Rozwiązania zadań do kursu stacjonarnego "Podstawy SQL" na platformie LNU.

12. 1. Pobierz z bazy dane na temat liczby książek napisanych przez autorów pochodzących z tego samego kraju. Wyświetl w wynikowej tabeli kolumny:
1. country - z nazwą kraju,
2. books - z liczbą książek znajdujących się w bazie danych, które zostały napisane przez autorów pochodzących z danego kraju.
Tabelę posortuj alfabetycznie po nazwach państw.

```
SELECT country, COUNT(*) AS books FROM books JOIN authors ON id = author_id
GROUP BY country ORDER BY country;
```

12. 2. Dla każdego autora z bazy danych wyświetl liczbę jego książek w bazie. Wynik zaprezentuj w tabeli zawierającej kolumny:
1. last_name - nazwisko autora
2. first_name - imię autora
3. books - liczba książek danego autora w bazie danych
Wynikową tabelę posortuj w pierwszej kolejności malejąco po liczbie książek, następnie alfabetycznie po nazwiskach i imionach autorów.

```
SELECT last_name, first_name, COUNT(ISBN) AS books FROM (SELECT last_name,
first_name, id, ISBN FROM authors LEFT JOIN books ON id = author_id) GROUP BY id
ORDER BY books DESC, last_name, first_name;
```

12. 3. Dla autorów, których książki znajdują się w bazie danych, wyświetl tabelę z kolumnami:
1. last_name - nazwisko autora
2. first_name - imię autora
3. publishers - liczba wydawców książek danego autora
Tabelę posortuj malejąco po liczbie wydawców, następnie alfabetycznie po nazwiskach i po imionach autorów.

```
SELECT last_name, first_name, COUNT(*) AS publishers FROM (SELECT DISTINCT
authors.id, last_name, first_name, name FROM authors JOIN books on authors.id =
author_id JOIN publishers ON publishers.id = publisher_id) GROUP BY id ORDER BY
publishers DESC, last_name, first_name;
```

12. 4. Tym razem wyświetl liczbę wydawców dla wszystkich autorów z bazy danych, również dla tych, których książek w niej nie ma. Wynikowa tabela powinna zawierać kolumny:
1. last_name - nazwisko autora
2. first_name - imię autora
3. publishers - liczba wydawców książek danego autora
Tabelę posortuj malejąco po liczbie wydawców, następnie alfabetycznie po nazwiskach i po imionach autorów.

```
SELECT last_name, first_name, COUNT(name) AS publishers FROM (SELECT DISTINCT
authors.id, last_name, first_name, name FROM authors LEFT JOIN books on
authors.id = author_id LEFT JOIN publishers ON publishers.id = publisher_id)
GROUP BY id ORDER BY publishers DESC, last_name, first_name;
```

12. 5. Wyświetl liczbę krajów, z których pochodzą autorzy książek znajdujących się w bazie danych. Wynikowa tabela powinna składać się z jednej kolumny o nazwie countries.

```
SELECT COUNT(DISTINCT country) AS countries FROM authors WHERE id IN ( SELECT
author_id FROM books );
```

12. 6. Tabele technic, city i friends zawierają dane dotyczące zestawów klocków LEGO z różnych serii. Wyświetl w wynikowej tabeli wszystkie dane z tabel technic oraz city.

```
SELECT * FROM technic UNION SELECT * FROM city;
```

12. 7. Z tabel technic, city oraz friends pobierz w wynikowej tabeli dane zestawów LEGO. Interesują nas tylko zestawy tańsze niż 200 dolarów. Wyświetl jedynie kolumny: item, name oraz price. Posortuj wynikową tabelę od najtańszych

do najdroższych zestawów.
```
SELECT item, name, price FROM technic WHERE price < 200 UNION SELECT item, name, price FROM city WHERE price < 200 UNION SELECT item, name, price FROM friends WHERE price < 200 ORDER BY price;
```

