WYŻSZA SZKOŁA HANDLOWA W RADOMIU

RADOM ACADEMY OF ECONOMICS

Wydział Studiów Strategicznych i Technicznych

Kierunek: Informatyka, rok II, semestr III (2021/2022)

BAZ DANYCH (SQL)

Prowadzący: dr hab. Filip Rudziński

Zespół laboratoryjny:

Magdalena Szafrańska, nr albumu: 18345

Spis treści

Wstęp teoretyczny **2**

Wykonanie projektu bazy danych **3**
- Cel projektu 3
- Użyte narzędzia 4
- Projekt bazy danych z użyciem diagramów ERD 4
  - Stworzenie struktury bazodanowej 4
  - Wprowadzenie relacji pomiędzy encjami 5
  - Generowanie kodu SQL 6
  - Uruchomienie serwera bazodanowego w narzędziu XAMPP 6
- Baza danych w serwerze lokalnym XAMPP 7
  - Uruchomienie narzędzia phpMyAdmin 7
  - Utworzenie bazy danych 7
  - Stworzenie encji 7
  - Stworzenie relacji pomiędzy encjami 9
- Wprowadzenie danych do bazy 10
  - Tabela “Country” 10
  - Tabela “Address” 12
  - Tabela “Product” 12
  - Tabela “Service” 12
  - Tabela “Influencers” 13
  - Tabela “Shipping” 13
  - Tabela “InfluencerAction” 13
  - Tabela “Salary” 13
- Zapytania SQL 14
  - Influencerzy posortowani malejąco wg liczby followersów (ORDER BY) 14
  - Akcje wykonane przez influencerów w listopadzie (BETWEEN) 14
  - Wszystkie produkty z “Reductor” w nazwie (LIKE) 15
  - Jaka wysyłka i kiedy trafiła do influencerów (AS) 15
  - Funkcje SQL 16
    - ilość wszystkich produktów (COUNT) 16
    - średni koszt możliwych do wykonania przez influencera akcji 16
  - Wysyłki konkretnych produktów do influencerów 17
    - wszyscy influencerzy, którym wysłano (LIKE + AND) 17
    - ograniczenie ilości do 5 (LIMIT) 18
  - Wszystkie wysyłki do określonego influencera 18
  - Aktualizacja imienia influencera (UPDATE) 19
  - Influencerzy spoza Polski (NOT IN) 20
  - Akcje wykonane przez influencerów 20
    - influencerzy, którzy nie wykonali żadnej akcji (LEFT JOIN) 20
    - influencerzy, którzy wykonali jakąkolwiek akcję (INNER JOIN) 21
  - Wyszukanie produktów nigdy nie wysłanych (NOT EXISTS) 21

Skrypt SQL do utworzenia bazy danych **23**

Wstęp teoretyczny

Podstawowym etapem powstawania bazy danych jest tworzenie jej projektu. Od decyzji podjętych na etapie projektowania zależeć będzie jakość i użyteczność stworzonej bazy danych. Baza danych, podobnie jak większość projektów komputerowych, jest modelem wycinka świata rzeczywistego, utworzonym tak, aby był możliwy do zapamiętania przez maszynę cyfrową i zawierał optymalną ilość informacji do zastosowania, jakiemu będzie służył.

Przy modelowaniu baz danych możemy posłużyć się notacją graficzną modelowania danych – diagramem związków encji ERD (ang. Entity-Relationship Diagram). Jest to model sieciowy opisujący na wysokim poziomie abstrakcji dane, które są przechowywane w systemie.

Każda nowo utworzona baza danych wymaga konsekwentnego etapowego działania. Cały proces projektowania bazy danych możemy podzielić na kilka etapów:

- planowanie bazy danych,
    - określenie występujących zbiorów encji,
    - określenie atrybutów przypisanych do poszczególnych encji
    - określenie dziedziny poszczególnych atrybutów
- tworzenie modelu konceptualnego (diagramu ERD),
- transformacja modelu konceptualnego na model relacyjny,
- proces normalizacji bazy danych,
- wybór struktur i określenie zasad dostępu do bazy danych.

Wykonanie projektu bazy danych

Zaprojektowałam bazę danych do serwisu monitorującego współpracę firmy z influencerami. Model związków encji zawiera 8 encji. Zaprojektowaną bazę planuję w przyszłości rozszerzyć i realnie wykorzystać w mojej pracy na co dzień, gdyż takiego właśnie narzędzia monitorującego brakuje w mojej obecnej firmie.

Cel projektu

Świat cyfrowy daje coraz więcej możliwości do współpracy online. Firmy zawierają porozumienia z osobami obecnymi w social mediach (influencerami) wynagradzając ich za promowanie produktów marki, wysyłając im swoje produkty do testowania czy choćby nadając im specjalne kody rabatowe na zniżkę w sklepie internetowym.
Moim celem jest przygotowanie modelu bazy danych, który będzie umożliwiał m.in.:

- dodawanie i usuwanie influencerów,
- gromadził informację jakie działania marketingowe na rzecz marki wykonał dany influencer,
- jakie wynagrodzenie marka powinna wypłacić influencerowi za określone działania,
- kiedy i jakie produkty zostały wysłane do influencera,
- użycia jego kodu zniżkowego,
- przeszukanie osób (influencerów) generujących dla marki największe zyski.

Użyte narzędzia

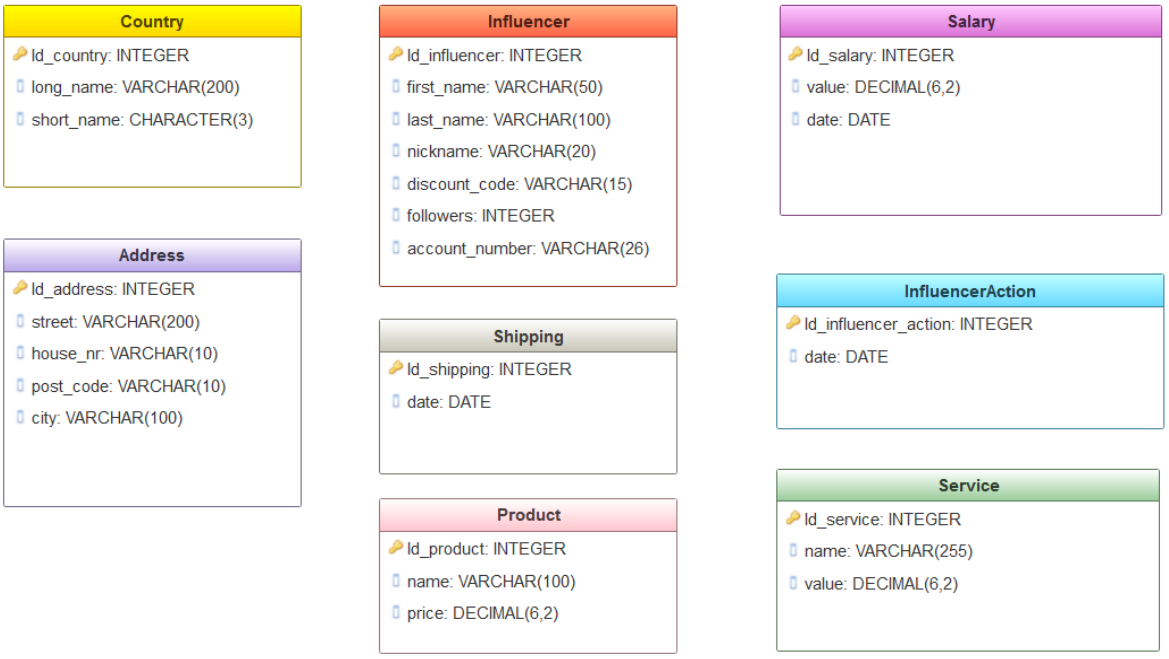
- język zapytań SQL
- lokalny serwer XAMPP w wersji v3.3.0
- silnik bazy danych MariaDB w wersji 10.4.21
- narzędzie phpMyAdmin służące do przetwarzania informacji znajdujących się w bazie danych i łatwego zarządzania bazą
- edytor GenMyModel do projektowanie baz danych (https://www.genmymodel.com/)

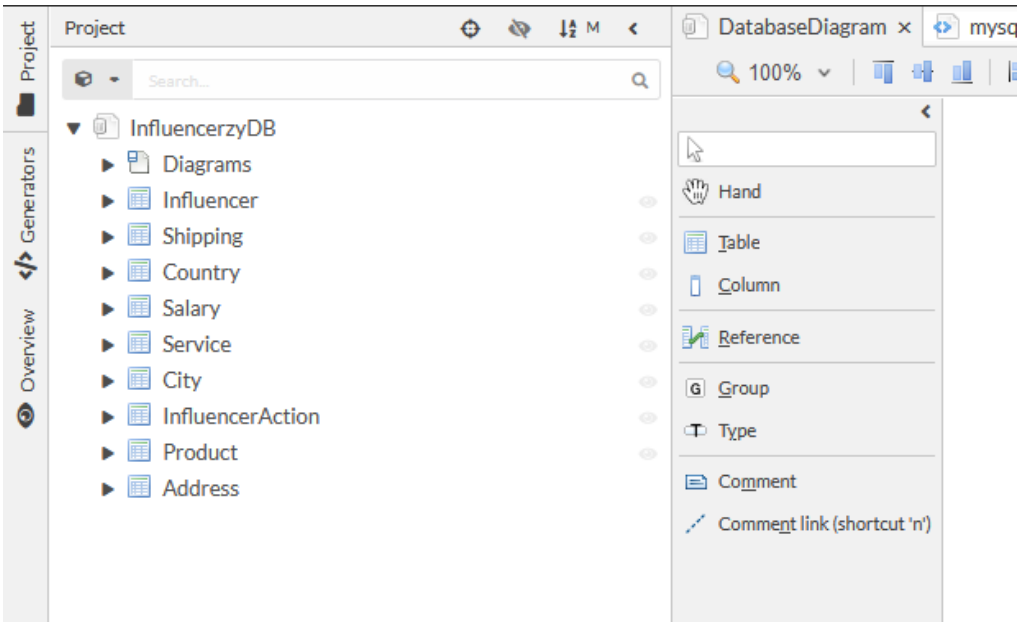
Projekt bazy danych z użyciem diagramów ERD

Opracowałam diagram związków encji, który będzie jednoznacznie i przejrzyście przedstawiał wymagania firmy w zakresie przetwarzanych przez nią danych oraz umożliwiał zbudowanie na jego podstawie relacyjnej bazy danych.
Do wykonania diagramu ERD użyłam edytora GenMyModel (https://www.genmymodel.com/) umożliwiającego projektowanie baz danych online na poziomie tabel i odniesień.

