Міністерство освіти та науки України
Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”
Факультет прикладної математики
Кафедра системного програмування і спеціалізованих комп’ютерних систем

Розрахункова графічна робота

з дисципліни

Бази даних та засоби управління

Виконав:
студент 3-го курсу,
групи КВ-23
Литвин Максим
Телеграм: @desp1c

Київ 2024

Постановка задачі:

1. Реалізувати функції перегляду, внесення, редагування та вилучення даних у таблицях бази даних, створених у лабораторній роботі №1, засобами консольного інтерфейсу.
2. Передбачити автоматичне пакетне генерування «рандомізованих» даних у базі.
3. Забезпечити реалізацію пошуку за декількома атрибутами з двох та більше сутностей одночасно: для числових атрибутів – у рамках діапазону, для рядкових – як шаблон функції LIKE оператора SELECT SQL, для логічного типу – значення True/False, для дат – у рамках діапазону дат.
4. Програмний код виконати згідно шаблону MVC (модель-подання-контролер).

Посилання на github

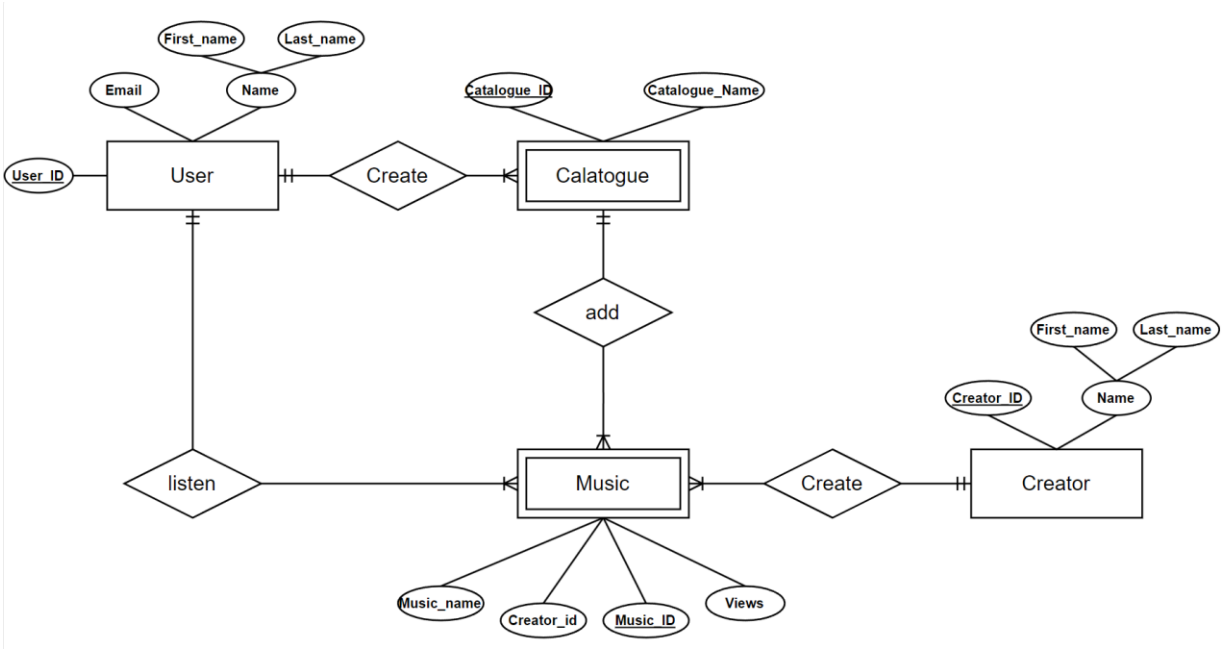
Структура бази даних:

User – користувач, який може створювати безліч каталогів, та має змогу додавати до кожного з них необхідну йому кількість музики до кожного каталогу.

Catalogue – каталог, який зберігає в собі посилання на довільну кількість музики, до якої через каталог має доступ користувач до прослуховування.

Music – музика, яка зберігає зв’язок між автором (творцем) та каталогом за яким вона закріплена в нашого користувача.

Creator – автор довільної кількості музики, яку може слухати наш користувач.



ER-діаграма в стилі «нотації Чена»

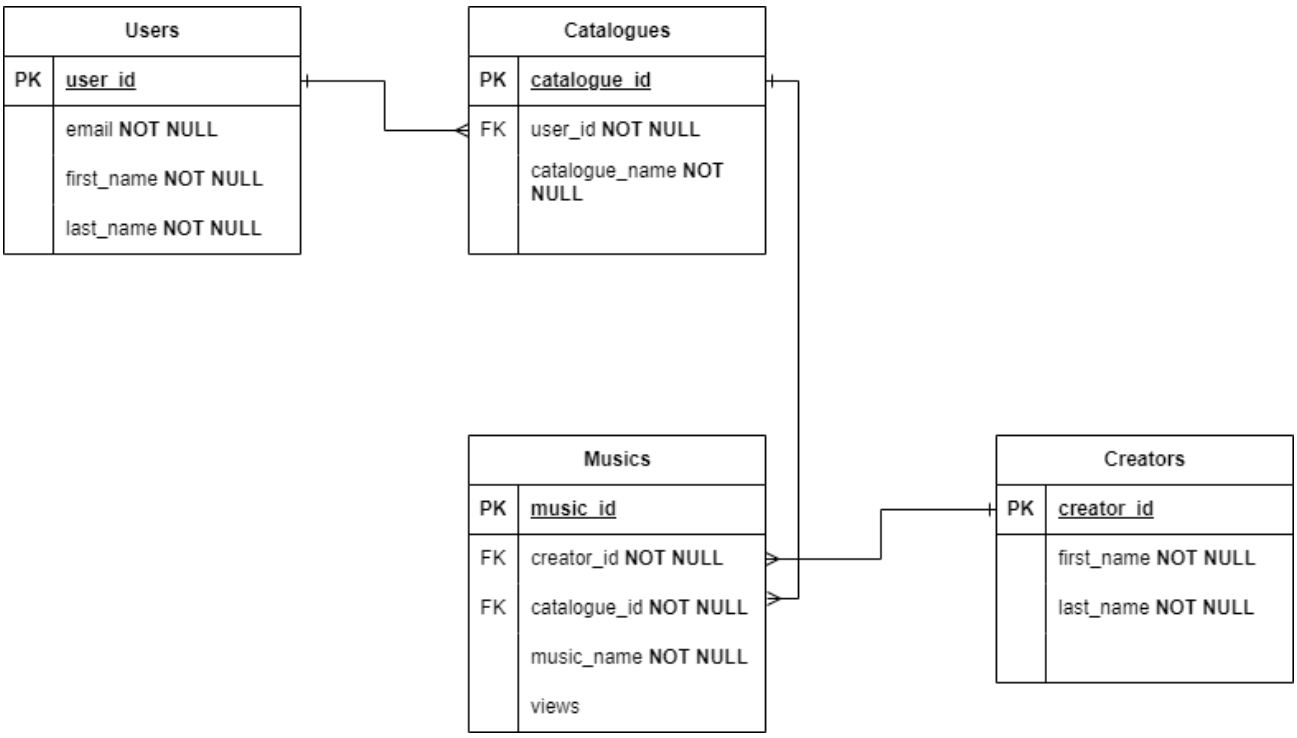
Зв’язки:

1:N – між user та catalogue (один користувач може створити безліч каталогів) – create(1)