12. 8. Tabele good_movies oraz movies zawierają dane dotyczące filmów z dwóch różnych portali filmowych.
Korzystając z rekordów w obu tabelach, wyświetl w tabeli wynikowej kolumny:
1. title - z tytułami filmów
2. actors - z głównymi aktorami
3. director - z reżyserami
4. rating - z ocenami filmów.
Interesują nas tylko filmy, w których jako aktor wystąpił Clint Eastwood (któż by inny). Tabelę posortuj malejąco po ocenach filmów.
```
SELECT title, [cast] as actors, director, rating FROM good_movies WHERE actors LIKE '%Clint Eastwood%' UNION SELECT title, actors, director, rating FROM movies WHERE actors LIKE '%Clint Eastwood%' ORDER BY rating DESC;
```

12. 9. Wyświetl dane najdroższego zestawu z każdej z trzech tabel: technic, city, friends. Wynikowa tabela powinna zawierać trzy kolumny:
1. item
2. name
3. price
Tabelę posortuj malejąco po cenie zestawów.
```
SELECT item, name, MAX(price) AS price FROM technic UNION SELECT item, name, MAX(price) AS price FROM city UNION SELECT item, name, MAX(price) AS price FROM friends ORDER BY price DESC;
```

12. 10. Zamierzasz wyszukać powtarzające się postaci w tabelach famous_cats oraz characters. W tym celu wyświetl wszystkie rekordy z obu tabel i posortuj alfabetycznie po nazwach postaci. Wynikowa tabela powinna zawierać dwie kolumny: name z nazwami postaci oraz cartoon z nazwami bajek.
```
SELECT name, cartoon FROM characters UNION ALL SELECT name, origin FROM famous_cats ORDER BY name;
```

12. 11. Z tabeli movies wyświetl tytuł i ocenę najwyżej oraz najniżej ocenianego filmu (w tej kolejności). Wynikowa tabela powinna zawierać kolumny: title oraz rating.
```
SELECT title, MAX(rating) as rating FROM movies UNION SELECT title, MIN(rating) as rating FROM movies ORDER BY rating DESC;
```

12. 12. Z trzech tabel w bazie danych wyświetl rekordy, biorąc pod uwagę kryteria:
1. Z zestawów technic interesują nas tylko takie, które są odpowiednie dla 10-latka.
2. Z zestawów friends interesują nas tylko zestawy tańsze niż 100 dolarów.
3. Z zestawów city interesuje nas tylko ten, który ma najkorzystniejszy stosunek ceny do liczby klocków (liczbę klocków zawiera kolumna pieces).
Wyświetl jedynie kolumny item, name i price. Wynikową tabelę posortuj od zestawów najtańszych do najdroższych.
```
SELECT item, name, price FROM technic WHERE CAST(TRIM(age, '+') AS INTEGER) <= 10 UNION SELECT item, name, price FROM friends WHERE price < 100 UNION SELECT item, name, price FROM city WHERE item = ( SELECT item FROM city ORDER BY ( price / pieces ) limit 1 ) ORDER BY price;
```

12. 13. Tabela crimes zawiera dane dotyczące ogólnej liczby przestępstw popełnionych w hrabstwie Cambridgeshire.
Kolumny y_12, y_13, y_14, y_15, y_16 zawierają liczby przestępstw popełnionych w kolejnych latach od 2012 do 2016. Kolumna ward zawiera nazwy okręgów, których dotyczą dane, kolumna district nazwy dystryktów, do których należą okręgi, zaś kolumna population liczbę mieszkańców okręgów.
Dla każdego z dystryktów wyświetl dane jego okręgów, w których w 2016 roku popełniono najwięcej i najmniej przestępstw w przeliczeniu na jednego mieszkańca.

