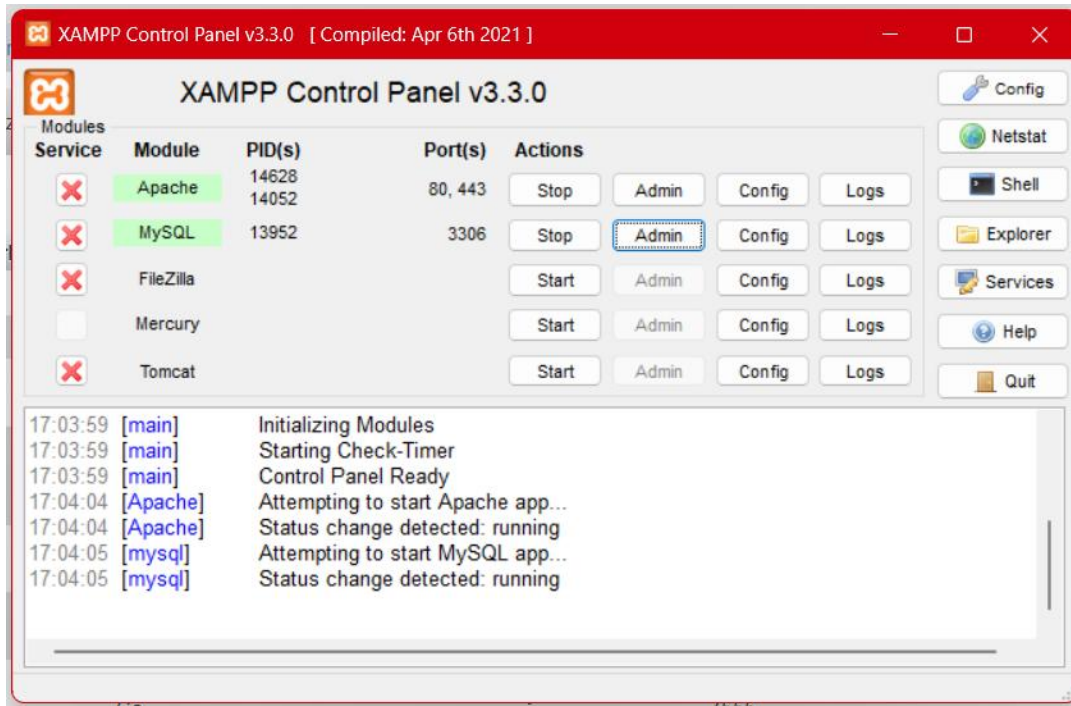
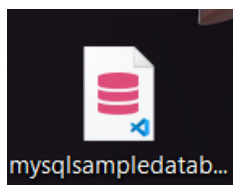


Sprawozdanie bazy danych 1

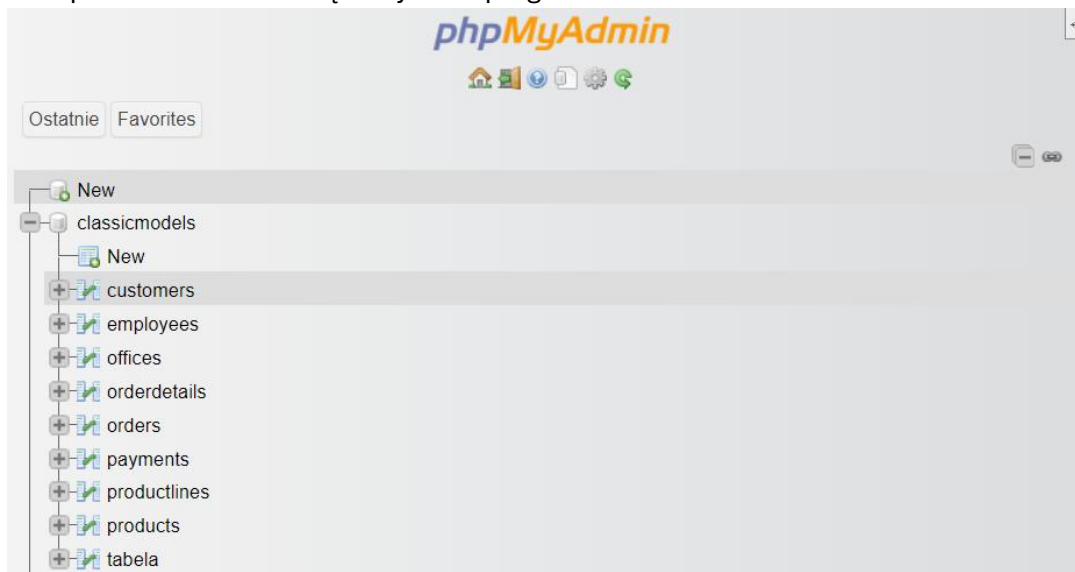
- 1) Pobrałem i zapoznałem się z wybranymi systemami zarządzania relacyjną bazą danych



- 2) Pobrałem z Internetu przykładową bazę danych



- 3) Zaimplementowałem bazę danych do programu



✓ Pokazano wiersze 0 - 24 (122 total, Wykonanie zapytania trwało 0,0010 sekund(y))

SELECT * FROM `customers`

1 > >> Show all Liczba wierszy: 25 Filter rows: Przeszukaj tę tabelę Sortuj wg kłucza: Zaden

Extra options

	customerNumber	customerName	contactLastName	contactFirstName	phone	addressLine1	addressLine2	city	state	postalCode	country	salesRepEmployeeNumber	credit
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	103	Atelier graphique	Schmitt	Carine	40 32 2555	54, rue Royale	NULL	Nantes	NULL	44000	France	1370	2'
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	112	Signal Gift Stores	King	Jean	7025551838	8489 Strong St.	NULL	Las Vegas	NV	83030	USA	1166	7'
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	114	Australian Collectors, Co.	Ferguson	Peter	03 9520 4555	636 St Kilda Road	Level 3	Melbourne	Victoria	3004	Australia	1611	11'
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	119	La Rochelle Gifts	Labruno	Janine	40 67 8555	67, rue des Cinquante Otages	NULL	Nantes	NULL	44000	France	1370	11'
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	121	Baane Mini Imports	Bergulfsen	Jonas	07 98 9555	Erling Skakkes gate 78	NULL	Stavern	NULL	4110	Norway	1504	8'
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	124	Mini Gifts Distributors Ltd.	Nelson	Susan	4155551450	5677 Strong St.	NULL	San Rafael	CA	97562	USA	1165	21'
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	125	Havel & Zbyszek	Piestrzeniewicz	Zbyszek	(26) 642-7555	ul. Filtrów 68	NULL	Warszawa	NULL	01-012	Poland	NULL	
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	128	Blauer See Auto, Co.	Keitel	Roland	+49 69 66 90 2555	Lyonerstr. 34	NULL	Frankfurt	NULL	60528	Germany	1504	5'
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	129	Mini Wheels Co.	Murphy	Julie	6505555787	5557 North Pendale Street	NULL	San Francisco	CA	94217	USA	1165	6'
<input type="checkbox"/> Edit <input type="checkbox"/> Copy <input type="checkbox"/> Delete	131	Land of Toys Inc.	Lee	Kwai	2125557818	897 Long Airport Avenue	NULL	NYC	NY	10022	USA	1323	11'

- 4) Wykonałem na pliku polecenia podane w zadaniu.

```

Console
Wciśnij Ctrl+Enter aby wykonać zapytanie

> SELECT * FROM `customers`
> SELECT * FROM `customers`
> SELECT * FROM `customers`
> SELECT * FROM `customers`
> CREATE TABLE przykład( kolumna1 int, kolumna2 varchar(255), kolumna3 int );
> ALTER TABLE przykład ADD kolumna4 int;
> ALTER TABLE przykład DROP kolumna4;
> DROP TABLE tabela;
>

```

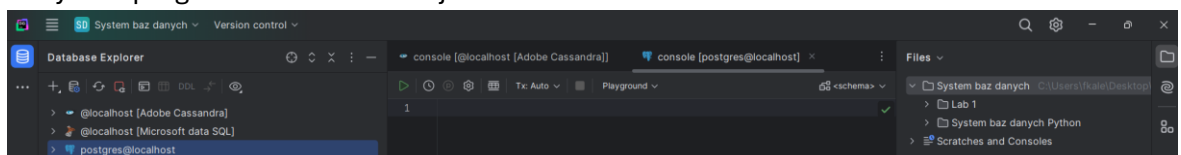
- 5) Połączyłem się do bazy danych za pomocą języka python i wykonałem proste zapytanie typu SELECT

```

Lab 1.py
1 import mysql.connector
2
3 conn = mysql.connector.connect(
4     host="localhost",
5     user="root",
6     database="classicmodels"
7 )
8
9
10 cursor = conn.cursor()
11
12 cursor.execute("SELECT phone FROM `customers`;")
13
14 results = cursor.fetchall()
15
16 for row in results:
17     print(row)
18
19 conn.close()
20

```

- 6*) Zainstalowałem również i zapoznałem się z 3 innymi systemami zarządzania bazą danych w programie DATAGRIP od JetBrains.