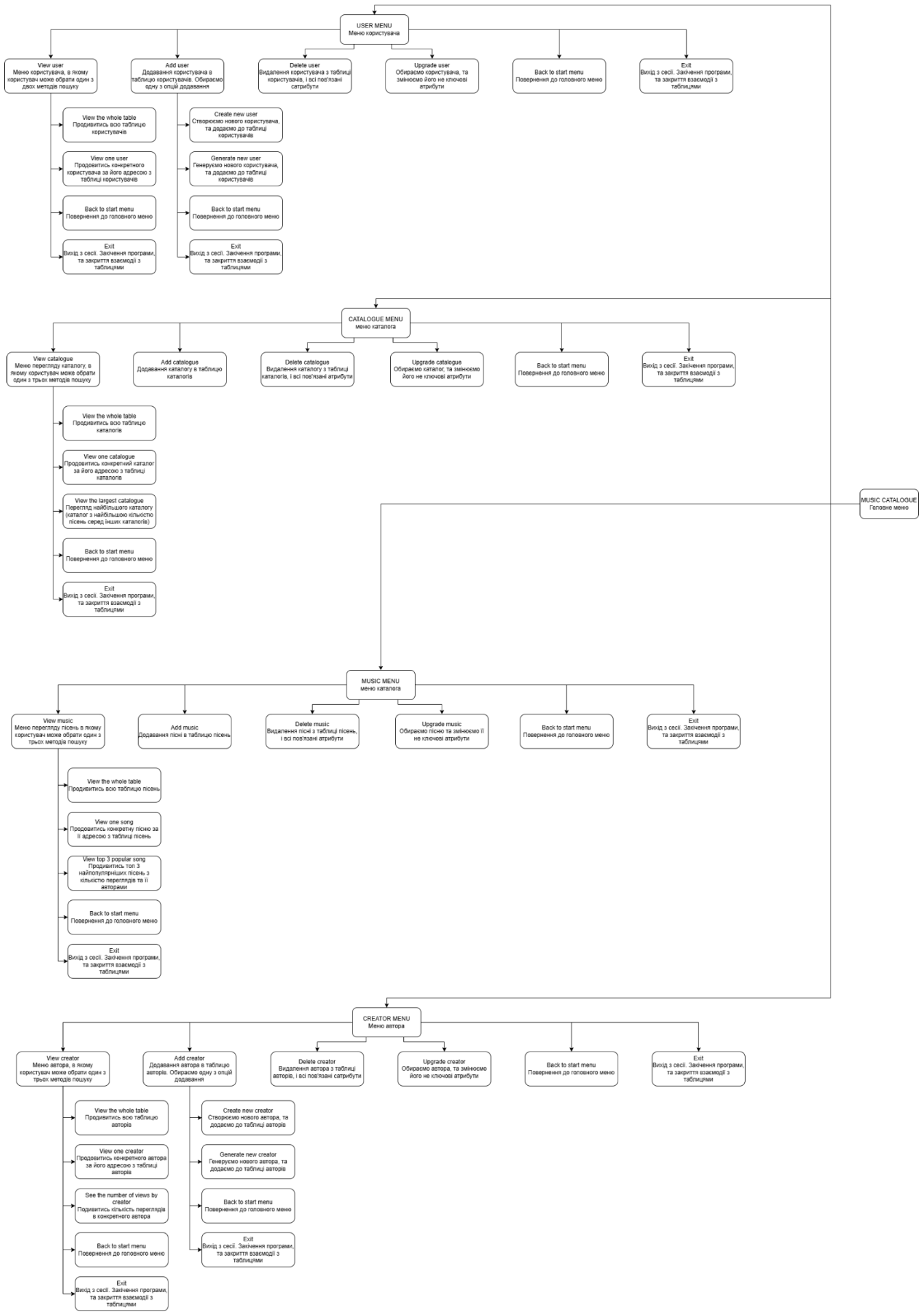
1:N – між user та music (один користувач може слухати безліч музики) – listen

1:N – між catalogue та music (один каталог може мати безліч музики) – add

1:N – між music та creator (один автор може створити безліч музики) – create(2)



Структура бази даних

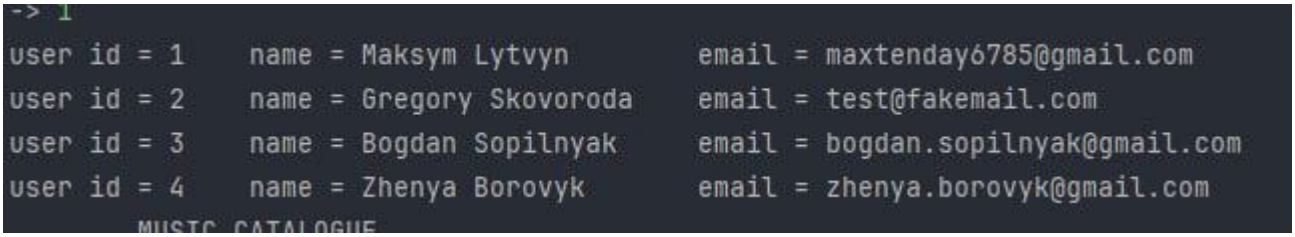

MUSIC CATALOGUE
Головне меню
USER MENU
Меню користувача
View user
Меню користувач, в якому користувач може обрати один з двох методів пошуку
Add user
Додавання користувача в таблицю користувачів. Обираємо одну з опцій додавання
Delete user
Видалення користувача з таблиці користувачів, і всі пов'язані сатрибути
Upgrade user
Обираємо користувача, та змінюємо його не ключові атрибути
Back to start menu
Повернення до головного меню
Exit
Вихід з сесії. Закінчення програми, та закриття взаємодії з таблицями
View the whole table
Продивитись всю таблицю користувачів
View one user
Продовитись конкретного користувача за його адресою з таблиці користувачів
Back to start menu
Повернення до головного меню
Exit
Вихід з сесії. Закінчення програми, та закриття взаємодії з таблицями
Create new user
Створюємо нового користувача, та додаємо до таблиці користувачів
Generate new user
Генеруємо нового користувача, та додаємо до таблиці користувачів
Back to start menu
Повернення до головного меню
Exit
Вихід з сесії. Закінчення програми, та закриття взаємодії з таблицями
CATALOGUE MENU
меню каталога
View catalogue
Меню перегляду каталогу, в якому користувач може обрати один з трьох методів пошуку
Add catalogue
Додавання каталогу в таблицю каталогів
Delete catalogue
Видалення каталогу з таблиці каталогів, і всі пов'язані атрибути
Upgrade catalogue
Обираємо каталог, та змінюємо його не ключові атрибути
Back to start menu
Повернення до головного меню
Exit
Вихід з сесії. Закінчення програми, та закриття взаємодії з таблицями
View the whole table
Продивитись всю таблицю каталогів
View one catalogue
Продовитись конкретний каталог за його адресою з таблиці каталогів
View the largest catalogue
Перегляд найбільшого каталогу (каталог з найбільшою кількістю пісень серед інших каталогів)
Back to start menu
Повернення до головного меню
Exit
Вихід з сесії. Закінчення програми, та закриття взаємодії з таблицями
MUSIC MENU
меню каталога
View music
Меню перегляду пісень в якому користувач може обрати один з трьох методів пошуку
Add music
Додавання пісні в таблицю пісень
Delete music
Видалення пісні з таблиці пісень, і всі пов'язані атрибути
Upgrade music
Обираємо пісню та змінюємо її не ключові атрибути
Back to start menu
Повернення до головного меню
Exit
Вихід з сесії. Закінчення програми, та закриття взаємодії з таблицями
View the whole table
Продивитись всю таблицю пісень
View one song
Продовитись конкретну пісню за її адресою з таблиці пісень
View top 3 popular song
Продивитись топ 3 найпопулярніших пісень з кількістю переглядів та її авторами
Back to start menu
Повернення до головного меню
Exit
Вихід з сесії. Закінчення програми, та закриття взаємодії з таблицями
CREATOR MENU
Меню автора
View creator
Меню автора, в якому користувач може обрати один з трьох методів пошуку
Add creator
Додавання автора в таблицю авторів. Обираємо одну з опцій додавання
Delete creator
Видалення автора з таблиці авторів, і всі пов'язані сатрибути
Upgrade creator
Обираємо автора, та змінюємо його не ключові атрибути
Back to start menu
Повернення до головного меню
Exit
Вихід з сесії. Закінчення програми, та закриття взаємодії з таблицями
View the whole table
Продивитись всю таблицю авторів
View one creator
Продовитись конкретного автора за його адресою з таблиці авторів
See the number of views by creator
Подивитись кількість переглядів в конкретного автора
Back to start menu
Повернення до головного меню
Exit
Вихід з сесії. Закінчення програми, та закриття взаємодії з таблицями
Create new creator
Створюємо нового автора, та додаємо до таблиці авторів
Generate new creator
Генеруємо нового автора, та додаємо до таблиці авторів
Back to start menu
Повернення до головного меню
Exit
Вихід з сесії. Закінчення програми, та закриття взаємодії з таблицями

