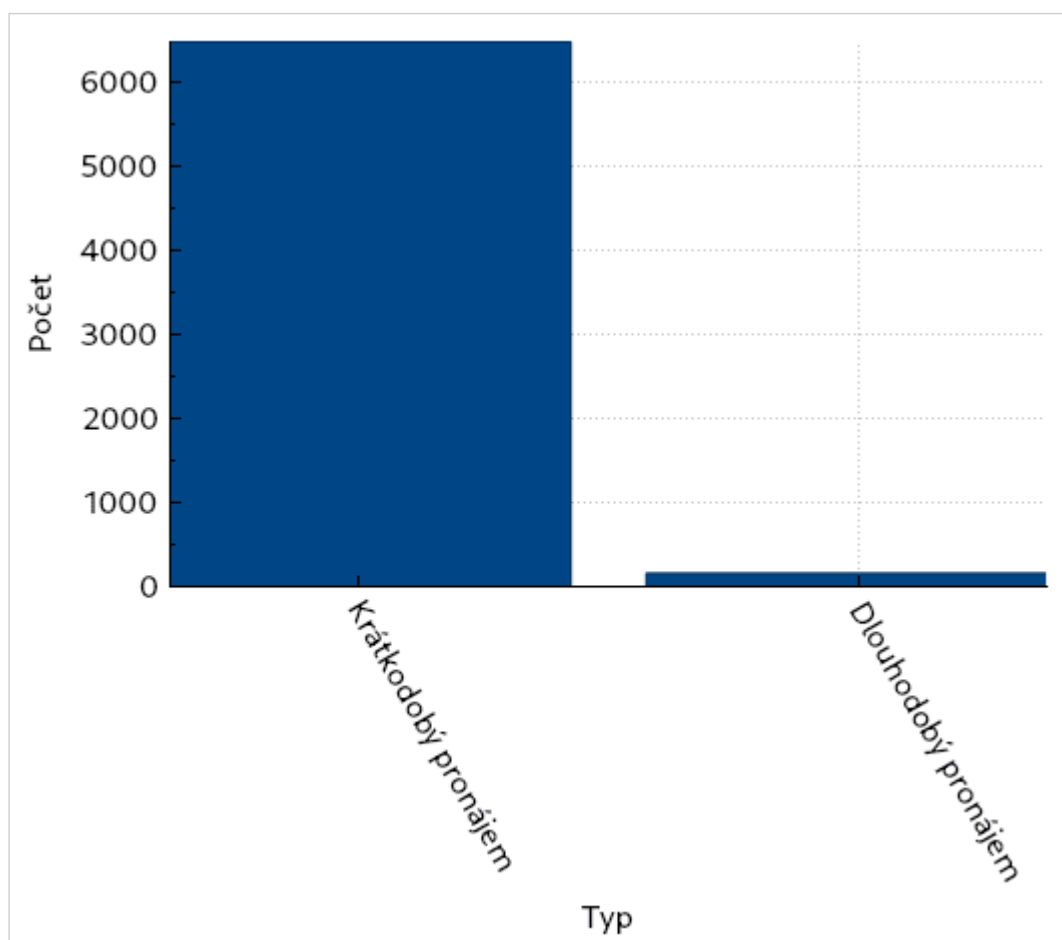
Airbnb definuje pobyty s minimální délkou pobytu do 30 dnů jako krátkodobé. Použijeme tedy stejnou definici.

```
1 WITH airbnb AS (  
2     SELECT minimum_nights,
```

```

3      CASE
4          WHEN minimum_nights < 30 THEN 'Krátkodobý
        pronájem'
5          ELSE 'Dlouhodobý pronájem'
6      END AS Typ
7  FROM "airbnb-prague-listings"
8  )
9  SELECT Typ, COUNT(*) AS "Počet"
10 FROM airbnb
11 GROUP BY Typ
12 ORDER BY "Počet" DESC
13 ;

```



Průměrná doba pobytu

Na první pohled je jasné, že nabídka krátkodobých pobytů převažuje. To není nic překvapivého, protože cílová skupina pronajímatelů jsou hlavně turisté, kteří nebudou v bytě bydlet dlouho.

Airbnb sice tvrdí, že se trh přesunul k dlouhodobějším pobytům, ale v případě Prahy to určitě neplatí.

10.12. Průměrná cena

Kolik stojí noc v Praze? Pokud bychom zahrnuli do výpočtu průměrné ceny všechny záznamy, byl by výsledek hodně zkreslený.

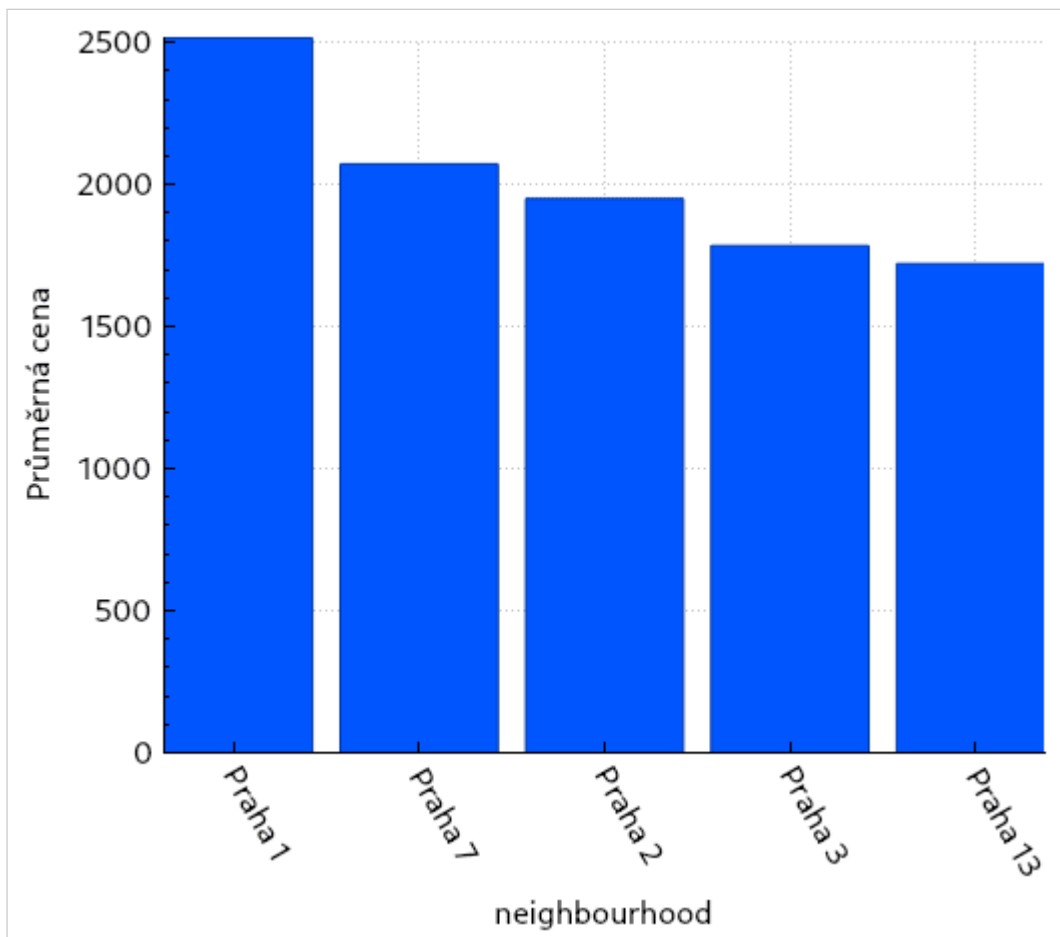
Proto pro výpočet uplatníme následující omezení:

- Ubytování, které bylo hodnoceno během posledních 6 měsíců
- Název čtvrti začíná na Praha
- Čtvrť má v nabídce více než 10 ubytování, aby nám příliš malý počet nezkruslil výsledky
- Pouze TOP 5 podle ceny

```
1 SELECT neighbourhood,  
2     COUNT(price) AS "Počet ubytování",  
3     CAST(AVG(price) AS INTEGER) AS "Průměrná cena"  
4 FROM "airbnb-prague-listings"  
5 WHERE last_review > '2021-10-01'  
6     AND neighbourhood LIKE 'Praha%'  
7 GROUP BY neighbourhood  
8 HAVING "Počet ubytování" > 10  
9 ORDER BY "Průměrná cena" DESC  
10 LIMIT 5  
11 ;
```

Výsledek

neighbourhood	Počet ubytování	Průměrná cena
Praha 1	1531	2515
Praha 7	158	2072
Praha 2	559	1950
Praha 3	341	1784
Praha 13	22	1721



Průměrná cena

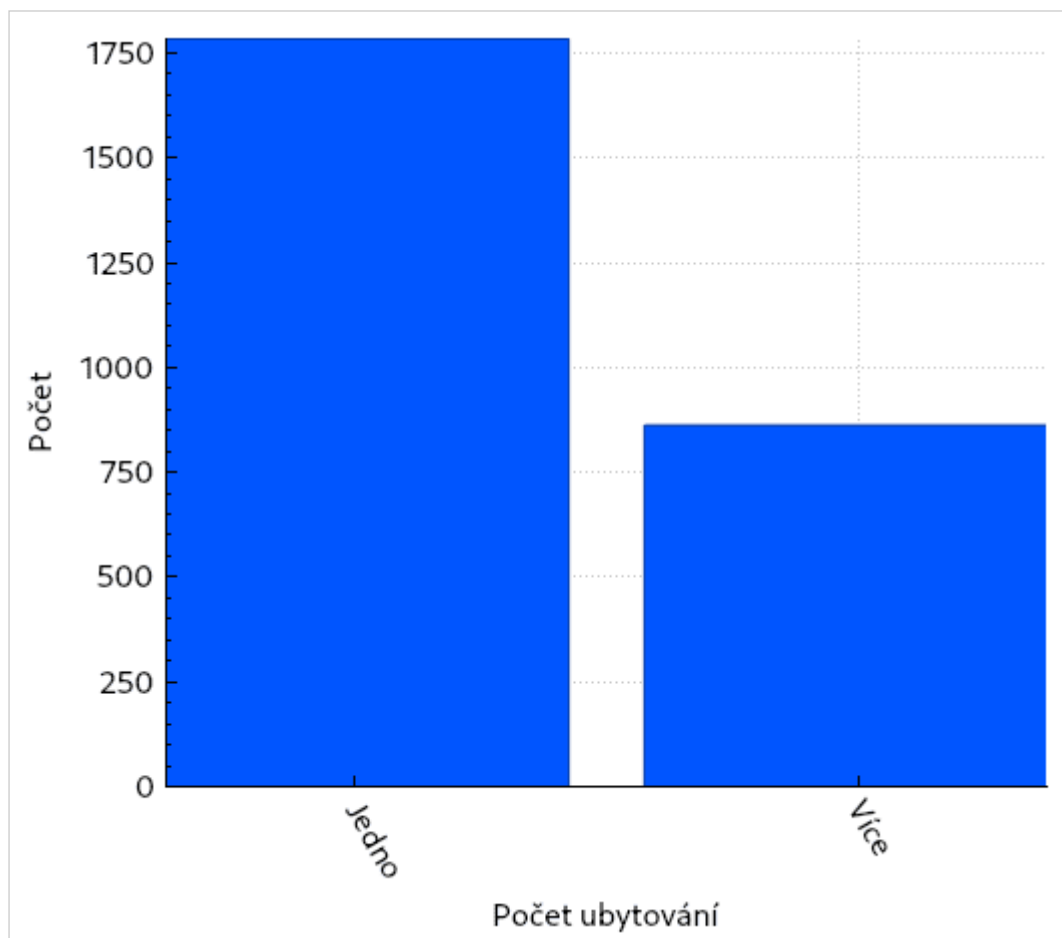
10.13. Počet nabídek jednotlivých hostitelů

Někteří hostitelé Airbnb nabízí více než jedno ubytování. Mohou nabízet samostatné pokoje v jednom bytě nebo více bytů či domů jako celek.

Hostitele s více nabídkami můžeme považovat za podnikatele, kteří ve všech nemovitostech určitě nebydlí.

```
1 WITH airbnb AS (  
2     SELECT DISTINCT host_name || ' (' || host_id || ')' AS  
   "Hostitel",  
3     calculated_host_listings_count,  
4     CASE calculated_host_listings_count WHEN 1 THEN  
   'Jedno' ELSE 'Více' END AS "Počet ubytování"  
5     FROM "airbnb-prague-listings"  
6 )
```

```
7 SELECT "Počet ubytování", COUNT(*) AS "Počet"  
8 FROM airbnb  
9 GROUP BY "Počet ubytování"  
10 ;
```



Počet nabídek

Výsledek ukazuje, že téměř polovina hostitelů nabízí více než jedno ubytování.