Sprawozdanie bazy danych 2

- 1) Stworzyłem tabelę „world” z atrybutami: country, population, continent

```
1 CREATE TABLE world (  
2   country varchar(255),  
3   population int(255),  
4   continent varchar(255)  
5 );
```

Dodałem do niej 20 państw i informacje o jej ludności i kontynencie,

```
1 INSERT INTO world (country, population, continent) VALUES  
2 ('Chiny', 1444216107, 'Azja'),  
3 ('Indie', 1393409038, 'Azja'),  
4 ('Stany Zjednoczone', 331893745, 'Ameryka Północna'),  
5 ('Indonezja', 276361783, 'Azja'),  
6 ('Pakistan', 225199937, 'Azja'),  
7 ('Nigeria', 211400708, 'Afryka'),  
8 ('Rosja', 145912025, 'Europa/Azja'),  
9 ('Meksyk', 130262216, 'Ameryka Północna'),  
10 ('Japonia', 125836021, 'Azja'),  
11 ('Egipt', 104258327, 'Afryka'),  
12 ('Grecja', 10724599, 'Europa'),  
13 ('Portugalia', 10305564, 'Europa'),  
14 ('Czechy', 10707897, 'Europa'),  
15 ('Boliwia', 11833158, 'Ameryka Południowa'),  
  
16 ('Kuba', 11326616, 'Ameryka Północna'),  
17 ('Tunezja', 11971805, 'Afryka'),  
18 ('Słowacja', 5463899, 'Europa'),  
19 ('Norwegia', 5431086, 'Europa'),  
20 ('Kostaryka', 5188262, 'Ameryka Północna'),  
21 ('Irlandia', 5023107, 'Europa');
```

- 2) Posegregowałem je malejąco według wielkości populacji

```
SELECT * FROM `world` ORDER BY population DESC;
```

☐ Profilowanie [[Edytuj w linii](#)] [[Edit](#)] [[Explain SQL](#)] [[Create](#)]

☐ Show all | Liczba wierszy: Filter rows

Extra options

country	population	continent
Chiny	1444216107	Azja
Indie	1393409038	Azja
Stany Zjednoczone	331893745	Ameryka Północna
Indonezja	276361783	Azja
Pakistan	225199937	Azja
Nigeria	211400708	Afryka
Rosja	145912025	Europa/Azja
Meksyk	130262216	Ameryka Północna
Japonia	125836021	Azja
Egipt	104258327	Afryka
Tunezja	11971805	Afryka
Boliwia	11833158	Ameryka Południowa
Kuba	11326616	Ameryka Północna
Grecja	10724599	Europa
Czechy	10707897	Europa
Portugalia	10305564	Europa
Słowacja	5463899	Europa
Norwegia	5431086	Europa
Kostaryka	5188262	Ameryka Północna
Irlandia	5023107	Europa

- 3) Rekord z najmniejszą populacją to Irlandia więc zmieniłem jej populację na 20 000 000

```
1 UPDATE world
2 SET population = 20000000
3 WHERE country = "Irlandia"
```

- 4) Wyszukałem rekordy z populacją poniżej 20 milionów

✔ Pokazano wiersze 0 - 8 (9 total, Wykonanie zapytania trwało 0,0013 sekund(y).)

```
SELECT * FROM `world` WHERE population < 20000000;
```

☐ Profilowanie [[Edytuj w linii](#)] [[Edit](#)] [[Explain SQL](#)] [[Create](#)]

☐ Show all | Liczba wierszy: 25 ▼ Filter rows:

Extra options

country	population	continent
Grecja	10724599	Europa
Portugalia	10305564	Europa
Czechy	10707897	Europa
Boliwia	11833158	Ameryka Południowa
Kuba	11326616	Ameryka Północna
Tunezja	11971805	Afryka
Słowacja	5463899	Europa
Norwegia	5431086	Europa
Kostaryka	5188262	Ameryka Północna

- 5) Usunąłem rekordy z populacją powyżej 20 milionów

✔ 10 rows deleted. (Wykonanie zapytania trwało 0,0013 sekund(y).)

```
DELETE FROM world WHERE population > 20000000;
```

- 6) Tabela na sam koniec ma takie rekordy

```
SELECT * FROM `world`
```

☐ Profilowanie [[Edytuj w linii](#)] [[Edit](#)] [[Explain](#)]

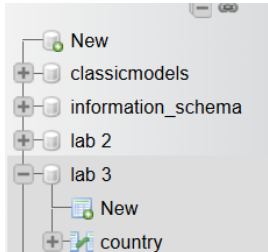
☐ Show all | Liczba wierszy: 25 ▼

Extra options

country	population	continent
Grecja	10724599	Europa
Portugalia	10305564	Europa
Czechy	10707897	Europa
Boliwia	11833158	Ameryka Południowa
Kuba	11326616	Ameryka Północna
Tunezja	11971805	Afryka
Słowacja	5463899	Europa
Norwegia	5431086	Europa
Kostaryka	5188262	Ameryka Północna
Irlandia	20000000	Europa

Sprawozdanie bazy danych 3

- 1) Stworzyłem tabelę „country” z wartościami podanymi w poleceniu



✓ MySQL zwrócił pusty wynik (zero wierszy). (Wykonanie zapytania trwało 0,0005 sekund(y).)

```
CREATE TABLE country ( Code CHAR(3) PRIMARY KEY NOT NULL DEFAULT 'UNK', Name VARCHAR(255) NOT NULL DEFAULT '', Code2 CHAR(2) NOT NULL DEFAULT '' UNIQUE );
```

[Edytuj w linii] [Edit] [Create PHP code]

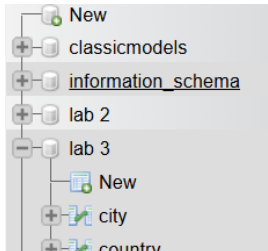
- 2) Zapewniłem aby kody znajdujące się w country. Code składały się z minimum trzech znaków

✓ MySQL zwrócił pusty wynik (zero wierszy). (Wykonanie zapytania trwało 0,0006 sekund(y).)

```
ALTER TABLE country ADD CONSTRAINT chk_code_length CHECK (LENGTH(Code) >=3);
```

[Edytuj w linii] [Edit] [Create PHP code]

- 3) Stworzyłem tabelę „city” z wartościami podanymi w poleceniu



✓ MySQL zwrócił pusty wynik (zero wierszy). (Wykonanie zapytania trwało 0,0008 sekund(y).)

```
CREATE TABLE city ( ID INT NOT NULL PRIMARY KEY, Name VARCHAR(255) NOT NULL DEFAULT '', CountryCode CHAR(3) NOT NULL DEFAULT '', District VARCHAR(255) NOT NULL DEFAULT '', Info JSON DEFAULT NULL );
```

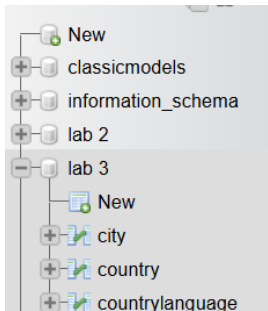
[Edytuj w linii] [Edit] [Create PHP code]

- 4) Dodałem klucz obcy do tabeli country związany z kolumną city.ID

✓ MySQL zwrócił pusty wynik (zero wierszy). (Wykonanie zapytania trwało 0,0014 sekund(y).)

```
ALTER TABLE country ADD Capital INT DEFAULT NULL, ADD CONSTRAINT fk_capital FOREIGN KEY (Capital) REFERENCES city(ID);
```

- 5) Stworzyłem tabelę „countrylanguage” z wartościami podanymi w poleceniu



✓ MySQL zwrócił pusty wynik (zero wierszy). (Wykonanie zapytania trwało 0,0008 sekund(y).)

```
CREATE TABLE countrylanguage ( CountryCode CHAR(3) NOT NULL DEFAULT '', Language CHAR(30) NOT NULL DEFAULT '', IsOfficial ENUM('T', 'F') NOT NULL DEFAULT 'F', Percentage DOUBLE PRECISION NOT NULL DEFAULT 0.0, PRIMARY KEY (CountryCode, Language), FOREIGN KEY (CountryCode) REFERENCES country(Code));
```

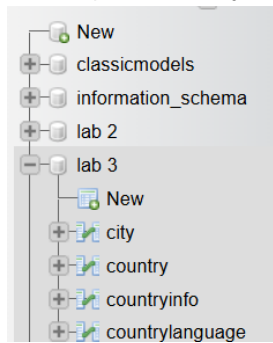
[Edytuj w linii] [Edit] [Create PHP code]

- 6) Stworzyłem index dla countrylanguage.CountryCode

✓ MySQL zwrócił pusty wynik (zero wierszy). (Wykonanie zapytania trwało 0,0008 sekund(y).)

```
CREATE INDEX idx_countrycode ON countrylanguage(CountryCode);
```





7) Stworzyłem tabelę „countryinfo” z wartościami podanymi w poleceniu



```
✓ MySQL zwrócił pusty wynik (zero wierszy). (Wykonanie zapytania trwało 0,0006 sekund(y).)
```

```
CREATE TABLE countryinfo ( _id VARCHAR(32) NOT NULL PRIMARY KEY DEFAULT (UUID()), doc  
JSON );
```

