

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  
Марій Павло Ярославович

Балів	Дата

Викладач:  
Мельникова Наталя Іванівна

## Зміст

1. Тема проекту. ....	3
2. Вступ. ....	4
3. Логічна схема БД проекту. ....	5
4. Опис структури БД. ....	6
5. Фізична модель БД. ....	9
6. Ділова модель. ....	15
7. Запити до БД. ....	16
8. Висновки. ....	19
9. Список використаних джерел інформації. ....	20

## 1. Тема проекту.

Темою проекту моєї команди ЕК-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, відсортуювши 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](#).

## 2. Вступ.

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

- **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.

- **Вони дають можливість набутти 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 даних 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.

Щодо причин вибору тематики кіно можна виділити наступні переваги:

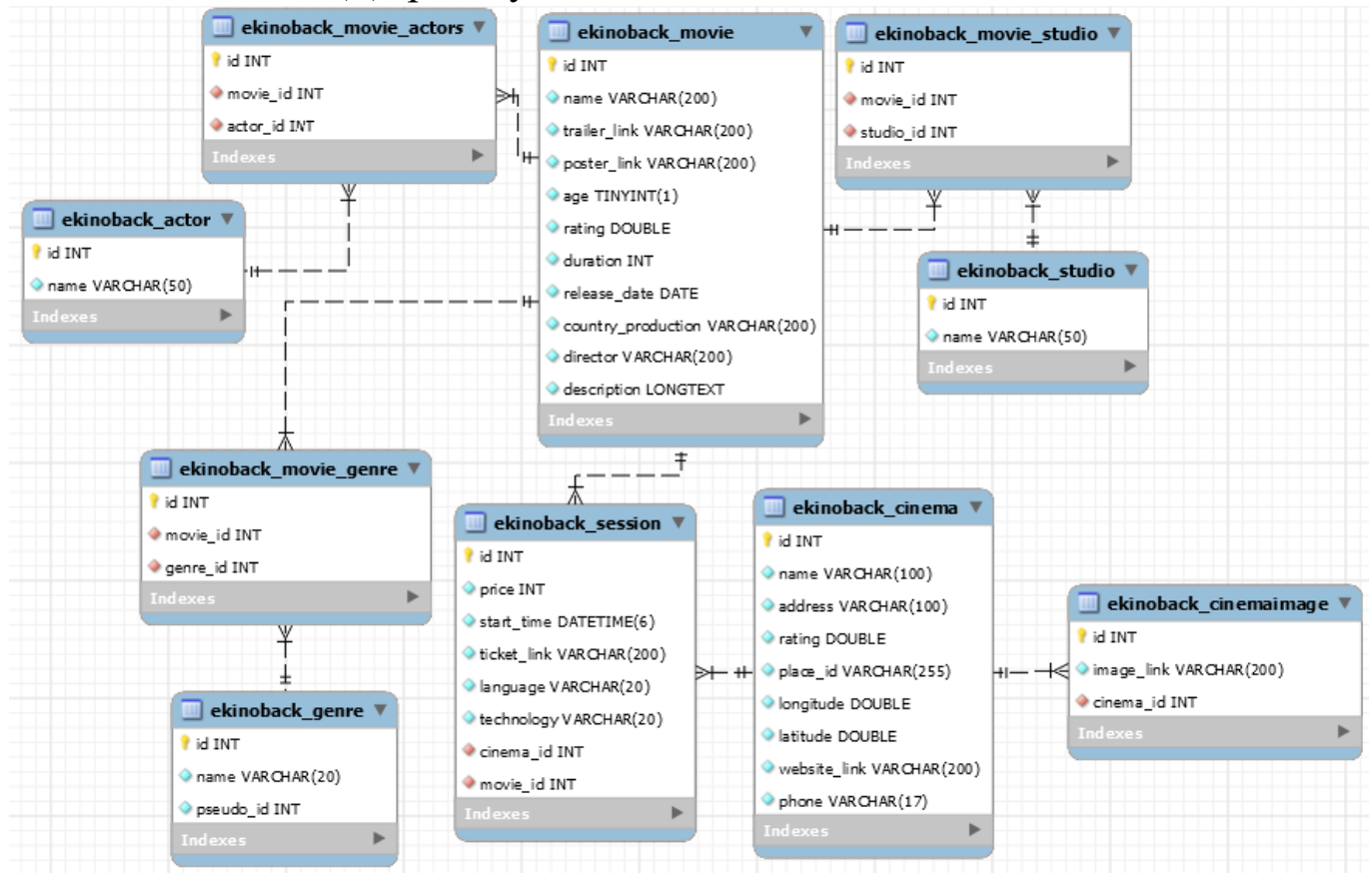
- **Активна аудиторія.** Обираючи кіно, на яке бажають сходити люди переважно звертаються до сервісів кінотеатрів чи інших веб-сервісів замість живого відвідування кінотеатрів чи інших засобів пошуку інформації (ЗМІ, газети, радіо чи живий обмін інформацією) і попит кінофільмів наразі значний.