Wynikowa tabela powinna zawierać trzy kolumny:
1. ward - z nazwą okręgu,
2. district - z nazwą dystryktu,
3. per capita - z obliczoną liczbą przestępstw w okręgu w przeliczeniu na jednego mieszkańca. Wartość tę zaokrąglij do czterech miejsc po przecinku.
Wynikowa tabela powinna być posortowana alfabetycznie po nazwach dystryktów, natomiast w ramach tych samych dystryktów jako pierwszy powinien być wyświetlony okręg, w którym liczba przestępstw na jednego mieszkańca była większa.

```
SELECT district, ward, ROUND(MAX(y_16/CAST(population AS FLOAT)),4) AS [per capita] FROM crimes GROUP BY district UNION SELECT district, ward, ROUND(MIN(y_16/CAST(population AS FLOAT)),4) AS [per capita] FROM crimes GROUP BY district ORDER BY district, [per capita] DESC;
```

12. 14. Wyświetl kolumny:
1. title - z tytułami książek,
2. last_name - z nazwiskami autorów
3. first_name - z imionami autorów
4. name - z nazwą wydawcy.
Uwzględnij wszystkich autorów i wydawców z bazy danych, nawet w przypadku gdy nie ma odpowiadających im danych w pozostałych tabelach.
Wynikową tabelę posortuj alfabetycznie w kolejności: po nazwiskach autorów, po imionach autorów, po nazwach wydawców i po tytułach książek.

```
SELECT title, last_name, first_name, name FROM authors LEFT JOIN books on authors.id = author_id LEFT JOIN publishers ON publishers.id = publisher_id UNION SELECT title, last_name, first_name, name FROM publishers LEFT JOIN books on publishers.id = publisher_id LEFT JOIN authors ON authors.id = author_id ORDER BY last_name, first_name, name, title;
```

13.1. Wyświetl tabelę fitness w taki sposób, aby wartości w kolumnie day_of_week zostały zamienione z wartości liczbowych na nazwy dni tygodnia wg zasady:
1 - Monday
2 - Tuesday
3 - Wednesday
4 - Thursday
5 - Friday
6 - Saturday
7 - Sunday

```
SELECT [#], class, time, CASE day_of_week WHEN 1 THEN 'Monday' WHEN 2 THEN 'Tuesday' WHEN 3 THEN 'Wednesday' WHEN 4 THEN 'Thursday' WHEN 5 THEN 'Friday' WHEN 6 THEN 'Saturday' WHEN 7 THEN 'Sunday' END AS day_of_week FROM fitness;
```

13.2. Wyświetl nazwy poszczególnych modeli telefonów oraz kraje z których pochodzą ich producenci.
1. Dla marki Samsung jako kraj producenta podaj wartość Korea
2. Dla marek Huawei i Xiaomi podaj wartość China
3. Dla marek Apple i Microsoft podaj wartość USA
4. Dla pozostałych marek podaj wartość unknown
Wynikowa tabela powinna zawierać dwie kolumny:
1. model - z nazwą modelu
2. country - z krajem pochodzenia producenta

```
SELECT model, CASE brand WHEN 'Samsung' THEN 'Korea' WHEN 'Huawei' THEN 'China' WHEN 'Apple' THEN 'USA' WHEN 'Xiaomi' THEN 'China' WHEN 'Microsoft' THEN 'USA' ELSE 'unknown' END AS country FROM cell_phones;
```

13.3. Stosując te same kryteria, co w poprzednim ćwiczeniu, czyli:
1. Dla marki Samsung jako kraj producenta podaj wartość Korea
2. Dla marek Huawei i Xiaomi podaj wartość China
3. Dla marek Apple i Microsoft podaj wartość USA
4. Dla pozostałych marek podaj wartość unknown
uzupełnione o dodatkową informację:
5. Dla marki Sony jako kraj producenta podaj wartość Japan

wyświetl dane o liczbie modeli telefonów z poszczególnych krajów.
Wynikowa tabela powinna zawierać dwie kolumny:
1. country - z krajem pochodzenia producenta
2. models - z liczbą modeli telefonów z danego kraju.
Tabelę posortuj alfabetycznie po nazwach krajów.

```
SELECT CASE brand WHEN 'Samsung' THEN 'Korea' WHEN 'Huawei' THEN 'China' WHEN
'Apple' THEN 'USA' WHEN 'Xiaomi' THEN 'China' WHEN 'Microsoft' THEN 'USA' WHEN
'Sony' THEN 'Japan' ELSE 'unknown' END AS country, COUNT(*) AS models FROM
cell_phones GROUP BY country ORDER BY country;
```