[Edytuj w linii] [Edit] [Create PHP code]

- | Nazwa | Stan | Data modyfikacji | Typ | Rozmiar |
|---|---|------------------|--------------------|---------|
|  initWorld |  | 11.11.2024 23:04 | JetBrains DataGrip | 7 KB |
|  Sprawozdanie Baz danych lab 4 |  | 11.11.2024 23:12 | Dokument progra... | 15 KB |

INSERT INTO country (Code, Name, Code2) VALUES ('POL', 'Polska', 'PL'), ('SWE', 'Szwecja', 'SE'), ('NOR', 'Norwegia', 'NO'), ('FIN', 'Finlandia', 'FI'), ('GRC', 'Grecja', 'GR'), ('CZE', 'Czechy', 'CZ'), ('IRL', 'Irlandia', 'IE'), ('PRT', 'Portugalia', 'PT'), ('AUT', 'Austria', 'AT'), ('BEL', 'Belgia', 'BE'), ('DNK', 'Dania', 'DK'), ('FRA', 'Francja', 'FR'), ('DEU', 'Niemcy', 'DE'), ('ESP', 'Hiszpania', 'ES'), ('ITA', 'Włochy', 'IT'), ('USA', 'Stany Zjednoczone', 'US'), ('CAN', 'Kanada', 'CA'), ('MEX', 'Meksyk', 'MX'), ('CUB', 'Kuba', 'CU'), ('HND', 'Honduras', 'HN'), ('BRA', 'Brazylia', 'BR'), ('ARG', 'Argentyna', 'AR'), ('CHL', 'Chile', 'CL'), ('COL', 'Kolumbia', 'CO'), ('PER', 'Peru', 'PE'), ('CHN', 'Chiny', 'CN'), ('JPN', 'Japonia', 'JP'), ('IND', 'Indie', 'IN'), ('SAU', 'Arabia Saudyjska', 'SA'), ('KOR', 'Korea Południowa', 'KR');

```
INSERT INTO city (ID, Name, CountryCode, District) VALUES (1, 'Warszawa', 'POL', 'Mazowieckie'), (2, 'Sztokholm', 'SWE', 'Sztokholm'), (3, 'Oslo', 'NOR', 'Oslo'), (4, 'Helsinki', 'FIN', 'Helsinki'), (5, 'Ateny', 'GRC', 'Attika'), (6, 'Praga', 'CZE', 'Praga'), (7, 'Dublin', 'IRL', 'Leinster'), (8, 'Lizbona', 'PRT', 'Lizbona'), (9, 'Wiedeń', 'AUT', 'Wiedeń'), (10, 'Bruksela', 'BEL', 'Bruksela'), (11, 'Kopenhaga', 'DNK', 'Hovedstaden'), (12, 'Paryż', 'FRA', 'Île-de-France'), (13, 'Berlin', 'DEU', 'Berlin'), (14, 'Madryt', 'ESP', 'Madryt'), (15, 'Rzym', 'ITA', 'Lacjum'), (16, 'Waszyngton', 'USA', 'Dystrykt Kolumbii'), (17, 'Ottawa', 'CAN', 'Ontario'), (18, 'Meksyk', 'MEX', 'Dystrykt Federalny'), (19, 'Hawana', 'CUB', 'Hawana'), (20, 'Tegucigalpa', 'HND', 'Francisco Morazán'), (21, 'Brasília', 'BRA', 'Dystrykt Federalny'), (22, 'Buenos Aires', 'ARG', 'Buenos Aires'), (23, 'San[])
```

```
INSERT INTO countrylanguage (CountryCode, Language, IsOfficial, Percentage) VALUES ('POL', 'Polski', 'T', 98.0), ('SWE', 'Szwedzki', 'T', 90.0), ('NOR', 'Norweski', 'T', 95.0), ('FIN', 'Fiński', 'T', 88.0), ('GRC', 'Grecki', 'T', 98.0), ('CZE', 'Czeski', 'T', 96.0), ('IRL', 'Angielski', 'T', 60.0), ('IRL', 'Irlandzki', 'T', 40.0), ('PRT', 'Portugalski', 'T', 100.0), ('AUT', 'Niemiecki', 'T', 98.0), ('BEL', 'Francuski', 'T', 40.0), ('BEL', 'Niderlandzki', 'T', 60.0), ('DNK', 'Duński', 'T', 100.0), ('FRA', 'Francuski', 'T', 100.0), ('DEU', 'Niemiecki', 'T', 100.0), ('ESP', 'Hiszpański', 'T', 90.0), ('ITA', 'Włoski', 'T', 100.0), ('USA', 'Angielski', 'T', 80.0), ('CAN', 'Angielski', 'T', 50.0), ('CAN', 'Francuski', 'T', 25.0), ('MEX', 'Hiszpański', 'T', 92.0), ('CUB', 'Hiszpański', 'T', 100.0), ('BRA', 'Portugalski', 'T', 98.0), ('ARG', 'Hiszpański', 'T', 95.0), [...]
```

```
INSERT INTO countryinfo (id, doc) VALUES ('POL', {'continent': "Europa", "president": "Andrzej Duda", "life_expectancy": 78, "gdp": 716000000000}), ('SWE', {'continent': "Europa", "president": "Król Karol XVI Gustav", "life_expectancy": 82, "gdp": 620000000000}), ('NOR', {'continent': "Europa", "president": "Król Harald V", "life_expectancy": 82, "gdp": 550000000000}), ('FIN', {'continent': "Europa", "president": "Sauli Niinistö", "life_expectancy": 81, "gdp": 320000000000}), ('GR', {'continent': "Europa", "president": "Katerini Sakellariopoulou", "life_expectancy": 81, "gdp": 240000000000}), ('CZ', {'continent': "Europa", "president": "Petr Pavel", "life_expectancy": 79, "gdp": 280000000000}), ('IRL', {'continent': "Europa", "president": "Michael D. Higgins", "life_expectancy": 82, "gdp": 500000000000}), ('PRT', {'continent': "Europa", "president": "Marcelo Rebelo de Sousa", "life_expectancy": 81, "gdp": 290000000000})[...]
```

```
SELECT country.Name AS CountryName, city.Name AS CapitalName, CAST(JSON_UNQUOTE(JSON_EXTRACT(countryinfo.doc, '$.GNP')) AS DECIMAL(15,2)) AS GNP FROM country JOIN city ON country.Capital = city.ID JOIN countryinfo ON country.Code = countryinfo.id ORDER BY GNP DESC LIMIT 1;
```

CountryName	CapitalName	GNP
United States	Washington D.C.	21427700.00

- ```
SELECT JSON_UNQUOTE(JSON_EXTRACT(countryinfo.doc, '$.Continent')) AS Continent, MIN(CAST(JSON_UNQUOTE(JSON_EXTRACT(countryinfo.doc, '$.GNP')) AS DECIMAL(15,2))) AS MinGNP,
MAX(CAST(JSON_UNQUOTE(JSON_EXTRACT(countryinfo.doc, '$.GNP')) AS DECIMAL(15,2))) AS MaxGNP, AVG(CAST(JSON_UNQUOTE(JSON_EXTRACT(countryinfo.doc, '$.GNP')) AS DECIMAL(15,2))) AS AvgGNP FROM countryinfo GROUP
BY Continent;
```

### Wynik zapytania



| Continent     | MinGNP     | MaxGNP      | AvgGNP         |
|---------------|------------|-------------|----------------|
| Africa        | 303170.00  | 351432.00   | 327301.000000  |
| Asia          | 761425.00  | 14722731.00 | 5014753.400000 |
| Europe        | 1318222.00 | 3845630.00  | 2392497.666667 |
| North America | 1220703.00 | 21427700.00 | 8128276.333333 |
| Oceania       | 208407.00  | 1382734.00  | 795570.500000  |
| South America | 449663.00  | 2055507.00  | 1252585.000000 |

- 4) Stworzyłem zapytanie zwracające wszystkie miasta znajdujące się w regionie North America (countryinfo)

✓ MySQL zwrócił pusty wynik (zero wierszy). (Wykonanie zapytania trwało 0,0009 sekund(y).)

```
SELECT city.Name AS CityName FROM city JOIN countryinfo ON city.CountryCode = countryinfo._id WHERE JSON_UNQUOTE(JSON_EXTRACT(countryinfo.doc, '$.Region')) = 'North America';
```

☐ Profilowanie [\[ Edytuj w linii \]](#) [\[ Edit \]](#) [\[ Explain SQL \]](#) [\[ Create PHP code \]](#) [\[ Refresh \]](#)

Wyniki w moim przypadku:

| CityName |
|----------|
|----------|

Brak

- 5) Stworzyłem zapytanie zwracające listę państw dla których wartość pola „HeadOfState” zawiera w sobie „Elisabeth”

✓ MySQL zwrócił pusty wynik (zero wierszy). (Wykonanie zapytania trwało 0,0009 sekund(y).)

```
SELECT country.Name AS CountryName FROM country JOIN countryinfo ON country.Code = countryinfo._id WHERE JSON_UNQUOTE(JSON_EXTRACT(countryinfo.doc, '$.HeadOfState')) LIKE 'Elisabeth%';
```

