МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

**НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»**

***Кафедра Систем Штучного Інтелекту***

**РОЗРАХУНКОВА РОБОТА**

з дисципліни «Організація баз даних та знань»

на тему:

«Веб-сервіс пошуку кіно»

Виконав:

студент групи КН-210

Кухар Ярослав Романович

|  |  |
| --- | --- |
| **Балів** | **Дата** |
|  |  |

Викладач:

Мельникова Наталя Іванівна

Львів – 2020

**Зміст**

[1. Тема проекту. 3](#_Toc40868254)

[2. Вступ. 4](#_Toc40868255)

[3. Логічна схема БД проекту. 5](#_Toc40868256)

[4. Опис структури БД. 6](#_Toc40868257)

[5. Фізична модель БД. 9](#_Toc40868258)

[6. Ділова модель. 16](#_Toc40868259)

[7. Запити до БД. 17](#_Toc40868260)

[8. Висновки. 21](#_Toc40868261)

[9. Список використаних джерел інформації. 22](#_Toc40868262)

# ТЕМА ПРОЕКТУ.

Темою проекту моєї команди EK-Team є веб-сервіс, який дозволяє користувачу легко знаходити потрібні сеанси в кінотеатрах Львова. За допомогою цього сервісу можна легко і швидко знаходити дешеві квитки в кіно. Крім цього, веб-сторінка надає інформацію про фільм, включаючи опис; ім’я режисера, акторів; назви студій, де знімали фільм; жанри фільму.

Користувач заходить на наш сайт, який можна переглянути за посиланням нижче, і бачить список фільмів, для яких на сьогодні є сеанси в кінотеатрах Львова. Він також може вибрати будь-яку дату, в межах одного тижня, бо кінотеатри планують свої сеанси в межах одного тижня. Також він може здійснити пошук певного фільму. Є блок анонсів, де зібрані фільми, які ще не вийшли в прокат, але скоро вийдуть. Ще є інформація про всі кінотеатри Львова, яку користувач може переглянути внизу сторінки. Інша сторінка – це інформація про фільм, наприклад, його постер, назва, трейлер, опис, рейтинг і тд. На цій сторінці можна переглянути сеанси на цей фільм у всіх кінотеатрах Львова, відсортувавши або відфільтрувавши ці сеанси за певними критеріями. Загалом, цей сервіс надає швидкий доступ до всіх сеансів, з зручними фільтрами і сортуванням.

Переглянути веб-сервіс та його функціонал можна за [посиланням](https://econom-kino.herokuapp.com)

# 2. ВСТУП.

Для даної розрахункової роботи було обрано тему саме веб-сервісів базуючись на поданих перевагах:

* **Вони дають можливість набути досвіду в різних аспектах розробки**

**залишаючи гнучкість для перерозподілу навантаження в процесі створення**. Це означає, що сервіс може включати як власний Back-End, так і Front-End з базою даний при цьому даючи можливість різних розподілів як навантаження так і обігу даних залежно від можливостей та потреб. Таким чином частиною даних база даних може не оперувати, якщо зручніше буде тримати це серед даних сервера Front-End-у чи Back-End-у.

* **Вони чудово масштабуються**. Сервіси можуть бути елементом як

цілком самостійним, так і ланкою складного проекту чи його основою, що робить його чудовою можливістю почати з малого і розвивати до великого і складного приєднуючи інші сервіси чи приєднуючись до інших сервісів.

Щодо причин вибору тематики кіно можна виділити наступні переваги:

* **Активна аудиторія**. Обираючи кіно, на яке бажають сходити люди

переважно звертаються до сервісів кінотеатрів чи інших веб-сервісів замість живого відвідування кінотеатрів чи інших засобів пошуку інформації (ЗМІ, газети, радіо чи живий обмін інформацією) і попит кінофільмів наразі значний.

