Żeby korzystać z sqliteów w jupyterze trzeba doinstalować kernel xeus sqlite :wink:Instalacja kernela xeus-sqlite do jupyterów:

* **z  użyciem mamby :**

mamba install xeus-sqlite -c conda-forge

* **z użyciem condy :**

conda install xeus-sqlite -c conda-forge

Tutaj dokumentacja:  
<https://xeus-sqlite.readthedocs.io/en/latest/installation.html>

[9:06](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649271968231349)

Przy okazji macierz różnic między konkretnymi DBMSami SQL :blush: :

* <https://troels.arvin.dk/db/rdbms/>

I lista przykładowych pytań rekrutacyjnych dla SQL:

* <https://www.plukasiewicz.net/Pytania_i_odpowiedzi/SQLQA>
* <https://www.guru99.com/sql-interview-questions-answers.html>

 (edited)

Guru99**Guru99**

[**Top 50 SQL Interview Questions and Answers for Experienced & Freshers (2022 Update)**](https://www.guru99.com/sql-interview-questions-answers.html)

SQL interview questions for Freshers and Experienced: Here is a list of the most popular SQL Server interview questions and their answers likely to be asked during the SQL interview basic to advance level SQL questions depending on their experience.

**Written by**

Richard Peterson

**Time to read**

14 minutes

Jun 28th, 2020

**Thursday, April 7th**

Operator LIKE  
Umożliwia przeszukiwanie danych tekstowych na podstawie wzorca podanego w klauzuli warunkowej. Może być to zarówno dokładny ciąg poszukiwanych znaków jak i wyrażenie stworzone przy pomocy następujących symboli:

* Symbol % zastępuje dowolny ciąg znaków
* Symbol \_ zastępuje dowolny pojedynczy znak

SELECT kolumna,kolumna,.. FROM tabela WHERE kolumna LIKE '\_kos%';

[9:13](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649488391072659)

**Ćwiczenie**  
**Wybierz id, nazwę i cenę produktów z tabeli Product które zawierają w nazwie literę “a”**  
Tabela **Product**:

[9:16](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649488580701629)

SELECT id,name,price FROM product WHERE name LIKE '%a%';

[9:17](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649488638500579)

Operator IN  
Operator IN sprawdza czy wartości z kolumny znajdują się w zbiorze podanym zaraz po klauzuli IN.

SELECT kolumna,kolumna,.. FROM tabela WHERE kolumna IN (11,23,50,20);

[9:18](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649488699191999)

**Przykład**  
**Wybieramy nazwy i ceny produktów których cena wynosi 60 lub 10 lub 25**  
Tabela **Product**:

[9:18](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649488739339779)

SELECT name,price FROM product WHERE price IN (60,10,25);

[9:19](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649488787941429)

**Ćwiczenie**  
**Wybierz nazwę, cenę i liczbę produktów z tabeli Product, których liczba to 3, 4, 5 lub 6**  
Tabela **Product**:

[9:22](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649488943108289)

SELECT name,price,amount FROM product WHERE amount IN (3,4,5,6);

[9:23](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649489011855979)

Operator BETWEEN  
Operator BETWEEN sprawdza czy wartości z kolumny znajdują się w przedziale podanym zaraz po klauzuli BETWEEN.

SELECT kolumna,kolumna,.. FROM tabela WHERE kolumna BETWEEN wartość1 AND wartość2;

[9:24](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649489074669969)

SELECT name,price FROM product WHERE price BETWEEN 40 AND 60;

[9:25](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649489100500459)

Operator IS NULL  
Umożliwia wybranie rekordów ze względu na nie-przypisaną wartość w jednej z jego kolumn.

SELECT kolumna,kolumna,.. FROM tabela WHERE kolumna IS NULL;

[9:27](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649489262516609)

Klauzula TOP  
Pozwala ona ograniczyć ilość wybieranych wierszy do podanej liczby. Umieszcza się ją bezpośrednio po instrukcji SELECT.

SELECT TOP liczba kolumna,kolumna,.. FROM tabela;

Możliwe jest również wybieranie części rekordów na podstawie procentowej.  
Pomiędzy instrukcjami TOP i PERCENT musimy wstawić liczbę określającą ile procent wyników chcemy otrzymać oraz użyć sortowania ORDER BY.