Wyniki w moim przypadku:

| CountryName |
|-------------|
|-------------|

Brak

- 6) Stworzyłem zapytanie zwracające ilość państw znajdujących się na każdym z kontynentów

✓ Pokazano wiersze 0 - 5 (6 total, Wykonanie zapytania trwało 0,0287 sekund(y).)

```
SELECT JSON_UNQUOTE(JSON_EXTRACT(countryinfo.doc, '$.Continent')) AS Continent, COUNT(*) AS CountryCount FROM countryinfo GROUP BY Continent;
```

☐ Profilowanie [\[ Edytuj w linii \]](#) [\[ Edit \]](#) [\[ Explain SQL \]](#) [\[ Create PHP code \]](#) [\[ Refresh \]](#)

Wyniki w moim przypadku:

| Continent     | CountryCount |
|---------------|--------------|
| Africa        | 2            |
| Asia          | 5            |
| Europe        | 6            |
| North America | 3            |
| Oceania       | 2            |
| South America | 2            |

- 7) Stworzyłem zapytanie zwracające nazwy 10 państw z największą oraz 10 z najmniejszą wartością pola LifeExpectancy (countryinfo)

✓ Pokazano wiersze 0 - 9 (20 total. Wykonanie zapytania trwało 0,0017 sekund(y)).

```
(SELECT country.Name AS CountryName, CAST(JSON_UNQUOTE(JSON_EXTRACT(countryinfo.doc, '$.LifeExpectancy')) AS DECIMAL(5,2)) AS LifeExpectancy FROM country JOIN countryinfo ON country.Code = countryinfo._id ORDER BY LifeExpectancy DESC LIMIT 10) UNION (SELECT country.Name AS CountryName, CAST(JSON_UNQUOTE(JSON_EXTRACT(countryinfo.doc, '$.LifeExpectancy')) AS DECIMAL(5,2)) AS LifeExpectancy FROM country JOIN countryinfo ON country.Code = countryinfo._id ORDER BY LifeExpectancy ASC LIMIT 10);
```

[Edytuj w linii](#) [\[ Edit \]](#) [\[ Create PHP code \]](#)

Wyniki w moim przypadku:

| CountryName    | LifeExpectancy | ▲ 1 |
|----------------|----------------|-----|
| Japan          | 84.50          |     |
| Spain          | 83.40          |     |
| South Korea    | 83.30          |     |
| Italy          | 83.10          |     |
| Australia      | 82.90          |     |
| France         | 82.40          |     |
| Canada         | 82.30          |     |
| United Kingdom | 81.80          |     |
| New Zealand    | 81.20          |     |
| Germany        | 81.00          |     |
| South Africa   | 64.10          |     |
| India          | 70.00          |     |
| Egypt          | 71.10          |     |
| Russia         | 72.60          |     |
| Brazil         | 75.00          |     |
| Mexico         | 76.40          |     |
| Argentina      | 76.70          |     |
| China          | 76.90          |     |
| Turkey         | 77.20          |     |
| United States  | 78.50          |     |

## Sprawozdanie bazy danych 5

- 1) Stwórz zapytanie zwracające wszystkie wartości pola „GovernmentForm” (tabela countryinfo, jeśli nie istnieje to stworzę tabelę) bez powtórzeń. Jeśli pole nie istnieje to rozszerz tabelę countryinfo o pole „GovernmentForm”. Wypełnij pole/tabelę danymi.

Tworzę nowe pole GovernmentForm do tabeli countryinfo:

Wykonanie zapytania/zapytań SQL do bazy danych lab 3: ?

```
1 ALTER TABLE countryinfo ADD GovernmentForm varchar(255);
```

Dodaję wartości do mojej tabeli:

Wykonanie zapytania/zapytań SQL do bazy danych lab 3: ?

```
1 UPDATE countryinfo SET GovernmentForm = 'Federal Republic' WHERE _id = 'ARG';
2 UPDATE countryinfo SET GovernmentForm = 'Constitutional Monarchy' WHERE _id = 'AUS';
3 UPDATE countryinfo SET GovernmentForm = 'Federal Republic' WHERE _id = 'BRA';
4 UPDATE countryinfo SET GovernmentForm = 'Constitutional Monarchy' WHERE _id = 'CAN';
5 UPDATE countryinfo SET GovernmentForm = 'Communist State' WHERE _id = 'CHN';
6 UPDATE countryinfo SET GovernmentForm = 'Federal Republic' WHERE _id = 'DEU';
7 UPDATE countryinfo SET GovernmentForm = 'Republic' WHERE _id = 'EGY';
8 UPDATE countryinfo SET GovernmentForm = 'Constitutional Monarchy' WHERE _id = 'ESP';
9 UPDATE countryinfo SET GovernmentForm = 'Republic' WHERE _id = 'FRA';
10 UPDATE countryinfo SET GovernmentForm = 'Constitutional Monarchy' WHERE _id = 'GBR';
11 UPDATE countryinfo SET GovernmentForm = 'Federal Republic' WHERE _id = 'IND';
12 UPDATE countryinfo SET GovernmentForm = 'Republic' WHERE _id = 'ITA';
13 UPDATE countryinfo SET GovernmentForm = 'Constitutional Monarchy' WHERE _id = 'JPN';
14 UPDATE countryinfo SET GovernmentForm = 'Republic' WHERE _id = 'KOR';
15 UPDATE countryinfo SET GovernmentForm = 'Federal Republic' WHERE _id = 'MEX';
```

Tworzę zapytanie:

Wykonanie zapytania/zapytań SQL do bazy danych lab 3: ?

```
1 SELECT DISTINCT GovernmentForm FROM countryinfo;
```

Wynik:

|                                           | GovernmentForm          |
|-------------------------------------------|-------------------------|
| <input type="checkbox"/> Edit Copy Delete | Federal Republic        |
| <input type="checkbox"/> Edit Copy Delete | Constitutional Monarchy |
| <input type="checkbox"/> Edit Copy Delete | Communist State         |
| <input type="checkbox"/> Edit Copy Delete | Republic                |

- 2) Stwórz zapytanie wypisujące najczęściej występującą wartość w bazie dla pola „Continent” (countryinfo). Jeśli pole nie istnieje to rozszerz tabelę o to pole oraz wypełnij je danymi.

Tworze nowe pole Continent do tabeli countryinfo:

Wykonanie zapytania/zapytań SQL do bazy danych lab 3: ?

```
1 ALTER TABLE countryinfo ADD Continent varchar(255);
```

Tworze zapytanie

Wykonanie zapytania/zapytań SQL do bazy danych lab 3: ?

```
1 SELECT JSON_UNQUOTE(JSON_EXTRACT(countryinfo.doc, '$.Continent')) AS Continent,
2 COUNT(*) AS CountContinent
3 FROM countryinfo
4 GROUP BY JSON_UNQUOTE(JSON_EXTRACT(countryinfo.doc, '$.Continent'))
5 ORDER BY CountContinent DESC
6 LIMIT 1;
```

Wynik:

| Continent | CountContinent |
|-----------|----------------|
| Europe    | 6              |

- 3) Stwórz zapytanie wypisujące nazwy państw wraz z wartością IndepYear w kolejności malejącej po polu „IndepYear” (countryinfo). Jeśli pole nie istnieje to rozszerz tabelę o to pole oraz wypełnij je danymi.

Tworze nowe pole do tabeli countryinfo:

Uruchom zapytanie SQL/zapytania w tabeli lab 3.countryinfo: ?

```
1 UPDATE countryinfo SET doc = JSON_SET(doc, '$.IndepYear', 1816) WHERE _id = 'ARG'; -- Argentina
2 UPDATE countryinfo SET doc = JSON_SET(doc, '$.IndepYear', 1901) WHERE _id = 'AUS'; -- Australia
3 UPDATE countryinfo SET doc = JSON_SET(doc, '$.IndepYear', 1822) WHERE _id = 'BRA'; -- Brazil
4 UPDATE countryinfo SET doc = JSON_SET(doc, '$.IndepYear', 1867) WHERE _id = 'CAN'; -- Canada
5 UPDATE countryinfo SET doc = JSON_SET(doc, '$.IndepYear', 1949) WHERE _id = 'CHN'; -- China
6 UPDATE countryinfo SET doc = JSON_SET(doc, '$.IndepYear', 1871) WHERE _id = 'DEU'; -- Germany
7 UPDATE countryinfo SET doc = JSON_SET(doc, '$.IndepYear', 1922) WHERE _id = 'EGY'; -- Egypt
8 UPDATE countryinfo SET doc = JSON_SET(doc, '$.IndepYear', 1492) WHERE _id = 'ESP'; -- Spain
9 UPDATE countryinfo SET doc = JSON_SET(doc, '$.IndepYear', 843) WHERE _id = 'FRA'; -- France
10 UPDATE countryinfo SET doc = JSON_SET(doc, '$.IndepYear', 1066) WHERE _id = 'GBR'; -- United Kingdom
11 UPDATE countryinfo SET doc = JSON_SET(doc, '$.IndepYear', 1947) WHERE _id = 'IND'; -- India
12 UPDATE countryinfo SET doc = JSON_SET(doc, '$.IndepYear', 1861) WHERE _id = 'ITA'; -- Italy
13 UPDATE countryinfo SET doc = JSON_SET(doc, '$.IndepYear', 660) WHERE _id = 'JPN'; -- Japan
14 UPDATE countryinfo SET doc = JSON_SET(doc, '$.IndepYear', 1948) WHERE _id = 'KOR'; -- South Korea
15 UPDATE countryinfo SET doc = JSON_SET(doc, '$.IndepYear', 1810) WHERE _id = 'MEX'; -- Mexico
```