* **Платоспроможна аудиторія**. В кінотеатри ходять переважно люди з

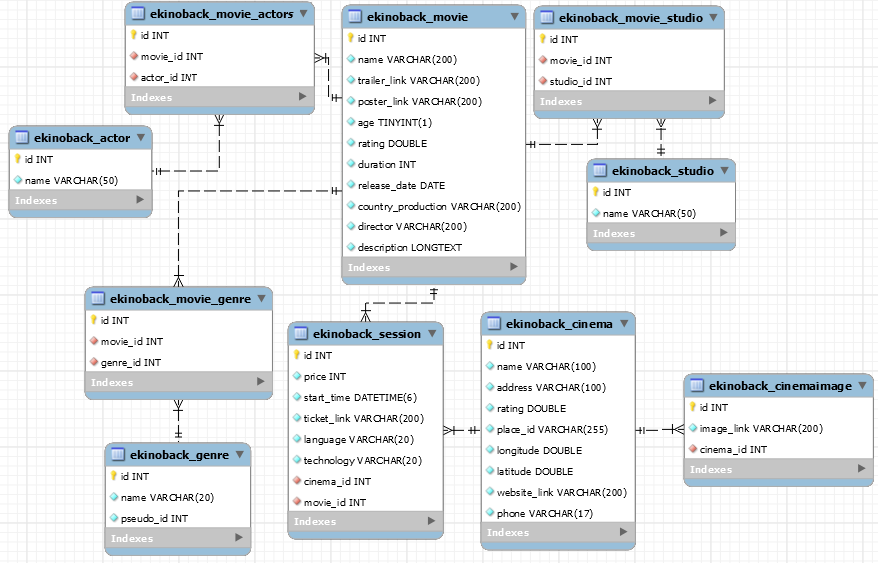
певним рівнем платоспроможності, що дозволяє в майбутньому продавати дорожчу рекламу чи за рахунок взаємодії з кінотеатрами монетизувати сервіс.

* **Незначна конкуренція**. Згідно з статистикою, найбільше люди обирають

кіно з сайтів кінотеатрів, хоча вже існують деякі сервіси з підбору кіно по різних кінотеатрах міст. Часто це пов’язано з незвичністю чи надмірною складністю інтерфейсу користувача, хоча наявність програм лояльності кінотеатрів теж знижує популярність таких сервісів.

З огляду на ці переваги, ми вирішили зробити веб-сервіс «Econom-Kino», який допомагає користувачу отримати усі сеанси з усіх кінотеатрв з багатьма гнучкими фільтр.

# 3. ЛОГІЧНА СХЕМА БД ПРОЕКТУ



*(Рис. 1. Схема БД Econom-Kino)*

# 4. ОПИС СТРУКТУРИ БД.

Логічна схема БД подана у попередньому пункті. Почнемо з однієї з головних таблиць у структурі БД – таблиці *ekinoback\_movie*. Структура цієї таблиці:

* id INT – Внутрішній ключ, цілочисельний унікальний ідентифікатор. Це поле використовується в кожній таблиці БД, тому в решті таблиць його не розглядатимемо.
* name varchar(200) – Назва фільму. Обов’язковий атрибут, обмеження – 200 символів. Всі атрибути даної БД є обов’язковими, тобто значенням поля не може бути NULL, тому в решті атрибутів таблиць цього не розглядатимемо.
* trailer\_link і poster\_link – Посилання відповідно на трейлер фільму та на постер фільму. Обмеження – по 200 символів.
* age – Вікове обмеження (True або False).
* rating – Рейтинг фільму за версією IMDB від 0 до 10.
* duration – Тривалість фільму в хвилинах.
* release\_date – Дата релізу фільму.
* country\_production – Країна-автор фільму.
* director – Режисер або режисери фільму.
* description – Опис фільму.