1. Stworzenie struktury bazodanowej

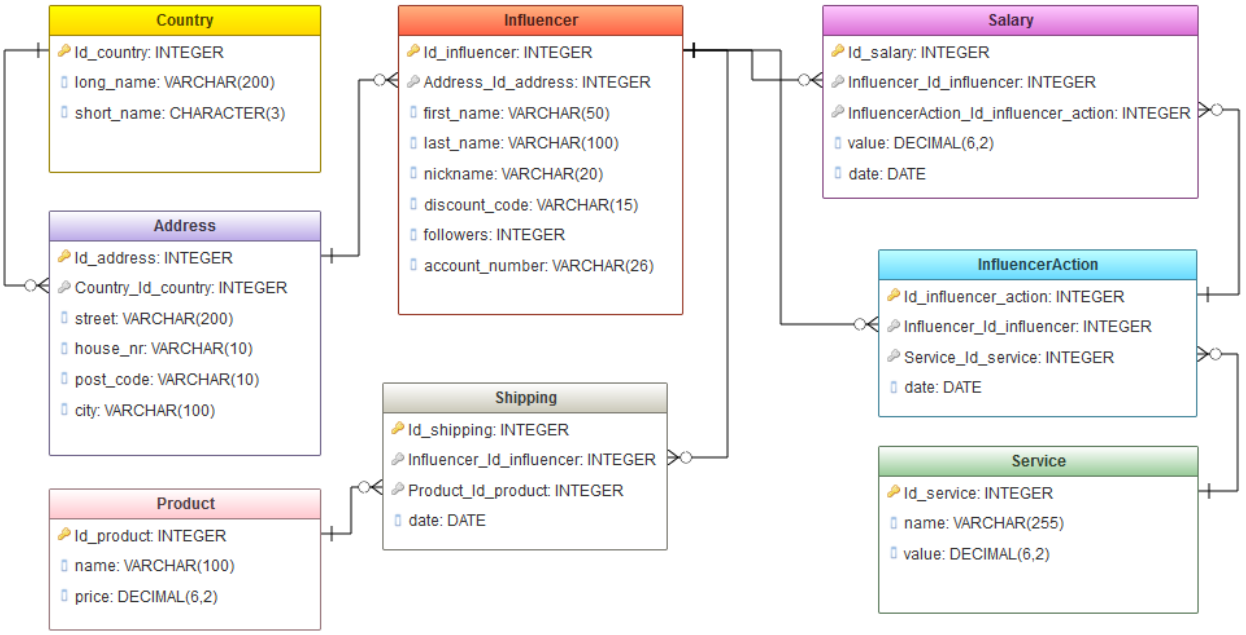
Diagramy ERD spotyka się w wielu różnych notacjach, np. Martina, Bachmana, Chena, IDEFIX. W moim laboratoryjnym przypadku narzędzie GenMyModel generuje diagram ERD w notacji Martina. Encje przedstawione są za pomocą prostokątów zawierających listę atrybutów. Klucze główne oznaczone są przez żółtą ikonę klucza





2. Wprowadzenie relacji pomiędzy encjami

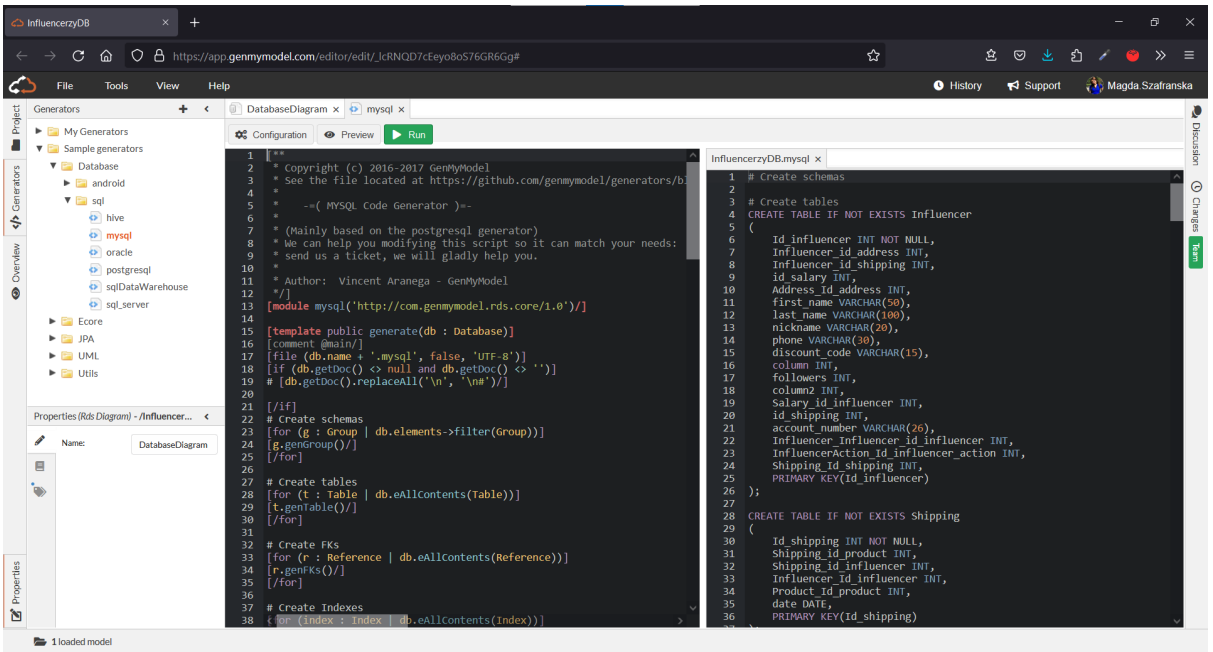
Po wprowadzeniu relacji pomiędzy tabelami zostały dodane klucze obce. Poniżej diagram po wprowadzeniu relacji wraz z opcjonalnością związku oraz pokazaniem rodzaju relacji.



Tak przygotowany diagram ERD pozwala na późniejszą weryfikację i optymalizację bazy danych, a także stanowi podstawową dokumentację projektowanej bazy danych.

3. Generowanie kodu SQL

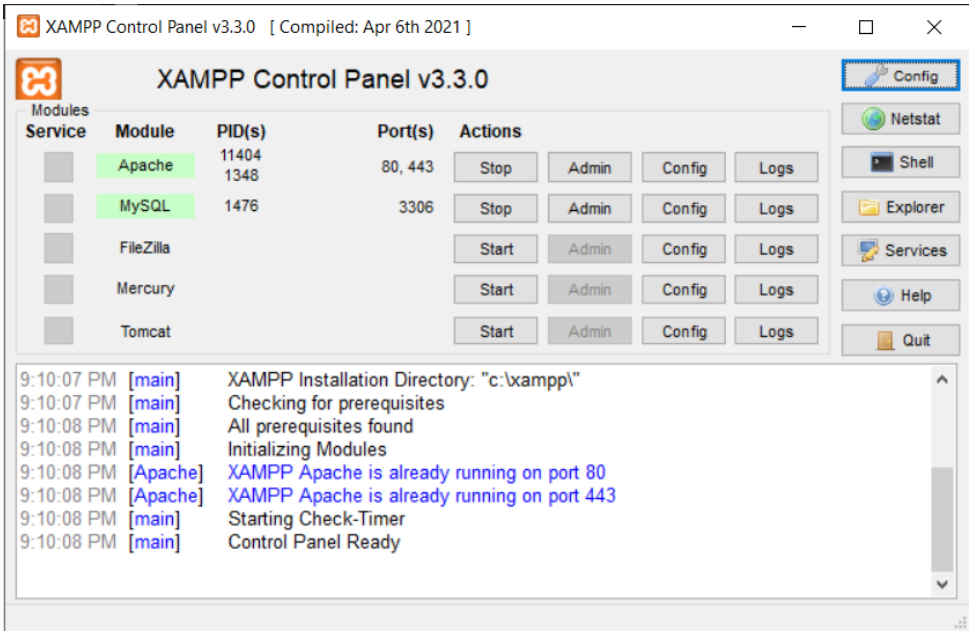
Po zakończeniu projektowania bazy danych generuję kod MySQL.



Program GenMyModel w bieżącej dostępnej wersji przechowuje w wygenerowanym skrypcie całą historię, kiedykolwiek dodane atrybuty tabel, ich relacje itp. Mając to na uwadzę kopiuję powstały kod i poprawiam powstałe w nim błędy. Całkowity kod po poprawkach znajduje się poniżej w sekcji .

4. Uruchomienie serwera bazodanowego w narzędziu XAMPP

W panelu kontrolnym pakietu XAMPP uruchamiam usługę “Apache” oraz “MySQL”. Po upewnieniu się, że serwer jest włączony wraz z usługą “MySQL”, zamykam panel bo on i tak pozostanie działający w tle. Następnie przechodzę do pakietu phpMyAdmin.



Baza danych w serwerze lokalnym XAMPP

1. Uruchomienie narzędzia phpMyAdmin

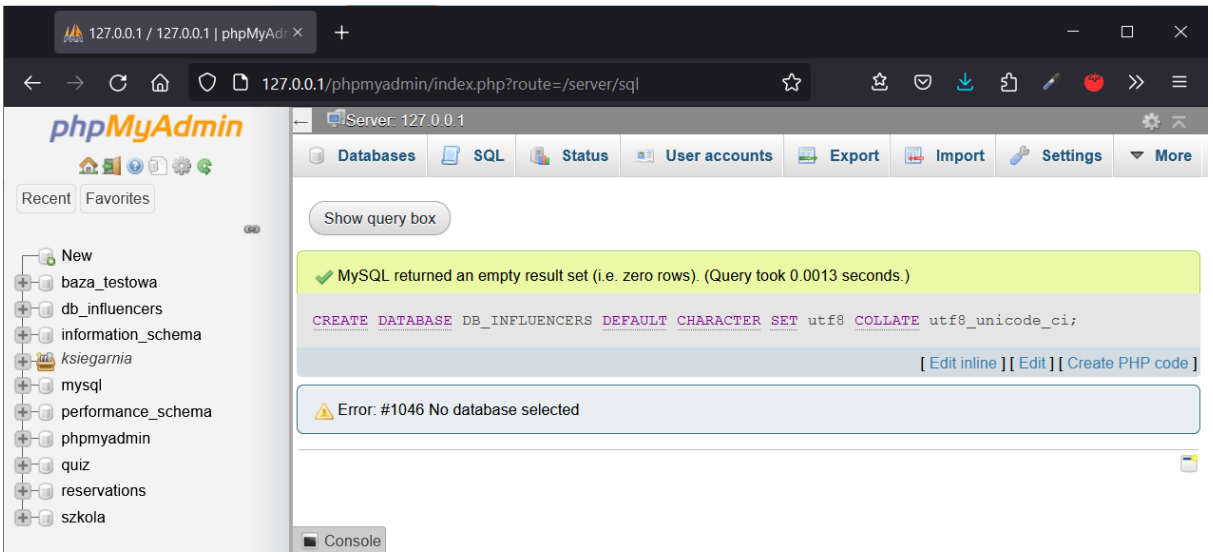
Do wykonania niniejszego projekty użyłam zainstalowanego na moim komputerze serwera lokalnego (a więc widocznego tylko dla mnie). XAMPP symuluje właśnie taki lokalny serwer dzięki specjalnemu adresowi sieciowemu w mojej karcie sieciowej: 127.0.0.1 (tzw. localhost). Wszystkie operacje odbywać się będą lokalnie na moim dysku, bez żadnych opóźnień.
Taki lokalny serwer posłuży mi idealnie do nauki treści z wykładów i wykonania projektu zaliczeniowego. Nic nie stoi na przeszkodzie aby później gotowy serwis przenieść z mojego lokalnego serwera na serwer działający w Internecie i dostępny już dla wszystkich internautów.

