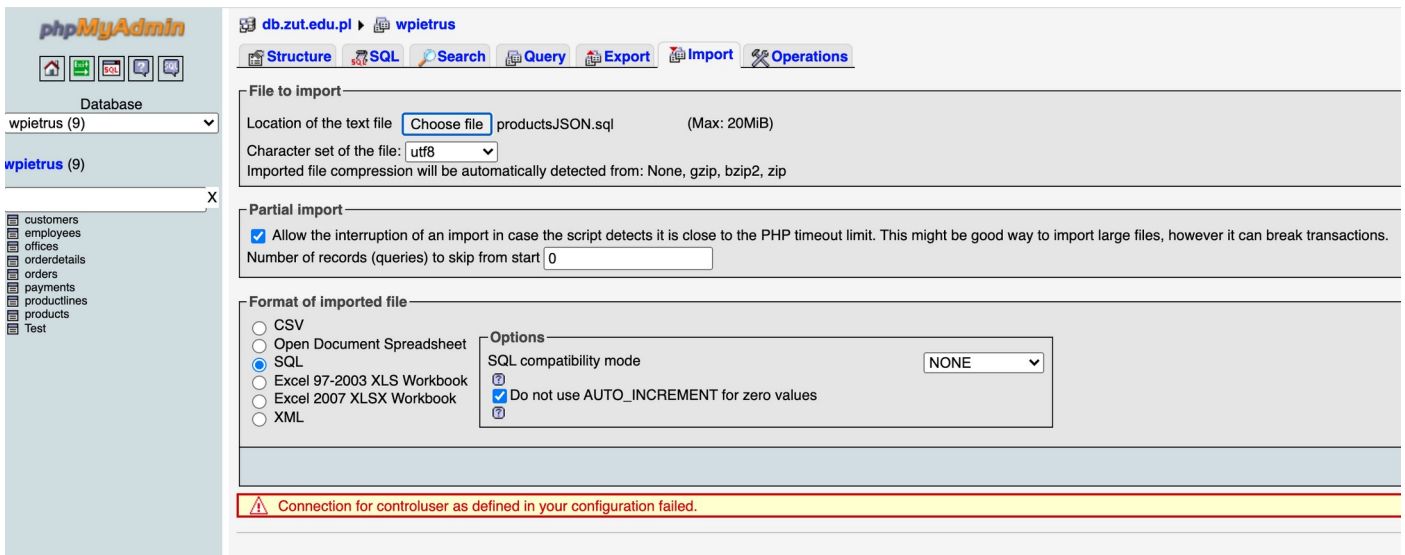


JSON – Java Script Object Notation

1. Zakres ćwiczenia i materiał do wykonania eksperymentów

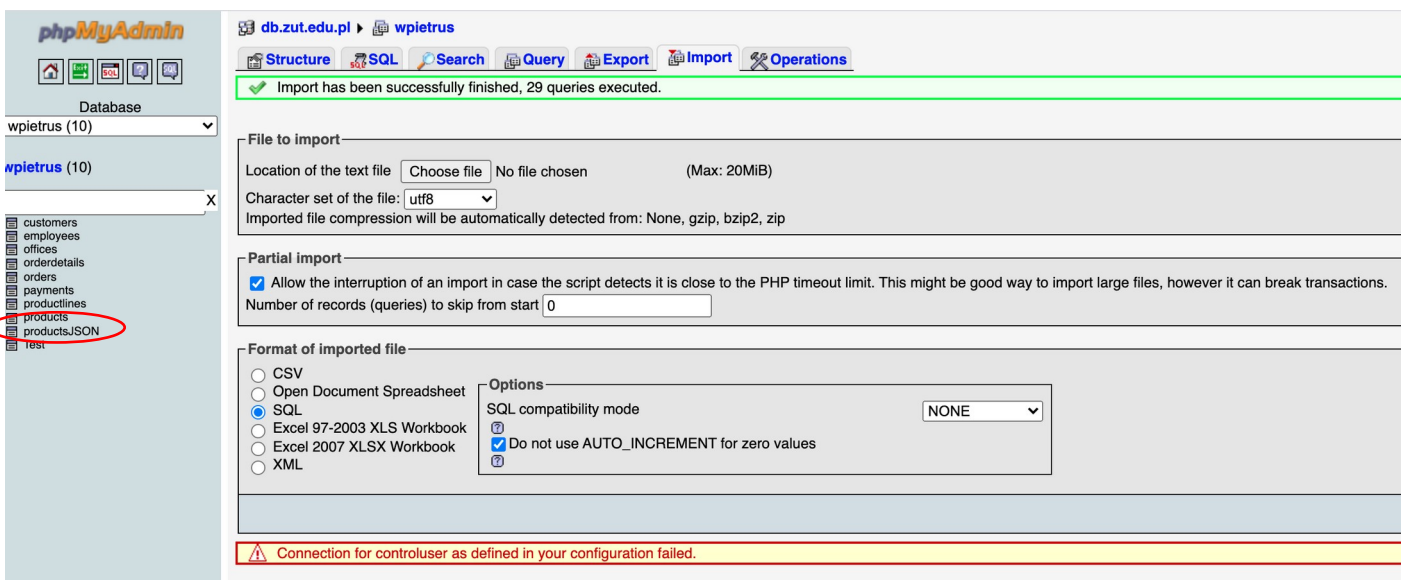
Ta część ćwiczenia laboratoryjnego wykorzystuje dane z bazy używanej we wcześniejszych laboratoriach. Informacje o produktach (tabele products i productlines) zostały zagregowane w jednej tabeli zawierającej struktury JSON.

Aby wykonać ćwiczenie, należy zaimportować plik productsJSON.sql do swojej bazy, w której pojawi się tabela productsJSON (patrz zrzuty ekranu poniżej).



phpMyAdmin interface showing the Import tab. The database selected is 'wpietrus (9)'. The file to import is 'productsJSON.sql'. The format of the imported file is set to 'SQL'. The options section shows 'SQL compatibility mode' set to 'NONE' and 'Do not use AUTO_INCREMENT for zero values' checked. A red error message at the bottom indicates: 'Connection for controluser as defined in your configuration failed.'

Import pliku wykonujemy do bazy używanej we wcześniejszych ćwiczeniach.