SELECT TOP liczba PERCENT kolumna,kolumna,.. FROM tabela ORDER BY kolumna;

[9:29](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649489385886439)

Odpowiednik klauzuli TOP w MySQL i SQLite to klauzula LIMIT  
Podobnie jak TOP, klauzula LIMIT zwróci nam podaną ilość oczekiwanych wyników, jednak w tym wypadku możemy podać również, od którego wiersza ma się to zwracanie rozpoczynać.

SELECT kolumna FROM tabela LIMIT od którego wiersza,liczba wierszy;

[9:31](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649489497518479)

SELECT id,name,price FROM product WHERE price>20 LIMIT 1;

[9:32](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649489562346089)

**Ćwiczenie**  
**Wybierz id i imię 4 pierwszych klientów korzystając z klauzuli LIMIT**  
Tabela **Customer**:

[9:37](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649489853755179)

Aliasy – zastępcze nazwy kolumn lub tabeli  
Alias to nazwa zastępcza, którą możemy zdefiniować dla konkretnej kolumny lub tabeli na początku zapytania, a następnie używać w całym zapytaniu jako wygodniejsza (zazwyczaj dużo krótsza) nazwa.  
Alias definiuje się używając klauzuli “AS” zaraz po nazwie kolumny lub tabeli w następujący sposób:

SELECT kolumna AS kol FROM tabela AS tab WHERE tab.kol=wartość;

Od momentu takiej definicji do kolumny kolumna wystarczy odwoływać się aliasem kol a tabeli tabela aliasem tab.

[9:41](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649490082394289)

COUNT()  
Zwraca liczbę przekazanych wartości. Jest to po prostu liczba elementów w podanym zbiorze

SELECT COUNT(kolumna) FROM tabela;

Może służyć do zliczenia ilości rekordów w tabeli.

[9:43](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649490200741089)

SELECT COUNT(name) as ilosc FROM product;

[9:45](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649490304298809)

**Przykład**  
**Liczymy ile produktów jest w tabeli Product**  
Tabela **Product**:  
In [9]:

SELECT \* FROM product;

Out[9]:  
idnamepriceamountdate5Spodnie100.052011-02-016Bluza60.022011-02-12  
Zapytanie i wynik zapytania:  
In [12]:

SELECT COUNT(name) AS ilosc FROM product;

Out[12]:  
ilosc2  
SUM()  
Zwraca sumę wartości podanego zbioru

SELECT SUM(kolumna);

Musimy uważać aby argumentem były zawsze liczby.

[9:46](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649490376176809)

SELECT SUM(amount) FROM product;

[9:47](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649490421690209)

**Ćwiczenie**  
**Znajdź sumę cen jednostkowych wszystkich produktów**

[9:48](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649490520860659)

SELECT SUM(price) AS cena\_sumaryczna FROM product;

[9:49](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649490555387209)

AVG()  
Zwraca średnią wartość z podanego zbioru

SELECT AVG(kolumna);

Musimy uważać aby argumentem były zawsze liczby.

[9:50](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649490611949439)

SELECT AVG(price) FROM product;

[9:50](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649490628649899)

SELECT AVG(price) as srednia\_cena FROM product;

[9:52](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649490770573939)

SELECT AVG(amount) FROM product WHERE price>40;

[9:54](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649490871509959)

MIN()  
Zwraca najmniejszą wartość z podanego zbioru

SELECT MIN(kolumna) FROM tabela;

Argumentami mogą być liczby, dane tekstowe i daty z czasem.

[9:54](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649490889400169)

**Przykład**  
**Wybieramy id i cenę produktu o najniższej cenie**

[9:55](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649490937284349)

SELECT id,MIN(price) as najtanszy FROM product;

[9:56](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649490988678769)

MAX()  
Zwraca największą wartość z podanego zbioru

SELECT MAX(kolumna) FROM tabela;

Argumentami mogą być liczby, dane tekstowe i daty z czasem.