2. Utworzenie bazy danych

W narzędziu phpMyAdmin utworzyłam nową bazę danych o nazwie DB_Influencers. Aby zapewnić poprawność wyświetlania polskich znaków zastosowałam dodatkowe polecenia SQL:

CREATE DATABASE DB_Influencers DEFAULT CHARACTER SET utf8 COLLATE utf8_unicode_ci;

Baza danych została prawidłowo utworzona.



3. Stworzenie encji

Skrypt wygenerowany w genMyModel.com podzieliłam na dwie części. Najpierw w oknie SQL wykonałam skrypt dotyczący tworzenia tabel. Kod przedstawiam poniżej. Po zatwierdzeniu tabele zostały pomyślnie utworzone, co również przedstawia poniższy zrzut ekranu.

```sql
# Create tables
CREATE TABLE IF NOT EXISTS Influencer
(
    Id_influencer INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    Address_Id_address INT,
    first_name VARCHAR(50),
    last_name VARCHAR(100),
    nickname VARCHAR(20),
    discount_code VARCHAR(15),
    followers INT,
    account_number VARCHAR(26)
);

CREATE TABLE IF NOT EXISTS Shipping
(
    Id_shipping INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    Influencer_Id_influencer INT,
    Product_Id_product INT,
    date DATE
);

CREATE TABLE IF NOT EXISTS Country
(
    Id_country INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    long_name VARCHAR(200),
    short_name CHARACTER(3)
);

CREATE TABLE IF NOT EXISTS Salary
(
    Id_salary INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    Influencer_Id_influencer INT,
    InfluencerAction_Id_influencer_action INT,
    value DECIMAL(6, 2),
    date DATE
);

CREATE TABLE IF NOT EXISTS Service
(
    Id_service INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(255),
    value DECIMAL(6, 2)
);

CREATE TABLE IF NOT EXISTS InfluencerAction
(
    Id_influencer_action INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    Influencer_Id_influencer INT,
    Service_Id_service INT,
    date DATE
);

CREATE TABLE IF NOT EXISTS Product
(
    Id_product INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100),
    price DECIMAL(6, 2)
);

CREATE TABLE IF NOT EXISTS Address
(
    Id_address INT NOT NULL PRIMARY KEY AUTO_INCREMENT,
    Country_Id_country INT,
    street VARCHAR(200),
    house_nr VARCHAR(10),
    post_code VARCHAR(10),
    city VARCHAR(100)
);
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0130 seconds.)

```
# Create tables CREATE TABLE IF NOT EXISTS Influencer ( Id_influencer INT NOT NULL PRIMARY KEY AUTO_INCREMENT, Address_Id_address INT,
first_name VARCHAR(50), last_name VARCHAR(100), nickname VARCHAR(20), discount_code VARCHAR(15), followers INT, account_number VARCHAR(26) );
```

[Edit inline] [Edit] [Create PHP code]

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0118 seconds.)

```
CREATE TABLE IF NOT EXISTS Shipping ( Id_shipping INT NOT NULL PRIMARY KEY AUTO_INCREMENT, Influencer_Id_influencer INT, Product_Id_product
INT, date DATE );
```

[Edit inline] [Edit] [Create PHP code]

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0115 seconds.)

```
CREATE TABLE IF NOT EXISTS Country ( Id_country INT NOT NULL PRIMARY KEY AUTO_INCREMENT, long_name VARCHAR(200), short_name CHARACTER(3) );
```

[Edit inline] [Edit] [Create PHP code]

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0177 seconds.)

```
CREATE TABLE IF NOT EXISTS Salary ( Id_salary INT NOT NULL PRIMARY KEY AUTO_INCREMENT, Influencer_Id_influencer INT,
InfluencerAction_Id_influencer_action INT, value DECIMAL(6, 2), date DATE );
```

[Edit inline] [Edit] [Create PHP code]

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0112 seconds.)

```
CREATE TABLE IF NOT EXISTS Service ( Id_service INT NOT NULL PRIMARY KEY AUTO_INCREMENT, name VARCHAR(255), value DECIMAL(6, 2) );
```

[Edit inline] [Edit] [Create PHP code]

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0100 seconds.)

```
CREATE TABLE IF NOT EXISTS InfluencerAction ( Id_influencer_action INT NOT NULL PRIMARY KEY AUTO_INCREMENT, Influencer_Id_influencer INT,
Service_Id_service INT, date DATE );
```

[Edit inline] [Edit] [Create PHP code]

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0101 seconds.)

```
CREATE TABLE IF NOT EXISTS Product ( Id_product INT NOT NULL PRIMARY KEY AUTO_INCREMENT, name VARCHAR(100), price DECIMAL(6, 2) );
```

[Edit inline] [Edit] [Create PHP code]

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0102 seconds.)

```
CREATE TABLE IF NOT EXISTS Address ( Id_address INT NOT NULL PRIMARY KEY AUTO_INCREMENT, Country_Id_country INT, street VARCHAR(200), house_nr
VARCHAR(10), post_code VARCHAR(10), city VARCHAR(100) );
```

[Edit inline] [Edit] [Create PHP code]

4. Stworzenie relacji pomiędzy encjami

Dodałam tę część wygenerowanego wcześniej skryptu odpowiadającą za utworzenie relacji pomiędzy tabelami. Kod poniżej. Po zatwierdzeniu relacje pomiędzy encjami zostały pomyślnie utworzone, co przedstawia poniższy zrzut ekranu.

```
# Create FKs

ALTER TABLE Influencer
    ADD    FOREIGN KEY (Address_Id_address) REFERENCES Address(Id_address)
;

ALTER TABLE Shipping
    ADD    FOREIGN KEY (Influencer_Id_influencer) REFERENCES Influencer(Id_influencer),
    ADD    FOREIGN KEY (Product_Id_product) REFERENCES Product(Id_product)
;

ALTER TABLE Salary
    ADD    FOREIGN KEY (InfluencerAction_Id_influencer_action) REFERENCES InfluencerAction(Id_influencer_action),
    ADD    FOREIGN KEY (Influencer_Id_influencer) REFERENCES Influencer(Id_influencer)
;

ALTER TABLE InfluencerAction
    ADD    FOREIGN KEY (Influencer_Id_influencer) REFERENCES Influencer(Id_influencer),
    ADD    FOREIGN KEY (Service_Id_service) REFERENCES Service(Id_service)
;

ALTER TABLE Address
    ADD    FOREIGN KEY (Country_Id_country) REFERENCES Country(Id_country)
;
```

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0452 seconds.)

```
# Create FKs ALTER TABLE Influencer ADD FOREIGN KEY (Address_Id_address) REFERENCES Address(Id_address);
```

[Edit inline] [Edit] [Create PHP code]

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0502 seconds.)

```
ALTER TABLE Shipping ADD FOREIGN KEY (Influencer_Id_influencer) REFERENCES Influencer(Id_influencer), ADD FOREIGN KEY (Product_Id_product)
REFERENCES Product(Id_product);
```

[Edit inline] [Edit] [Create PHP code]

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0431 seconds.)

```
ALTER TABLE Salary ADD FOREIGN KEY (InfluencerAction_Id_influencer_action) REFERENCES InfluencerAction(Id_influencer_action), ADD FOREIGN KEY
(Influencer_Id_influencer) REFERENCES Influencer(Id_influencer);
```

[Edit inline] [Edit] [Create PHP code]

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0450 seconds.)

```
ALTER TABLE InfluencerAction ADD FOREIGN KEY (Influencer_Id_influencer) REFERENCES Influencer(Id_influencer), ADD FOREIGN KEY
(Service_Id_service) REFERENCES Service(Id_service);
```

[Edit inline] [Edit] [Create PHP code]

MySQL returned an empty result set (i.e. zero rows). (Query took 0.0423 seconds.)