phpMyAdmin interface showing the Import tab after successful completion. A green message at the top states: 'Import has been successfully finished, 29 queries executed.' The database selected is 'wpietrus (10)'. The file to import is 'No file chosen'. The format of the imported file is set to 'SQL'. The options section shows 'SQL compatibility mode' set to 'NONE' and 'Do not use AUTO_INCREMENT for zero values' checked. The 'productsJSON' table is highlighted in the left sidebar. A red error message at the bottom indicates: 'Connection for controluser as defined in your configuration failed.'

Po imporcie w bazie powinna pojawić się tabela productsJSON.

The screenshot shows the phpMyAdmin interface for the 'productsJSON' table. The table structure is as follows:

Field	Type	Collation	Attributes	Null	Default	Extra	Action
productCode	varchar(15)	utf8mb4_general_ci		No	None		
productInformation	longtext	utf8mb4_bin		Yes	NULL		

Indexes:

Action	Keyname	Type	Unique	Packed	Field	Cardinality	Collation	Null	Comment
	PRIMARY	BTREE	Yes	No	productCode	110	A		
	productCode_UNIQUE	BTREE	Yes	No	productCode	110	A		

A warning message is displayed: "The indexes PRIMARY and productCode_UNIQUE seem to be equal and one of them could possibly be removed."

Jej wewnętrzna struktura to 2 kolumny (uwaga – obsługa JSON wykonywana jest przez MariaDB na typie tekstowym LONGTEXT, typ JSON jest jego aliasem).