13.4. Wyświetl kolumny:
1. title - z tytułem filmu
2. type - z kategorią filmu
3. recommendation - z rekomendacją dla filmu, zawierającą wartości:
excellent - jeżeli ocena filmu jest większa niż 8.4
good - jeżeli ocena filmu jest większa niż 7.6 i mniejsza lub równa 8.4
OK - jeżeli ocena filmu jest większa niż 7.0 i mniejsza lub równa 7.6
poor - jeżeli ocena jest mniejsza lub równa 7.0.
Posortuj tabelę malejąco po ocenach filmów (czyli wartościach z kolumny rating).

```
SELECT title, type, CASE WHEN rating > 8.4 THEN 'excellent' WHEN rating > 7.6
THEN 'good' WHEN rating >= 7.0 THEN 'OK' ELSE 'poor' END AS recommendation FROM
movies ORDER BY rating DESC;
```

13.5. Z tabeli crimes wyświetl kolumny:
1. ward - z nazwami okręgów
2. y_15 - z liczbą przestępstw popełnionych w okręgu w 2015 r.
3. y_16 - z liczbą przestępstw popełnionych w okręgu w 2016 r.
4. diff - z różnicą pomiędzy liczbą przestępstw popełnionych w 2016 r. a liczbą przestępstw popełnionych w 2015 r.
5. change - z oznaczeniem wzrostu, spadku lub braku zmiany w liczbie przestępstw popełnionych w roku 2016 w porównaniu do roku 2015. W przypadku wzrostu powinien zostać wyświetlony znak z tabeli UNICODE o numerze 9650, w przypadku spadku powinien zostać wyświetlony znak z tabeli UNICODE o numerze 9660. Jeżeli liczba przestępstw nie zmieniła się, powinien zostać wyświetlony komunikat NO CHANGE

```
SELECT ward, y_15, y_16, y_16 - y_15 AS diff, CASE WHEN y_16 - y_15 > 0 THEN
CHAR(9650) WHEN y_16 - y_15 < 0 THEN CHAR(9660) ELSE 'NO CHANGE' END AS change
FROM crimes;
```

13.6. Kolumny A i B w tabeli pressure zawierają pomiary ciśnień. Posortuj tabelę malejąco w pierwszej kolejności po tej kolumnie, która ma większą średnią wartość pomiarów, a następnie po kolumnie, która ma mniejszą średnią wartość pomiarów. Jeżeli średnie są sobie równe, posortuj tabelę najpierw po kolumnie A, a następnie po B.
Uwaga: aby zapewnić uniwersalność zapytania niezależnie od faktycznych danych w tabeli, w klauzuli ORDER BY użyj instrukcji CASE.

```
SELECT * FROM pressure ORDER BY CASE WHEN ( SELECT AVG(A) FROM pressure ) >=
( SELECT AVG(B) FROM pressure ) THEN A ELSE B END DESC, CASE WHEN ( SELECT
AVG(A) FROM pressure ) <= ( SELECT AVG(B) FROM pressure ) THEN A ELSE B END
DESC;
```

13.7. Z tabeli employees wyświetl kolumny:
1. id
2. last_name
3. first_name
4. sex - zawierającą oznaczenie płci pracownika: literę W - w przypadku kobiety lub literę M - w przypadku mężczyzny.
Listę imion żeńskich i męskich zawierają odpowiednio tabele W_names i M_names.
Wynikową tabelę posortuj alfabetycznie po nazwisko, a następnie po imieniu pracownika.

```
SELECT id, last_name, first_name, CASE WHEN first_name IN ( SELECT * FROM
W_names ) THEN 'W' ELSE 'M' END AS sex FROM employees ORDER BY last_name,
first_name;
```