Крім цього, характеристиками сутності Фільм є ще Жанри фільму,

Актори, які знімались у фільмі, а також Студії, які знімали цей фільм. Для цього створено таблиці:

*ekinoback\_genre:*

* name – Назва жанру. Обмеження – 20 символів.
* pseudo\_id – Специфічний ідентифікатор жанру, потрібен для правильного функціонування парсерів, які вносять дані у базу даних.

*ekinoback\_actor:*

* name – Ім’я і прізвище актора. Обмеження – 50 символів.

*ekinoback\_studio:*

* name – Назва студії. Обмеження – 50 символів.

Всі ці сутності відносяться до сутності Фільму. Зв’язок між цими таблицями і таблицею movie – Багато до багатьох (many-to-many), адже, наприклад, в одному фільмі може зніматись кілька акторів, і один актор може зніматись в кількох фільмах. Аналогічна ситуація для Студій і Жанрів. Тому всі ці зв’язки організовані у вигляді many-to-many за допомогою проміжних таблиць:

* *ekinoback\_movie\_genre;*
* *ekinoback\_movie\_studio;*
* *ekinoback\_movie\_actor;*

які мають однакову структуру – Ідентифікатор фільму та ідентифікатор жанру/студії/актора.

Наступною таблицею, яку ми розглянемо, буде *ekinoback\_cinema* – Кінотеатр:

* name – Назва кінотеатру. Обмеження – 100 символів.
* address – Адреса кінотеатру. Обмеження – 100 символів.
* rating – Рейтинг кінотеатру, оцінка цього місця на Google Maps.
* place\_id – Ідентифікатор кінотеатру, який використовується в Google Maps. Обмеження – 255 символів.
* longitude – Довгота адреси кінотеатру.
* latitude – Широта адреси кінотеатру.
* website\_link – Посилання на веб-сайт кінотеатру.Обмеження – 200 символів.
* phone – Номер телефону кінотуатру. Обмеження – 17 символів.

Ця таблиця майже повністю описує сутність Кінотеатр, проте для правильної роботи нашого застосунку кінотеатр повинен мати його фотографії, які також мають зберігатись у базі даних. Для цього створюємо ще одну таблицю, *ekinoback\_cinemaimage*, яка відповідатиме за зберігання посилань на фотографії певного кінотеатру:

* image\_link – Посилання на фотографію певного кінотеатру. Обмеження – 200 символів.
* cinema\_id – Зовнішній ключ, який посилається на Кінотеатр, якому належить це зображення.

Це зв’язок Один до багатьох (one-to-many), який реалізується за допомогою зовнішнього ключа.

Наступна, яка водночас є і останньою, таблиця – це *ekinoback\_session*, яка відповідає за сутність Сеанс:

* price – Ціна квитка.
* start\_time – Дата і час початку сеансу.
* ticket\_link – Посилання на сторінку, де можна придбати квиток саме на цей сеанс. Обмеження – 200 символів.
* language – Мова показу фільму. Обмеження – 20 символів.
* technology – Технологія показу фільму, для прикладу, 2D, 3D, 4DX, Cinetech+ і тд.
* cinema\_id – Зовнішній ключ, який посилається на кінотеатр, в якому проводитиметься цей сеанс.
* movie\_id – Зовнішній ключ, який посилається на фільм, який показуватиметься під час цього сеансу.

Крім цього, розглянемо індекси, які є у цій базі даних, крім внутрішніх ключів.

* movie.name – цей індекс потрібен тому, що часто здійснюється пошук фільму за назвою.
* movie.rating – цей індекс потрібен тому, що часто проводиться сортування та фільтрація за рейтингом.
* cinema.name та cinema.rating – за тих же самих причин, що й попередні пункти відповідно.
* session.price – цей індекс потрібен, бо часто здійснюється фільтрування або сортування за ціною.