```
ALTER TABLE Address ADD FOREIGN KEY (Country_Id_country) REFERENCES Country(Id_country);
```

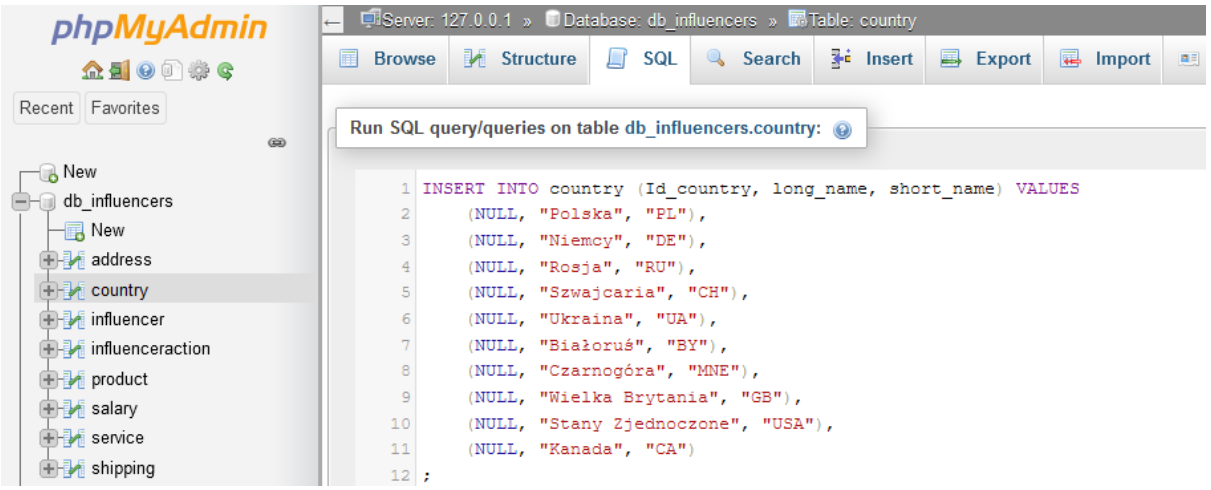
[Edit inline] [Edit] [Create PHP code]

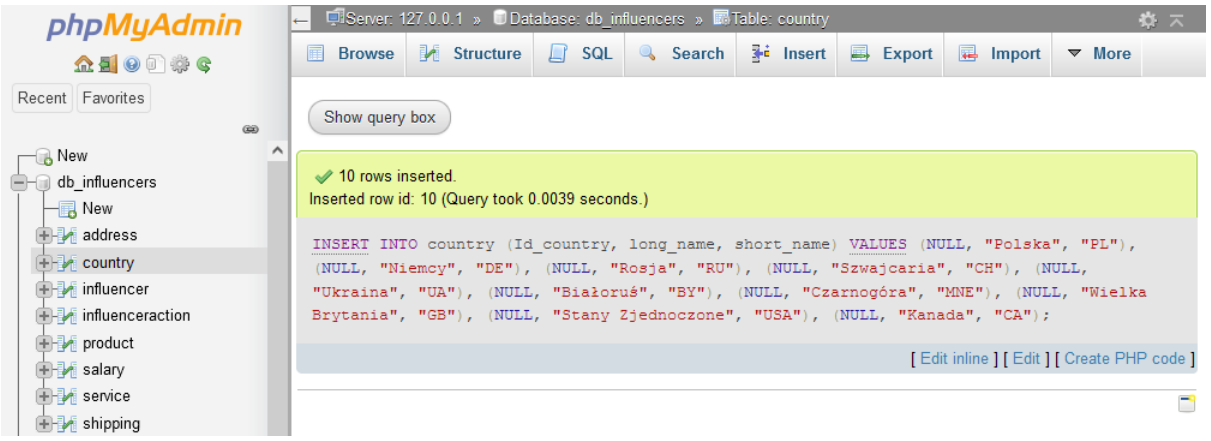
Wprowadzenie danych do bazy

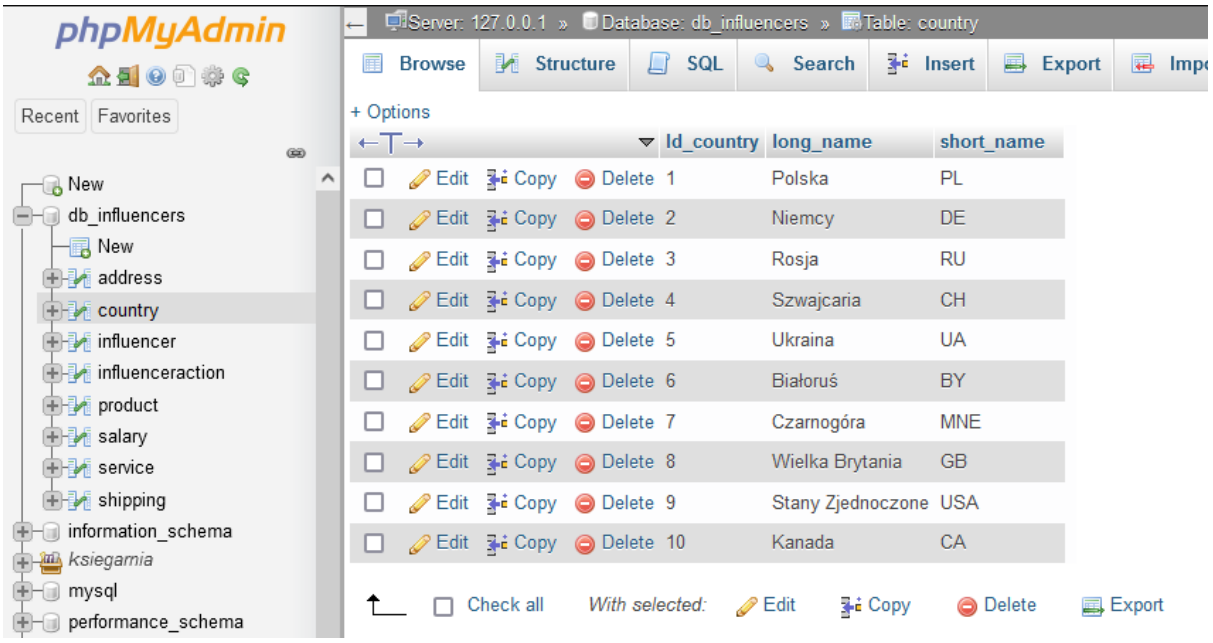
Używając języka SQL uzupełniłam danymi zaprojektowane tabele. Do każdej z nich wprowadzam co najmniej po 10 rekordów.

- **Tabela “Country”**

Po wprowadzeniu skrypty SQL dotyczącego tabeli “Country”, wprowadzone dane zostały poprawnie dodane. Skrypt oraz zrzuty ekranu poniżej. Dla pozostałych tabel proces był analogiczny więc zamieszczę jedynie skrypt SQL dla każdej z nich.








```
# Create data in Country

INSERT INTO country (Id_country, long_name, short_name) VALUES
    (NULL, "Polska", "PL"),
    (NULL, "Niemcy", "DE"),
    (NULL, "Rosja", "RU"),
    (NULL, "Szwajcaria", "CH"),
    (NULL, "Ukraina", "UA"),
    (NULL, "Białoruś", "BY"),
    (NULL, "Czarnogóra", "MNE"),
    (NULL, "Wielka Brytania", "GB"),
    (NULL, "Stany Zjednoczone", "USA"),
    (NULL, "Kanada", "CA")
;
```

- **Tabela “Address”**

```
INSERT INTO `address` (`Id_address`, `Country_Id_country`, `street`, `house_nr`, `post_code`, `city`) VALUES
(1, 1, 'Puławska', '38/15', '03-232', 'Warszawa'),
(2, 2, 'Isern-Hinnerk-Weg', '14B', '22457', 'Hamburg'),
(3, 8, 'High Park Road', '20A', 'PR9 7QL', 'Southport'),
(4, 1, 'Jurajska', '4/59', '02-699', 'Warszawa'),
(5, 1, 'Kadłubka', '42/5', '71-524', 'Szczecin'),
(6, 1, 'Aleja Krakowska', '49', '05-090', 'Sękocin Stary'),
(7, 1, 'Nowowiejska', '15', '06-500', 'Mława'),
(8, 8, 'Saxonbury Way', '94', 'PE2 9FB', 'Cambridgeshire'),
(9, 1, 'Marii Dąbrowskiej', '96/9', '97-300', 'Piotrków Trybunalski'),
(10, 1, 'Niemeńska', '66', '60-412', 'Poznań');
```

- **Tabela “Product”**

```sql
INSERT INTO `product` (`Id_product`, `name`, `price`) VALUES
(1, 'Modify Reductor', '129.00'),
(2, 'Modify Femibra', '139.00'),
(3, 'Gumka do włosów - lniana ', '29.00'),
(4, 'Gumka do włosów - lniana XL', '49.00'),
(5, 'Gumka do włosów - welurowa', '19.00'),
(6, 'Gumka do włosów - welurowa XL', '29.00'),
(7, 'Pełna kuracja Modify Reductor', '387.00'),
(8, 'Pełna kuracja Modify Femibra', '417.00'),
(9, 'Zestaw świąteczny Modify Reductor', '199.00'),
(10, 'Zestaw świąteczny Modify Femibra', '256.00');
```

- **Tabela “Service”**

```sql
INSERT INTO `service` (`Id_service`, `name`, `value`) VALUES
(1, 'Relacja ', '50.00'),
(2, 'Relacja + swipe', '150.00'),
(3, 'Post bez produktu', '100.00'),
(4, 'Post z produktem', '250.00'),
(5, 'Udział w evencie marki', '500.00'),
(6, 'Logo marki na stronie', '200.00'),
(7, 'Oznaczenie na story', '100.00'),
(8, 'Użycie kodu influencera', '30.00'),
(9, 'Polecenie zakupu bez kodu', '40.00'),
(10, 'Pakiet (post + relacja + swipe)', '350.00');
```

- **Tabela “Influencers”**

```sql
INSERT INTO `influencer` (`Id_influencer`, `Address_Id_address`, `first_name`, `last_name`, `nickname`, `discount_code`, `followers`, `account_number`) VALUES
(1, 4, 'Ismena', 'Stelmaszczyk', 'ismena_stelmaszczyk', 'ismena15', 47600, NULL),
(2, 2, 'Paulina', 'Guzińska', 'paulina_guzinska', 'paulina15', 264000, NULL),
(3, 5, 'Bartosz', 'Smęda', 'smeda.triathlon', 'triathlon15', 801, NULL),
(4, 3, 'John', 'Smith', 'john_smith', 'john15', 45000, '12445588237595145212547719'),
(5, 1, 'Maria', 'Wrześniak', 'mariaaa', 'marysia15', 53000, NULL),
(6, 6, 'Klaudia', 'Michalak', 'klaudia_michalak', 'klaudia15', 4123, NULL),
(7, 7, 'Aleksander', 'Mąkosa', 'alexi', 'alexi15', 89000, NULL),
(8, 8, 'Aleksa', 'Woźnicki', 'aleksa_woznikci', 'woznikci15', 632, NULL),
(9, 9, 'Rafał', 'Piotrowski', 'rafal_p', 'rafal15', 7522, NULL),
(10, 10, 'Marcin', 'Wiesiuk', 'marcinek_w', 'marcinek15', 65002, NULL);
```

- **Tabela “Shipping”**

```sql
INSERT INTO `shipping` (`Id_shipping`, `Influencer_Id_influencer`, `Product_Id_product`, `date`) VALUES
(1, 1, 7, '2021-09-08'),
(2, 2, 7, '2021-11-07'),
(3, 3, 9, '2021-08-18'),
(4, 4, 1, '2021-11-19'),
(5, 5, 2, '2021-10-18'),
(6, 5, 5, '2021-10-18'),
(7, 6, 10, '2020-12-12'),
(8, 7, 9, '2021-11-17'),
(9, 8, 2, '2021-07-11'),
(10, 9, 1, '2021-11-06'),
(11, 10, 9, '2021-11-15'),
(12, 1, 4, '2021-11-08');
```

- **Tabela “InfluencerAction”**

```sql
INSERT INTO `influenceraction` (`Id_influencer_action`, `Influencer_Id_influencer`, `Service_Id_service`, `date`) VALUES
(1, 1, 8, '2021-09-30'),
(2, 1, 8, '2021-09-30'),
(3, 1, 1, '2021-09-15'),
(4, 2, 7, '2021-10-13'),
(5, 4, 10, '2021-11-02'),
(6, 3, 10, '2021-11-14'),
(7, 5, 3, '2021-10-19'),
(8, 7, 5, '2021-11-02'),
(9, 10, 9, '2021-11-03'),
(10, 8, 6, '2021-08-10');
```

- **Tabela “Salary”**

```sql
INSERT INTO `salary` (`Id_salary`, `Influencer_Id_influencer`, `InfluencerAction_Id_influencer_action`, `value`, `date`) VALUES
(1, 1, 1, '30.00', '2021-09-30'),
(2, 1, 2, '30.00', '2021-09-30'),
(3, 1, 3, '50.00', '2021-09-15'),
(4, 2, 4, '100.00', '2021-10-13'),
(5, 4, 5, '350.00', '2021-11-12'),
(6, 3, 6, '350.00', '2021-11-14'),
(7, 5, 7, '100.00', '2021-10-19'),
(8, 7, 8, '500.00', '2021-11-02'),
(9, 10, 9, '40.00', '2021-11-03'),
(10, 8, 10, '200.00', '2021-08-10');
```

Zapytania SQL

Używając języka SQL stworzyłam 10 dowolnych zapytań do zaprojektowanej bazy. Podczas tworzenia zapytań złożonych dostarczyłam listę wszystkich relacji, które zachodzą pomiędzy używanymi w tym zapytaniu tabelami. Tę listę umieszczałam po klauzuli WHERE. W przykładach użyłam zarówno zapytań złożonych jak i skorelowanych.

1. Influencerzy posortowani malejąco wg liczby followersów (ORDER BY)

POLECENIE SQL:
SELECT inf.first_name, inf.last_name, inf.followers FROM influencer AS inf ORDER BY inf.followers DESC;

Showing rows 0 - 9 (10 total, Query took 0.0004 seconds.) [followers: **264000... - 632...**]