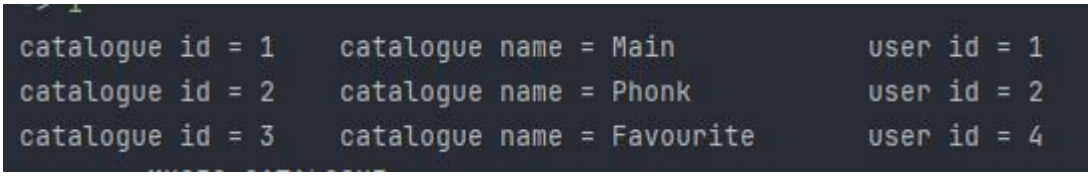
Структура меню користувача

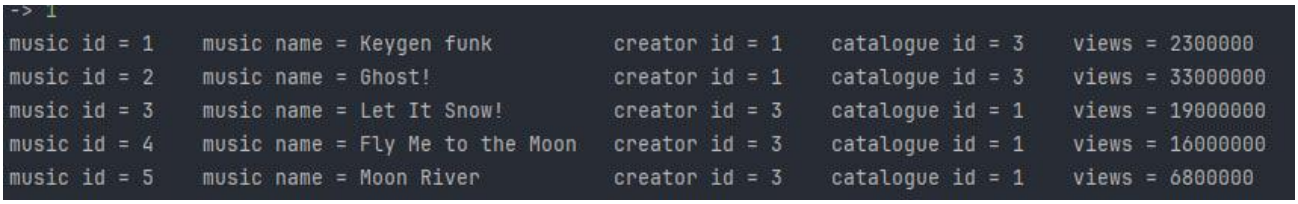
Мова програмування та додаткові бібліотеки:
Java, та базовий набір бібліотек.
Додаткові бібліотеки: postgresql

Пункт №1. Скріншоти

Видалення користувача, та каскадне видалення вмісту дочірніх таблиць:







```
-> 1
user id = 1     name = Maksym Lytvyn        email = maxtenday6785@gmail.com
user id = 2     name = Gregory Skovoroda    email = test@fakemail.com
user id = 3     name = Bogdan Sopilnyak     email = bogdan.sopilnyak@gmail.com
```

```
-> 1
catalogue id = 1     catalogue name = Main          user id = 1
catalogue id = 2     catalogue name = Phonk         user id = 2
        MUSIC CATALOGUE
```

```
-> 1
music id = 3     music name = Let It Snow!         creator id = 3     catalogue id = 1     views = 19000000
music id = 4     music name = Fly Me to the Moon   creator id = 3     catalogue id = 1     views = 16000000
music id = 5     music name = Moon River           creator id = 3     catalogue id = 1     views = 6800000
        MUSIC CATALOGUE
```

Додавання пісні:

```
        MUSIC MENU
Choose what you wish to do:
1. View music.
2. Add music.
3. Delete music.
4. Update music.
9. Back to start menu.
0. Exit
-> 2
Enter music id:
-> 1
Enter music name:
-> Imagination
Enter creator id:
-> 1
Enter catalogue id:
-> 2
Enter number of views:
-> 1000000
Music successfully added!
```

```
Select one of the methods for viewing the table:
1. View the whole table.
2. View one song.
3. View top 3 popular song.
9. Back to start menu.
0. Exit
-> 1
music id = 1     music name = Imagination          creator id = 1     catalogue id = 2     views = 1000000
music id = 3     music name = Let It Snow!         creator id = 3     catalogue id = 1     views = 19000000
music id = 4     music name = Fly Me to the Moon   creator id = 3     catalogue id = 1     views = 16000000
music id = 5     music name = Moon River           creator id = 3     catalogue id = 1     views = 6800000
```

Пункт №2. Згенеровані таблиці

Генеруємо дані таблиці авторів:

```
        CREATOR MENU
Choose what you wish to do:
1. View creator.
2. Add creator.
3. Delete creator.
4. Update creator.
9. Back to start menu.
0. Exit
-> 2
Select one of the methods for adding to the table:
1. Create new creator.
2. Generate new creator.
9. Back to start menu.
0. Exit
-> 2
Enter how many creators you want to generate:
-> 100000
100000 records inserted successfully.
```

```
Select one of the methods for viewing the table:
1. View the whole table.
2. View one creator.
3. See the number of views by creator.
9. Back to start menu.
0. Exit
-> 1
creator id = 1     name = Ivan Nikolaev
creator id = 2     name = Taylor Switch
creator id = 3     name = Francis Sinatra
creator id = 4     name = James Morgan
creator id = 5     name = Harry Williams
creator id = 6     name = Mia Morgan
creator id = 7     name = Max Carry
creator id = 8     name = John Evans
creator id = 9     name = Bob Smith
creator id = 10    name = Freya Williams
creator id = 11    name = Stephen Peters
creator id = 12    name = Richard Roberts
creator id = 13    name = Leo Davis
creator id = 14    name = Lily Campbell
creator id = 15    name = Johan Brown
creator id = 16    name = Oliver Harry
creator id = 17    name = Laura Campbell
creator id = 18    name = Mia Nelson
creator id = 19    name = Johan Davies
creator id = 20    name = Harry Lebron
creator id = 21    name = Johan Smith
creator id = 22    name = Laura Lewis
```

SQL-запит:

```
SELECT "first_name", "last_name"
FROM unnest(array['Max', 'Ann', 'John', 'Ava', 'Bob', 'Tomas', 'Mia',
'Richard',
'Freya', 'Florence', 'Oliver', 'Noah', 'Laura', 'Mike', 'Johan', 'James',
 'Arthur', 'Leo', 'Harry', 'Oscar', 'Harry', 'Amelia', 'Lily',
'Stephen']) AS "first_name"
CROSS JOIN unnest(array['Lebron', 'Nelson', 'Smith', 'Jones', 'Williams',
'Brown', 'Taylor',
'Davies', 'Evans', 'Williams', 'Stone', 'Bell', 'Campbell', 'Morgan',
'Lewis', 'Roberts',
'Evans', 'Adams', 'Gibson', 'Peters', 'Shelby', 'Harry', 'Carry',
'Davis']) AS "last_name"
ORDER BY random()
LIMIT ?;
```