Tworze zapytanie do wyników:

Wykonanie zapytania/zapytań SQL do bazy danych lab 3: ?


```
1 SELECT country.Name AS Name, CAST(JSON_UNQUOTE(JSON_EXTRACT(countryinfo.doc, '$.IndepYear'))) AS UNSIGNED AS IndepYear FROM country JOIN countryinfo ON
country.Code = countryinfo._id ORDER BY 'IndepYear' DESC;
```

Wynik:

| Name           | IndepYear |
|----------------|-----------|
| Argentina      | 1816      |
| Australia      | 1901      |
| Brazil         | 1822      |
| Canada         | 1867      |
| China          | 1949      |
| Germany        | 1871      |
| Egypt          | 1922      |
| Spain          | 1492      |
| France         | 843       |
| United Kingdom | 1066      |
| India          | 1947      |
| Italy          | 1861      |
| Japan          | 660       |
| South Korea    | 1948      |
| Mexico         | 1810      |
| New Zealand    | 1840      |
| Russia         | 1991      |
| Turkey         | 1923      |
| United States  | 1776      |
| South Africa   | 1910      |

- 4) Stwórz zapytanie wypisujące języki oraz ilokrotnie są językami urzędowymi w kolejności malejącej)

Tworze zapytanie:

Wykonanie zapytania/zapytań SQL do bazy danych lab 3: 


```
1 SELECT Language, COUNT(*) AS OfficialCount FROM countrylanguage WHERE IsOfficial = 'T' GROUP BY Language ORDER BY OfficialCount DESC;
```

Wyniki:

| Language   | OfficialCount |
|------------|---------------|
| English    | 6             |
| Spanish    | 3             |
| French     | 2             |
| Chinese    | 1             |
| Arabic     | 1             |
| Italian    | 1             |
| Russian    | 1             |
| German     | 1             |
| Hindi      | 1             |
| Japanese   | 1             |
| Portuguese | 1             |

- 5) Stwórz zapytanie wypisujące języki oraz ilość ludzi posługujących się nimi na całym świecie w kolejności malejącej

Dodaje pole population do countryinfo

Wykonanie zapytania/zapytań SQL do bazy danych lab 3: 

```
1 UPDATE countryinfo SET doc = JSON_SET(doc, '$.Population', 45195774) WHERE _id = 'ARG'; -- Argentina
2 UPDATE countryinfo SET doc = JSON_SET(doc, '$.Population', 25687041) WHERE _id = 'AUS'; -- Australia
3 UPDATE countryinfo SET doc = JSON_SET(doc, '$.Population', 216422446) WHERE _id = 'BRA'; -- Brazil
4 UPDATE countryinfo SET doc = JSON_SET(doc, '$.Population', 38905000) WHERE _id = 'CAN'; -- Canada
5 UPDATE countryinfo SET doc = JSON_SET(doc, '$.Population', 1444216107) WHERE _id = 'CHN'; -- China
6 UPDATE countryinfo SET doc = JSON_SET(doc, '$.Population', 83900000) WHERE _id = 'DEU'; -- Germany
7 UPDATE countryinfo SET doc = JSON_SET(doc, '$.Population', 109262178) WHERE _id = 'EGY'; -- Egypt
8 UPDATE countryinfo SET doc = JSON_SET(doc, '$.Population', 47450795) WHERE _id = 'ESP'; -- Spain
9 UPDATE countryinfo SET doc = JSON_SET(doc, '$.Population', 67186638) WHERE _id = 'FRA'; -- France
10 UPDATE countryinfo SET doc = JSON_SET(doc, '$.Population', 67530172) WHERE _id = 'GBR'; -- United Kingdom
11 UPDATE countryinfo SET doc = JSON_SET(doc, '$.Population', 1417173173) WHERE _id = 'IND'; -- India
12 UPDATE countryinfo SET doc = JSON_SET(doc, '$.Population', 60244639) WHERE _id = 'ITA'; -- Italy
13 UPDATE countryinfo SET doc = JSON_SET(doc, '$.Population', 125440000) WHERE _id = 'JPN'; -- Japan
14 UPDATE countryinfo SET doc = JSON_SET(doc, '$.Population', 52000000) WHERE _id = 'KOR'; -- South Korea
15 UPDATE countryinfo SET doc = JSON_SET(doc, '$.Population', 128932753) WHERE _id = 'MEX'; -- Mexico
```

Zapytanie:

Wykonanie zapytania/zapytań SQL do bazy danych lab 3: 

```
1 SELECT
2 countrylanguage.Language,
3 CAST(JSON_EXTRACT(countryinfo.doc, '$.Population') AS UNSIGNED) * countrylanguage.Percentage / 100 AS Amount
4 FROM
5 countrylanguage
6 JOIN
7 countryinfo
8 ON
9 countrylanguage.CountryCode = countryinfo._id;
```

Wynik:

| Language   | Amount       |
|------------|--------------|
| Spanish    | 42935985.3   |
| English    | 24402688.95  |
| Portuguese | 212093997.08 |
| English    | 23343000     |
| French     | 15562000     |
| Chinese    | 1299794496.3 |
| German     | 75510000     |
| Arabic     | 103799069.1  |
| Spanish    | 46501779.1   |
| French     | 63827306.1   |
| English    | 64153663.4   |
| English    | 283434634.6  |
| Hindi      | 566869269.2  |
| Italian    | 59642192.61  |
| Japanese   | 124185600    |
| Spanish    | 116039477.7  |
| English    | 4780131.3    |
| Russian    | 124245362.75 |
| English    | 268753602.4  |
| English    | 6069817.9    |

- 6) Stwórz zapytanie wypisujące kraje które znajdują się w pierwszej dwudziestce pod względem długości życia oraz jednocześnie znajdują się w pierwszej dwudziestce krajów z największą wartością GNP

Zapytanie:

```
1 WITH TopLifeExpectancy AS (
2 SELECT
3 country.Code AS CountryCode
4 FROM
5 country
6 JOIN
7 countryinfo
8 ON
9 country.Code = countryinfo._id
10 ORDER BY
11 CAST(JSON_UNQUOTE(JSON_EXTRACT(countryinfo.doc, '$.LifeExpectancy')) AS UNSIGNED) DESC
12 LIMIT 20
13),
14 TopGNP AS (
15 SELECT
16 country.Code AS CountryCode
17 FROM
18 country
19 JOIN
20 countryinfo
21 ON
22 country.Code = countryinfo._id
23 ORDER BY
24 CAST(JSON_UNQUOTE(JSON_EXTRACT(countryinfo.doc, '$.GNP')) AS UNSIGNED) DESC
25 LIMIT 20
26)
27 SELECT
28 country.Name,
29 CAST(JSON_UNQUOTE(JSON_EXTRACT(countryinfo.doc, '$.LifeExpectancy')) AS UNSIGNED) AS LifeExpectancy,
30 CAST(JSON_UNQUOTE(JSON_EXTRACT(countryinfo.doc, '$.GNP')) AS UNSIGNED) AS GNP
31 FROM
32 country
33 JOIN
34 countryinfo
35 ON
36 country.Code = countryinfo._id
37 WHERE
38 country.Code IN (SELECT CountryCode FROM TopLifeExpectancy)
39 AND
40 country.Code IN (SELECT CountryCode FROM TopGNP);
```

Wyniki:

| Name           | LifeExpectancy | GNP      |
|----------------|----------------|----------|
| Argentina      | 76             | 449663   |
| Australia      | 82             | 1382734  |
| Brazil         | 75             | 2055507  |
| Canada         | 82             | 1736426  |
| China          | 76             | 14722731 |
| Germany        | 81             | 3845630  |
| Egypt          | 71             | 303170   |
| Spain          | 83             | 1318222  |
| France         | 82             | 2715518  |
| United Kingdom | 81             | 2825208  |
| India          | 70             | 2875142  |
| Italy          | 83             | 1937894  |
| Japan          | 84             | 5081770  |
| South Korea    | 83             | 1632699  |
| Mexico         | 76             | 1220703  |
| New Zealand    | 81             | 208407   |
| Russia         | 72             | 1712514  |
| Turkey         | 77             | 761425   |
| United States  | 78             | 21427700 |
| South Africa   | 64             | 351432   |

# Sprawozdanie bazy danych 6

- 1) Zaprojektuj (postać logiczną) swoją bazę danych w wybranym przez siebie narzędziu (zalecane narzędzie SQL Data Modeler), baza danych powinna spełniać następujące wymagania:
  - posiadać co najmniej 5 encji
  - posiadać co najmniej jedną relację „one to many”
  - posiadać przynajmniej jedną relację „many to many”
  - posiadać w przynajmniej jednej tabeli klucz złożony składający się co najmniej z dwóch kolumn
  - posiadać co najmniej jedną tabelę, w której przynajmniej jedna kolumna będzie typu Enum
- 2) Stworzyłem bazę danych która zawiera 5 encji i posiada jedną tabelę z przynajmniej jedną kolumną typu Enum, kluczem złożonym i posiada wymagane relacje z polecenia które pokaże w podpunkcie 3).