[9:58](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649491086895079)

Klauzula DISTINCT  
Zwraca w wyniku rekordy bez powtórzeń

SELECT DISTINCT(kolumna) FROM tabela;

[10:00](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649491257264859)

Użycie w agregatach  
Może być także użyte w agregatach, np.:

SELECT COUNT(DISTINCT kolumna) FROM tabela;

[10:01](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649491309130799)

Użycie w agregatach  
Może być także użyte w agregatach, np.:

SELECT COUNT(DISTINCT kolumna) FROM tabela;

[10:02](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649491325442729)

SELECT COUNT(DISTINCT city) FROM customer;

[10:03](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649491397659839)

Funkcje daty SQL  
Umożliwiają operacje na dacie i czasie w trakcie wykonywania zapytania. Typowe funkcje to:  
DAY(), MONTH(), YEAR()  
Zwracają kolejno dzień, miesiąc, rok z podanej w argumencie daty

SELECT DAY(data);

[10:06](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649491614137799)

Funkcje daty SQLite  
Umożliwiają operacje na dacie i czasie w trakcie wykonywania zapytania. Typowe funkcje to:  
strftime('%d'), strftime('%m'), strftime('%Y')  
Zwracają kolejno dzień, miesiąc, rok z podanej w argumencie daty

SELECT strftime('%d',data);

[10:09](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649491758802819)

SELECT id,name,date FROM customer WHERE STRFTIME('%m',date)='02';

[10:10](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649491857257309)

**Ćwiczenie**  
**Wybierz nazwę, cenę i datę dodania produktów z tabeli Product, których dzień dodania należy do przedziału od 01 do 10**  
Tabela **Product**:

[10:15](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649492140193049)

SELECT name,price,date FROM product WHERE STRFTIME('%d',date) BETWEEN "01" AND "10";

[10:17](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649492249982639)

DATE('now')  
Zwraca bieżącą datę i czas - SQLite

SELECT DATE('now');

[10:17](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649492257992499)

GETDATE()  
Zwraca bieżącą datę i czas - SQL

SELECT GETDATE();

[10:18](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649492325422129)

DATEDIFF()

Zwraca liczbę podanych jednostek pomiędzy podanymi datami - SQL

SELECT DATEDIFF(DAY,'20110210','20110310');

[10:20](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649492438140669)

JULIANDAY()-JULIANDAY()  
Zwraca liczbę podanych dni pomiędzy podanymi datami - SQLite

SELECT JULIANDAY('2011-02-10')-JULIANDAY('2011-03-10');

[10:22](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649492562136919)

Klauzula GROUP BY  
Klauzula GROUP BY umożliwia wykonywanie funkcji grupujących na grupie wartości wyodrębnionych na podstawie wartości innej kolumny.  
Aby użyć kolumny do grupowania, musimy pamiętać, aby umieścić ją w liście kolumn, zaraz po słowie SELECT. W przeciwnym razie nie będzie ona dostępna dla klauzuli grupującej GROUP BY.

SELECT SUM(kolumna),inna\_kolumna FROM tabela GROUP BY inna\_kolumna;

Oblicza agregaty - funkcje obliczane dla zbiorów wierszy.

[10:23](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649492606910789)

Dodatkowa baza danych do zaimportowania w programie DB Browser znajduje się w plikach simple\_library\_db.csv oraz library\_db.sql  
Schemat bazy danych z pliku library\_db.sql znajdziesz w pliku library\_db\_scheme.png

[10:25](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649492752796809)

<https://github.com/pkociepka/sql/blob/master/simple_library_db.csv>

[**simple\_library\_db.csv**](https://github.com/pkociepka/sql/blob/master/simple_library_db.csv)

```  
ID,Title,Author\_surname,Author\_name,Registration\_date  
1,Solaris,Lem,Stanisław,2002-01-30  
2,Solaris,Lem,Stanisław,1986-03-26  
3,Solaris,Lem,Stanisław,2017-03-02  
Show more

[pkociepka/sql](https://github.com/pkociepka/sql) | Added by [GitHub](https://codebrainersds.slack.com/services/B02PQ9FDDFC)

[10:25](https://codebrainersds.slack.com/archives/C036J3BS3H6/p1649492759732979)

<https://github.com/pkociepka/sql/blob/master/library_db.sql>

[**library\_db.sql**](https://github.com/pkociepka/sql/blob/master/library_db.sql)

```  
BEGIN TRANSACTION;  
CREATE TABLE IF NOT EXISTS "users" (  
    "ID"    INTEGER,  
    "surname"    TEXT,  
Show more

[pkociepka/sql](https://github.com/pkociepka/sql) | Added by [GitHub](https://codebrainersds.slack.com/services/B02PQ9FDDFC)

Różnice SQL

<https://troels.arvin.dk/db/rdbms/>

<http://www.galaxy.agh.edu.pl/~pamalino/programowanie/mysql/>

http://zasoby.open.agh.edu.pl/~11smdrobniak/