Генеруємо дані таблиці авторів:

```
        USER MENU
Choose what you wish to do:
1. View user.
2. Add user.
3. Delete user.
4. Update user.
9. Back to start menu.
0. Exit
-> 2
Select one of the methods for adding to the table:
1. Create new user.
2. Generate new user.
9. Back to start menu.
0. Exit
-> 2
Enter how many users you want to generate:
-> 100000
100000 records inserted successfully.
```

```
Select one of the methods for viewing the table:
1. View the whole table.
2. View one creator.
9. Back to start menu.
0. Exit
-> 1
user id = 1     name = Maksym Lytvyn        email = maxtenday6785@gmail.com
user id = 2     name = Gregory Skovoroda    email = test@fakemail.com
user id = 3     name = Bogdan Sopilnyak     email = bogdan.sopilnyak@gmail.com
user id = 4     name = Arthur Evans         email = arthur.evans@gmail.com
user id = 5     name = Noah Roberts         email = noah.roberts@gmail.com
user id = 6     name = Leo Smith            email = leo.smith@gmail.com
user id = 7     name = Richard Williams     email = richard.williams@gmail.com
user id = 8     name = Johan Davies         email = johan.davies@gmail.com
user id = 9     name = Mia Davis            email = mia.davis@gmail.com
user id = 10    name = Max Harry            email = max.harry@gmail.com
user id = 11    name = Ann Campbell         email = ann.campbell@gmail.com
user id = 12    name = Amelia Gibson        email = amelia.gibson@gmail.com
user id = 13    name = Amelia Davies        email = amelia.davies@gmail.com
user id = 14    name = Mike Campbell        email = mike.campbell@gmail.com
user id = 15    name = Richard Nelson       email = richard.nelson@gmail.com
user id = 16    name = Stephen Brown        email = stephen.brown@gmail.com
user id = 17    name = Ava Peters           email = ava.peters@gmail.com
user id = 18    name = Harry Lewis          email = harry.lewis@gmail.com
```

SQL-запит:

```
SELECT "first_name", "last_name",
       LOWER("first_name" || '.' || "last_name") || '@gmail.com' as
"email"
FROM unnest(array['Max', 'Ann', 'John', 'Ava', 'Bob', 'Tomas', 'Mia',
'Richard',
'Freya', 'Florence', 'Oliver', 'Noah', 'Laura', 'Mike', 'Johan', 'James',
 'Arthur', 'Leo', 'Harry', 'Oscar', 'Harry', 'Amelia', 'Lily',
'Stephen']) AS "first_name"
CROSS JOIN unnest(array['Lebron', 'Nelson', 'Smith', 'Jones', 'Williams',
'Brown', 'Taylor',
'Davies', 'Evans', 'Williams', 'Stone', 'Bell', 'Campbell', 'Morgan',
'Lewis', 'Roberts',
'Evans', 'Adams', 'Gibson', 'Peters', 'Shelby', 'Harry', 'Carry',
'Davis']) AS "last_name"
ORDER BY random()
LIMIT ?;
```

Пункт №3. Додаткові запити

GetCreatorViews:

```
        CREATOR MENU
Choose what you wish to do:
1. View creator.
2. Add creator.
3. Delete creator.
4. Update creator.
9. Back to start menu.
0. Exit
-> 1
Select one of the methods for viewing the table:
1. View the whole table.
2. View one creator.
3. See the number of views by creator.
9. Back to start menu.
0. Exit
-> 3
Enter the creator id of the creator you want to view:
-> 3
Creator: Francis Sinatra. Total views: 41800000
Time spent: 2017700 nanoseconds
```

SQL-запит:

```
SELECT c."first_name", c."last_name", SUM(m."views") AS total_views
FROM public."musics" m
INNER JOIN public."creators" c ON c."creator_id" = m."creator_id"
WHERE c."creator_id" =?
GROUP BY c."first_name", c."last_name"
ORDER BY total_views DESC
```

GetBiggestCatalogue:

```
        CATALOGUE MENU
Choose what you wish to do:
1. View catalogue.
2. Add catalogue.
3. Delete catalogue.
4. Update catalogue.
9. Back to start menu.
0. Exit
-> 1
Select one of the methods for viewing the table:
1. View the whole table.
2. View one catalogue.
3. View the largest catalogue.
9. Back to start menu.
0. Exit
-> 3
Owner: Maksym Lytvyn. Catalogue: Main. Total music: 3
Time spent: 2832100 nanoseconds
```

SQL-запит:

```
SELECT u."first_name", u."last_name", c."catalogue_name",
COUNT(m."music_id") AS music_count
FROM "catalogues" c
LEFT JOIN "musics" m ON c."catalogue_id" = m."catalogue_id"
INNER JOIN "users" u ON u."user_id" = c."user_id"
GROUP BY u."first_name", u."last_name", c."catalogue_name"
ORDER BY music_count DESC
LIMIT 1
```

GetPopularMusic:

```
        MUSIC MENU
Choose what you wish to do:
1. View music.
2. Add music.
3. Delete music.
4. Update music.
9. Back to start menu.
0. Exit
-> 1
Select one of the methods for viewing the table:
1. View the whole table.
2. View one song.
3. View top 3 popular song.
9. Back to start menu.
0. Exit
-> 3
Owner: Francis Sinatra. Music: Let It Snow!. Views: 19000000
Owner: Francis Sinatra. Music: Fly Me to the Moon. Views: 16000000
Owner: Francis Sinatra. Music: Moon River. Views: 6800000
Time spent: 1902900 nanoseconds
```

SQL-запит:

```
SELECT c."first_name", c."last_name", m."music_name", m."views"
FROM "musics" m
LEFT JOIN "creators" c ON c."creator_id" = m."creator_id"
GROUP BY c."first_name", c."last_name", m."music_name", m."views"
ORDER BY m."views" DESC
LIMIT 3
```

Пункт №4. Ілюстрація модуля.

Код модуля:

Module.java

```
package model;

import java.sql.*;
import java.util.ArrayList;
import SQLError.DataBaseException;
import entity.*;
import entity.additional.*;
import util.Error;
import validation.Validation;


public class Model {
    private final Connection connection;

    public Model(Connection connection){
        this.connection = connection;
    }

    public ArrayList<User> getAllUsers(){
        final String sql = "SELECT * FROM \"users\"\n" + "ORDER BY
\"user_id\"";
        ArrayList<User> user = new ArrayList<>();

        try(PreparedStatement statement =
connection.prepareStatement(sql)){
            ResultSet resultSet = statement.executeQuery();

            while(resultSet.next()){
                int id = resultSet.getInt("user_id");
                String email = resultSet.getString("email");
                String firstName = resultSet.getString("first_name");
                String lastName = resultSet.getString("last_name");

                User user1 = new User(id, email, firstName, lastName);
                user.add(user1);
            }
        } catch(SQLException e){
            throw new DataBaseException("SQL Error! Unexpected error
related to users data retrieval");
        }

        return user;
    }

    public User getUser(int userId){
        if(!Validation.isID(userId)){
            System.out.println("Your ID entered incorrectly.");
            return null;
        }
```

```
        if(!Error.isUserIdTaken(connection, userId)){
            System.out.println("This user id is not exist.");
            return null;
        }

        final String sql = "SELECT \"email\", \"first_name\",
\"last_name\" FROM \"users\" WHERE \"user_id\" = ?";

        try(PreparedStatement statement =
connection.prepareStatement(sql)){
            statement.setInt(1, userId);
            ResultSet resultSet = statement.executeQuery();
            if(resultSet.next()){
                String email = resultSet.getString("email");
                String firstName = resultSet.getString("first_name");
                String lastName = resultSet.getString("last_name");

                return new User(userId, email, firstName, lastName);
            }
            System.out.println("Error! Table is empty!");
            return null;
        } catch(SQLException e){
            throw new DataBaseException("SQL Error! Unexpected error
related to user data retrieval");
        }
    }

    public void addUser(User user) {
        if(user == null) {
            System.out.println("Error! You have not entered the data");
            return;
        }
        if(!Validation.isName(user.getFirstName())){
            System.out.println("Your name entered incorrectly.");
            return;
        }
        if(!Validation.isEmail(user.getEmail())){
            System.out.println("Your email entered incorrectly.");
            return;
        }
        if(!Validation.isID(user.getUserId())){
            System.out.println("Your ID entered incorrectly.");
            return;
        }

        if(Error.isUserIdTaken(connection, user.getUserId())){
            System.out.println("This user id is already taken.");
            return;
        }

        final String sql = "INSERT INTO \"users\"(user_id, email,
first_name, last_name) VALUES (?,?,?,?)";
        try(PreparedStatement statement =
```

```
connection.prepareStatement(sql)){
            statement.setInt(1, user.getUserId());
            statement.setString(2, user.getEmail());
            statement.setString(3, user.getFirstName());
            statement.setString(4, user.getLastName());

            if(statement.executeUpdate() == 0)
                System.out.println("Error! The \"users\" table has not
changed.");
            else System.out.println("User successfully added!");
        } catch(SQLException e){
            throw new DataBaseException("SQL Error! Unexpected error
related to adding a user");
        }
    }

    public void deleteUser(int userId){
        if(!Validation.isID(userId)){
            System.out.println("User ID is not entered correctly or this
ID does not exist.");
            return;
        }
        if(!Error.isUserIdTaken(connection, userId)){
            System.out.println("This user id is not exist.");
            return;
        }

        final String sql = "DELETE FROM \"users\" WHERE \"user_id\" = ?";
        try(PreparedStatement statement =
connection.prepareStatement(sql)){
            statement.setInt(1, userId);
            if(statement.executeUpdate() == 0)
                System.out.println("Error! Table is empty!");
            else System.out.println("User successfully deleted!");
        } catch(SQLException e){
            throw new DataBaseException("SQL Error! Unexpected error
related to user deletion");
        }
    }

    public void updateUser(User user){
        if(user == null) {
            System.out.println("Error! You have not entered the data");
            return;
        }

        if(!Validation.isID(user.getUserId())){
            System.out.println("User ID entered incorrectly.");
            return;
        }

        if(!Validation.isName(user.getFirstName())){
            System.out.println("Your name entered incorrectly.");
            return;
```

```
        }

        if(!Validation.isEmail(user.getEmail())){
            System.out.println("Your email entered incorrectly.");
            return;
        }

        if(!Error.isUserIdTaken(connection, user.getUserId())){
            System.out.println("This user id is not exist.");
            return;
        }

        final String sql = "UPDATE \"users\" SET \"email\"= ?,
\"first_name\"= ?, \"last_name\" = ? WHERE \"user_id\"= ?";
        try(PreparedStatement statement =
connection.prepareStatement(sql)){
                statement.setInt(4, user.getUserId());
                statement.setString(1, user.getEmail());
                statement.setString(2, user.getFirstName());
                statement.setString(3, user.getLastName());

            if(statement.executeUpdate() == 0)
                System.out.println("Error! Table is empty!");

            else System.out.println("User successfully updated!");
        } catch(SQLException e){
            throw new DataBaseException("SQL Error! Unexpected error
related to user data change");
        }
    }

    public ArrayList<Catalogue> getAllCatalogue(){
        ArrayList<Catalogue> catalogue = new ArrayList<>();
        final String sql = "SELECT * FROM catalogues\n" + "ORDER BY
\"catalogue_id\"";
        try(PreparedStatement statement =
connection.prepareStatement(sql)){
            ResultSet resultSet = statement.executeQuery();
            while(resultSet.next()){
                int catalogueId = resultSet.getInt("catalogue_id");
                String name = resultSet.getString("catalogue_name");
                int userId = resultSet.getInt("user_id");

                Catalogue catalogue1 = new Catalogue(catalogueId, name,
userId);
                catalogue.add(catalogue1);
            }
            return catalogue;
        } catch(SQLException e){
            throw new DataBaseException("SQL Error! Unexpected error
related to catalogues data retrieval");
        }
    }
```

```
    public Catalogue getCatalogue(int catalogueId){
        if(!Validation.isID(catalogueId)){
            System.out.println("Catalogue ID entered incorrectly.");
            return null;
        }

        if(!Error.isCatalogueIdTaken(connection, catalogueId)){
            System.out.println("This catalogue id is not exist.");
            return null;
        }

        final String sql = "SELECT \"catalogue_name\", \"user_id\" FROM
\"catalogues\" WHERE \"catalogue_id\" = ?";
        try(PreparedStatement statement =
connection.prepareStatement(sql)){
            statement.setInt(1, catalogueId);
            ResultSet resultSet = statement.executeQuery();
            if(resultSet.next()){
                int userId = resultSet.getInt("user_id");
                String catalogueName =
resultSet.getString("catalogue_name");

                return new Catalogue(catalogueId, catalogueName, userId);
            }

            System.out.println("Error! Table is empty!");
            return null;
        } catch(SQLException e){
            throw new DataBaseException(" SQL Error! Unexpected error
related to catalogue data retrieval");
        }
    }
    public void addCatalogue(Catalogue catalogue){
        if(catalogue == null) {
            System.out.println("Error! You have not entered the data");
            return;
        }
        if(!Validation.isName(catalogue.getCatalogueName())){
            System.out.println("Catalogue name entered incorrectly.");
            return;
        }
        if(!Validation.isID(catalogue.getCatalogueId())){
            System.out.println("Catalogue ID entered incorrectly.");
            return;
        }
        if(!Validation.isID(catalogue.getUserId())){
            System.out.println("User ID entered incorrectly.");
            return;
        }

        if(Error.isCatalogueIdTaken(connection,
catalogue.getCatalogueId())){
            System.out.println("This catalogue id is already taken.");
            return;
```

```
        }

        if(!Error.isUserIdTaken(connection, catalogue.getUserId())){
            System.out.println("This user id is not exist.");
            return;
        }

        final String sql = "INSERT INTO \"catalogues\"(catalogue_id,
catalogue_name, user_id) VALUES (?,?,?)";
        try(PreparedStatement statement =
connection.prepareStatement(sql)){
            statement.setInt(1, catalogue.getCatalogueId());
            statement.setString(2, catalogue.getCatalogueName());
            statement.setInt(3, catalogue.getUserId());

            if(statement.executeUpdate() == 0) {
                System.out.println("Error! The \"catalogue\" table has
not changed.");
            }
            else System.out.println("Catalogue successfully added!");
        } catch(SQLException e){
            throw new DataBaseException("SQL Error! Unexpected error
related to adding a catalogue");
        }
    }
    public void deleteCatalogue(int catalogueId){
        if(!Validation.isID(catalogueId)){
            System.out.println("Catalogue ID entered incorrectly.");
            return;
        }

        if(!Error.isCatalogueIdTaken(connection, catalogueId)){
            System.out.println("This catalogue id is not exist.");
            return;
        }

        final String sql = "DELETE FROM \"catalogues\" WHERE
\"catalogue_id\" = ?";
        try(PreparedStatement statement =
connection.prepareStatement(sql)){
            statement.setInt(1, catalogueId);
            if(statement.executeUpdate() == 0)
                System.out.println("Error! Table is empty!");
            else System.out.println("Catalogue successfully deleted!");
        } catch(SQLException e){
            throw new DataBaseException("SQL Error! Unexpected error
related to catalogue deletion");
        }
    }

    public void updateCatalogue(Catalogue catalogue){
        if(catalogue == null) {
            System.out.println("Error! You have not entered the data");
            return;
```

```
        }
        if(!Validation.isName(catalogue.getCatalogueName())){
            System.out.println("Catalogue name entered incorrectly.");
            return;
        }
        if(!Validation.isID(catalogue.getCatalogueId())){
            System.out.println("Catalogue ID entered incorrectly.");
            return;
        }
        if(!Validation.isID(catalogue.getUserId())){
            System.out.println("User ID entered incorrectly.");
            return;
        }

        if(!Error.isCatalogueIdTaken(connection,
catalogue.getCatalogueId())){
            System.out.println("This catalogue id is not exist.");