# 5. ФІЗИЧНА МОДЕЛЬ БД.

Текст файлу створення БД з оголошенням обмежень, індексів та ключів:

SET @OLD\_UNIQUE\_CHECKS=@@UNIQUE\_CHECKS, UNIQUE\_CHECKS=0;

SET @OLD\_FOREIGN\_KEY\_CHECKS=@@FOREIGN\_KEY\_CHECKS, FOREIGN\_KEY\_CHECKS=0;

SET @OLD\_SQL\_MODE=@@SQL\_MODE, SQL\_MODE='ONLY\_FULL\_GROUP\_BY,STRICT\_TRANS\_TABLES,NO\_ZERO\_IN\_DATE,NO\_ZERO\_DATE,ERROR\_FOR\_DIVISION\_BY\_ZERO,NO\_ENGINE\_SUBSTITUTION';

CREATE SCHEMA IF NOT EXISTS `ekinobase` DEFAULT CHARACTER SET utf8mb4 COLLATE utf8mb4\_0900\_ai\_ci ;

USE `ekinobase` ;

CREATE TABLE IF NOT EXISTS `ekinobase`.`ekinoback\_actor` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(50) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 1

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE TABLE IF NOT EXISTS `ekinobase`.`ekinoback\_cinema` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(100) NOT NULL,

`address` VARCHAR(100) NOT NULL,

`rating` DOUBLE NOT NULL,

`place\_id` VARCHAR(255) NOT NULL,

`longitude` DOUBLE NOT NULL,

`latitude` DOUBLE NOT NULL,

`website\_link` VARCHAR(200) NOT NULL,

`phone` VARCHAR(17) NOT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `name` (`name` ASC),

INDEX `cinema\_rating` (`rating` ASC))

ENGINE = InnoDB

AUTO\_INCREMENT = 1

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE TABLE IF NOT EXISTS `ekinobase`.`ekinoback\_cinemaimage` (

`id` INT NOT NULL AUTO\_INCREMENT,

`image\_link` VARCHAR(200) NOT NULL,

`cinema\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `ekinoback\_cinemaimage\_cinema\_id\_30f3e107\_fk\_ekinoback\_cinema\_id` (`cinema\_id` ASC),

CONSTRAINT `ekinoback\_cinemaimage\_cinema\_id\_30f3e107\_fk\_ekinoback\_cinema\_id`

FOREIGN KEY (`cinema\_id`)

REFERENCES `ekinobase`.`ekinoback\_cinema` (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 1

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE TABLE IF NOT EXISTS `ekinobase`.`ekinoback\_genre` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(20) NOT NULL,

`pseudo\_id` INT NOT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `name` (`name` ASC) VISIBLE,

INDEX `ekinoback\_genre\_pseudo\_id\_820c15b3` (`pseudo\_id` ASC) )

ENGINE = InnoDB

AUTO\_INCREMENT = 1

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE TABLE IF NOT EXISTS `ekinobase`.`ekinoback\_movie` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(200) NOT NULL,

`trailer\_link` VARCHAR(200) NOT NULL,

`poster\_link` VARCHAR(200) NOT NULL,

`age` TINYINT(1) NOT NULL,

`rating` DOUBLE NOT NULL,

`duration` INT NOT NULL,

`release\_date` DATE NOT NULL,

`country\_production` VARCHAR(200) NOT NULL,

`director` VARCHAR(200) NOT NULL,

`description` LONGTEXT NOT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `name` (`name` ASC) ,

INDEX `movie\_rating` (`rating` ASC) )

ENGINE = InnoDB

AUTO\_INCREMENT = 1

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE TABLE IF NOT EXISTS `ekinobase`.`ekinoback\_movie\_actors` (

`id` INT NOT NULL AUTO\_INCREMENT,

`movie\_id` INT NOT NULL,

`actor\_id` INT NOT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `ekinoback\_movie\_actors\_movie\_id\_actor\_id\_2f821621\_uniq` (`movie\_id` ASC, `actor\_id` ASC) ,

INDEX `ekinoback\_movie\_actors\_actor\_id\_51113f26\_fk\_ekinoback\_actor\_id` (`actor\_id` ASC) ,