10.14. Pražští ubytovací magnáti

Kdo má v nabídce nejvíc ubytování? A o jakém počtu nabídek za jednoho hostitele se vůbec bavíme? Jsou to jednotky, desítky nebo ještě víc?

Pro zajímavost se také podíváme, za jakou průměrnou cenu za noc ubytování nabízí.

```
1 SELECT host_name || ' (' || host_id || ')' AS "Hostitel",
```

```

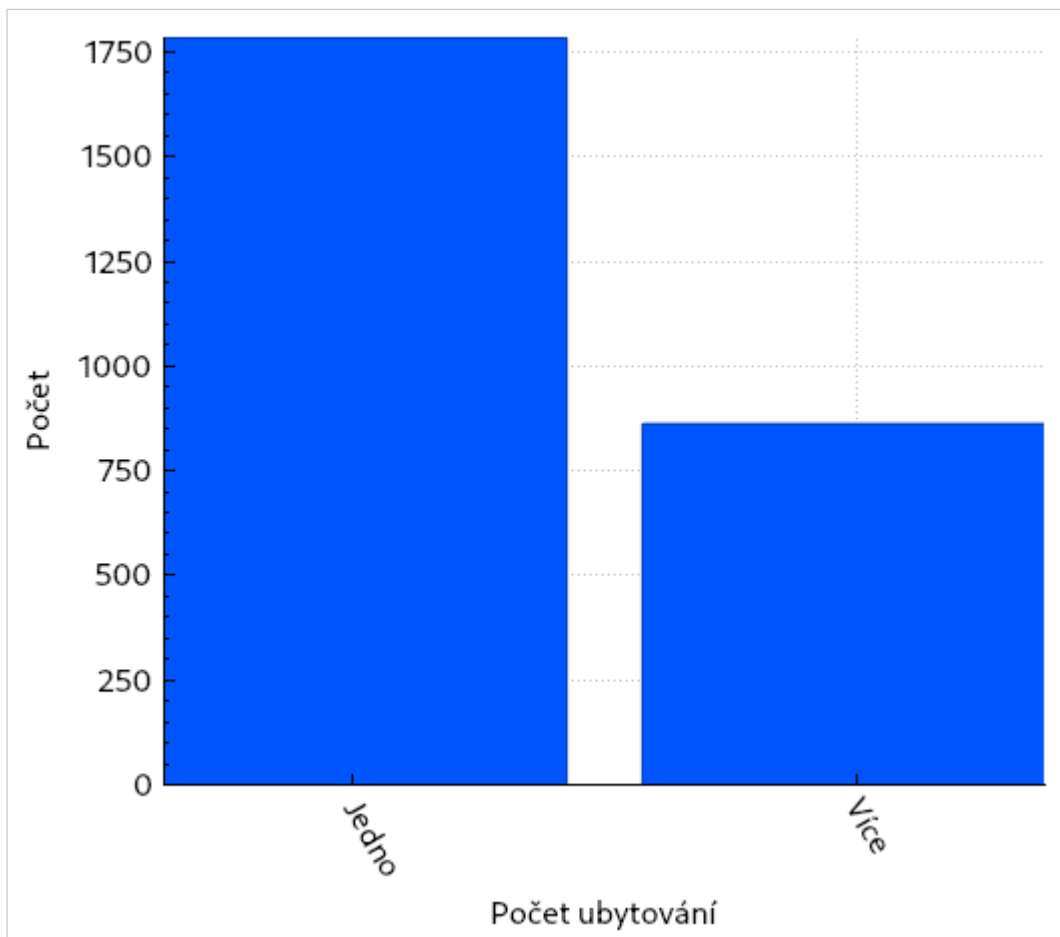
2     calculated_host_listings_count,
3     CAST(AVG(price) AS INTEGER) AS "Průměrná cena/noc"
4 FROM "airbnb-prague-listings"
5 GROUP BY Hostitel, calculated_host_listings_count
6 ORDER BY calculated_host_listings_count DESC
7 LIMIT 5
8 ;

```

Výsledek

Hostitel	calculated_host_listings_count
Průměrná cena/noc	

Josef (8087351)	86
2113	
Michal & Friends (52431987)	76
2537	
Anton & Vika (9550037)	65
2060	
Prague For You (1167781)	58
4405	
Klara (5282)	55
3970	



Ubytovací magnáti

POZNÁMKA

Pro analýzu nebyly použity žádné soukromé informace. Jména, nabídky a další údaje o recenzích jsou veřejně přístupné na stránkách Airbnb.

10.15. Závěrem

Jako při každé analýze, je potřeba vzít v úvahu nejen data samotná, ale i data která nejsou k dispozici a také širší souvislosti. Profesionální analytik nikdy nesklouzne k rychlým a zjednodušujícím interpretacím.

Nejlepší analytici proto vždy kromě dat **používají selský rozum**.

11. Pět fontů, které se skvěle hodí pro SQL

11.1. Správný výběr fontu

Pokud pracujete s počítačem často a trávíte u obrazovky i několik hodin denně, je výběr správného fontu důležitý. Font výrazně ovlivňuje schopnost rychlé orientace v textu a namáhání očí.

Jsou určité fonty, které dobře vypadají, ale nehodí se pro delší používání.

Následujících 5 fontů je k dispozici zdarma a navíc se perfektně hodí pro zobrazení kódu. Jsou to totiž fonty, které byly přímo stvořeny pro programátory.

Přinášíme jejich porovnání nejen v klasickém světlém (light) tématu, ale i v dnes tolik populárnímu tmavém (dark) módu. Velikost je pokaždé 16px.