            return;
        }

        final String sql = "UPDATE \"catalogues\" SET \"catalogue_name\"=
?, \"user_id\" = ? WHERE \"catalogue_id\"= ?";
        try(PreparedStatement statement =
connection.prepareStatement(sql)){
            statement.setInt(3, catalogue.getCatalogueId());
            statement.setString(1, catalogue.getCatalogueName());
            statement.setInt(2, catalogue.getUserId());

            if(statement.executeUpdate() == 0) {
                System.out.println("Error! The \"catalogues\" table has
not changed.");
            }
            else System.out.println("Catalogue successfully updated!");
        } catch(SQLException e){
            throw new DataBaseException("SQL Error! Unexpected error
related to accessing data in a table");
        }
    }

    public ArrayList<Music> getAllMusic(){
        final String sql = "SELECT * FROM \"musics\"\n " + "ORDER BY
\"music_id\"";
        ArrayList<Music> music = new ArrayList<>();

        try(PreparedStatement statement =
connection.prepareStatement(sql)){
            ResultSet resultSet = statement.executeQuery();

            while(resultSet.next()) {
                int musicId = resultSet.getInt("music_id");
                String musicName = resultSet.getString("music_name");
                int creatorId = resultSet.getInt("creator_id");
                int catalogueId = resultSet.getInt("catalogue_id");
                int views = resultSet.getInt("views");
```

```
                Music music1 = new Music(musicId, musicName, creatorId,
catalogueId, views);
                music.add(music1);
            }

            return music;
        } catch(SQLException e) {
            throw new DataBaseException("QL Error! Unexpected error
related to musics data retrieval");
        }
    }
    public Music getMusic(int musicId){
        if(!Validation.isID(musicId)){
            System.out.println("Music ID entered incorrectly.");
            return null;
        }

        if(!Error.isMusicIdTaken(connection, musicId)){
            System.out.println("This music id is not exist.");
            return null;
        }

        final String sql = "SELECT \"music_name\", \"creator_id\",
\"catalogue_id\", \"views\" FROM \"musics\" WHERE \"music_id\" = ?";
        try(PreparedStatement statement =
connection.prepareStatement(sql)){
            statement.setInt(1, musicId);
            ResultSet resultSet = statement.executeQuery();
            if(resultSet.next()){
                String musicName = resultSet.getString("music_name");
                int creatorId = resultSet.getInt("creator_id");
                int catalogueId = resultSet.getInt("catalogue_id");
                int views = resultSet.getInt("views");
                return new Music(musicId, musicName, creatorId,
catalogueId, views);
            }
            System.out.println("Error! Table is empty!");
            return null;
        } catch(SQLException e){
            throw new DataBaseException("SQL Error! Unexpected error
related to music data retrieval");
        }
    }
    public void addMusic(Music music){
        if(music == null) {
            System.out.println("Error! You have not entered the data");
            return;
        }
        if(!Validation.isName(music.getMusicName())){
            System.out.println("Music name entered incorrectly.");
            return;
        }
        if(!Validation.isID(music.getMusicId())){
```

```
            System.out.println("Music ID entered incorrectly.");
            return;
        }

        if(!Validation.isID(music.getCatalogueId())){
            System.out.println("Catalogue ID entered incorrectly.");
            return;
        }

        if(!Validation.isID(music.getCreatorId())){
            System.out.println("Creator ID entered incorrectly.");
            return;
        }

        if(!Validation.isView(music.getViews())){
            System.out.println("Views entered incorrectly.");
            return;
        }

        if(Error.isMusicIdTaken(connection, music.getMusicId())){
            System.out.println("This music id is already taken.");
            return;
        }

        final String sql = "INSERT INTO \"musics\"(music_id, music_name,
creator_id, catalogue_id, views) VALUES (?,?,?,?,?)";
        try(PreparedStatement statement =
connection.prepareStatement(sql)){
            statement.setInt(1, music.getMusicId());
            statement.setString(2, music.getMusicName());
            statement.setInt(3, music.getCreatorId());
            statement.setInt(4, music.getCatalogueId());
            statement.setInt(5, music.getViews());

            if(statement.executeUpdate() == 0) {
                System.out.println("Error! The \"musics\" table has not
changed.");
            }
            else System.out.println("Music successfully added!");
        } catch(SQLException e){
            throw new DataBaseException("SQL Error! Unexpected error
related to adding a music");
        }
    }
    public void deleteMusic(int musicId){
        if(!Validation.isID(musicId)){
            System.out.println("Music ID entered incorrectly.");
            return;
        }

        if(!Error.isMusicIdTaken(connection, musicId)){
            System.out.println("This music id is not exist.");
            return;
        }
```

```
        final String sql = "DELETE FROM \"musics\" WHERE \"music_id\" =
?";
        try(PreparedStatement statement =
connection.prepareStatement(sql)){
            statement.setInt(1, musicId);
            if(statement.executeUpdate() == 0) {
                System.out.println("Error! Table is empty!");
            }
            else System.out.println("Music successfully deleted!");
        } catch(SQLException e){
            throw new DataBaseException("SQL Error! Unexpected error
related to music deletion");
        }
    }
    public void updateMusic(Music music){
        if(music == null) {
            System.out.println("Error! You have not entered the data");
            return;
        }

        if(!Validation.isID(music.getMusicId())){
            System.out.println("Music ID entered incorrectly.");
            return;
        }

        if(!Validation.isName(music.getMusicName())){
            System.out.println("Music name entered incorrectly.");
            return;
        }

        if(!Validation.isID(music.getCatalogueId())){
            System.out.println("Catalogue ID entered incorrectly.");
            return;
        }

        if(!Validation.isID(music.getCreatorId())){
            System.out.println("Creator ID entered incorrectly.");
            return;
        }

        if(!Validation.isView(music.getViews())){
            System.out.println("Views entered incorrectly.");
        }

        if(!Error.isMusicIdTaken(connection, music.getMusicId())){
            System.out.println("This music id is not exist.");
            return;
        }

        final String sql = "UPDATE \"musics\" SET \"music_name\"= ?,
\"creator_id\"= ?, \"catalogue_id\"= ?, " +
                "\"views\" = ? WHERE \"music_id\"= ?\n";
        try(PreparedStatement statement =
```

```
connection.prepareStatement(sql)){
            statement.setInt(5, music.getMusicId());
            statement.setString(1, music.getMusicName());
            statement.setInt(2, music.getCreatorId());
            statement.setInt(3, music.getCatalogueId());
            statement.setInt(4, music.getViews());

            if(statement.executeUpdate() == 0) {
                System.out.println("Error! The \"musics\" table has not
changed.");
            }
            else System.out.println("Music successfully updated!");
        } catch(SQLException e){
            throw new DataBaseException(" SQL Error! Unexpected error
related to music data change");
        }
    }

    public ArrayList<Creator> getAllCreator(){
        final String sql = "SELECT * FROM \"creators\"\n" + "ORDER BY
\"creator_id\"";
        ArrayList<Creator> creators = new ArrayList<>();
        try(PreparedStatement statement =
connection.prepareStatement(sql)){
            ResultSet resultSet = statement.executeQuery();
            while(resultSet.next()){
                int creatorId = resultSet.getInt("creator_id");
                String firstName = resultSet.getString("first_name");
                String lastName = resultSet.getString("last_name");

                creators.add(new Creator(creatorId, firstName,
lastName));
            }
            return creators;
        } catch(SQLException e){
            throw new DataBaseException("SQL Error! Unexpected error
related to creators data retrieval");
        }
    }
    public Creator getCreator(int creatorId){
        if(!Validation.isID(creatorId)){
            System.out.println("Creator ID entered incorrectly.");
            return null;
        }

        if(!Error.isCreatorIdTaken(connection, creatorId)){
            System.out.println("This creator id is not exist.");
            return null;
        }

        final String sql = "SELECT \"first_name\",  \"last_name\" FROM
\"creators\" WHERE \"creator_id\"= ?";
        try(PreparedStatement statement =
connection.prepareStatement(sql)){
```

```
            statement.setInt(1, creatorId);
            ResultSet resultSet = statement.executeQuery();
            if(resultSet.next()){
                String firstName = resultSet.getString("first_name");
                String lastName = resultSet.getString("last_name");
                return new Creator(creatorId, firstName, lastName);
            }
            System.out.println("Error! Table is empty!");
            return null;
        } catch(SQLException e){
            throw new DataBaseException("SQL Error! Unexpected error
related to creator data retrieval");
        }
    }

    public void addCreator(Creator creator){
        if(creator == null) {
            System.out.println("Error! You have not entered the data");
            return;
        }
        if(!Validation.isName(creator.getFirstName())){
            System.out.println("Creator first name entered
incorrectly.");
            return;
        }
        if(!Validation.isName(creator.getLastName())){
            System.out.println("Creator last name entered incorrectly.");
            return;
        }
        if(!Validation.isID(creator.getCreatorId())){
            System.out.println("Creator ID entered incorrectly.");
            return;
        }

        if(Error.isCreatorIdTaken(connection, creator.getCreatorId())){
            System.out.println("This creator id is already taken.");
            return;
        }

        final String sql = "INSERT INTO \"creators\"(creator_id,
first_name, last_name) VALUES(?,?,?)";
        try(PreparedStatement statement =
connection.prepareStatement(sql)){
            statement.setInt(1, creator.getCreatorId());
            statement.setString(2, creator.getFirstName());
            statement.setString(3, creator.getLastName());
            if(statement.executeUpdate() == 0) {
                System.out.println("Error! The \"users\" table has not
changed.");
            }
            else System.out.println("Creator successfully added!");
        } catch(SQLException e){
            throw new DataBaseException("SQL Error! Unexpected error
related to adding a creator");
```

```
        }
    }

    public void deleteCreator(int creatorId){
        if(!Validation.isID(creatorId)){
            System.out.println("Creator ID entered incorrectly.");
            return;
        }

        if(!Error.isCreatorIdTaken(connection, creatorId)){
            System.out.println("This creator id is not exist.");
            return;
        }

        final String sql = "DELETE FROM \"creators\" WHERE
\"creator_id\"= ?";
        try(PreparedStatement statement =
connection.prepareStatement(sql)){