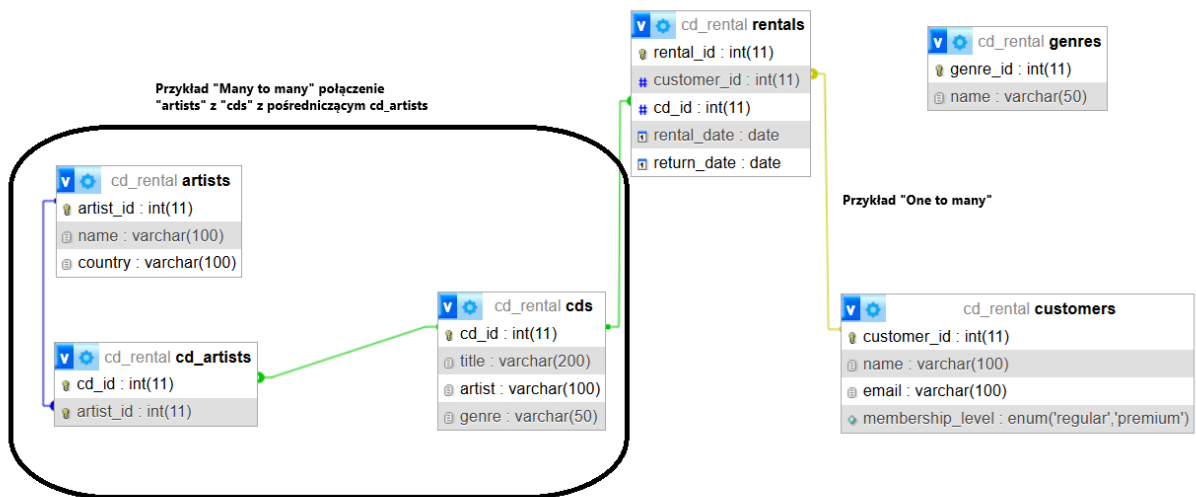
Uruchom zapytanie/zapytania SQL na serwerze "127.0.0.1":

```
1 CREATE DATABASE CD_Rental;
2
3 USE CD_Rental;
4
5 CREATE TABLE Customers (
6 customer_id INT AUTO_INCREMENT PRIMARY KEY,
7 name VARCHAR(100),
8 email VARCHAR(100),
9 membership_level ENUM('regular', 'premium') NOT NULL
10);
11
12 CREATE TABLE CDs (
13 cd_id INT AUTO_INCREMENT PRIMARY KEY,
14 title VARCHAR(200),
15 artist VARCHAR(100),
16 genre VARCHAR(50)
17);
18
19 CREATE TABLE Rentals (
20 rental_id INT AUTO_INCREMENT PRIMARY KEY,
21 customer_id INT,
22 cd_id INT,
23 rental_date DATE,
24 return_date DATE,
25 FOREIGN KEY (customer_id) REFERENCES Customers(customer_id),
26 FOREIGN KEY (cd_id) REFERENCES CDs(cd_id)
27);
28
29 CREATE TABLE Artists (
30 artist_id INT AUTO_INCREMENT PRIMARY KEY,
31 name VARCHAR(100),
32 country VARCHAR(100)
33);
34
35 CREATE TABLE Genres (
36 genre_id INT AUTO_INCREMENT PRIMARY KEY,
37 name VARCHAR(50)
38);
39
40 -- Tabela pośrednia dla relacji many-to-many:
41 CREATE TABLE CD_Artists (
42 cd_id INT,
43 artist_id INT,
44 PRIMARY KEY (cd_id, artist_id),
45 FOREIGN KEY (cd_id) REFERENCES CDs(cd_id),
46 FOREIGN KEY (artist_id) REFERENCES Artists(artist_id)
47);
```



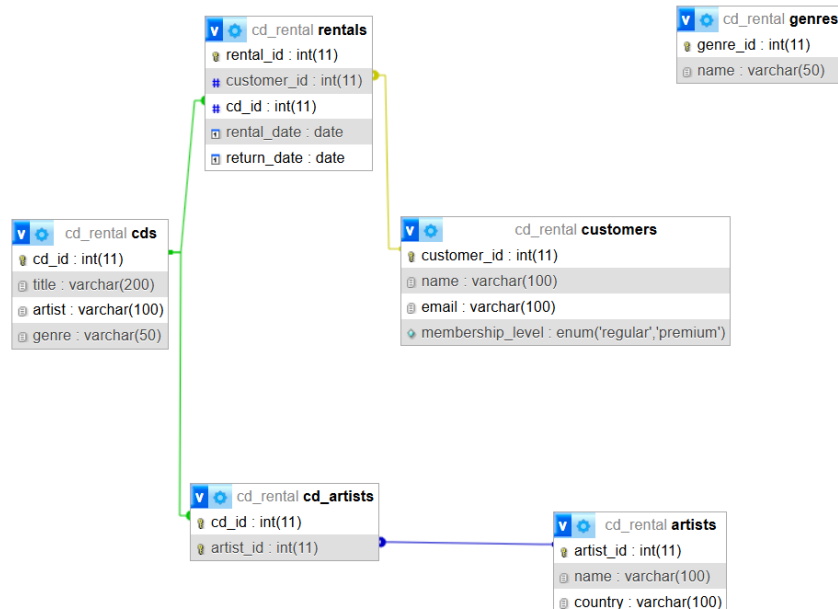
### 3) Relacje w moim DataSecie

- Relacje „One to many”, przykład:  
relacja „customers(customer\_id)” z „rentals(customer\_id)”
- Relacje „Many to many”, przykład:  
relacja „cds(cd\_id)” z „artists(cd\_id)”. Wymagana tabela pośrednia to cd\_artists



# Sprawozdanie bazy danych 7

## 1) Moja baza danych dla listy 6



- 2) Zaprojektowaną bazę (dla listy 6) przekształć do pierwszej postaci normalnej, podkreśl elementy które muszą być spełnione, aby baza znajdowała się w pierwszej postaci normalnej

### Pierwsza postać normalna (1NF)

Baza danych jest w 1NF, jeśli wszystkie kolumny zawierają atomowe wartości, a w tabelach nie ma powtarzających się grup. Moja baza danych spełnia te wymagania, ponieważ każda wartość w tabelach jest niepodzielna, a struktura tabel eliminuje redundancję danych.

- 3) Zaprojektowaną bazę przekształć do drugiej postaci normalnej, podkreśl elementy które muszą być spełnione, aby baza znajdowała się w drugiej postaci normalnej

### Druga postać normalna (2NF)

Aby osiągnąć 2NF, należy spełnić warunki 1NF oraz zapewnić, że każdy atrybut niekluczowy jest w pełni zależny od klucza głównego. W mojej bazie danych relacje są tak zaprojektowane, że wszystkie kolumny niekluczowe zależą w pełni od kluczy głównych swoich tabel.

- 4) Zaprojektowaną bazę przekształć do trzeciej postaci normalnej, podkreśl elementy które muszą być spełnione, aby baza znajdowała się w trzeciej postaci normalnej

#### **Trzecia postać normalna (3NF)**

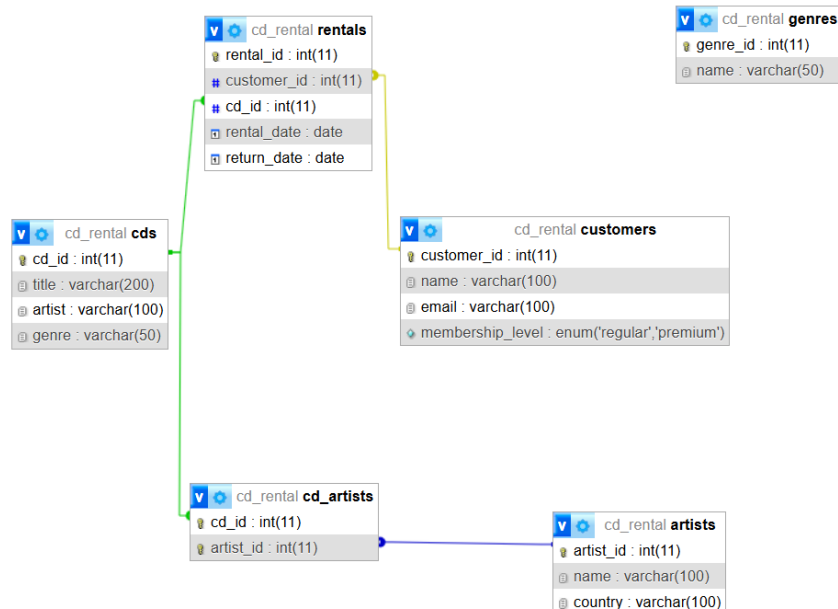
W 3NF eliminowane są przechodnie zależności, czyli sytuacje, w których atrybuty niekluczowe zależą od innych atrybutów niekluczowych. Moja baza danych spełnia te wymagania, ponieważ każdy atrybut zależy bezpośrednio od klucza głównego tabeli, a struktura nie zawiera redundantnych danych.