```
SELECT inf.first_name, inf.last_name, inf.followers FROM influencer AS inf ORDER BY inf.followers DESC;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

←T→			first_name	last_name	followers 1
Edit	Copy	Delete	Paulina	Guzińska	264000
Edit	Copy	Delete	Aleksander	Mąkosa	89000
Edit	Copy	Delete	Marcin	Wiesiuk	65002
Edit	Copy	Delete	Maria	Wrześniak	53000
Edit	Copy	Delete	Ismena	Stelmaszczyk	47600
Edit	Copy	Delete	John	Smith	45000
Edit	Copy	Delete	Rafał	Piotrowski	7522
Edit	Copy	Delete	Klaudia	Michalak	4123
Edit	Copy	Delete	Bartosz	Smęda	801
Edit	Copy	Delete	Aleksa	Woźnicki	632

2. Akcje wykonane przez influencerów w listopadzie (BETWEEN)

POLECENIE SQL:

SELECT * FROM influenceraction WHERE influenceraction.date BETWEEN "2021-11-01" AND "2021-11-30";

Showing rows 0 - 3 (4 total, Query took 0.0013 seconds.)

```
SELECT * FROM influenceraction WHERE influenceraction.date BETWEEN "2021-11-01" AND
"2021-11-30";
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

	Id_influencer_action	Influencer_Id_influencer	Service_Id_service	date
Edit Copy Delete	5	4	10	2021-11-02
Edit Copy Delete	6	3	10	2021-11-14
Edit Copy Delete	8	7	5	2021-11-02
Edit Copy Delete	9	10	9	2021-11-03

Check all *With selected:* Edit Copy Delete Export

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

3. Wszystkie produkty z “Reductor” w nazwie (LIKE)

POLECENIE SQL:
SELECT * FROM product WHERE product.name LIKE "%Reductor%";

Showing rows 0 - 2 (3 total, Query took 0.0004 seconds.)

```
SELECT * FROM product WHERE product.name LIKE "%Reductor%";
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table S

+ Options

	Id_product	name	price
Edit Copy Delete	1	Modify Reductor	129.00
Edit Copy Delete	7	Pełna kuracja Modify Reductor	387.00
Edit Copy Delete	9	Zestaw świąteczny Modify Reductor	199.00

Check all *With selected:* Edit Copy Delete

4. Jaka wysyłka i kiedy trafiła do influencerów (AS)

POLECENIE SQL:
SELECT sh.Id_shipping, inf.first_name, inf.last_name, sh.Product_Id_product, sh.date
FROM influencer AS inf, shipping AS sh

WHERE inf.Id_influencer = sh.Influencer_Id_influencer;

Showing rows 0 - 11 (12 total, Query took 0.0005 seconds.)

```
SELECT sh.Id_shipping, inf.first_name, inf.last_name,
sh.Product_Id_product, sh.date FROM influencer AS inf, shipping AS sh
WHERE inf.Id_influencer = sh.Influencer_Id_influencer;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

Id_shipping	first_name	last_name	Product_Id_product	date
1	Ismena	Stelmaszczyk	7	2021-09-08
2	Paulina	Guzińska	7	2021-11-07
3	Bartosz	Smęda	9	2021-08-18
4	John	Smith	1	2021-11-19
5	Maria	Wrześniak	2	2021-10-18
6	Maria	Wrześniak	5	2021-10-18
7	Klaudia	Michalak	10	2020-12-12
8	Aleksander	Mąkosa	9	2021-11-17
9	Aleksa	Woźnicki	2	2021-07-11
10	Rafał	Piotrowski	1	2021-11-06
11	Marcin	Wiesiuk	9	2021-11-15
12	Ismena	Stelmaszczyk	4	2021-11-08

Show all | Number of rows: 25 Filter rows: Search this table

5. Funkcje wbudowane SQL

a. ilość wszystkich produktów (COUNT)

POLECENIE SQL:
SELECT COUNT(*) FROM product;

Your SQL query has been executed successfully.

```
SELECT COUNT(*) FROM product;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

+ Options

COUNT(*)
10

b. średni koszt możliwych do wykonania przez influencera akcji

POLECENIE SQL:
SELECT AVG(value) FROM service;

Showing rows 0 - 0 (1 total, Query took 0.0004 seconds.)

```
SELECT AVG(value) FROM service;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

AVG(value)
177.000000

6. Wysyłki konkretnych produktów do influencerów

a. wszyscy influencerzy, którym wysłano (LIKE + AND)

POLECENIE SQL:

SELECT inf.first_name, inf.last_name, p.name, sh.date
FROM influencer AS inf, product AS p, shipping AS sh
WHERE (p.name LIKE "%Reductor%" OR p.name LIKE "%Femibra%")
AND inf.Id_influencer = sh.Influencer_Id_influencer
AND sh.Product_Id_product = p.Id_product
LIMIT 5;

Showing rows 0 - 9 (10 total, Query took 0.0005 seconds.)

```
SELECT inf.first_name, inf.last_name, p.name, sh.date FROM influencer
AS inf, product AS p, shipping AS sh WHERE (p.name LIKE "%Reductor%"
OR p.name LIKE "%Femibra%") AND inf.Id_influencer =
sh.Influencer_Id_influencer AND sh.Product_Id_product = p.Id_product;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

first_name	last_name	name	date
Ismena	Stelmaszczyk	Pełna kuracja Modify Reductor	2021-09-08
Paulina	Guzińska	Pełna kuracja Modify Reductor	2021-11-07
Bartosz	Smęda	Zestaw świąteczny Modify Reductor	2021-08-18
John	Smith	Modify Reductor	2021-11-19
Maria	Wrześniak	Modify Femibra	2021-10-18
Klaudia	Michalak	Zestaw świąteczny Modify Femibra	2020-12-12
Aleksander	Mąkosa	Zestaw świąteczny Modify Reductor	2021-11-17
Aleksa	Woźnicki	Modify Femibra	2021-07-11
Rafał	Piotrowski	Modify Reductor	2021-11-06
Marcin	Wiesiuk	Zestaw świąteczny Modify Reductor	2021-11-15

Show all | Number of rows: 25 Filter rows: Search this table

b. ograniczenie ilości do 5 (LIMIT)

POLECENIE SQL:

SELECT inf.first_name, inf.last_name, p.name, sh.date
FROM influencer AS inf, product AS p, shipping AS sh
WHERE (p.name LIKE "%Reductor%" OR p.name LIKE "%Femibra%")
AND inf.Id_influencer = sh.Influencer_Id_influencer
AND sh.Product_Id_product = p.Id_product;

Showing rows 0 - 4 (5 total, Query took 0.0004 seconds.)

```
SELECT inf.first_name, inf.last_name, p.name, sh.date FROM influencer AS inf, product AS p,
shipping AS sh WHERE (p.name LIKE "%Reductor%" OR p.name LIKE "%Femibra%") AND
inf.Id_influencer = sh.Influencer_Id_influencer AND sh.Product_Id_product = p.Id_product LIMIT
5;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

+ Options

first_name	last_name	name	date
Ismenka	Stelmaszczyk	Pełna kuracja Modify Reductor	2021-09-08
Paulina	Guzińska	Pełna kuracja Modify Reductor	2021-11-07
Bartosz	Smęda	Zestaw świąteczny Modify Reductor	2021-08-18
John	Smith	Modify Reductor	2021-11-19
Maria	Wrześniak	Modify Femibra	2021-10-18

Query results operations

7. Wszystkie wysyłki do określonego influencera

POLECENIE SQL:

SELECT inf.first_name, inf.last_name, sh.date, p.name
FROM influencer AS inf, shipping AS sh, product AS p
WHERE inf.Id_influencer = 1
AND inf.Id_influencer = sh.Influencer_Id_influencer
AND sh.Product_Id_product = p.Id_product;

Showing rows 0 - 1 (2 total, Query took 0.0006 seconds.)

```
SELECT inf.first_name, inf.last_name, sh.date, p.name FROM influencer AS inf,
shipping AS sh, product AS p WHERE inf.Id_influencer = 1 AND
inf.Id_influencer = sh.Influencer_Id_influencer AND sh.Product_Id_product =
p.Id_product;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

first_name	last_name	date	name
Ismena	Stelmaszczyk	2021-09-08	Pełna kuracja Modify Reductor
Ismena	Stelmaszczyk	2021-11-08	Gumka do włosów - Iniana XL

Show all | Number of rows: 25 Filter rows: Search this table

8. Aktualizacja imienia influencera (UPDATE)

POLECENIE SQL:
UPDATE influencer SET influencer.first_name = "Ismenka"
WHERE influencer.Id_influencer = 1;

1 row affected. (Query took 0.0033 seconds.)

```
UPDATE influencer SET influencer.first_name = "Ismenka"
WHERE influencer.Id_influencer = 1;
```

[Edit inline] [Edit] [Create PHP code]

Wykonanie ponownie skryptu z poprzedniego punktu dla sprawdzenia.

Showing rows 0 - 1 (2 total, Query took 0.0007 seconds.)

```
SELECT inf.first_name, inf.last_name, sh.date, p.name FROM influencer AS inf,
shipping AS sh, product AS p WHERE inf.Id_influencer = 1 AND inf.Id_influencer
= sh.Influencer_Id_influencer AND sh.Product_Id_product = p.Id_product;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

first_name	last_name	date	name
Ismenka	Stelmaszczyk	2021-09-08	Pełna kuracja Modify Reductor
Ismenka	Stelmaszczyk	2021-11-08	Gumka do włosów - Iniana XL

Show all | Number of rows: 25 Filter rows: Search this table

9. Influencerzy spoza Polski (NOT IN)

POLECENIE SQL:
SELECT inf.first_name, inf.last_name, a.street, a.house_nr, a.post_code, a.city, c.long_name, c.short_name
FROM influencer AS inf, country AS c, address AS a
WHERE c.short_name NOT IN ("PL")
AND inf.Address_Id_address = a.Id_address
AND a.Country_Id_country = c.Id_country;

Showing rows 0 - 2 (3 total, Query took 0.0005 seconds.)