            statement.setInt(1, creatorId);
            if(statement.executeUpdate() == 0) {
                System.out.println("Error! Table is empty!");
            }
            else System.out.println("Creator successfully deleted!");
        } catch(SQLException e){
            throw new DataBaseException("SQL Error! Unexpected error
related to creator deletion");
        }
    }
    public void updateCreator(Creator creator){
        if(creator == null) {
            System.out.println("Error! You have not entered the data");
            return;
        }
        if(!Validation.isName(creator.getFirstName())){
            System.out.println("Creator first name entered
incorrectly.");
            return;
        }
        if(!Validation.isName(creator.getLastName())){
            System.out.println("Creator last name entered incorrectly.");
            return;
        }
        if(!Validation.isID(creator.getCreatorId())){
            System.out.println("Creator ID entered incorrectly.");
            return;
        }

        if(!Error.isCreatorIdTaken(connection, creator.getCreatorId())){
            System.out.println("This creator id is not exist.");
            return;
        }

        final String sql = "UPDATE \"creators\" SET \"first_name\"= ?,
\"last_name\"= ? WHERE \"creator_id\"= ?";
```

```
        try(PreparedStatement statement =
connection.prepareStatement(sql)){
            statement.setInt(3, creator.getCreatorId());
            statement.setString(1, creator.getFirstName());
            statement.setString(2, creator.getLastName());

            if(statement.executeUpdate() == 0) {
                System.out.println("Error! The \"creators\" table has not
changed.");
            }
            else System.out.println("Creator successfully updated!");
        } catch(SQLException e){
            throw new DataBaseException("SQL Error! Unexpected error
related to creator data change");
        }
    }

    public void randomGenerateUser(int recordCount) {
        String generateSql = """
            SELECT "first_name", "last_name",
                   LOWER("first_name" || '.' || "last_name") ||
'@gmail.com' as "email"
            FROM unnest(array['Max', 'Ann', 'John', 'Ava', 'Bob',
'Tomas', 'Mia', 'Richard',
            'Freya', 'Florence', 'Oliver', 'Noah', 'Laura', 'Mike',
'Johan', 'James',
             'Arthur', 'Leo', 'Harry', 'Oscar', 'Harry', 'Amelia',
'Lily', 'Stephen']) AS "first_name"
            CROSS JOIN unnest(array['Lebron', 'Nelson', 'Smith', 'Jones',
'Williams', 'Brown', 'Taylor',
            'Davies', 'Evans', 'Williams', 'Stone', 'Bell', 'Campbell',
'Morgan', 'Lewis', 'Roberts',
            'Evans', 'Adams', 'Gibson', 'Peters', 'Shelby', 'Harry',
'Carry', 'Davis']) AS "last_name"
            ORDER BY random()
            LIMIT ?;
            """;
        String maxIdSql = "SELECT COALESCE(MAX(user_id), 0) FROM users";
        String insertSql = "INSERT INTO \"users\" (\"user_id\",
\"first_name\", \"last_name\", \"email\") VALUES (?, ?, ?, ?)";

        int startingId = 0;
        try (Statement statement = connection.createStatement();
             ResultSet rs = statement.executeQuery(maxIdSql)) {
            if (rs.next()) {
                startingId = rs.getInt(1);
            }
        }
        catch(SQLException e){
            throw new DataBaseException("SQL error! Your number: " +
recordCount + " is not allowed.");
        }

        try (PreparedStatement generatedStatement =
```

```
connection.prepareStatement(generateSql);
             PreparedStatement insertStatement =
connection.prepareStatement(insertSql)) {

            generatedStatement.setInt(1, recordCount);

            ResultSet rs = generatedStatement.executeQuery();

            while (rs.next()) {
                String firstName = rs.getString("first_name");
                String lastName = rs.getString("last_name");
                String email = rs.getString("email");

                insertStatement.setInt(1, ++startingId);
                insertStatement.setString(2, firstName);
                insertStatement.setString(3, lastName);
                insertStatement.setString(4, email);

                insertStatement.executeUpdate();
            }
            System.out.println(recordCount + " records inserted
successfully.");
        } catch (SQLException e) {
            throw new DataBaseException("SQL error! Unexpected error
related to data generation for users table");
        }
    }
    public void randomGenerateCreator(int recordCount){
        String generateSql = """
            SELECT "first_name", "last_name"
            FROM unnest(array['Max', 'Ann', 'John', 'Ava', 'Bob',
'Tomas', 'Mia', 'Richard',
            'Freya', 'Florence', 'Oliver', 'Noah', 'Laura', 'Mike',
'Johan', 'James',
             'Arthur', 'Leo', 'Harry', 'Oscar', 'Harry', 'Amelia',
'Lily', 'Stephen']) AS "first_name"
            CROSS JOIN unnest(array['Lebron', 'Nelson', 'Smith', 'Jones',
'Williams', 'Brown', 'Taylor',
            'Davies', 'Evans', 'Williams', 'Stone', 'Bell', 'Campbell',
'Morgan', 'Lewis', 'Roberts',
            'Evans', 'Adams', 'Gibson', 'Peters', 'Shelby', 'Harry',
'Carry', 'Davis']) AS "last_name"
            ORDER BY random()
            LIMIT ?;
            """;
        String maxIdSql = "SELECT COALESCE(MAX(creator_id), 0) FROM
creators";
        String insertSql = "INSERT INTO \"creators\" (\"creator_id\",
\"first_name\", \"last_name\") VALUES (?, ?, ?)";

        int startingId = 0;
        try (Statement stmt = connection.createStatement();
             ResultSet rs = stmt.executeQuery(maxIdSql)) {
            if (rs.next()) {
```

```
                startingId = rs.getInt(1);
            }
        }
        catch(SQLException e){
            throw new DataBaseException("SQL error! Your number: " +
recordCount + " is not allowed.");
        }

        try (PreparedStatement generatedStatement =
connection.prepareStatement(generateSql);
             PreparedStatement insertStatement =
connection.prepareStatement(insertSql)) {

            generatedStatement.setInt(1, recordCount);

            ResultSet resultSet = generatedStatement.executeQuery();

            while (resultSet.next()) {
                String firstName = resultSet.getString("first_name");
                String lastName = resultSet.getString("last_name");

                insertStatement.setInt(1, ++startingId);
                insertStatement.setString(2, firstName);
                insertStatement.setString(3, lastName);

                insertStatement.executeUpdate();
            }
            System.out.println(recordCount + " records inserted
successfully.");
        } catch (SQLException e) {
            throw new DataBaseException("SQL error!");
        }
    }

    public void getCreatorViews(int creatorId){
        if(!Validation.isID(creatorId)){
            System.out.println("Creator ID entered incorrectly.");
            return;
        }

        if(!Error.isCreatorIdTaken(connection, creatorId)){
            System.out.println("This creator id is not exist.");
            return;
        }

        long startTime, endTime, duration;
        startTime = System.nanoTime();
        ArrayList<CreatorStats> creatorStats = new ArrayList<>();
        final String sql = """
                SELECT c."first_name", c."last_name", SUM(m."views") AS
total_views
                FROM public."musics" m
                INNER JOIN public."creators" c ON c."creator_id" =
m."creator_id"
```

```
                WHERE c."creator_id" =?
                GROUP BY c."first_name", c."last_name"
                ORDER BY total_views DESC
                """;
        try(PreparedStatement statement =
connection.prepareStatement(sql)){
            statement.setInt(1, creatorId);
            ResultSet resultSet = statement.executeQuery();
            while(resultSet.next()){
                String name = resultSet.getString("first_name");
                String lastName = resultSet.getString("last_name");
                int views = resultSet.getInt("total_views");

                creatorStats.add(new CreatorStats(name, lastName,
views));
            }
        }catch(SQLException e){
            throw new DataBaseException("SQL Error! Unexpected error
related to data generation for creators table");
        }

        for(CreatorStats cs : creatorStats) cs.show();

        endTime = System.nanoTime();
        duration = endTime - startTime;
        System.out.println("Time spent: " + (duration) + " nanoseconds");
    }

    public void getBiggestCatalogue(){
        long startTime, endTime, duration;
        startTime = System.nanoTime();
        BiggestCatalogue catalogue;
        final String sql = """
                SELECT u."first_name", u."last_name", c."catalogue_name",
COUNT(m."music_id") AS music_count
                FROM "catalogues" c
                LEFT JOIN "musics" m ON c."catalogue_id" =
m."catalogue_id"
                INNER JOIN "users" u ON u."user_id" = c."user_id"
                GROUP BY u."first_name", u."last_name",
c."catalogue_name"
                ORDER BY music_count DESC
                LIMIT 1
                """;
        try(PreparedStatement statement =
connection.prepareStatement(sql)){
            ResultSet resultSet = statement.executeQuery();
            if(resultSet.next()){
                String firstName = resultSet.getString("first_name");
                String lastName = resultSet.getString("last_name");
                String catalogueName =
resultSet.getString("catalogue_name");
                int musicCount = resultSet.getInt("music_count");
```

```
                catalogue = new BiggestCatalogue(firstName, lastName,
catalogueName, musicCount);
            }
            else{
                System.out.println("Error! Data is not exist.");
                return;
            }
        }catch(SQLException e){
            throw new DataBaseException("SQL Error! Unexpected error
related to catalogue data retrieval");
        }
        catalogue.show();
        endTime = System.nanoTime();
        duration = endTime - startTime;
        System.out.println("Time spent: " + (duration) + " nanoseconds");
}

    public void getPopularMusic(){
        long startTime, endTime, duration;
        startTime = System.nanoTime();

        ArrayList<PopularMusic> musics = new ArrayList<>();
        final String sql = """
                SELECT c."first_name", c."last_name", m."music_name",
m."views"
                FROM "musics" m
                LEFT JOIN "creators" c ON c."creator_id" = m."creator_id"
                GROUP BY c."first_name", c."last_name", m."music_name",
m."views"
                ORDER BY m."views" DESC
                LIMIT 3
                """;
        try(PreparedStatement statement =
connection.prepareStatement(sql)){
            ResultSet resultSet = statement.executeQuery();
            while(resultSet.next()){
                String firstName = resultSet.getString("first_name");
                String lastName = resultSet.getString("last_name");
                String musicName = resultSet.getString("music_name");
                int views = resultSet.getInt("views");

                PopularMusic music = new PopularMusic(firstName,
lastName, musicName, views);
                musics.add(music);
            }
                if(!musics.isEmpty())
                for(PopularMusic music : musics) music.show();
                else System.out.println("Error! Table is empty!\n");

        }catch(SQLException e){
            throw new DataBaseException("SQL Error! Unexpected error
related to receive data from musics and/or creator.");
        }
        endTime = System.nanoTime();
```

```
        duration = endTime - startTime;
        System.out.println("Time spent: " + (duration) + " nanoseconds");

    }
}
```