#### **Podsumowanie**

Moja baza danych została zaprojektowana zgodnie z zasadami 1NF, 2NF i 3NF. Nie wymaga dodatkowych modyfikacji, aby spełnić te postacie normalne.

# Sprawozdanie bazy danych 8

## 1) Moja baza danych dla listy 6



- 2) Zaprojektowaną bazę (dla listy 6) przekształć do czwartej postaci normalnej, podkreśl elementy które muszą być spełnione, aby baza znajdowała się w czwartej postaci normalnej

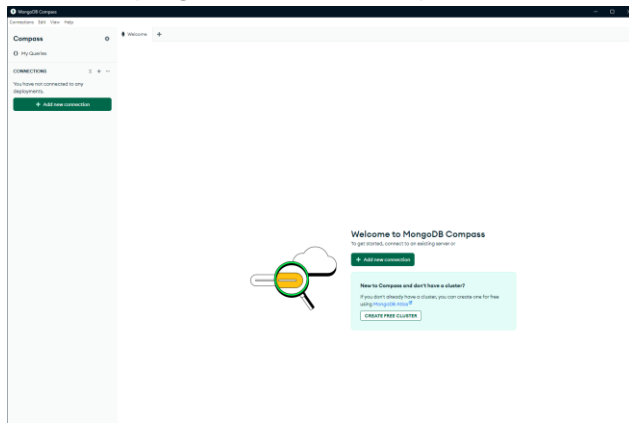
### Czwarta postać normalna (4NF)

Baza danych spełnia 4NF, jeśli jest w 3NF oraz eliminuje wielowartościowe zależności. Wielowartościowa zależność występuje wtedy, gdy jedna kolumna w tabeli może być powiązana z wieloma wartościami innej kolumny w sposób niezależny od pozostałych kolumn w tabeli. Aby zapobiec takim zależnościom, konieczne jest rozdzielenie danych do osobnych tabel.

W mojej bazie danych relacje wiele-do-wielu są poprawnie znormalizowane przy użyciu tabeli pośredniej CD\_Artists. Dzięki temu dane są podzielone w sposób eliminujący wielowartościowe zależności. Struktura bazy danych spełnia wszystkie wymagania 4NF i nie wymaga dodatkowych modyfikacji.

# Sprawozdanie bazy danych 9

- 1) Zainstaluj bazę danych MongoDB na swoim komputerze  
Pobrałem program na moim komputerze



- 2) Odpowiednikami dla jakich pojęć w relacyjnej bazie danych są pojęcia: kolekcja (Collection), dokument (Document), pole (Field), dokumenty wbudowane (Embedded Documents), klucz główny (Primary Key)

Kolekcja → Tabela

Dokument → Wiersz/rekord

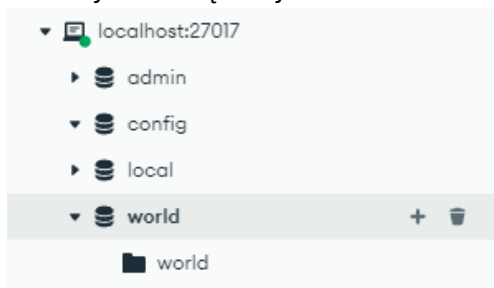
Pole → Kolumna

Dokumenty wbudowane → Zagnieżdżone tabele/klucze obce

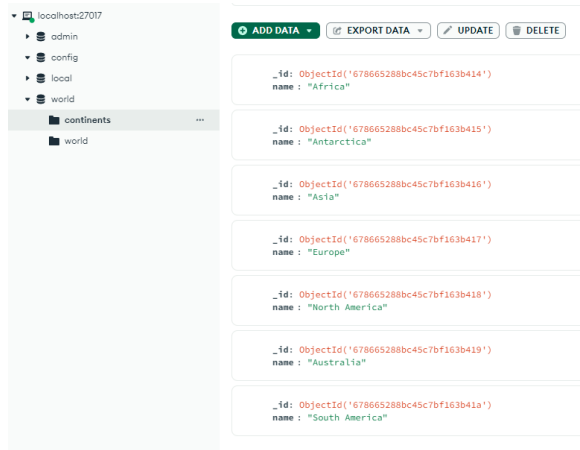
Klucz główny → Primary Key (\_id)

- 3) Stwórz bazę danych „world”

Stworzyłem bazę danych world:



4) W bazie stwórz kolekcję „continents”, następnie uzupełnij ją listą kontynentów



5) Usuń kolekcję „continents”

```
> db.continents.drop()
< true
```

6) Usuń bazę danych „world”

```
> db.world.drop()
< true
```

# Sprawozdanie bazy danych 10

- 1) Zaimportuj kolekcję „restaurants” z pliku restaurants.json

```
PS C:\Users\fkale> & "C:\Users\fkale\Desktop\mongodb-database-tools-windows-x86_64-100.10.0\bin\mongoimport.exe" --db world --collection restaurants --file "C:\Users\fkale\Desktop\OneDrive - Politechnika Wroclawska\Studia\Semestr 3\System b az danych\Lab 10\restaurants.json" --jsonArray
2025-01-14T15:19:38.716+0100 connected to: mongodb://localhost/
2025-01-14T15:19:38.740+0100 20 document(s) imported successfully. 0 document(s) failed to import.
PS C:\Users\fkale> |
```

- 2) Wypisz wszystkie dokumenty znajdujące się w kolekcji

Mój plik zawiera 20 przykładów na screen'ie umieściłem pierwszą stronę po wpisaniu w terminalu find()

```
> db.restaurants.find().pretty()
< {
 "_id": ObjectId("6786727a9885742e9ef1dad"),
 "address": {
 "building": "2003",
 "coord": [
 -73.971111,
 40.881111000000004
],
 "street": "New Street D",
 "zipcode": "2003"
 },
 "borough": "Bronx",
 "cuisine": "Mexican",
 "grades": [
 {
 "date": 2014-03-16T20:05:00.000Z,
 "grade": "A",
 "score": 6
 },
 {
 "date": 2013-09-24T05:25:00.000Z,
 "grade": "C",
 "score": 9
 },
 {
 "date": 2013-05-31T11:30:20.000Z,
 "grade": "B",
 "score": 12
 }
],
 "name": "Restaurant 3",
 "restaurant_id": "40000003"
}
{
 "_id": ObjectId("6786727a9885742e9ef1dae"),
 "address": {
 "building": "2012",
 "coord": [
 -73.881110999999999,
 40.881111
],
 "street": "New Street D",
 "zipcode": "2003"
 },
 "borough": "Bronx",
 "cuisine": "Mexican",
 "grades": [
 {
 "date": 2014-03-16T20:05:00.000Z,
 "grade": "A",
 "score": 6
 },
 {
 "date": 2013-09-24T05:25:00.000Z,
 "grade": "C",
 "score": 9
 },
 {
 "date": 2013-05-31T11:30:20.000Z,
 "grade": "B",
 "score": 12
 }
],
 "name": "Restaurant 3",
 "restaurant_id": "40000003"
}
```

- 3) Wypisz name, borough i cuisine dla wszystkich dokumentów znajdujących się w kolekcji  
Ten sam problem co poprzednio plik ma 20 dokumentów wystąpił pierwszą stronę wyników

```
> db.restaurants.find({}, {name: 1, borough: 1, cuisine: 1})
< {
 "_id": ObjectId("6786727a9885742e9ef1dad"),
 "borough": "Bronx",
 "cuisine": "Mexican",
 "name": "Restaurant 3"
}
{
 "_id": ObjectId("6786727a9885742e9ef1dae"),
 "borough": "Manhattan",
 "cuisine": "Thai",
 "name": "Restaurant 12"
}
{
 "_id": ObjectId("6786727a9885742e9ef1daf"),
 "borough": "Queens",
 "cuisine": "Japanese",
 "name": "Restaurant 1"
}
{
 "_id": ObjectId("6786727a9885742e9ef1db0"),
 "borough": "Brooklyn",
 "cuisine": "French",
 "name": "Restaurant 15"
}
{
 "_id": ObjectId("6786727a9885742e9ef1db1"),
 "borough": "Staten Island",
 "cuisine": "Italian",
 "name": "Restaurant 14"
}
{
 "_id": ObjectId("6786727a9885742e9ef1db2"),
 "borough": "Queens",
 "cuisine": "Japanese",
 "name": "Restaurant 16"
}
}
```

- 4) Wypisz name, borough, address. street i cuisine wyłączając id (\_id) dla wszystkich dokumentów znajdujących się w kolekcji

```
> db.restaurants.find({}, {name: 1, borough: 1, "address.street": 1, cuisine: 1, _id: 0})
< {
 address: {
 street: 'New Street D'
 },
 borough: 'Bronx',
 cuisine: 'Mexican',
 name: 'Restaurant 3'
}
{
 address: {
 street: 'New Street M'
 },
 borough: 'Manhattan',
 cuisine: 'Thai',
 name: 'Restaurant 12'
}
{
 address: {
 street: 'New Street B'
 },
 borough: 'Queens',
 cuisine: 'Japanese',
 name: 'Restaurant 1'
}
{
 address: {
 street: 'New Street P'
 },
 borough: 'Brooklyn',
 cuisine: 'French',
 name: 'Restaurant 18'
}
```