Ještě k tomu tmavému módu. Údajně nemá na oči žádný vliv, jak někteří věří. Podle jiné studie dokonce většina normálně vidících uživatelů vykazuje vyšší výkon při použití světlého tématu. Na druhou stranu tmavá témata mohou šetřit baterii. Tak si vyberte. 😊

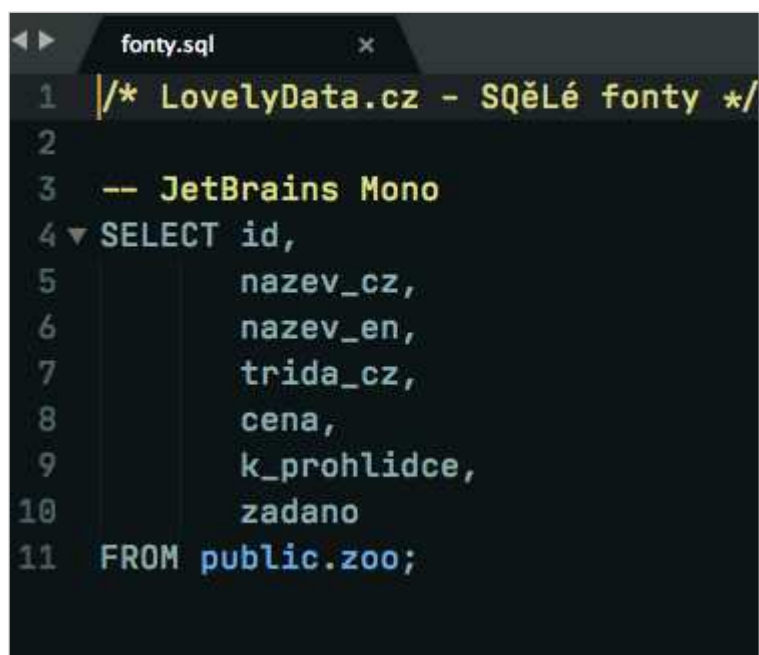
11.2. JetBrains Mono



A screenshot of a code editor window titled 'fonty.sql'. The editor displays SQL code with a light background and yellow highlighting for comments. The code is as follows:

```
1  /* LovelyData.cz - SQěLé fonty */
2
3  -- JetBrains Mono
4  SELECT id,
5         nazev_cz,
6         nazev_en,
7         trida_cz,
8         cena,
9         k_prohlidce,
10        zadano
11  FROM public.zoo;
```

Světlé schéma




A screenshot of a code editor window titled 'fonty.sql'. The editor displays the same SQL code as the previous image, but with a dark background and light-colored text. The code is as follows:

```
1  /* LovelyData.cz - SQěLé fonty */
2
3  -- JetBrains Mono
4  SELECT id,
5         nazev_cz,
6         nazev_en,
7         trida_cz,
8         cena,
9         k_prohlidce,
10        zadano
11  FROM public.zoo;
```

Tmavé schéma

www.jetbrains.com/lp/mono/

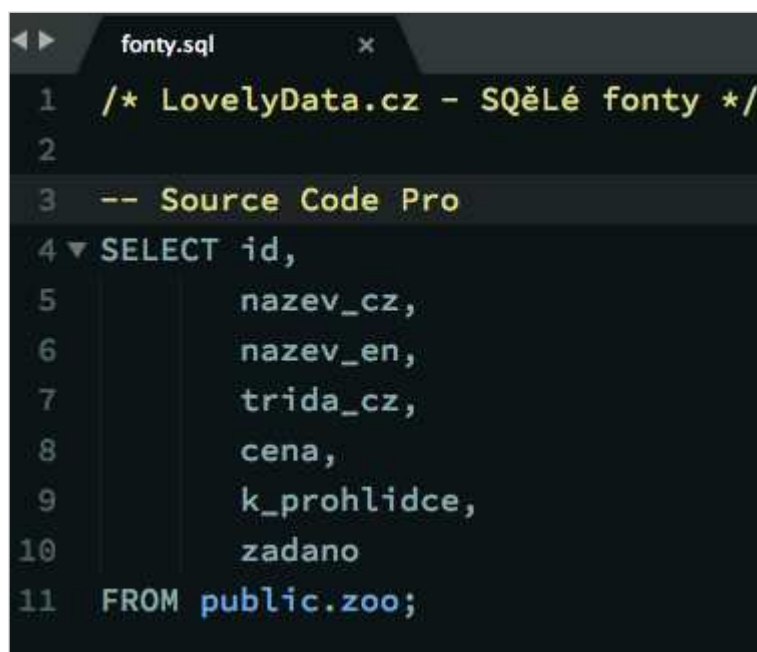
11.3. Source Code Pro



A screenshot of a SQL editor window titled 'fonty.sql'. The editor displays a SQL query using the Source Code Pro font in a light theme. The query is as follows:

```
1  /* LovelyData.cz - SQěLé fonty */
2
3  -- Source Code Pro
4  SELECT id,
5         nazev_cz,
6         nazev_en,
7         trida_cz,
8         cena,
9         k_prohlidce,
10        zadano
11  FROM public.zoo;
```

Světlé schéma



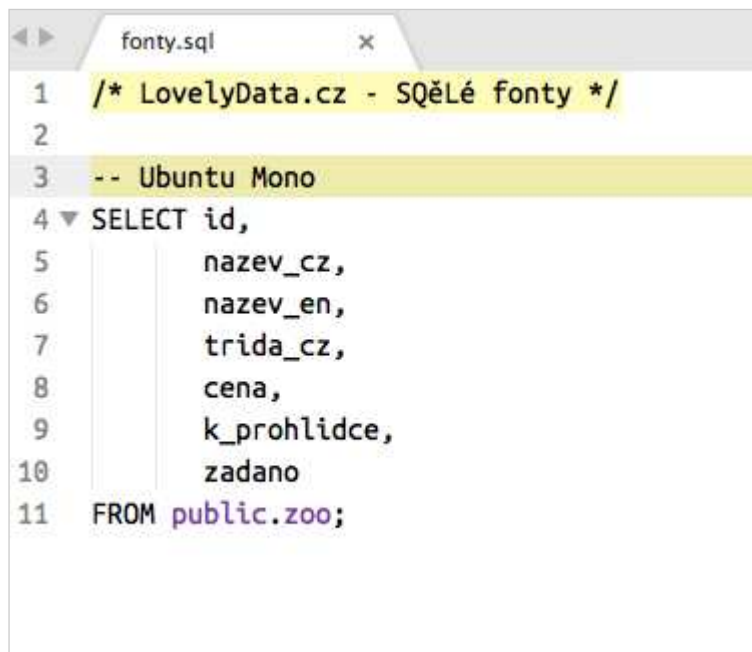
A screenshot of a SQL editor window titled 'fonty.sql'. The editor displays the same SQL query as the previous image, but using the Source Code Pro font in a dark theme. The query is as follows:

```
1  /* LovelyData.cz - SQěLé fonty */
2
3  -- Source Code Pro
4  SELECT id,
5         nazev_cz,
6         nazev_en,
7         trida_cz,
8         cena,
9         k_prohlidce,
10        zadano
11  FROM public.zoo;
```

Tmavé schéma

github.com/adobe-fonts/source-code-pro

11.4. Ubuntu Mono



```
fonty.sql x
1  /* LovelyData.cz - SQĚLé fonty */
2
3  -- Ubuntu Mono
4  ▼ SELECT id,
5      nazev_cz,
6      nazev_en,
7      trida_cz,
8      cena,
9      k_prohlidce,
10     zadano
11  FROM public.zoo;
```