13.8. Z tabeli authors wyświetl rekordy dotyczące żyjących autorów (czyli tych,

którzy w polu date_of_death mają wartość NULL).
```
SELECT * FROM authors WHERE date_of_death IS NULL;
```

13.9. Z tabeli authors wyświetl dane tych autorów, którzy mają drugie imię (pole middle_name).
```
SELECT * FROM authors WHERE middle_name IS NOT NULL;
```

13.10. Z tabeli cell_phones wyświetl kolumny:
1. brand
2. model
3. system
4. price - zawierającej cenę telefonu: jeżeli w kolumnie discount_price podana została cena promocyjna, wyświetl tę cenę, jeżeli zaś ceny promocyjnej nie ma, wyświetl cenę z kolumny regular_price.

Użyj funkcji IFNULL()
```
SELECT brand, model, system, IFNULL( discount_price, regular_price ) AS price
FROM cell_phones;
```

13.11. Wyświetl preferowane dane kontaktowe dla wszystkich sławnych kotów. Najbardziej pożądanym sposobem kontaktu jest telefon, drugim w kolejności email, trzecim - fax.
Wynikowa tabela powinna mieć dwie kolumny:
1. name
2. contact - z numerem telefony, emailem lub faksem wg wskazanego wyżej priorytetu i dostępności tych danych w tabeli.

Użyj funkcji COALESCE()
```
SELECT name, COALESCE( tel, email, fax ) AS contact FROM famous_cats;
```

13.12. Wyświetl w pojedynczej kolumnie o nazwie not equal liczbę rekordów, w których wartości w polu A i B nie są takie same.
Użyj NULLIF()
```
SELECT COUNT(NULLIF(A,B)) AS [not equal] FROM pressure;
```

13.13. Z tabeli tickets wyświetl rekordy wszystkich mandatów wystawionych w Chicago pod warunkiem, że tabela vehicles zawiera dane samochodu o numerze rejestracyjnym HCA11A9
Użyj EXISTS
```
SELECT * FROM tickets WHERE EXISTS ( SELECT * FROM vehicles WHERE licence_plate
= 'HCA11A9' ) AND location = 'Chicago';
```

13.14. Utwórz index na kolumnie title w tabeli movies.
```
CREATE INDEX title_index ON movies(title);
```

13.15. Utwórz indeksy na kolumnie last_name z tabeli authors oraz na kolumnie name z tabeli publishers.
```
CREATE INDEX abc ON authors(last_name);
CREATE INDEX def ON publishers(name);
```

14.1. Uruchom zapytanie:
SELECT title, artist, year_of_birth FROM Artworks JOIN Artists ON name = artist WHERE artist IN ("Powell, Joseph", "Wharncliffe, Lady", "Crisp, Fiona");
Zwróć uwagę na czas przetwarzania zapytania przez system (Czas CPU w rubryce nad przyciskiem Uruchom).
Następnie utwórz indeksy na obu tych polach. Wypróbuj ponownie zapytanie łączące tabele i zaobserwuj różnicę w czasie CPU.
```
SELECT title, artist, year_of_birth FROM Artworks JOIN Artists ON name = artist
WHERE artist IN ("Powell, Joseph", "Wharncliffe, Lady", "Crisp, Fiona");
CREATE INDEX name_index ON Artists(name);
CREATE INDEX artist_index ON Artworks(artist);
SELECT title, artist, year_of_birth FROM Artworks JOIN Artists ON name = artist
```

```
WHERE artist IN ("Powell, Joseph", "Wharncliffe, Lady", "Crisp, Fiona");
```

14.2. W bazie danych zdefiniowano indeksy:
1. title_index na kolumnie title z tabeli Artworks,
2. artist_index na kolumnie artist z tabeli Artworks,
3. name_index na kolumnie name z tabeli Artists.
Zakładając, że najczęstszym typem zapytania do bazy danych jest:
SELECT name, title FROM Artists JOIN Artworks ON name = artist WHERE artist IN ('Pocock, Nicholas', 'Varley, John', 'Hussey, Giles');
przeanalizuj sposób realizacji zapytania przez system zarządzania bazą danych i usuń niepotrzebne indeksy.

```
EXPLAIN QUERY PLAN SELECT name, title FROM Artists JOIN Artworks ON name =
artist WHERE artist IN ('Pocock, Nicholas', 'Varley, John', 'Hussey, Giles');
DROP INDEX title_index;
```