2. Podstawy JSON

JSON jest lekkim formatem wymiany danych (ustandaryzowanym przez RFC 8259) bazującym na podzbiorze języka JavaScript. Nie limituje to zastosowania tego formatu jedynie do oprogramowania bazującego na JavaScript, gdyż z powodu swoich zalet obsługa JSON została wprowadzona do innych popularnych języków programowania. Ze względu na istniejące liczne oprogramowanie wspierające przetwarzanie oraz rozwiązania uzupełniające jego możliwości - JSON Path i JSON Schema (w fazie standaryzacji) - JSON traktowany jest często całościowo jako rodzina technologii.

JSON jest konkurencyjnym do XML formatem wymiany danych między różnymi programami. Oba rozwiązania posiadają istotne podobieństwa m.in.:

- reprezentują dane o strukturze hierarchicznej,
- są neutralne językowo/narzędziowo (możliwe jest ich użycie w różnych językach programowania czy narzędziach),
- samodokumentują strukturę danych wraz z możliwością wykonania manualnej analizy dokumentu (przy ograniczonym stopniu jego komplikacji).

JSON jednak różni się od XML ze względu na to, że:

- nie posiada zdefiniowanej wzorcowej struktury dokumentu, struktura wynika z zawartości przygotowanego dokumentu (JSON Schema w 2021 wciąż jest nieoficjalnym rozwiązaniem) – XML posiada XML Schema opisujące wzorcową strukturę dokumentu (aplikacji XML);
- bezpośrednio dostosowany do użycia w rozwiązaniach Webowych (poprzez obsługę w JavaScript silników przeglądarek) – XML może mieć dodatkowe wymagania (choć jest powszechnie stosowany w licznych bibliotekach dostępnych do wiodących języków programowania) oraz został utworzony poza konkretnym językiem programowania;
- dokument JSON posiada mniejszy rozmiar od XML-owego odpowiednika (używa prostszych znaczników struktury).

Oczywistymi sytuacjami użycia JSON jest import lub eksport takiej formy dostarczanej/oczekiwanej przez inne systemy informatyczne. Jego użycie pozwala na rozwiązanie pewnych ograniczeń relacyjnego modelu BD:

- przechowywanie informacji o danych posiadających odmienną strukturę między rekordami (przypadek 1);
- utworzenie struktury, która w przyszłości może podlegać łatwym modyfikacjom (przypadek 2).

W przypadku 1 użycie modelu relacyjnego wymaga wprowadzenia nadzbioru kolumn pokrywającego wszystkie występujące atrybuty, co przy dużej zmienności struktury wpływa na rozrost poziomy bazy i niskim stopień jej wypełnienia. Natomiast przypadek 2 wymusza częste modyfikacje struktury (oznaczające przegenerowanie tabel i indeksów) lub dodawanie kolumn „na zapas”.

Analizę budowy dokumentów JSON zaczniemy od wyjaśnienia podstawowych elementów składających się na taki format danych. JSON, będący tablicą asocjacyjną języka JavaScript, bazuje na podstawowych elementach:

- "nazwa_wartości": "wartość tekstowa",
"nazwa_wartości": wartość numeryczna
lub "nazwa_obiektu": obiekt/tablica;
- {} definiuje obiekt (może zawierać pola proste lub zagnieżdżone obiekty/tablice);
- [] definiuje tablicę zawierającą obiekt(-y oddzielone przecinkiem ,) .