Světlé schéma

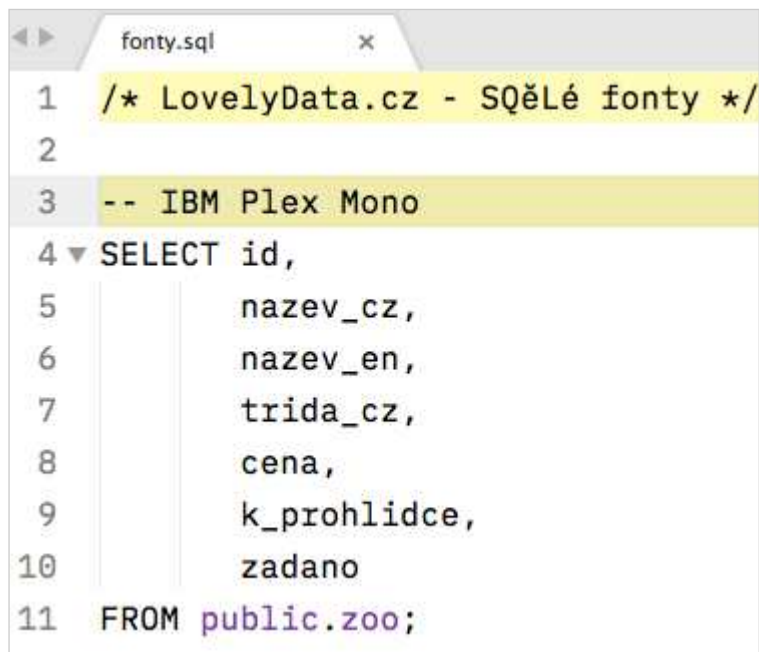


```
fonty.sql x
1  /* LovelyData.cz - SQĚLé fonty */
2
3  -- Ubuntu Mono
4  ▼ SELECT id,
5      nazev_cz,
6      nazev_en,
7      trida_cz,
8      cena,
9      k_prohlidce,
10     zadano
11  FROM public.zoo;
```

Tmavé schéma

design.ubuntu.com/font/

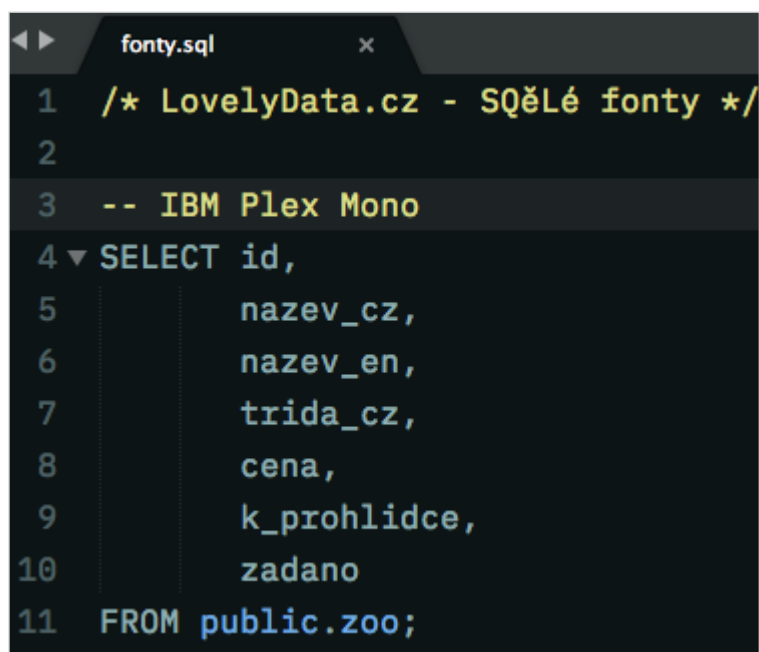
11.5. IBM Plex Mono



A screenshot of a SQL editor window titled 'fonty.sql'. The editor displays a SQL query using the IBM Plex Mono font in a light theme. The query is as follows:

```
1  /* LovelyData.cz - SQĚLé fonty */  
2  
3  -- IBM Plex Mono  
4  ▼ SELECT id,  
5      nazev_cz,  
6      nazev_en,  
7      trida_cz,  
8      cena,  
9      k_prohlidce,  
10     zadano  
11  FROM public.zoo;
```

Světlé schéma



A screenshot of the same SQL editor window titled 'fonty.sql', but with a dark theme. The SQL query is identical to the one in the light theme screenshot:

```
1  /* LovelyData.cz - SQĚLé fonty */  
2  
3  -- IBM Plex Mono  
4  ▼ SELECT id,  
5      nazev_cz,  
6      nazev_en,  
7      trida_cz,  
8      cena,  
9      k_prohlidce,  
10     zadano  
11  FROM public.zoo;
```

Tmavé schéma

www.ibm.com/plex/

11.6. Cascadia Code



A screenshot of a SQL editor window titled 'fonty.sql'. The code is displayed in a light-themed font. The first line is a comment: '/* LovelyData.cz - SQĚLé fonty */'. The second line is empty. The third line is a comment: '-- Cascadia Code'. The fourth line starts with 'SELECT id,'. The fifth line has 'nazev_cz,'. The sixth line has 'nazev_en,'. The seventh line has 'trida_cz,'. The eighth line has 'cena,'. The ninth line has 'k_prohlidce,'. The tenth line has 'zadano'. The eleventh line is 'FROM public.zoo;'. The font is a clean, modern sans-serif typeface.

```
1  /* LovelyData.cz - SQĚLé fonty */
2
3  -- Cascadia Code
4  SELECT id,
5      nazev_cz,
6      nazev_en,
7      trida_cz,
8      cena,
9      k_prohlidce,
10     zadano
11  FROM public.zoo;
```

Světlé schéma



A screenshot of a SQL editor window titled 'fonty.sql'. The code is displayed in a dark-themed font. The first line is a comment: '/* LovelyData.cz - SQĚLé fonty */'. The second line is empty. The third line is a comment: '-- Cascadia Code'. The fourth line starts with 'SELECT id,'. The fifth line has 'nazev_cz,'. The sixth line has 'nazev_en,'. The seventh line has 'trida_cz,'. The eighth line has 'cena,'. The ninth line has 'k_prohlidce,'. The tenth line has 'zadano'. The eleventh line is 'FROM public.zoo;'. The font is a clean, modern sans-serif typeface.

```
1  /* LovelyData.cz - SQĚLé fonty */
2
3  -- Cascadia Code
4  SELECT id,
5      nazev_cz,
6      nazev_en,
7      trida_cz,
8      cena,
9      k_prohlidce,
10     zadano
11  FROM public.zoo;
```

Tmavé schéma

github.com/microsoft/cascadia-code

12. SQL nebo Python? Návod, jak si správně vybrat.

12.1. SQL nebo Python?

Je lepší začít s [SQL](#) nebo s [Pythonem](#)? To je otázka, kterou v LovelyData dostáváme často. Poradíme vám nejen s kterým jazykem začít, ale i jaké programy používat, abyste oba jazyky využili na maximum.