CONSTRAINT `ekinoback\_movie\_actors\_actor\_id\_51113f26\_fk\_ekinoback\_actor\_id`

FOREIGN KEY (`actor\_id`)

REFERENCES `ekinobase`.`ekinoback\_actor` (`id`),

CONSTRAINT `ekinoback\_movie\_actors\_movie\_id\_ebe45bce\_fk\_ekinoback\_movie\_id`

FOREIGN KEY (`movie\_id`)

REFERENCES `ekinobase`.`ekinoback\_movie` (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 1

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE TABLE IF NOT EXISTS `ekinobase`.`ekinoback\_movie\_genre` (

`id` INT NOT NULL AUTO\_INCREMENT,

`movie\_id` INT NOT NULL,

`genre\_id` INT NOT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `ekinoback\_movie\_genre\_movie\_id\_genre\_id\_1a7e02ed\_uniq` (`movie\_id` ASC, `genre\_id` ASC) ,

INDEX `ekinoback\_movie\_genre\_genre\_id\_377ee090\_fk\_ekinoback\_genre\_id` (`genre\_id` ASC),

CONSTRAINT `ekinoback\_movie\_genre\_genre\_id\_377ee090\_fk\_ekinoback\_genre\_id`

FOREIGN KEY (`genre\_id`)

REFERENCES `ekinobase`.`ekinoback\_genre` (`id`),

CONSTRAINT `ekinoback\_movie\_genre\_movie\_id\_0879da91\_fk\_ekinoback\_movie\_id`

FOREIGN KEY (`movie\_id`)

REFERENCES `ekinobase`.`ekinoback\_movie` (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 1

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE TABLE IF NOT EXISTS `ekinobase`.`ekinoback\_studio` (

`id` INT NOT NULL AUTO\_INCREMENT,

`name` VARCHAR(50) NOT NULL,

PRIMARY KEY (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 1

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE TABLE IF NOT EXISTS `ekinobase`.`ekinoback\_movie\_studio` (

`id` INT NOT NULL AUTO\_INCREMENT,

`movie\_id` INT NOT NULL,

`studio\_id` INT NOT NULL,

PRIMARY KEY (`id`),

UNIQUE INDEX `ekinoback\_movie\_studio\_movie\_id\_studio\_id\_0ea4d2e7\_uniq` (`movie\_id` ASC, `studio\_id` ASC),

INDEX `ekinoback\_movie\_studio\_studio\_id\_0c5af49e\_fk\_ekinoback\_studio\_id` (`studio\_id` ASC),

CONSTRAINT `ekinoback\_movie\_studio\_movie\_id\_cbf548d0\_fk\_ekinoback\_movie\_id`

FOREIGN KEY (`movie\_id`)

REFERENCES `ekinobase`.`ekinoback\_movie` (`id`),

CONSTRAINT `ekinoback\_movie\_studio\_studio\_id\_0c5af49e\_fk\_ekinoback\_studio\_id`

FOREIGN KEY (`studio\_id`)

REFERENCES `ekinobase`.`ekinoback\_studio` (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 1

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

CREATE TABLE IF NOT EXISTS `ekinobase`.`ekinoback\_session` (

`id` INT NOT NULL AUTO\_INCREMENT,

`price` INT NOT NULL,

`start\_time` DATETIME(6) NOT NULL,

`ticket\_link` VARCHAR(200) NOT NULL,

`language` VARCHAR(20) NOT NULL,

`technology` VARCHAR(20) NOT NULL,

`cinema\_id` INT NOT NULL,

`movie\_id` INT NOT NULL,

PRIMARY KEY (`id`),

INDEX `ekinoback\_session\_cinema\_id\_d7882bf8\_fk\_ekinoback\_cinema\_id` (`cinema\_id` ASC),

INDEX `ekinoback\_session\_movie\_id\_17d4bf2c\_fk\_ekinoback\_movie\_id` (`movie\_id` ASC),