Przykład JSON – Odpowiedź usług Jednolitego Pliku Kontrolnego (źródło: mf.gov.pl)

```
{
  "ReferenceNumber": " ef7d17780087346e0000004c0c7982ec",
  "TimeoutInSec": 900,
  "RequestToUploadFileList": [
    {
      "BlobName": "094951bc-ba54-404e-b2c8-df2591ad0e17",
      "FileName": "JPK-VAT-TEST-0001.xml.zip.aes",

      "Url": "https://taxdocumentstorage03tst.blob.core.windows.net/ef7d17780087346e0000004c0c7982ec/094951bc-ba54-404e-b2c8-df2591ad0e17?sv=2015-07-08&sr=b&si=ef7d17780087346e0000004c0c7982ec&sig=kN7LlprYkIP9uxod%2F1gcaDGN8WjbEbFDIA4 GXuuz0mk%3D",
      "Method": "PUT",
      "HeaderList": [
        {
          "Key": "Content-MD5",
          "Value": "5YnivEH4gz5Wg5E8M2XwAQ=="
        },
        {
          "Key": "x-ms-blob-type",
          "Value": "BlockBlob"
        }
      ]
    }
  ]
}
```

Tabela używana w tym laboratorium zawiera kolumnę productInformation będącą w postaci JSON. Zawartość przykładowego wiersza znajduje się poniżej. Uwaga: standardowy widok PHPMyAdmin pokazuje jedynie częściową zawartość pól tekstowych (w tym i JSON). Pokazanie całości informacji wymaga włączenia odpowiedniej opcji – pełnego tekstu i aktualizacji wyników.

Showing rows 0 - 0 (~1 total, Query took 0.0003 sec)

SELECT

FROM productsJSON

LIMIT 1

Profiling

Edit

Explain SQL

Create PHP Code

Refresh

Show : 30 row(s) starting from record # 0

in horizontal mode and repeat headers after 100 cells

Options

☐ Full texts
☒ Full texts

☒ Show binary contents
☐ Hide Browser transformation

☐ Show BLOB contents
☒ Show binary contents as HEX

Go

	productCode	productInformation
<input checked="" type="checkbox"/>	S10_1678	<pre>{ "productName": "1969 Harley Davidson Ultimate Chopper", "productLine": "Motorcycles", "productScale": "1:10", "productVendor": "Min Lin Diecast", "productDescription": "This replica features working kickstand, front suspension, gear-shift lever, footbrake lever, drive chain, wheels and steering. All parts are particularly delicate due to their precise scale and require special care and attention.", "quantityInStock": 7933, "buyPrice": 48.81, "MSRP": 95.7, "textDescription": "Our motorcycles are state of the art replicas of classic as well as contemporary motorcycle legends such as Harley Davidson, Ducati and Vespa. Models contain stunning details such as official logos, rotating wheels, working kickstand, front suspension, gear-shift lever, footbrake lever, and drive chain. Materials used include diecast and plastic. The models range in size from 1:10 to 1:50 scale and include numerous limited edition and several out-of-production vehicles. All models come fully assembled and ready for display in the home or office. Most include a certificate of authenticity." }</pre>

Check All / Uncheck All

With selected:

Oficjalna dokumentacja JSON dostępna jest pod <https://www.json.org/json-en.html>

3. Funkcje MariaDB obsługujące JSON

Lista funkcji MariaDB dot. funkcji obsługujących JSON wraz z ich opisem oraz przykładami dostępna jest pod <https://mariadb.com/kb/en/json-functions/>

Uwaga: Proszę zwrócić uwagę na różnice w działaniu funkcji JSON_EXTRACT i JSON_VALUE.

```
SET @SampleJSON='{ "sampleText": "123ABC", "sampleNumber": 3.14 }';
SELECT
  JSON_EXTRACT(@SampleJSON, '$.sampleText'),
  JSON_EXTRACT(@SampleJSON, '$.sampleNumber'),
  JSON_VALUE(@SampleJSON, '$.sampleText'),
  JSON_VALUE(@SampleJSON, '$.sampleNumber');
```

JSON_EXTRACT(@SampleJSON, '\$.sampleText')	"123ABC"
JSON_EXTRACT(@SampleJSON, '\$.sampleNumber')	3.14
JSON_VALUE(@SampleJSON, '\$.sampleText')	123ABC
JSON_VALUE(@SampleJSON, '\$.sampleNumber')	3.14

4. Podstawy JSONPath

Operacje na dokumencie JSON wymagające podania ścieżki korzystają z wyrażeń podawanych w postaci JSONPath. Wyrażenia te bazują na 3 grupach elementów.