- **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, було вирішено створити веб-сервіс, що регулярно збиратиме дані з кінотеатрів та подаватиме ці дані користувачу з максимально простим, але гнучким пошуком за різними фільтрами.

### 3. Логічна схема БД проекту.



#### 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 кінотеатру. 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_cinemainimage`
(
  `id` INT NOT NULL AUTO_INCREMENT,
  `image_link` VARCHAR(200) NOT NULL,
  `cinema_id` INT NOT NULL,
  PRIMARY KEY (`id`),
  INDEX
`ekinoback_cinemainimage_cinema_id_30f3e107_fk_ekinoback_cinema_
id` (`cinema_id` ASC),
  CONSTRAINT
`ekinoback_cinemainimage_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_i
d` (`actor_id` ASC) ,
  CONSTRAINT
`ekinoback_movie_actors_actor_id_51113f26_fk_ekinoback_actor_i
d`
  FOREIGN KEY (`actor_id`)
  REFERENCES `ekinobase`.`ekinoback_actor` (`id`),

```

```

CONSTRAINT
`ekinoback_movie_actors_movie_id_ebe45bce_fk_ekinoback_movie_i
d`
    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.

Таблиця Функція	Кінотеатр	Фільм	Сеанс	Актор	Жанр	Студія	Фото кінотеатру
Класифікація за жанром		*			*		
Класифікація за складом		*		*			
Розподіл за місцем	*	*	*				
Розподіл за часом		*	*				
Класифікація за студією		*				*	
Пошук анонсів		*	*				
Розподіл за прокатом	*	*	*				
Перегляд інформації про кінотеатр	*						*

## 7. Запити до БД.

1. Отримаємо всі фільми:

```
SELECT * FROM ekinoback_movie;
```

	id	name	trailer_link	poster_link	age	rating	duration	release_date	country_productio	director	description
▶	1	Вірю в ко...	https://w...	http://im...	0	6.5	115	2020-03-12	United States o...	Andrew ...	Коли Джеремі зуст...
	2	Встанови...	null	http://im...	0	7.2	15	2020-04-18	Iceland,Norway	Bobbie P...	Готуючись до вис...
	3	Втеча з П...	https://w...	http://im...	1	6.7	102	2020-03-06	Australia,Canad...	Francis A...	Francis Annan
	4	В Чорній,...	https://w...	http://im...	1	8.8	82	2020-03-26	Ukraine	Denis So...	Неподалік від стол...
	5	Герой Са...	https://w...	http://im...	0	8.2	77	2020-02-05	Belgium,France	Tanguy d...	Анімаційний фільм, ...
*	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

2. Отримаємо всі фільми, які належать до жанру «Документальний»:

```
SELECT G.name, M.name FROM ekinoback_genre G
INNER JOIN ekinoback_movie_genre GM
ON G.id = GM.genre_id
INNER JOIN ekinoback_movie M
ON GM.movie_id = M.id
WHERE G.name = "Документальний";
```

	name	name
▶	Документальний	Вірю в кохання
	Документальний	Встановити прапор

3. Підрахуємо, скільки сеансів є для фільмів, в яких знімався актор Anton Sokol:

```
SELECT COUNT(*) AS amount FROM ekinoback_session S
INNER JOIN ekinoback_movie M
ON S.movie_id = M.id
INNER JOIN ekinoback_movie_actors MA
ON M.id = MA.movie_id
INNER JOIN ekinoback_actor A
ON MA.actor_id = A.id
WHERE A.name = "Anton Sokol";
```

	amount
▶	6



4. Визначимо, які кінотеатри є найдешевшими, тобто середня ціна квитка є найменшою:

```
SELECT C.name, AVG(S.price) AS avg_price
FROM ekinoback_session S
INNER JOIN ekinoback_cinema C
    ON S.cinema_id = C.id
GROUP BY C.id
ORDER BY avg_price;
```

	name	avg_price
►	кінокомплекс Кінопалац	84.0000
	Multiplex Spartak	105.0000
	Multiplex	110.0000
	Кінопалац ім. О.Довженка	125.0000
	Планета Кіно 4DX	158.0000
	Планета Кіно	162.0000