- 5) Wypisz 5 pierwszych restauracji znajdujących się na Bronx  
Restaurant 3, Restaurant 18, Restaurant 13 i Restaurant 8 więcej restauracji z Bronx mój plik nie posiada.

```
> db.restaurants.find({borough: "Bronx"}, {limit: 5})
< {
 _id: ObjectId('6786727a9885742e9ef1dad'),
 address: {
 building: '2003',
 coord: [
 -73.971111,
 40.881110000000004
],
 street: 'New Street D',
 zipcode: '2003'
 },
 borough: 'Bronx',
 cuisine: 'Mexican',
 grades: [
 {
 date: 2014-03-16T20:05:00.000Z,
 grade: 'A',
 score: 6
 },
 {
 date: 2013-09-24T05:25:00.000Z,
 grade: 'C',
 score: 9
 },
 {
 date: 2013-05-31T11:38:20.000Z,
 grade: 'B',
 score: 12
 }
],
 name: 'Restaurant 3',
 restaurant_id: '40000003'
}
{
 _id: ObjectId('6786727a9885742e9ef1db5'),
 address: {
 building: '2018',
 coord: [
 -73.82110999999999,
 41.031111
],
 street: 'New Street S',
 zipcode: '20018'
 },
 borough: 'Bronx',
 cuisine: 'Mexican',
 grades: [
 {
 date: 2014-03-16T20:38:00.000Z,
 grade: 'A',

```



- 6) Wypisz 5 kolejnych restauracji znajdujących się na Bronx (pomiń pierwsze 5 i wyświetl kolejne 5)

Tak jak pisałem powyżej mój plik ma tylko 4 restauracje z Bronxu dlatego wynik jest pusty.

```
> db.restaurants.find({borough: "Bronx"}).skip(5).limit(5)
<
world> |
```

- 7) Wypisz restauracje których score wynosi pomiędzy 5 a 10

```
> db.restaurants.find({'grades.score' : {$gte: 5, $lte: 10}})
< {
 _id: ObjectId('6786727a9885742e9e9f1dad'),
 address: {
 building: '2003',
 coord: [
 -73.971111,
 40.88111000000004
],
 street: 'New Street D',
 zipcode: '2003'
 },
 borough: 'Bronx',
 cuisine: 'Mexican',
 grades: [
 {
 date: 2014-03-16T20:05:00.000Z,
 grade: 'A',
 score: 6
 },
 {
 date: 2013-09-24T05:25:00.000Z,
 grade: 'C',
 score: 9
 },
 {
 date: 2013-05-31T11:38:20.000Z,
 grade: 'B',
 score: 12
 }
],
 name: 'Restaurant 3',
 restaurant_id: '40000003'
}
{
 _id: ObjectId('6786727a9885742e9e9f1dae'),
 address: {
 building: '2012',
 coord: [
 -73.88111099999999,
 40.971111
],
 street: 'New Street M',
 zipcode: '20012'
 },
 borough: 'Manhattan',
 cuisine: 'Thai',
 grades: [
 {
 date: 2014-03-16T20:20:00.000Z,
 grade: 'A',
 score: 10
 }
],
 name: 'Restaurant 4',
 restaurant_id: '40000004'
}
```

# Sprawozdanie bazy danych 11

- 1) Używając kolekcji „restaurants” wykonaj poniższe zadania



- 2) Wypisz wszystkie dokumenty których grade wynosi A, score wynosi 9 i data ma wartość ISODate("2014-08-11T00:00:00Z")

W moim pliku nie występuję taki wynik:

```
> db.restaurants.find({
 "grades": {
 $elemMatch: {
 grade: "A",
 score: 9,
 date: ISODate("2014-08-11T00:00:00Z")
 }
 }
})
<
world>
```

- 3) Wyświetl wszystkie restauracje w kolejności alfabetycznej po nazwie restauracji

```
> db.restaurants.find().sort({ name: 1 })
{
 "_id": ObjectId("678672a9885742e9ef1dcf"),
 "address": {
 "building": "2000",
 "coord": [
 -74.001111,
 40.801111
],
 "street": "New Street A",
 "zipcode": "2000"
 },
 "borough": "Brooklyn",
 "cuisine": "French",
 "grades": [
 {
 "date": "2014-02-16T20:00:00.000Z",
 "grade": "A",
 "score": 0
 },
 {
 "date": "2013-09-24T05:20:00.000Z",
 "grade": "C",
 "score": 3
 },
 {
 "date": "2013-05-31T11:33:20.000Z",
 "grade": "B",
 "score": 6
 }
],
 "name": "Restaurant 0",
 "restaurant_id": "40080000"
},
{
 "_id": ObjectId("678672a9885742e9ef1daf"),
 "address": {
 "building": "2001",
 "coord": [
 -73.99110999999999,
 40.861111
],
 "street": "New Street B",
 "zipcode": "2001"
 },
 "borough": "Queens",
 "cuisine": "Japanese",
 "grades": [
 {
 "date": "2014-03-16T20:01:40.000Z",
 "grade": "B",
 "score": 6
 }
],
 "name": "Restaurant 1",
 "restaurant_id": "40080001"
}
```

- 4) Wypisz wszystkie restauracje które nie posiadają wartości dla pola address.building

Mój plik nie posiada takiego wyniku

```
> db.restaurants.find({ "address.building": { $exists: false } })
<
world>
```

- 5) Wypisz wszystkie restauracje których nazwa rozpoczyna się od słowa „Shop”, niezależnie od wielkości liter

Nazwy restauracji w moim pliku zostały nazwane Restaurants 1,2,3,4 więc żaden wynik nie posiada słowa Shop w sobie.

```
> db.restaurants.find({ name: { $regex: /^Shop/i } })
<
world>
```

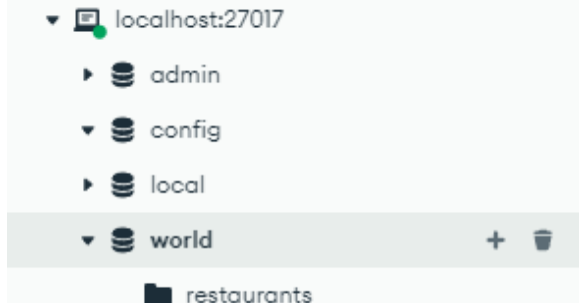
- 6) Wypisz wszystkie restauracje których nazwa zawiera słowa: „Shop” oraz „Pizza”, niezależnie od wielkości liter

Sytuacja opisana powyżej nazwy w wygenerowanym pliku nie posiadają takich słów w sobie.

```
> db.restaurants.find({
 name: { $regex: /Shop/i },
 name: { $regex: /Pizza/i }
})
<
```

# Sprawozdanie bazy danych 12

- 1) Używając kolekcji „restaurants” wykonaj poniższe zadania



- 2) Dodaj własną restaurację „Student Culture Zone” używając polecenia „insertOne”

```
> db.restaurants.insertOne({
 "address": {
 "building": "123",
 "coord": [-73.123456, 40.654321],
 "street": "Student Street",
 "zipcode": "54321"
 },
 "borough": "Manhattan",
 "cuisine": "International",
 "grades": [],
 "name": "Student Culture Zone",
 "restaurant_id": "99999999"
})
< {
 acknowledged: true,
 insertedId: ObjectId('6786f9486bed53efe59bb871')
}
world>
```

- 3) Zmień nazwę dodanej restauracji na „Student Culture Zone PWR”

```
> db.restaurants.updateOne(
 { name: "Student Culture Zone" },
 { $set: { name: "Student Culture Zone PWR" } }
)
< {
 acknowledged: true,
 insertedId: null,
 matchedCount: 1,
 modifiedCount: 1,
 upsertedCount: 0
}
world>
```

- 4) Usuń dodaną restaurację

```
> db.restaurants.deleteOne({ name: "Student Culture Zone PWR" })
< {
 acknowledged: true,
 deletedCount: 1
}
world>
```

- 5) Zmień nazwę pola „name” na „restaurant\_name”

```
> db.restaurants.updateMany(
 {},
 { $rename: { "name": "restaurant_name" } }
)
< {
 acknowledged: true,
 insertedId: null,
 matchedCount: 20,
 modifiedCount: 20,
 upsertedCount: 0
}
world>
```

- 6) Dodaj pole warunkowe „A class”: true jeżeli restauracja posiada przynajmniej jeden grade „A”, używając „\$cond”

```
> db.restaurants.updateMany(
 {},
 [
 { $set: { "A class": { $cond: { if: { $gt: [{ $size: { $filter: { input: "$grades", as: "grade", cond: { $eq: ["$$grade.grade", "A"] } } }], 0 } }, then: true, else: false } } }
]
)
< {
 acknowledged: true,
 insertedId: null,
 matchedCount: 20,
 modifiedCount: 20,
 upsertedCount: 0
}
world>
```