I. Operatorach

Operator	Znaczenie
\$	Główny element (root) zapytania, rozpoczyna wszystkie wyrażenia
@	Węzeł bieżący, przetwarzany przez filtr
*	Wzorzec uniwersalny
..	Głębokie skanowanie (dotyczy wszystkich potomków węzła)

.<nazwa>	Węzeł-dziecko (notacja kropkowa)
['<nazwa>' (, '<nazwa>')]	Węzeł(y)-dzieko(ci) (notacja nawiasowa)
[<numer> (, <numer>)]	Indeks(y) tablicy (liczone od 0)
[początek:koniec(:krok)]	Wycinek tablicy (wyłączając koniec)
?(<wyrażenie>)	Wyrażenie filtrujące generujące wartość logiczną

II. Filtrach

Operator	Znaczenie
==	Lewa strona jest równa prawej (również występuje zgodność typów)
!=	Lewa strona nie jest równa prawej
<	Lewa strona jest mniejsza od prawej
<=	Lewa strona jest równa lub mniejsza od prawej
>	Lewa strona jest większa od prawej
>=	Lewa strona jest równa lub większa od prawej
=~	Lewa strona jest odpowiada wyrażeniu regularnemu z prawej
in	Lewa strona występuje w prawej
nin	Lewa strona nie występuje w prawej
subsetof	Lewa strona jest podzbiorem prawej
anyof	Lewa strona ma część wspólną z prawą
noneof	Lewa strona nie ma części wspólnej z prawą
size	Rozmiar lewej strony odpowiada prawej
empty	Lewa strona jest pusta

III. Funkcjach

Fukcja	Opis
min()	Minimalna wartość elementów tablicy numerycznej
max()	Maksymalna wartość elementów tablicy numerycznej
avg()	Średnia wartość elementów tablicy numerycznej
stddev()	Odchylenie standardowe wartości elementów tablicy numerycznej
length()	Długość tablicy
sum()	Suma elementów tablicy numerycznej
first()	
keys()	Zwraca klucze elementu (inna forma to użycie ~)
concat(X)	Zwraca złączenie elementów
append(X)	Dodaje nowy element do tablicy

UWAGA: Wsparcie oferowane przez serwer MariaDB (bieżąca wersja 10.5.10) technologii JSON (w tym JSONPath) jest w fazie rozwoju. Nie są dostępne wszystkie elementy JSONPath, a implementacja obsługi JSON różni się częściowo od tej dostępnej w MySQL. Obsługa JSON przez SQL jest w fazie ustandaryzowania i należy spodziewać się jej ujednolicania w różnych mechanizmach baz relacyjnych.

5. Przykładowe zadania

Problem 1: Podstawowe odczytanie informacji z JSON - pokazanie informacji o nazwie i linii produktu.

```
SELECT
    productCode,
    JSON_VALUE(productInformation, '$.productName') AS productName,
    JSON_VALUE(productInformation, '$.productLine') AS productLine
FROM
    productsJSON;
```