14.3. Dodaj do tabeli books nowy rekord o wartościach:
ISBN: 978-1905460762
title: The Daltons in the Blizzard
author_id: 19
publisher_id: 20
Następnie wyświetl tabelę, zawierającą tytuły wszystkich książek z tabeli books, nazwiska ich autorów oraz nazwy wydawnictw.
Wynikowa tabela powinna zawierać trzy kolumny:
1. title - z tytułem książki
2. last_name - z nazwiskiem autora
3. name - z nazwą wydawnictwa

```
INSERT INTO books VALUES ('978-1905460762', 'The Daltons in the Blizzard', 19,
20);
SELECT title, last_name, name FROM books LEFT JOIN authors ON author_id =
authors.id LEFT JOIN publishers ON publisher_id = publishers.id;
```

14.4. Spróbuj dodać do tabeli books nowy rekord o wartościach:
ISBN: 978-1905460762
title: The Daltons in the Blizzard
author_id: 19
publisher_id: 20
Gdyby był z tym kłopot, być może przydadzą się informacje o autorze:
imię i nazwisko: Rene Goscinny
kraj: France
ur. 14 sierpnia 1926 r.
zm. 5 listopada 1977 r.
oraz o wydawcy:
nazwa: Cinebook
kraj: Great Britain
Następnie - podobnie jak w poprzednim ćwiczeniu - wyświetl tabelę, zawierającą tytuły wszystkich książek z tabeli books, nazwiska ich autorów oraz nazwy wydawnictw.
Wynikowa tabela powinna zawierać trzy kolumny:
1. title - z tytułem książki
2. last_name - z nazwiskiem autora
3. name - z nazwą wydawnictwa

```
INSERT INTO authors VALUES(19,'Rene',NULL,'Goscinny','France','1926-08-
14','1977-11-05');
INSERT INTO publishers VALUES(20,'Cinebook','Great Britain');
INSERT INTO books VALUES('978-1905460762','The Daltons in the Blizzard',19,20);
SELECT title, last_name, name FROM books LEFT JOIN authors ON author_id =
authors.id LEFT JOIN publishers ON publisher_id = publishers.id;
```

14.5. Spróbuj usunąć tabele authors oraz publishers.

```
DROP TABLE books;
DROP TABLE authors;
DROP TABLE publishers;
```

14.6. Znormalizuj tabelę cell_phones do pierwszej postaci normalnej.

Po normalizacji tabela cell_phones powinna składać się z pięciu kolumn:
1. id - będącą kluczem głównym tabeli, zawierającą kolejne liczby naturalne począwszy od liczby 1.
2. brand - zawierającą nazwę producenta
3. model - zawierającą nazwę modelu
4. price - zawierającą cenę netto
5. plus_tax - zawierającą cenę z podatkiem.
Wszystkim kolumnom zadeklaruj odpowiednie typy danych. Za wyjątkiem kolumny id wszystkim kolumnom nadaj constraint NOT NULL.

```
CREATE TABLE phones ( id INTEGER PRIMARY KEY AUTOINCREMENT, brand TEXT NOT NULL, model TEXT NOT NULL, price FLOAT NOT NULL, plus_tax FLOAT NOT NULL);
INSERT INTO phones (brand, model, price, plus_tax) SELECT SUBSTR(model, 1, INSTR(model,' ')-1), SUBSTR(model, INSTR(model,' ')+1), price, plus_tax FROM cell_phones;
DROP TABLE cell_phones;
ALTER TABLE phones RENAME TO cell_phones;
```