```
SELECT inf.first_name, inf.last_name, a.street, a.house_nr, a.post_code, a.city, c.long_name,
c.short_name FROM influencer AS inf, country AS c, address AS a WHERE c.short_name NOT IN ("PL")
AND inf.Address_Id_address = a.Id_address AND a.Country_Id_country = c.Id_country;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

first_name	last_name	street	house_nr	post_code	city	long_name	short_name
Paulina	Guzińska	Isern-Hinnerk-Weg	14B	22457	Hamburg	Niemcy	DE
John	Smith	High Park Road	20A	PR9 7QL	Southport	Wielka Brytania	GB
Aleksa	Woźnicki	Saxonbury Way	94	PE2 9FB	Cambridgeshire	Wielka Brytania	GB

Show all | Number of rows: 25 Filter rows: Search this table

10. Akcje wykonane przez influencerów

a. influencerzy, którzy nie wykonali żadnej akcji (LEFT JOIN)

POLECENIE SQL:
SELECT i.Id_influencer, i.first_name, i.last_name

FROM influencer AS i LEFT JOIN influenceraction AS ia
ON i.Id_influencer = ia.Influencer_Id_influencer
WHERE ia.Influencer_Id_influencer IS NULL;

Showing rows 0 - 1 (2 total, Query took 0.0004 seconds.)

```
SELECT i.Id_influencer, i.first_name, i.last_name FROM influencer AS i LEFT JOIN influenceraction AS ia
ON i.Id_influencer = ia.Influencer_Id_influencer WHERE ia.Influencer_Id_influencer IS NULL;
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

Id_influencer	first_name	last_name
6	Klaudia	Michalak
9	Rafał	Piotrowski

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

b. influencerzy, którzy wykonali jakąkolwiek akcję (INNER JOIN)

POLECENIE SQL:
SELECT * FROM influencer AS i
INNER JOIN influenceraction AS ia
ON i.Id_influencer = ia.Influencer_Id_influencer;

Showing rows 0 - 9 (10 total, Query took 0.0004 seconds.) [Id_influencer: **1... - 10...**]

```
SELECT i.Id_influencer, i.first_name, i.last_name, ia.Service_Id_service, ia.date FROM influencer AS i
INNER JOIN influenceraction AS ia ON i.Id_influencer = ia.Influencer_Id_influencer ORDER BY
`i`.`Id_influencer` ASC
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

Id_influencer ▲ 1	first_name	last_name	Service_Id_service	date
1	Ismenka	Stelmaszczyk	8	2021-09-30
1	Ismenka	Stelmaszczyk	8	2021-09-30
1	Ismenka	Stelmaszczyk	1	2021-09-15
2	Paulina	Guzińska	7	2021-10-13
3	Bartosz	Smęda	10	2021-11-14
4	John	Smith	10	2021-11-02
5	Maria	Wrześniak	3	2021-10-19
7	Aleksander	Mąkosa	5	2021-11-02
8	Aleksa	Woźnicki	6	2021-08-10
10	Marcin	Wiesiuk	9	2021-11-03

Show all | Number of rows: 25 Filter rows: Search this table

11. Wyszukanie produktów nigdy nie wysłanych (NOT EXISTS)

POLECENIE SQL:
SELECT p.name, p.price FROM product AS p WHERE NOT EXISTS
(SELECT * FROM shipping AS s WHERE s.Product_Id_product = p.Id_product);

Showing rows 0 - 2 (3 total, Query took 0.0014 seconds.)

```
SELECT p.name, p.price FROM product AS p WHERE NOT EXISTS (SELECT * FROM shipping AS s
WHERE s.Product_Id_product = p.Id_product);
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table Sort by key: None

+ Options

←T→				name	price
	Edit	Copy	Delete	Gumka do włosów - lniana	29.00
	Edit	Copy	Delete	Gumka do włosów - welurowa XL	29.00
	Edit	Copy	Delete	Pełna kuracja Modify Femibra	417.00

Check all *With selected:* Edit Copy Delete Export

Widoki

Widoki (perspektywy) w języku SQL to wirtualne tabele tworzone na podstawie zapytań. Składają się z kolumn i wierszy pobranych z prawdziwych tabel. Pokazywane w widoku dane są zawsze aktualne, ponieważ widoki są tworzone w momencie wykonania zapytania. Widoki nie przechowują zapisanych w tabelach danych.

Aby zaprezentować tę funkcjonalność stworzyłam widok z odpowiednich kolumn oparty na następujących tabelach:

- tabela “influencer”
    - kolumna “first_name”
    - kolumna “last_name”
- tabela “address”
    - kolumna “street”
    - kolumna “house_nr”
    - kolumna “post_code”
    - kolumna “city”
- tabela “country”
    - kolumna “long_name”

Kod do wygenerowanie widoku:
SELECT i.first_name AS first_name, i.last_name AS last_name, a.street AS street, a.house_nr AS house_nr, a.post_code AS post_code, a.city AS city, c.long_name AS long_name
FROM ((db_influencers_konw.influencer i left join db_influencers_konw.address a on(i.Address_Id_address = a.Id_address)) left join db_influencers_konw.country c on(a.Country_Id_country = c.Id_country))

Stworzony widok wygenerował mi tabelę jak na poniższym zrzucie ekranu.
POLECENIE SQL:
SELECT * FROM full_address_data

Showing rows 0 - 9 (10 total, Query took 0.0005 seconds.)

```
SELECT * FROM `full_address_data`
```

Profiling [Edit inline] [Edit] [Explain SQL] [Create PHP code] [Refresh]

Show all | Number of rows: 25 Filter rows: Search this table

+ Options

	first_name	last_name	street	house_nr	post_code	city	long_name
Edit Copy Delete	Ismenka	Stelmaszczyk	Jurajska	4/59	02-699	Warszawa	Polska
Edit Copy Delete	Paulina	Guzińska	Isern-Hinnerk-Weg	14B	22457	Hamburg	Niemcy
Edit Copy Delete	Bartosz	Smęda	Kadłubka	42/5	71-524	Szczecin	Polska
Edit Copy Delete	John	Smith	High Park Road	20A	PR9 7QL	Southport	Wielka Brytania
Edit Copy Delete	Maria	Wrześniak	Puławska	38/15	03-232	Warszawa	Polska
Edit Copy Delete	Klaudia	Michalak	Aleja Krakowska	49	05-090	Sękocin Stary	Polska
Edit Copy Delete	Aleksander	Mąkosa	Nowowiejska	15	06-500	Mława	Polska
Edit Copy Delete	Aleksa	Woźnicki	Saxonbury Way	94	PE2 9FB	Cambridgeshire	Wielka Brytania
Edit Copy Delete	Rafał	Piotrowski	Marii Dąbrowskiej	96/9	97-300	Piotrków Trybunalski	Polska
Edit Copy Delete	Marcin	Wiesiuk	Niemeńska	66	60-412	Poznań	Polska

Check all *With selected:* Edit Copy Delete Export

Zdarzenia (Triggers)

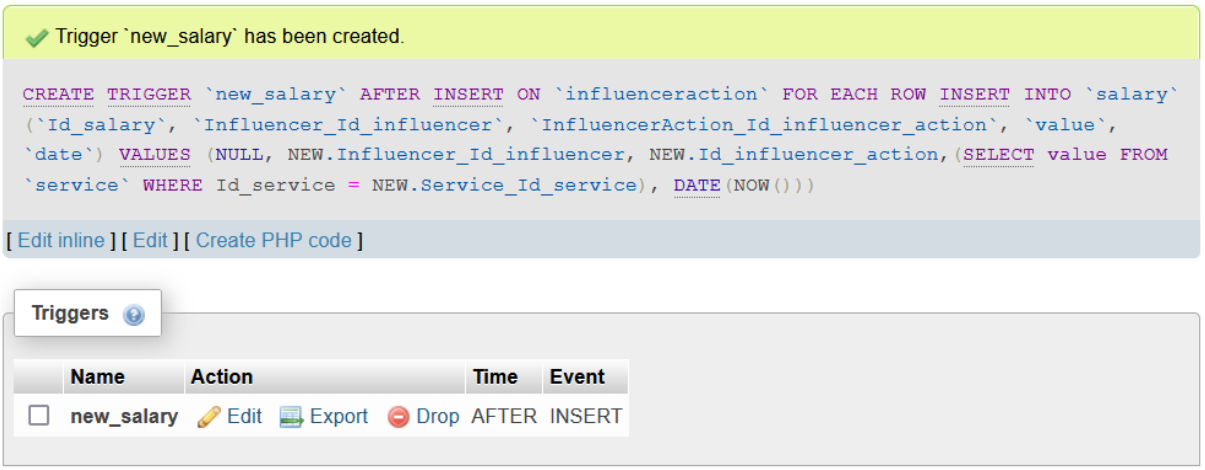
Trigger (zdarzenie) to automatycznie wykonywany program (na skutek zdarzenia w bazie).

Zdarzeniami w mojej bazie będą:

- utworzenie rekordu w tabeli “address” po dodaniu nowego influencera do tabeli “influencer”
- wysłanie id utworzonego rekordu do tabeli “influencer”

Dodaję zdarzenie tworzenia nowego rekordu w tabeli “Salary” przy dodaniu każdej dodanej akcji influencera.

POLECENIE SQL:
CREATE TRIGGER `new_salary`
AFTER INSERT ON `influenceraction`
FOR EACH ROW
INSERT INTO `salary` (`Id_salary`, `Influencer_Id_influencer`, `InfluencerAction_Id_influencer_action`, `value`, `date`)
VALUES (NULL, NEW.Influencer_Id_influencer, NEW.Id_influencer_action, (SELECT value FROM `service` WHERE Id_service = NEW.Service_Id_service), DATE(NOW()))



Dla sprawdzenia działania dodałam rekord w influenceraction. Zdarzenie wygenerowało w tabeli “Salary” rekord z automatycznie zaciągniętą wartością usługi z tabeli “service” co świadczy o poprawności działania skryptu zdarzenia.

Funkcje

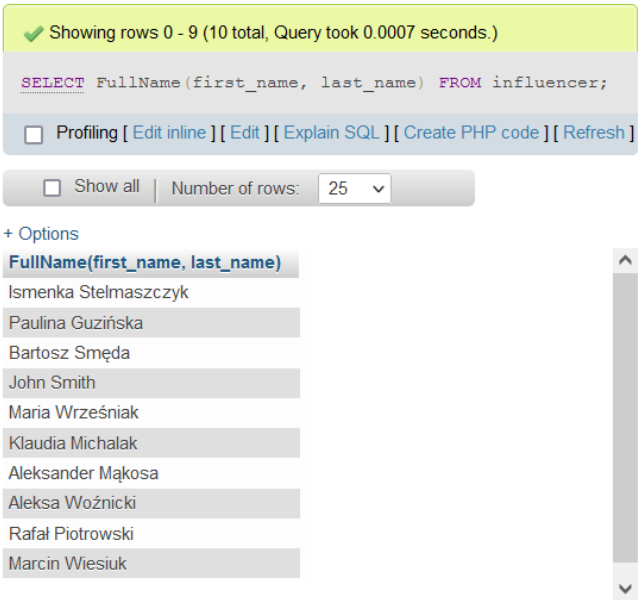
Funkcji używa się, aby tworzyć dedykowane rozwiązania. Zarówno funkcja jak i procedura to ręcznie wykonywane programy. Można je wyjaśnić na podstawie analogii: jeśli chciałabym odczytać pensję influencera, to skorzystałabym z funkcji. Jeśli jednak chciałabym ją zmodyfikować, to skorzystałabym z procedury. To tak jak *gettery* i *settery* w językach programowania.
Funkcje od procedur różnią się jednak tym, że funkcja musi zwracać wartość (w klauzuli return), a procedura może (za pomocą parametru out) ale nie musi. Dodatkowo, funkcje mogą być wywoływane z procedury ale procedura nie może być wywołana z funkcji.

1. Funkcja FullName

Wyświetlanie imienia i nazwiska. Często używam tej funkcji w bazie więc zdecydowałam się na funkcję skracająca mi czas wywoływania imienia i nazwiska osobno za każdym razem.

POLECENIE SQL:

```
CREATE FUNCTION FullName2(first_name VARCHAR(20), last_name VARCHAR(30))
RETURNS VARCHAR(50)
DETERMINISTIC
RETURN CONCAT(first_name, ' ',  last_name);
```



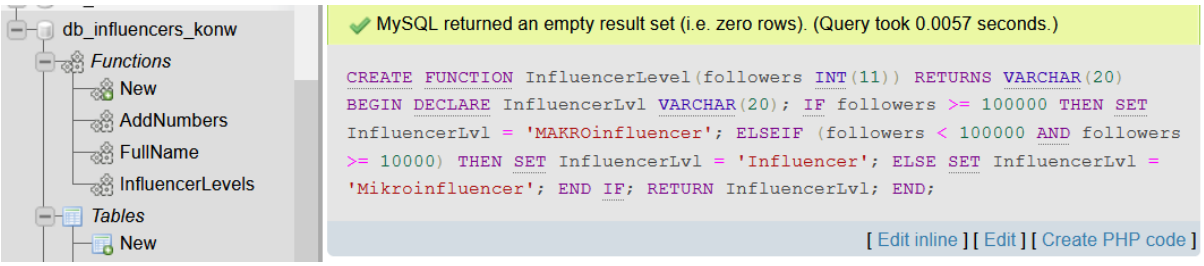
2. Funkcja etykietująca liczebność influencera

Przypisanie influencerowi odpowiedniej “etykiety” w zależności od liczebności jego followersów.

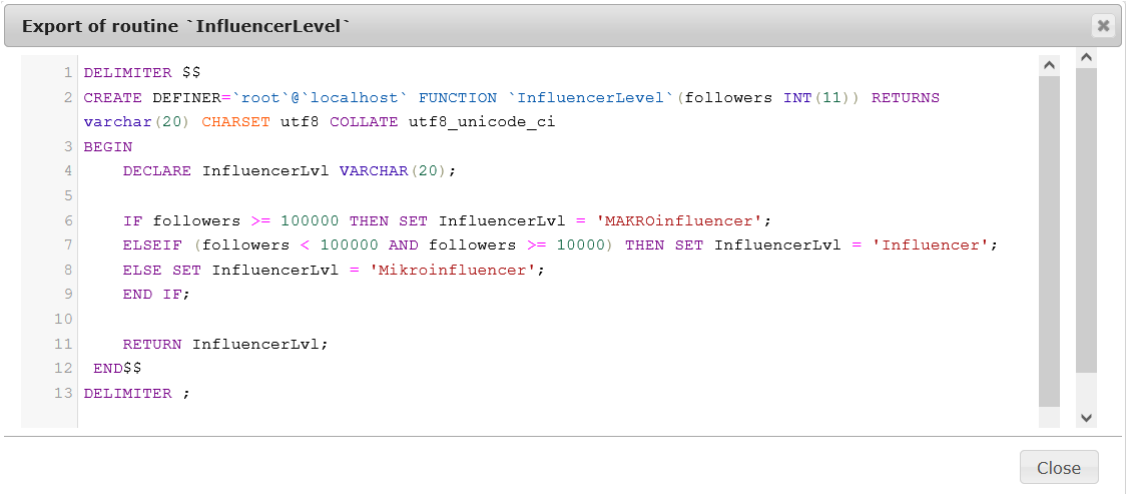
POLECENIE SQL:

```
DELIMITER //
CREATE FUNCTION InfluencerLevel(followers INT(11))
RETURNS VARCHAR(20)

BEGIN
	DECLARE InfluencerLvl VARCHAR(20);

	IF followers >= 100000 THEN SET InfluencerLvl = 'MAKROinfluencer';
	ELSEIF (followers < 100000 AND followers >= 10000) THEN SET InfluencerLvl = 'Influencer';
	ELSE SET InfluencerLvl = 'Mikroinfluencer';
	END IF;

	RETURN InfluencerLvl;
 END //
DELIMITER ;
```

Kod został zaimplementowany pomyślnie.

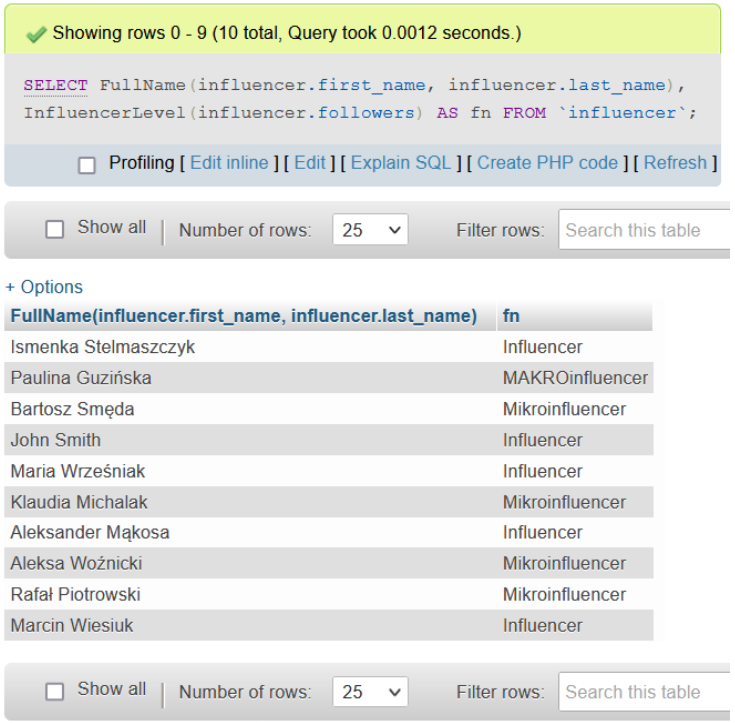


Poniżej dodatkowo podgląd wyeksportowanej funkcji.



Połączenie dwóch powyższych funkcji pozwoliło mi wygenerować zapytanie złożone. Korzystając w nim z pierwszej funkcji wyświetliłam imiona i nazwiska wszystkich influencerów z tabeli Influencerzy. Druga funkcja natomiast przyporządkowała danego influencera w osobnej kolumnie do jednej z trzech grup liczności (w zależności od liczby followersów tegoż influencera).

SELECT FullName(influencer.first_name, influencer.last_name), InfluencerLevel(influencer.followers) AS fn FROM `influencer`;



3. aa

obliczanie wypłaty w kursie euro - w funkcji przekazuję wartość i podaję (wymyślam) na bieżąco kurs euro.

Procedury

Tworzenie konta użytkownika

Skrypt SQL do utworzenia bazy danych

```
-- phpMyAdmin SQL Dump
-- version 5.1.1
-- https://www.phpmyadmin.net/
--
-- Host: 127.0.0.1
-- Generation Time: Nov 18, 2021 at 09:44 PM
-- Server version: 10.4.21-MariaDB
-- PHP Version: 8.0.11

CREATE TABLE `address` (
  `Id_address` int(11) NOT NULL,
  `Country_Id_country` int(11) DEFAULT NULL,
  `street` varchar(200) COLLATE utf8_unicode_ci DEFAULT NULL,
  `house_nr` varchar(10) COLLATE utf8_unicode_ci DEFAULT NULL,
  `post_code` varchar(10) COLLATE utf8_unicode_ci DEFAULT NULL,
  `city` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

--
-- Dumping data for table `address`
--

INSERT INTO `address` (`Id_address`, `Country_Id_country`, `street`, `house_nr`,
`post_code`, `city`) VALUES
(1, 1, 'Puławska', '38/15', '03-232', 'Warszawa'),
(2, 2, 'Isern-Hinnerk-Weg', '14B', '22457', 'Hamburg'),
(3, 8, 'High Park Road', '20A', 'PR9 7QL', 'Southport'),
(4, 1, 'Jurajska', '4/59', '02-699', 'Warszawa'),
(5, 1, 'Kadłubka', '42/5', '71-524', 'Szczecin'),
(6, 1, 'Aleja Krakowska', '49', '05-090', 'Sękocin Stary'),
(7, 1, 'Nowowiejska', '15', '06-500', 'Mława'),
(8, 8, 'Saxonbury Way', '94', 'PE2 9FB', 'Cambridgeshire'),
(9, 1, 'Marii Dąbrowskiej', '96/9', '97-300', 'Piotrków Trybunalski'),
(10, 1, 'Niemeńska', '66', '60-412', 'Poznań');

-- --------------------------------------------------------
```

```
--
-- Table structure for table `country`
--

CREATE TABLE `country` (
  `Id_country` int(11) NOT NULL,
  `long_name` varchar(200) COLLATE utf8_unicode_ci DEFAULT NULL,
  `short_name` char(3) COLLATE utf8_unicode_ci DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

--
-- Dumping data for table `country`
--

INSERT INTO `country` (`Id_country`, `long_name`, `short_name`) VALUES
(1, 'Polska', 'PL'),
(2, 'Niemcy', 'DE'),
(3, 'Rosja', 'RU'),
(4, 'Szwajcaria', 'CH'),
(5, 'Ukraina', 'UA'),
(6, 'Białoruś', 'BY'),
(7, 'Czarnogóra', 'MNE'),
(8, 'Wielka Brytania', 'GB'),
(9, 'Stany Zjednoczone', 'USA'),
(10, 'Kanada', 'CA');

-- --------------------------------------------------------

--
-- Table structure for table `influencer`
--

CREATE TABLE `influencer` (
  `Id_influencer` int(11) NOT NULL,
  `Address_Id_address` int(11) DEFAULT NULL,
  `first_name` varchar(50) COLLATE utf8_unicode_ci DEFAULT NULL,
  `last_name` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
  `nickname` varchar(20) COLLATE utf8_unicode_ci DEFAULT NULL,
  `discount_code` varchar(15) COLLATE utf8_unicode_ci DEFAULT NULL,
  `followers` int(11) DEFAULT NULL,
  `account_number` varchar(26) COLLATE utf8_unicode_ci DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

--
-- Dumping data for table `influencer`
--

INSERT INTO `influencer` (`Id_influencer`, `Address_Id_address`, `first_name`, `last_name`,
`nickname`, `discount_code`, `followers`, `account_number`) VALUES
(1, 4, 'Ismenka', 'Stelmaszczyk', 'ismena_stelmaszczyk', 'ismena15', 47600, NULL),
```