productCode	productName	productLine
S10_1678	1969 Harley Davidson Ultimate Chopper	Motorcycles
S10_1949	1952 Alpine Renault 1300	Classic Cars
S10_2016	1996 Moto Guzzi 1100i	Motorcycles
S10_4698	2003 Harley-Davidson Eagle Drag Bike	Motorcycles
S10_4757	1972 Alfa Romeo GTA	Classic Cars
S10_4962	1962 LanciaA Delta 16V	Classic Cars
S12_1099	1968 Ford Mustang	Classic Cars
S12_1108	2001 Ferrari Enzo	Classic Cars
S12_1666	1958 Setra Bus	Trucks and Buses
S12_2823	2002 Suzuki XREO	Motorcycles
S12_3148	1969 Corvair Monza	Classic Cars
S12_3380	1968 Dodge Charger	Classic Cars
S12_3891	1969 Ford Falcon	Classic Cars
S12_3990	1970 Plymouth Hemi Cuda	Classic Cars
S12_4473	1957 Chevy Pickup	Trucks and Buses
S12_4675	1969 Dodge Charger	Classic Cars
S18_1097	1940 Ford Pickup Truck	Trucks and Buses
S18_1129	1993 Mazda RX-7	Classic Cars
S18_1342	1937 Lincoln Berline	Vintage Cars
S18_1367	1936 Mercedes-Benz 500K Special Roadster	Vintage Cars
S18_1589	1965 Aston Martin DB5	Classic Cars
S18_1662	1980s Black Hawk Helicopter	Planes
S18_1749	1917 Grand Touring Sedan	Vintage Cars
S18_1889	1948 Porsche 356-A Roadster	Classic Cars
S18_1984	1995 Honda Civic	Classic Cars
S18_2238	1998 Chrysler Plymouth Prowler	Classic Cars
S18_2248	1911 Ford Town Car	Vintage Cars
S18_2319	1964 Mercedes Tour Bus	Trucks and Buses
S18_2325	1932 Model A Ford J-Coupe	Vintage Cars
S18_2432	1926 Ford Fire Engine	Trucks and Buses

Problem 2: Odczytanie informacji z JSON z użyciem jej - pokazanie nazwy i linii produktu pochodzących od "Classic Metal Creations".

```
SELECT
    productCode,
    JSON_VALUE(productInformation, '$.productVendor') AS productVendor,
    JSON_VALUE(productInformation, '$.productName') AS productName,
    JSON_VALUE(productInformation, '$.productLine') AS productLine
FROM
    productsJSON
WHERE
    JSON_VALUE(productInformation, '$.productVendor') = 'Classic Metal Creations';
```

productCode	productVendor	productName	productLine
S10_1949	Classic Metal Creations	1952 Alpine Renault 1300	Classic Cars
S18_1589	Classic Metal Creations	1965 Aston Martin DB5	Classic Cars
S18_4721	Classic Metal Creations	1957 Corvette Convertible	Classic Cars
S24_1785	Classic Metal Creations	1928 British Royal Navy Airplane	Planes
S24_2022	Classic Metal Creations	1938 Cadillac V-16 Presidential Limousine	Vintage Cars
S24_2766	Classic Metal Creations	1949 Jaguar XK 120	Classic Cars
S24_3856	Classic Metal Creations	1956 Porsche 356A Coupe	Classic Cars
S24_4620	Classic Metal Creations	1961 Chevrolet Impala	Classic Cars
S32_2509	Classic Metal Creations	1954 Greyhound Scenicruiser	Trucks and Buses
S50_1514	Classic Metal Creations	1962 City of Detroit Streetcar	Trains

Problem 3: Łączenie informacji z JSON z informacjami pochodzącymi z innych tabel - podanie sumy wszystkich zamówionych modeli "1965 Aston Martin DB5".

```
SELECT
    SUM(quantityOrdered)
FROM
    orderdetails,
    (SELECT
        productCode
    FROM
        productsJSON
    WHERE
        JSON_VALUE(productInformation, '$.productName') = '1965 Aston Martin DB5') AS
    selectedproduct
WHERE
    orderdetails.productCode = selectedproduct.productCode;
```