User:

ArrayList<User> getAllUser() – повертає масив об’єктів User, який тримає в собі всі дані таблиці users

User getUser(int userId) – повертає об’єкт User, в якому дані *користувача*, якого обрав наш користувач

void addUser(User user) – додаємо до таблиці users нового *користувача*, якого ввів наш користувач

void deleteUser(int userId) – видаляємо *користувача* з таблиці users, якого обрав наш користувач

void updateUser(User user) – поновлюємо дані *користувача* (окрім foreign key), якого обрав наш користувач

Catalogue:

ArrayList<Catalogue> getAllCatalogue() – повертає масив об’єктів Catalogue, який тримає в собі всі дані таблиці catalogues

Catalogue getCatalogue(int catalogueId) – повертає каталог, який обрав наш користувач по ID

void getBiggestCatalogue() – повертає найбільший за кількістю пісень каталог, та *користувача* який володіє цим каталогом.

void addCatalogue (Catalogue catalogue) – додаємо до таблиці catalogues новий каталог

void deleteCatalogue (int ID) – видаляє каталог обраний користувачем

void updateCatalogue (Catalogue catalogue) –поновлюємо каталог обраний користувачем

Music:

ArrayList<Music> getAllMusic() – повертає всі дані з таблиці musics

Music getMusic(int musicId) – повертає пісню, який обрав наш користувач по ID

Void getPopularMusic() – отримати 3 найбільш популярні музики

void addMusic(Music music) – додає нову пісню, яку обрав користувач

void deleteMusic(int musicId) – видаляє стару пісню, яку обрав користувач (не чипаючи автора, бо пісня залежить від автора, а не навпаки)

void updateMusic(Music music) – оновлює дані обраної пісні користувачем

Creator:

ArrayList<Creator> getAllCreator() – повертає всі дані з таблиці creators

Creator getCreator(int creatorId) – повертає автора

void getCreatorViews() – отримуємо автора, якого обрав користувач, і переглядаємо сумарну кількість переглядів за всі пісні автора

void addCreator (Creator creator) – додає автора до таблиці creators

void deleteCreator(int creatorId) – видаляти автора

void updateCreator (Creator creator) – оновити автора, якого обрав наш користувач