14.7. Znormalizuj tabelę orders do drugiej postaci normalnej, eliminując redundantne dane. W tym celu podziel tabelę na dwie osobne table: jedną z danymi zamówienia, drugą z danymi klientów.
Tabela z danymi klientów powinna nazywać się clients i zawierać kolumny:
1. client_id - będącą kluczem głównym, zawierającym kolejne liczby naturalne począwszy od 1.
2. name - z imieniem i nazwiskiem klienta
3. address - z adresem klienta
4. email - z emailem klienta
5. phone - z numerem telefony klienta, dane telefonu w formacie TEXT.
Kolumny name, address, email powinny mieć nadany constraint NOT NULL.
Kolejność klientów w tabeli powinna być zgodna z kolejnością klientów występujących w tabeli orders.
Tabela z danymi zamówień powinna nazywać się orders i zawierać kolumny:
1. order_id - będącą kluczem głównym
2. value - z wartością zamówienia
3. client_id - będącą kluczem obcym odwołującym się do klucza głównego tabeli clients.

```
CREATE TABLE clients (client_id INTEGER PRIMARY KEY, name TEXT NOT NULL, address TEXT NOT NULL, email TEXT NOT NULL, phone TEXT);
INSERT INTO clients (name, address, email, phone) SELECT DISTINCT client, address, email, phone FROM orders;
CREATE TABLE orders1 ( order_id INTEGER PRIMARY KEY, value FLOAT NOT NULL, client_id INTEGER NOT NULL, FOREIGN KEY (client_id) REFERENCES clients(client_id));
INSERT INTO orders1 (value, client_id) SELECT value, client_id FROM orders, clients WHERE clients.name = client AND clients.address = orders.address AND clients.email = orders.email;
DROP TABLE orders;
ALTER TABLE orders1 RENAME TO orders;
```

14.8. Znormalizuj tabelę cell_phones do trzeciej postaci normalnej, usuwając z niej kolumnę plus_tax, która powstała przez dodanie do wartości z pola price 23% podatku VAT. W tabeli cell_phones wszystkim kolumnom nadano constraint NOT NULL.

```
CREATE TABLE cell_phones2 (id INTEGER NOT NULL PRIMARY KEY AUTOINCREMENT, brand TEXT NOT NULL, model TEXT NOT NULL, price FLOAT NOT NULL);
INSERT INTO cell_phones2 SELECT id, brand, model, price FROM cell_phones;
DROP TABLE cell_phones;
ALTER TABLE cell_phones2 RENAME TO cell_phones;
```

14.9. Utwórz w bazie danych tabele:
authors, zawierającą kolumny:
1. id - będącą kluczem głównym, zawierającym liczby całkowite
2. first_name
3. middle_name
4. last_name

5. country
Wszystkie kolumny poza middle_name powinny mieć nadany constrain NOT NULL.
publishers, zawierającą kolumny:
1. id - będącą kluczem głównym, zawierającym liczby całkowite
2. name
3. country
Wszystkie kolumny powinny mieć nadany constrain NOT NULL.
books, zawierającą kolumny:
1. ISBN - będącą kluczem głównym, zawierającym wartości typu TEXT,
2. title
3. author_id - będącą kluczem obcym odwołującym się do klucza głównego tabeli authors
4. publisher_id - będącą kluczem obcym odwołującym się do klucza głównego tabeli publishers
Wszystkie kolumny powinny mieć nadany constrain NOT NULL.

```
CREATE TABLE authors ( id INTEGER NOT NULL PRIMARY KEY, first_name TEXT NOT NULL, middle_name TEXT, last_name TEXT NOT NULL, country TEXT NOT NULL);
CREATE TABLE publishers (id INTEGER NOT NULL PRIMARY KEY, name TEXT NOT NULL, country TEXT NOT NULL);
CREATE TABLE books (ISBN TEXT NOT NULL PRIMARY KEY, title TEXT NOT NULL, author_id INTEGER NOT NULL, publisher_id INTEGER NOT NULL, FOREIGN KEY(author_id) REFERENCES authors(id), FOREIGN KEY(publisher_id) REFERENCES publishers(id));
```