SUM(quantityOrdered)
914

Problem 4: Modyfikacja struktury JSON - dodanie atrybutu material do wybranych produktów (produkt S24_2011 = "Wood", produkt S18_2581 = "Metal")

```
UPDATE productsJSON
SET
    productInformation = JSON_INSERT(productInformation, '$.material', 'Wood')
WHERE
    productCode = 'S24_2011';
UPDATE productsJSON
SET
    productInformation = JSON_INSERT(productInformation, '$.material', 'Metal')
WHERE
    productCode = 'S18_2581';
```

```
UPDATE productsJSON SET productInformation = JSON_DETACHED(JSON_INSERT(productInformation, '$.material', 'Wood')) WHERE productCode = 'S24_2011';# 1 row(s) affected.

UPDATE productsJSON SET productInformation = JSON_DETACHED(JSON_INSERT(productInformation, '$.material', 'Metal')) WHERE productCode = 'S18_2581';# 1 row(s) affected.
```

Problem 5: Wyszukanie wierszy posiadających wybrany atrybut - znalezienie produktów posiadających informacje o materiale.

```
SELECT
    *
FROM
    productsJSON
WHERE
    JSON_CONTAINS_PATH(productInformation, 'one', '$.material');
```

			productCode	productInformation
<input type="checkbox"/>			S18_2581	{ "productName": "P-51-D Mustang", "produc...
<input type="checkbox"/>			S24_2011	{ "productName": "18th century schooner", ...

Problem 6: Przygotowanie dla każdego klienta (podać nazwę) tablicy JSON (customerPayments) zawierającej informację o dokonanych płatnościach (data i kwota) – informacje należy zapisać w nowej tabeli customerspayments.

```
CREATE TABLE customerspayments AS (
    SELECT
        customerName,
        CONCAT( '[',
            GROUP_CONCAT( JSON_OBJECT( 'paymentDate',
                                    paymentDate,
                                    'amount',
                                    amount )
                            SEPARATOR ",\n" ),
            ']' ) AS customerPayments
    FROM
        customers,
        payments
    WHERE
        customers.customerNumber = payments.customerNumber
    GROUP BY customers.customerNumber );
```

customerName	customerPayments
Atelier graphique	[{"paymentDate": "2004-10-19", "amount": 6066.78}, {"paymentDate": "2003-06-05", "amount": 14571.44}, {"paymentDate": "2004-12-18", "amount": 1676.14}]
Signal Gift Stores	[{"paymentDate": "2004-12-17", "amount": 14191.12}, {"paymentDate": "2003-06-06", "amount": 32641.98}, {"paymentDate": "2004-08-20", "amount": 33347.88}]
Australian Collectors, Co.	[{"paymentDate": "2003-05-20", "amount": 45864.03}, {"paymentDate": "2004-12-15", "amount": 82261.22}, {"paymentDate": "2003-05-31", "amount": 7565.08}, {"paymentDate": "2004-03-10", "amount": 44894.74}]
La Rochelle Gifts	[{"paymentDate": "2004-11-14", "amount": 19501.82}, {"paymentDate": "2004-08-08", "amount": 47924.19}, {"paymentDate": "2003-06-08", "amount": 45556.01}]

Uwaga: W nowszych wersjach MariaDB (od 10.5.0) dostępna jest funkcja JSON_ARRAYAGG.

6. Zadania do wykonania (rozwiązania w punkcie 8)

Używając tabeli productsJSON przygotuj zapytania:

Zadanie 1. Pokaż informacje o nazwie, stanie magazynowym i cenie zakupu produktów (wskazówka – JSON_VALUE/JSON_EXTRACT).

Zadanie 2. Pokaż wszystkie produkty o cenie zakupu wyższej od 100 (wskazówka – JSON_VALUE/JSON_EXTRACT).

Zadanie 3. Podaj liczbę produktów w każdej kategorii produktów ((wskazówka – JSON_VALUE/JSON_EXTRACT + COUNT).

Zadanie 4. Do wybranych produktów dodaj atrybut numeryczny length zawierający długość produktu (produkt S10_1949 = 38,5; produkt S18_1589 = 25,0) (wskazówka – JSON_INSERT).