INDEX `ekinoback\_session\_price\_10262086` (`price` ASC),

CONSTRAINT `ekinoback\_session\_cinema\_id\_d7882bf8\_fk\_ekinoback\_cinema\_id`

FOREIGN KEY (`cinema\_id`)

REFERENCES `ekinobase`.`ekinoback\_cinema` (`id`),

CONSTRAINT `ekinoback\_session\_movie\_id\_17d4bf2c\_fk\_ekinoback\_movie\_id`

FOREIGN KEY (`movie\_id`)

REFERENCES `ekinobase`.`ekinoback\_movie` (`id`))

ENGINE = InnoDB

AUTO\_INCREMENT = 1

DEFAULT CHARACTER SET = utf8mb4

COLLATE = utf8mb4\_0900\_ai\_ci;

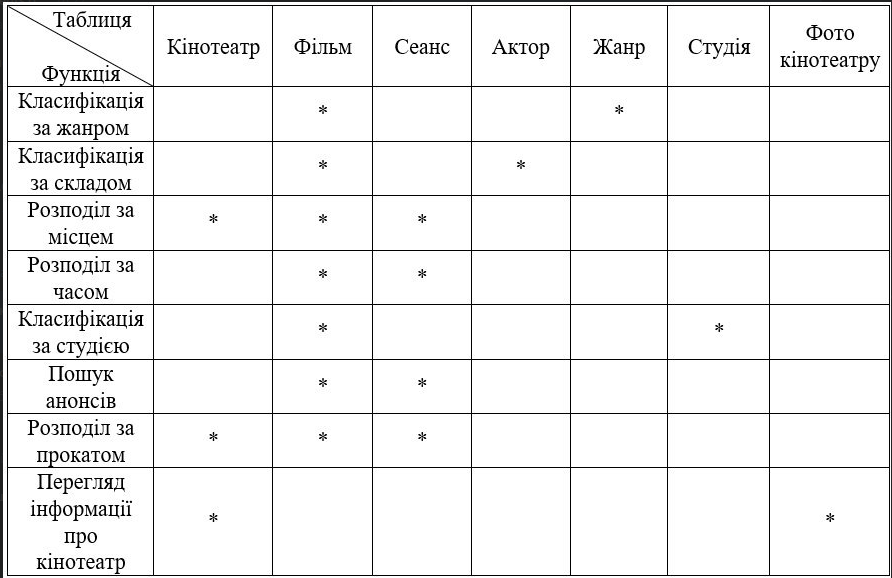
SET SQL\_MODE=@OLD\_SQL\_MODE;

SET FOREIGN\_KEY\_CHECKS=@OLD\_FOREIGN\_KEY\_CHECKS;

SET UNIQUE\_CHECKS=@OLD\_UNIQUE\_CHECKS;

# 6. ДІЛОВА МОДЕЛЬ.

Тут відображена ділова модель бази даних. Вона встановлює відповідність між певною функцією, яку виконує наш сервіс, та сутністю в базі даних, яка бере участь в запиті.



*(Рис. 2. Ділова модель БД)*

# 

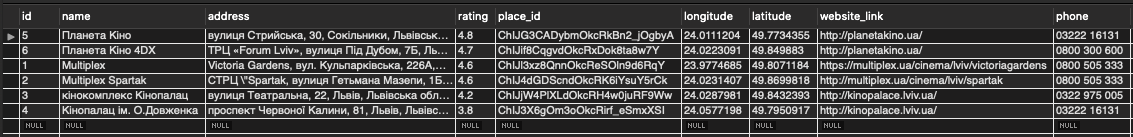
# 7. ЗАПИТИ ДО БД.