12.2. TLDR

Pokud chcete pracovat jako analytik, tester, programátor, datový vědec (data scientist) - prostě kdokoliv, kdo potřebuje získávat informace z databází - musíte se naučit SQL. SQL je základní a univerzální jazyk bez kterého se neobejdete.

Pokud vás láká práce s daty a chcete umět data analyzovat, čistit a vizualizovat, budete se muset ještě naučit Python. S tímto jazykem si poradíte v každé situaci, protože umí pracovat s Excelem, textovými soubory, SQL databázemi, rozhraními API a tím jeho možnosti ani zdaleka nekončí.

12.3. Trocha historie

Možná vás překvapí, že SQL i Python jsou technologie, které zajišťují dlouhodobou - a lukrativní - kariéru. Důvod? Používají se už několik desítek let a jejich popularita vůbec neklesá. Spíše naopak.

12.3.1. SQL

Jazyk SQL má za sebou historii, která je delší než 50 let. Už přes půl století je tento dotazovací jazyk oblíbeným nástrojem pro práci s daty v tabulkách, které jsou uloženy v databázích.

Má ale i další výhodu - stal se standardním a univerzálním jazykem, kterému rozumí relační databázové systémy (RDBMS) od různých výrobců. Pokud se tedy naučíte SQL na bezplatném PostgreSQL, můžete většinu příkazů použít třeba na komerčním SQL Serveru od Microsoftu. Většina SQL příkazů totiž funguje úplně stejně na všech databázích.

Pár zajímavostí:

- Zkratka SQL znamená Structured Query Language.
- Vznikl v roce 1970 ve společnosti IBM.
- Původní název SEQUEL (Structured English Query Language).
- 2 možnosti výslovnosti [es kjú el] nebo [síkwl].
- Standardní jazyk pro relační databáze (RDBMS).
- Používají všechny nejrozšířenější databázové systémy jako např.:
 - [PostgreSQL](#),
 - [MySQL](#),
 - [Microsoft SQL Server](#),
 - [Oracle Database](#),
 - [SQLite](#),
 - [IBM DB2](#),
 - [Microsoft Access](#),
 - a další.

12.3.2. Python

[Python](#) patří mezi nejpoblárnější jazyky současnosti a zcela zaslouženě. Má široké možnosti využití a dvě velké výhody - je jednoduchý a výkonný. Jednoduchostí je zde myšlena jeho syntaxe, protože kód v Pythonu se *téměř* čte jako angličtina. Proto se jedná o vhodný jazyk i pro začátečníky.

Data se dnes vyskytují v mnoha podobách a formátech a ne pouze v SQL databázích. Data můžete získat v souborech CSV, v textu, na webu, ve formátech jako je XML nebo JSON a to jsme zdaleka nevyjmenovali všechny. Právě v této spleti dat vyniká Python, který nabízí spoustu možností, jak data analyzovat, čistit anebo vizualizovat.

Python se používá pro vývoj aplikací, webů, pro datovou vědu, Internet of things (IoT) nebo třeba pro systémovou administraci. Poptávka po lidech, kteří ho ovládají, každým rokem stoupá.

Pár zajímavostí:

- Pojmenován po britské komediální skupině *Monty Python*.
- Publikován v roce 1991.
- 2 možnosti výslovnosti [paj tn] nebo [paj ton].
- Zaměření na dobrou čitelnost kódu.
- Má otevřený zdrojový kód.
- Je k dispozici zdarma pro mnoho operačních systémů (Windows, macOS, Linux a spoustu dalších).

12.4. Který jazyk je jednodušší pro začátečníky?

Jako jazyk je SQL rozhodně snadnější než Python. Zejména základy SQL zvládnete opravdu rychle. Protože se jedná převážně o dotazovací jazyk (query language), obsahuje mnohem jednodušší a menší sadu příkazů než jiné jazyky.

12.5. Porovnání SQL a Pythonu

Níže najdete 3 příklady, které dělají totéž. Jeden je napsaný v SQL, druhý v Pythonu a třetí v Pythonu s využitím knihovny Pandas. I kdybyste neměli žádné znalosti ani jednoho z těchto jazyků, tak se vsadíme, že budete v principu vědět co dělají.

Pro vyzkoušení prvních dvou příkladů můžete využít online editor na replit.com. Tento IDE (Integrated Development Environment) funguje v prohlížeči, podporuje velké množství jazyků a hlavně - nemusíte nic instalovat. Třetí příklad na Replit bohužel nepůjde, protože knihovna Pandas tam (zatím) není k dispozici.

12.5.1. SQL

Spusťte SQL online editor na replit.com/languages/sqlite. Do levého okna zkopírujte následující SQL příkazy a spusťte je.

```
1 /* Vytvoř tabulku BAND */
2 CREATE TABLE BAND
3 (
4     id INT,
5     name VARCHAR(50)
```

```

6 );
7
8 /* Vlož záznamy */
9 INSERT INTO BAND VALUES(1, 'John Lennon');
10 INSERT INTO BAND VALUES(2, 'Paul McCartney');
11 INSERT INTO BAND VALUES(3, 'George Harrison');
12 INSERT INTO BAND VALUES(4, 'Ringo Starr');
13
14 /* Vyber Ringa */
15 SELECT *
16 FROM BAND
17 WHERE name = 'Ringo Starr';

```

Výsledek

```
4|Ringo Starr
```

12.5.2. Python

Spusťte Python online editor na replit.com/languages/python3. Do levého okna zkopírujte následující Python kód a spusťte ho.

```

1 # Vytvoř seznam BAND
2 band = list()
3
4 # Vlož záznamy
5 band.append([1, 'John Lennon'])
6 band.append([2, 'Paul McCartney'])
7 band.append([3, 'George Harrison'])
8 band.append([4, 'Ringo Starr'])
9
10 # Vyber Ringa
11 for member in band:
12     id, name = member
13     if name == 'Ringo Starr':
14         print(member)

```

Výsledek

```
[4, 'Ringo Starr']
```

12.5.3. Python a Pandas

Pandas je jedna z nejpoblárnějších knihoven Pythonu, která umožňuje pracovat s daty v mnoha různých formátech. Ať už jsou data uložena v textových souborech, Excelovských sešitech nebo SQL databázích - Pandas s nimi pracuje snadno, rychle a efektivně. Proto je tato knihovna tak oblíbená mezi analytiku a datovými vědci.