Zadanie 5. Znajdź wszystkie produkty posiadające informacje o ich długości (wskazówka – JSON_CONTAINS_PATH).

Zadanie 6. Przygotuj tablicę JSON zawierającą informację o pracownikach każdego z oddziałów. Podaj lokalizację oddziału (miasto) oraz imię i nazwisko każdego z pracowników tam pracujących (wskazówka – JSON_OBJECT) – zastawienie zapisz w nowej tabeli officeemployees.

7. Uwagi dot. zastosowania JSON w relacyjnych bazach danych

JSON jest uniwersalną formą zapisu danych ustrukturyzowanych w postaci hierarchicznej. Ze względu na swoją elastyczność może zostać użyty w bazach danych do przechowywania informacji o zmiennej strukturze, dając nowe możliwości wykraczające poza tradycyjny model relacyjny. Należy jednak pamiętać, że:

- nadmiernie wykorzystując JSON w bazie relacyjnej odchodzimy od tego modelu organizacji bazy danych (denormalizujemy model);
- relacyjne bazy są zoptymalizowane do tego modelu danych i użycie JSON (w szczególności operując na jego zawartości) wpływa na wydajność zapytań;
- język SQL został utworzony z myślą o atrybutach zdefiniowanych w bazie (układ tabelaryczno-kolumnowy), a nie ukrytych w środku innych struktur;

- standardowo tabela bazy relacyjnej nie indeksuje pól zawartych w kolumnach JSON (możliwość dostępna w niektórych RDBMSach obsługujących natywny typ JSON) oraz nie może użyć ich jako kluczy (rozwiązaniem pośrednim jest użycie kolumn generowanych/wirtualnych);
- jeżeli brak jest stabilnej struktury bazy (różni się między rekordami lub zmienia się w czasie) lepiej jest od razu rozważyć użycie NoSQL (element kursu Zarządzanie informacją 2).

8. Przykładowe rozwiązania zadań

Zadanie 1.

```
SELECT
    productCode,
    JSON_VALUE(productInformation, '$.productName') AS productName,
    JSON_VALUE(productInformation, '$.quantityInStock') AS quantityInStock,
    JSON_VALUE(productInformation, '$.buyPrice') AS buyPrice
FROM
    productsJSON;
```

Zadanie 2.

```
SELECT
    productCode,
    JSON_VALUE(productInformation, '$.productName') AS productName,
    JSON_VALUE(productInformation, '$.buyPrice') AS buyPrice
FROM
    productsJSON
WHERE
    JSON_VALUE(productInformation, '$.buyPrice') > 100;
```

Zadanie 3.

```
SELECT
    JSON_VALUE(productInformation, '$.productLine') AS productLine,
    COUNT(*) AS productsInLine
FROM
    productsJSON
GROUP BY JSON_VALUE(productInformation, '$.productLine');
```

Zadanie 4.

```
UPDATE productsJSON
SET
    productInformation = JSON_INSERT(productInformation, '$.length', 38.5)
WHERE
    productCode = 'S10_1949';
UPDATE productsJSON
SET
    productInformation = JSON_INSERT(productInformation, '$.length', 25.0)
WHERE
    productCode = 'S18_1589';
```

Zadanie 5.

```
SELECT
    *
FROM
    productsJSON
WHERE
    JSON_CONTAINS_PATH(productInformation, 'one', '$.length');
```

Zadanie 6.

```
CREATE TABLE officeemployees AS (
    SELECT
        city,
        CONCAT('[',
            GROUP_CONCAT(JSON_OBJECT('firstName',
                                    firstName,
                                    'lastName',
                                    lastName)
                            SEPARATOR ",\n"),
```

```
        ']) AS officeEmployees
FROM
    offices,
    employees
WHERE
    offices.officeCode = employees.officeCode
GROUP BY offices.officeCode);
```