1. Отримаємо всі кінотеатри відсортовані за рейтингом:

select \*

from ekinoback\_cinema

order by rating desc;



1. Отримаємо всі фото, які належать до кінотеатру «Multiplex»:

select image.id as Id,

cinema.name as Cinema,

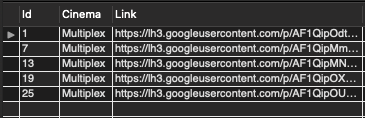
image.image\_link as Link

from ekinoback\_cinemaimage as image

inner join ekinoback\_cinema as cinema

on image.cinema\_id = cinema.id

where cinema.name = "Multiplex";



1. Визначимо, які фільми є найдешевшими, тобто середня ціна квитка є

найменшою:

select movie.name as movie,

avg(session.price) as avg\_price

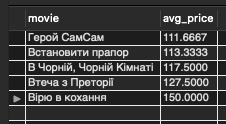
from ekinoback\_session as session

inner join ekinoback\_movie as movie

on session.movie\_id = movie.id

group by movie.id

order by avg\_price;



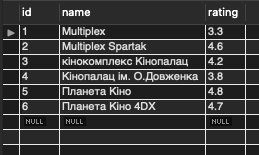
1. Змінюю рейтинг кінотеатру “Multiplex”:

update ekinoback\_cinema

set rating = 3.3

where name = "Multiplex";

select id, name, rating from ekinoback\_cinema;



1. Видалимо застарілі сеанси, тобто ті, дата яких менша за сьогоднішню:

delete from ekinoback\_session

where start\_time < curdate();

select \* from ekinoback\_session;



1. Визначимо акторів, що грали у філмах жанру “Комедія”:

select a.name as actor

from ekinoback\_movie\_genre as mg

inner join ekinoback\_genre as g

on g.id = mg.genre\_id

inner join ekinoback\_movie\_actors ma

on ma.movie\_id = mg.movie\_id

inner join ekinoback\_actor as a

on a.id = ma.actor\_id

where g.name = "Комедія";



1. Визначимо, по скільки сеансів має кожен кінотеатр.

select cinema.name as cinema,

count(session.id) as sessions

from ekinoback\_cinema as cinema

inner join ekinoback\_session as session

ON cinema.id = session.cinema\_id

group by cinema.id



# ВИСНОВОК.

Під час виконання цієї розрахункової роботи я навчився працювати з

реляційними базами даних, навчився проектувати схеми баз даних за допомогою ER-діаграм. Твкож навчився записувати дані в БД, і діставати їх простими запитами. Крім цього, була проведена робота над оптимізацією, що включала в себе створення індексів та застосування певних обмежень.

Сам проект побудований на основі Django Rest Framework, Python, який надає високорівневий доступ до більшості функцій бази даних, наприклад, створення, вибірки даних та запису даних. Тому вся робота над проектом була зроблена без використання синтаксису SQL. Для виконання розрахункової роботи я створив аналогічну модель бази даних в середовищі MySQL для відображення роботи створеної командою бази даних.

9. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ІНФОРМАЦІЇ.

1. Пасічник В.В., Резніченко В.А. Організація баз даних та знань - К.: Видавнича група BHV, 2006. — 384 с.: іл. — ISBN 966-552-156-Х.

2. Coronel C., Morris S. Database Systems: Design, Implementation, and Management. 12th ed. – Cengage Learning, 2017. – 818 p.

3. Connolly T.M., Begg C.E. Database Systems: A Practical Approach to Design, Implementation and Management: Global Edition. – 6th Edition. – Pearson Education, 2015. – 1440 p.

4. Kroenke D.M., Auer D.J. Database Processing: Fundamentals, Design, and Implementation. 14th ed. – Pearson Education Ltd., 2016. – 638 p.

5. <https://www.w3schools.com/sql/>

6. <https://www.tutorialspoint.com/sql/index.htm>

7. <http://www.sql-tutorial.ru/>

8. <https://www.codecademy.com/learn/learn-sql>

9. <https://www.mysqltutorial.org/>

10. <https://www.tutorialspoint.com/mysql/index.htm>