Následující kód dělá totéž jako předchozí dva příklady, jen k tomu používá Python spolu s knihovnou Pandas. Výsledný kód je tak kratší.

```
1 import pandas as pd
2
3 # Vytvoř DataFrame BAND a vlož záznamy
4 data = {'name': ['John Lennon', 'Paul McCartney',
5                 'George Harrison', 'Ringo Starr']}
6 band = pd.DataFrame(data)
7
8 # Vyber Ringa
9 ringo = band['name'] == 'Ringo Starr'
10 print(band[ringo])
```

Výsledek

```
      name
3  Ringo Starr
```

12.6. Jazyky pro lidi

Předchozí příklady názorně ukazují, jak se oba jazyky podobají angličtině a jsou díky tomu dobře čitelné. Jejich základy tak snadno zvládnou i uživatelé, kteří s nimi nemají předchozí zkušenosti.

Popularita obou jazyků také zaručuje dostupnost velkého množství výukových

materiálů. Ať už se jedná o různé návody, videa nebo [kurzy](#). Pokud se někdo chce naučit jeden nebo oba jazyky, může začít klidně hned.

12.7. Závěr

Jak SQL, tak i Python, jsou jazyky prověřené časem. Používají se na celém světě a jejich základy se dají naučit snadno.

Proto rozhodně nešlápnete vedle, když si vyberete jeden nebo druhý jazyk. Pokud vám jako LovelyData můžeme poradit, tak doporučujeme naučit se oba jazyky - alespoň jejich základy. Otevřete si tak dveře k novým příležitostem, ze kterých si budete vybírat.

13. V čem psát kód

13.1. Úvod

I když můžete SQL a Python kód psát klidně v tom nejjednodušším editoru (např. Notepad nebo TextEdit), pro skutečnou práci budete chtít využívat propracovanější software. Naštěstí pro oba jazyky existuje spousta aplikací, které jsou skvělé a mnoho z nich je navíc úplně zdarma.

13.1.1. Textový editor vs IDE

Pokud se začnete zajímat o programování a editory, tak narazíte na pojem IDE (Integrated Development Environment). Jaký je mezi nimi rozdíl?

Textový editor má na rozdíl od vývojového prostředí (IDE) méně funkcí. Mnoho editorů je ale snadno rozšiřitelných pomocí různých pluginů a rozdíly mezi textovým editorem a IDE se tak začínají stírat.

Obecně lze říci, že integrované vývojové prostředí je primárně zaměřené na jeden programovací jazyk a díky tomu obsahuje (integruje) všechno, co programátor potřebuje pro práci s tímto jazykem. Z toho vyplývá, že IDE bude mít strmější učitelskou křivku.

Pro začátečníky je proto lepší začít s textovým editorem, který se naučí rychleji a který navíc využijí pro více než jeden programovací jazyk.

13.1.2. SQL

Pro psaní SQL je ideální použít *SQL editor*, který v sobě spojuje textový editor a zároveň se umí připojit k databázi. Mnoho těchto editorů dokáže pracovat hned s několika RDBMS a vy tak pohodlně ovládáte data z jednoho místa.

SQL Workbench/J	sql-workbench.eu	Zdarma, více RDBMS
DB Browser for SQLite	sqlitebrowser.org	Zdarma, pouze SQLite
DBeaver	dbeaver.io	Zdarma, více RDBMS
HeidiSQL	heidisql.com	Zdarma, více RDBMS

SQL Server Management Studio (SSMS)	docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms	Zdarma, pouze MS SQL
MySQL Workbench	mysql.com/products/workbench	Zdarma, pouze MySQL
Oracle SQL Developer	oracle.com/database/sql-developer/	Zdarma, pouze Oracle
SquirrelL	squirrel-sql.sourceforge.net	Zdarma, více RDBMS
Adminer	adminer.org/cs	Zdarma, více RDBMS, jediný PHP soubor
DbVisualizer	dbvis.com	Free verze, více RDBMS
VS Code	code.visualstudio.com	Zdarma, více jazyků
Beekeeper Studio	beekeeperstudio.io	Free verze, více RDBMS

13.1.3. SQL v cloudu

Trendem posledních let, alespoň mezi dodavateli takových nástrojů, je snaha **přesunout vše do cloudu**. To má samozřejmě svoje výhody i nevýhody. Velkou výhodou při používání takových cloudových SQL klientů je fakt, že si uživatelé nemusí nic instalovat na svoje PC. Stačí jen otevřít prohlížeč a přihlásit se. To šetří čas nejen uživatelům, ale i správcům firemních PC.

PopSQL	popsql.com/	Free verze, více RDBMS
Hue	gethue.com	Zdarma, více RDBMS
CloudBeaver	cloudbeaver.io	Free verze, více RDBMS
SQL.BAND	hub.docker.com/r/sqliteonlinecom/sqlband	Zdarma

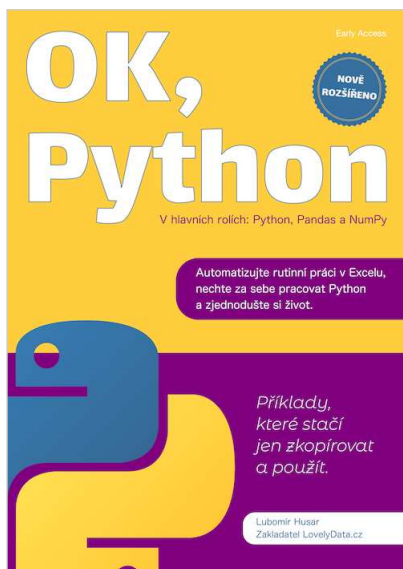
13.1.4. Python

Pro Python máte k dispozici nepřeberné množství textových editorů a IDE. Níže najdete ty nejpopulárnější.