```
(2, 2, 'Paulina', 'Guzińska', 'paulina_guzinska', 'paulina15', 264000, NULL),
(3, 5, 'Bartosz', 'Smęda', 'smeda.triathlon', 'triathlon15', 801, NULL),
(4, 3, 'John', 'Smith', 'john_smith', 'john15', 45000, '124455882375951452125477 19'),
(5, 1, 'Maria', 'Wrześniak', 'mariaaa', 'marysia15', 53000, NULL),
(6, 6, 'Klaudia', 'Michalak', 'klaudia_michalak', 'klaudia15', 4123, NULL),
(7, 7, 'Aleksander', 'Mąkosa', 'alexi', 'alexi15', 89000, NULL),
(8, 8, 'Aleksa', 'Woźnicki', 'aleksa_woznikci', 'woznikci15', 632, NULL),
(9, 9, 'Rafał', 'Piotrowski', 'rafal_p', 'rafal15', 7522, NULL),
(10, 10, 'Marcin', 'Wiesiuk', 'marcinek_w', 'marcinek15', 65002, NULL);

-- --------------------------------------------------------

--
-- Table structure for table `influenceraction`
--

CREATE TABLE `influenceraction` (
  `Id_influencer_action` int(11) NOT NULL,
  `Influencer_Id_influencer` int(11) DEFAULT NULL,
  `Service_Id_service` int(11) DEFAULT NULL,
  `date` date DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

--
-- Dumping data for table `influenceraction`
--

INSERT INTO `influenceraction` (`Id_influencer_action`, `Influencer_Id_influencer`,
`Service_Id_service`, `date`) VALUES
(1, 1, 8, '2021-09-30'),
(2, 1, 8, '2021-09-30'),
(3, 1, 1, '2021-09-15'),
(4, 2, 7, '2021-10-13'),
(5, 4, 10, '2021-11-02'),
(6, 3, 10, '2021-11-14'),
(7, 5, 3, '2021-10-19'),
(8, 7, 5, '2021-11-02'),
(9, 10, 9, '2021-11-03'),
(10, 8, 6, '2021-08-10');

-- --------------------------------------------------------

--
-- Table structure for table `product`
--

CREATE TABLE `product` (
  `Id_product` int(11) NOT NULL,
  `name` varchar(100) COLLATE utf8_unicode_ci DEFAULT NULL,
  `price` decimal(6,2) DEFAULT NULL
```

```
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

--
-- Dumping data for table `product`
--

INSERT INTO `product` (`Id_product`, `name`, `price`) VALUES
(1, 'Modify Reductor', '129.00'),
(2, 'Modify Femibra', '139.00'),
(3, 'Gumka do włosów - lniana ', '29.00'),
(4, 'Gumka do włosów - lniana XL', '49.00'),
(5, 'Gumka do włosów - welurowa', '19.00'),
(6, 'Gumka do włosów - welurowa XL', '29.00'),
(7, 'Pełna kuracja Modify Reductor', '387.00'),
(8, 'Pełna kuracja Modify Femibra', '417.00'),
(9, 'Zestaw świąteczny Modify Reductor', '199.00'),
(10, 'Zestaw świąteczny Modify Femibra', '256.00');

-- --------------------------------------------------------

--
-- Table structure for table `salary`
--

CREATE TABLE `salary` (
  `Id_salary` int(11) NOT NULL,
  `Influencer_Id_influencer` int(11) DEFAULT NULL,
  `InfluencerAction_Id_influencer_action` int(11) DEFAULT NULL,
  `value` decimal(6,2) DEFAULT NULL,
  `date` date DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

--
-- Dumping data for table `salary`
--

INSERT INTO `salary` (`Id_salary`, `Influencer_Id_influencer`,
`InfluencerAction_Id_influencer_action`, `value`, `date`) VALUES
(1, 1, 1, '30.00', '2021-09-30'),
(2, 1, 2, '30.00', '2021-09-30'),
(3, 1, 3, '50.00', '2021-09-15'),
(4, 2, 4, '100.00', '2021-10-13'),
(5, 4, 5, '350.00', '2021-11-12'),
(6, 3, 6, '350.00', '2021-11-14'),
(7, 5, 7, '100.00', '2021-10-19'),
(8, 7, 8, '500.00', '2021-11-02'),
(9, 10, 9, '40.00', '2021-11-03'),
(10, 8, 10, '200.00', '2021-08-10');

-- --------------------------------------------------------
```

```
--
-- Table structure for table `service`
--

CREATE TABLE `service` (
  `Id_service` int(11) NOT NULL,
  `name` varchar(255) COLLATE utf8_unicode_ci DEFAULT NULL,
  `value` decimal(6,2) DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

--
-- Dumping data for table `service`
--

INSERT INTO `service` (`Id_service`, `name`, `value`) VALUES
(1, 'Relacja ', '50.00'),
(2, 'Relacja + swipe', '150.00'),
(3, 'Post bez produktu', '100.00'),
(4, 'Post z produktem', '250.00'),
(5, 'Udział w evencie marki', '500.00'),
(6, 'Logo marki na stronie', '200.00'),
(7, 'Oznaczenie na story', '100.00'),
(8, 'Użycie kodu influencera', '30.00'),
(9, 'Polecenie zakupu bez kodu', '40.00'),
(10, 'Pakiet (post + relacja + swipe)', '350.00');

-- --------------------------------------------------------

--
-- Table structure for table `shipping`
--

CREATE TABLE `shipping` (
  `Id_shipping` int(11) NOT NULL,
  `Influencer_Id_influencer` int(11) DEFAULT NULL,
  `Product_Id_product` int(11) DEFAULT NULL,
  `date` date DEFAULT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci;

--
-- Dumping data for table `shipping`
--

INSERT INTO `shipping` (`Id_shipping`, `Influencer_Id_influencer`, `Product_Id_product`,
`date`) VALUES
(1, 1, 7, '2021-09-08'),
(2, 2, 7, '2021-11-07'),
(3, 3, 9, '2021-08-18'),
(4, 4, 1, '2021-11-19'),
```

```
(5, 5, 2, '2021-10-18'),
(6, 5, 5, '2021-10-18'),
(7, 6, 10, '2020-12-12'),
(8, 7, 9, '2021-11-17'),
(9, 8, 2, '2021-07-11'),
(10, 9, 1, '2021-11-06'),
(11, 10, 9, '2021-11-15'),
(12, 1, 4, '2021-11-08');

--
-- Indexes for dumped tables
--

--
-- Indexes for table `address`
--
ALTER TABLE `address`
  ADD PRIMARY KEY (`Id_address`),
  ADD KEY `Country_Id_country` (`Country_Id_country`);

--
-- Indexes for table `country`
--
ALTER TABLE `country`
  ADD PRIMARY KEY (`Id_country`);

--
-- Indexes for table `influencer`
--
ALTER TABLE `influencer`
  ADD PRIMARY KEY (`Id_influencer`),
  ADD KEY `Address_Id_address` (`Address_Id_address`);

--
-- Indexes for table `influenceraction`
--
ALTER TABLE `influenceraction`
  ADD PRIMARY KEY (`Id_influencer_action`),
  ADD KEY `Influencer_Id_influencer` (`Influencer_Id_influencer`),
  ADD KEY `Service_Id_service` (`Service_Id_service`);

--
-- Indexes for table `product`
--
ALTER TABLE `product`
  ADD PRIMARY KEY (`Id_product`);

--
-- Indexes for table `salary`
--
```

```
ALTER TABLE `salary`
  ADD PRIMARY KEY (`Id_salary`),
  ADD KEY `InfluencerAction_Id_influencer_action`
(`InfluencerAction_Id_influencer_action`),
  ADD KEY `Influencer_Id_influencer` (`Influencer_Id_influencer`);

--
-- Indexes for table `service`
--
ALTER TABLE `service`
  ADD PRIMARY KEY (`Id_service`);

--
-- Indexes for table `shipping`
--
ALTER TABLE `shipping`
  ADD PRIMARY KEY (`Id_shipping`),
  ADD KEY `Influencer_Id_influencer` (`Influencer_Id_influencer`),
  ADD KEY `Product_Id_product` (`Product_Id_product`);

--
-- AUTO_INCREMENT for dumped tables
--

--
-- AUTO_INCREMENT for table `address`
--
ALTER TABLE `address`
  MODIFY `Id_address` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=11;

--
-- AUTO_INCREMENT for table `country`
--
ALTER TABLE `country`
  MODIFY `Id_country` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=11;

--
-- AUTO_INCREMENT for table `influencer`
--
ALTER TABLE `influencer`
  MODIFY `Id_influencer` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=11;

--
-- AUTO_INCREMENT for table `influenceraction`
--
ALTER TABLE `influenceraction`
  MODIFY `Id_influencer_action` int(11) NOT NULL AUTO_INCREMENT,
AUTO_INCREMENT=11;

--
```

```
-- AUTO_INCREMENT for table `product`
--
ALTER TABLE `product`
  MODIFY `Id_product` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=11;

--
-- AUTO_INCREMENT for table `salary`
--
ALTER TABLE `salary`
  MODIFY `Id_salary` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=11;

--
-- AUTO_INCREMENT for table `service`
--
ALTER TABLE `service`
  MODIFY `Id_service` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=11;

--
-- AUTO_INCREMENT for table `shipping`
--
ALTER TABLE `shipping`
  MODIFY `Id_shipping` int(11) NOT NULL AUTO_INCREMENT, AUTO_INCREMENT=13;

--
-- Constraints for dumped tables
--

--
-- Constraints for table `address`
--
ALTER TABLE `address`
  ADD CONSTRAINT `address_ibfk_1` FOREIGN KEY (`Country_Id_country`)
REFERENCES `country` (`Id_country`);

--
-- Constraints for table `influencer`
--
ALTER TABLE `influencer`
  ADD CONSTRAINT `influencer_ibfk_1` FOREIGN KEY (`Address_Id_address`)
REFERENCES `address` (`Id_address`);

--
-- Constraints for table `influenceraction`
--
ALTER TABLE `influenceraction`
  ADD CONSTRAINT `influenceraction_ibfk_1` FOREIGN KEY (`Influencer_Id_influencer`)
REFERENCES `influencer` (`Id_influencer`),
  ADD CONSTRAINT `influenceraction_ibfk_2` FOREIGN KEY (`Service_Id_service`)
REFERENCES `service` (`Id_service`);
```

```
--
-- Constraints for table `salary`
--
ALTER TABLE `salary`
  ADD CONSTRAINT `salary_ibfk_1` FOREIGN KEY
(`InfluencerAction_Id_influencer_action`) REFERENCES `influenceraction`
(`Id_influencer_action`),
  ADD CONSTRAINT `salary_ibfk_2` FOREIGN KEY (`Influencer_Id_influencer`)
REFERENCES `influencer` (`Id_influencer`);

--
-- Constraints for table `shipping`
--
ALTER TABLE `shipping`
  ADD CONSTRAINT `shipping_ibfk_1` FOREIGN KEY (`Influencer_Id_influencer`)
REFERENCES `influencer` (`Id_influencer`),
  ADD CONSTRAINT `shipping_ibfk_2` FOREIGN KEY (`Product_Id_product`)
REFERENCES `product` (`Id_product`);
COMMIT;
```