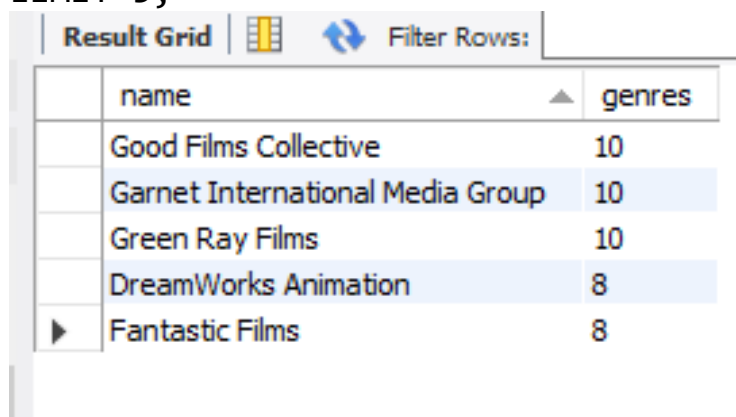
5. Визначимо, в яких кінотеатрах показуються фільми студії BBC Films, і виведемо результат по рейтингу кінотеатрів:

```
SELECT ST.name AS studio_name,
C.name AS cinema_name,
C.rating AS cinema_rating
FROM ekinoback_studio ST
INNER JOIN ekinoback_movie_studio MS
    ON ST.id = MS.studio_id
INNER JOIN ekinoback_movie M
    ON MS.movie_id = M.id
INNER JOIN ekinoback_session SS
    ON M.id = SS.movie_id
INNER JOIN ekinoback_cinema C
    ON SS.cinema_id = C.id
WHERE ST.name = "BBC Films"
ORDER BY C.rating DESC;
```

	studio_name	cinema_name	cinema_rating
►	BBC Films	Планета Кіно	4.8
	BBC Films	Планета Кіно 4DX	4.7
	BBC Films	Multiplex	4.6
	BBC Films	Multiplex Spartak	4.6
	BBC Films	кінокомплекс Кінопалац	4.2
	BBC Films	Кінопалац ім. О.Довженка	3.8

6. Визначимо 5 студій, які випускають фільми найширшого жанрового діапазону.

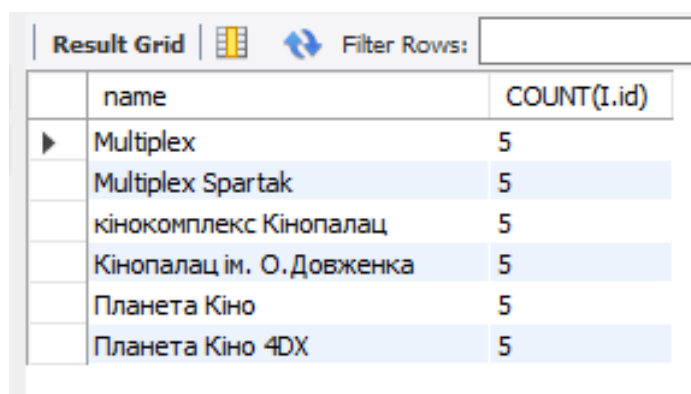
```
SELECT S.name, COUNT(DISTINCT(G.id)) AS genres
FROM ekinoback_studio S
INNER JOIN ekinoback_movie_studio MS
    ON S.id = MS.studio_id
INNER JOIN ekinoback_movie M
    ON MS.movie_id = M.id
INNER JOIN ekinoback_movie_genre MG
    ON M.id = MG.movie_id
INNER JOIN ekinoback_genre G
    ON MG.genre_id = G.id
GROUP BY S.id
ORDER BY genres DESC
LIMIT 5;
```



	name	genres
	Good Films Collective	10
	Garnet International Media Group	10
	Green Ray Films	10
	DreamWorks Animation	8
▶	Fantastic Films	8

7. Визначимо, по скільки фотографій має кожен кінотеатр.

```
SELECT C.name, COUNT(I.id) FROM ekinoback_cinema C
INNER JOIN ekinoback_cinemainimage I
    ON C.id = I.cinema_id
GROUP BY C.id
```



	name	COUNT(I.id)
▶	Multiplex	5
	Multiplex Spartak	5
	кінокомплекс Кінопалац	5
	Кінопалац ім. О.Довженка	5
	Планета Кіно	5
	Планета Кіно 4DX	5

## 8. Висновки.

Виконуючи цю розрахункову роботу, а також весь проект в цілому, я навчився працювати з реляційними базами даних. Спершу навчився проектувати схеми баз даних за допомогою ER-діаграм (Entity-Relation Diagram). Після цього – генерувати скрипт створення Баз даних за допомогою Forward Engineering в середовищі MySQL Workbench. В цей згенерований код слід вносити правки, створювати потрібні індекси, перевіряти правильність зовнішніх ключів. Після створення бази даних 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-X.
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>