VS Code	code.visualstudio.com	Zdarma, více jazyků
Notepad++	notepad-plus-plus.org	Zdarma, více jazyků
Sublime Text	sublimetext.com	Zkušební verze zdarma, více jazyků
Jupyter Notebook	jupyter.org	Zdarma, více jazyků
PyCharm	jetbrains.com/pycharm	Free Community verze, pouze Python
Spyder	www.spyder-ide.org	Zdarma, pouze Python
Thonny	thonny.org	Zdarma, pouze Python
Vim	www.vim.org	Zdarma, více jazyků

14. Mohlo by vás zajímat

14.1. Vyšlo



OK, Python

Automatizujte rutinní práci v Excelu, nechte za sebe pracovat Python a zjednodušte si život. V hlavních rolích: Python, Pandas a NumPy. Příklady, které stačí jen zkopírovat a použít.

[Knihu můžete koupit zde](#)

14.2. Připravujeme



Python pro každý den

Nejlepší návody a příklady, které vám ušetří spoustu času. Pro všechny, kteří si chtějí usnadnit práci a místo sebe raději *zaměstnat* Python.

[Nechte se informovat, až kniha vyjde](#)

Rejstřík

A

Adminer, [82](#)
agregační funkce, [47](#), [62](#)
Airbnb, [53](#), [60](#)
analytical functions, [62](#)
analytické funkce, [62](#)
analytik, [75](#)
analýza dat, [44](#)
API, [75](#)
a Python, [81](#)

B

Beekeeper Studio, [82](#)
BETWEEN, [18](#), [19](#), [20](#), [23](#)

C

Cascadia Code, [74](#)
CASE, [24](#), [24](#), [27](#)
cloud
 SQL cloud, [44](#)
cloud SQL klient, [82](#)
CloudBeaver, [82](#)
Comma Separated Values, [12](#)
Common Table Expressions, [40](#), [46](#), [48](#),
 [50](#), [51](#)
CSV, [9](#), [53](#), [54](#), [55](#), [76](#)
CTE, [46](#), [48](#)

D

Dark mode, [69](#)
Data Dictionary, [54](#)
datová analýza, [44](#), [53](#), [68](#)
datová sada, [18](#), [60](#)
datové sady, [53](#)
datový analytik, [6](#)
datový vědec, [75](#)

DB Browser for SQLite, [9](#), [54](#), [81](#)
DB4S, [12](#)
DBeaver, [81](#)
DbVisualizer, [82](#)
DELETE, [52](#)
DISTINCT, [21](#), [23](#)
dotazovací jazyk
 Query language, [77](#)

E

Escape, [29](#)
Excel, [9](#), [12](#), [47](#), [75](#), [79](#), [84](#)
EXCEPT, [38](#)

F

Font, [69](#)

G

graf, [55](#)

H

HeidiSQL, [81](#)
Hue, [82](#)

I

IBM Plex Mono, [73](#)
IDE, [77](#), [81](#), [82](#)
IF-THEN-ELSE, [24](#)
INSERT, [52](#)
integrované vývojové prostředí, [81](#)
Internet of things, [76](#)
INTERSECT, [37](#)

J

JetBrains Mono, [70](#)
JSON, [76](#)
Jupyter Notebook, [83](#)

L

Light mode, [69](#)
LIKE, [29](#)
LIMIT, [44](#), [46](#)
Linux, [9](#)
Lovely blog, [6](#)
LovelyData, [75](#)
LovelyData.cz, [6](#)

M

macOS, [9](#)
MAX, [44](#), [45](#)
metriky, [53](#)
Microsoft SQL Server, [40](#)
MIN, [44](#), [45](#)
MINUS, [38](#), [39](#)
množinové operace, [34](#)
MySQL, [6](#), [7](#), [18](#), [40](#)
MySQL Workbench, [82](#)

N

Nested query, [48](#)
NOT, [20](#), [23](#)
Notepad, [81](#)
Notepad++, [83](#)
NULL, [20](#)
NumPy, [84](#)

O

OK, Python, [84](#)
online editor, [78](#)
Operace s množinami, [34](#)
Oracle, [6](#), [7](#)
Oracle DB, [39](#)
Oracle SQL Developer, [82](#)
ORDER BY, [21](#), [23](#), [46](#)

P

Pandas, [79](#), [84](#)
PC, [9](#), [54](#)
Plot, [55](#)
poddotaz, [45](#), [48](#)
Poddotazy, [48](#)
poddotazy, [51](#)
PopSQL, [82](#)
PostgreSQL, [6](#), [7](#), [9](#), [18](#), [40](#)
programátor, [75](#)
PyCharm, [83](#)
Python, [75](#), [79](#), [84](#)

R

RDBMS

Relational Database Management
System

Relační databázový systém, [6](#), [39](#),
[43](#), [48](#), [62](#), [75](#), [81](#)

rekurzivní, [48](#)

ROW_NUMBER, [44](#), [46](#)

S

SELECT, [35](#), [52](#)
selský rozum, [68](#)
SET operations, [34](#)
Source Code Pro, [71](#)
Spyder, [83](#)
SQL, [6](#), [9](#), [24](#), [29](#), [40](#), [44](#), [48](#), [60](#), [75](#), [75](#),
[81](#)
SQL editor, [9](#), [81](#)
SQL hack, [40](#)
SQL operátor, [18](#)
SQL Server, [6](#), [7](#), [18](#), [48](#)
SQL Server Management Studio, [82](#)
SQL standard, [43](#)
SQL Table Builder, [7](#)
SQL Workbench/J, [81](#)

SQL.BAND, [82](#)
sqlfiddle.com, [44](#)
SQLite, [6](#), [7](#), [40](#), [54](#), [54](#), [62](#)
sqliteonline.com, [44](#)
SquirrelL, [82](#)
Sublime Text, [83](#)
Subqueries, [48](#)
Subquery, [48](#)
Subselect, [48](#)
SUM, [62](#)

T

tester, [75](#)
TextEdit, [81](#)
Thonny, [83](#)
Titanic, [18](#)
TOP, [44](#), [46](#), [64](#)

U

Ubuntu Mono, [72](#)
UNION, [35](#)
UNION ALL, [36](#)
UPDATE, [52](#)

V

Vim, [83](#)
vizualizace, [53](#), [55](#)
VS Code, [82](#), [83](#)

W

WHEN, [25](#)
window functions, [62](#)
Windows, [9](#)
WITH, [48](#)

X

XML, [76](#)



SQL

pro každý den

Jak si užít SQL a nezabloudit v kódu

Lubomír Husar

 LovelyData