**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МОЭВМ**

отчет

**по практической работе**

**по дисциплине «Основы технологий хранения данных»**

Тема: **Создание базы данных в MySQL**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6304 |  | Ковынев М.В. |
| Преподаватель: |  | Борисенко К.А. |

Санкт-Петербург

2019

# **Цель работы.**

# Приобретение навыков написания запросов на языке SQL на примерах собственной реализации базы данных в MySQL

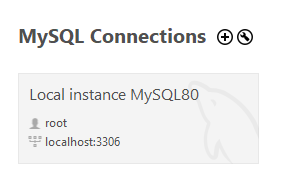
**Порядок выполнения работы.**

1. Выбрать предметную область.
2. Создать структуру (определение) каждой таблицы.
3. Заполнить таблицы тестовыми данными.
4. Написать 15 любых запросов к базе данных на языке SQL
5. Выполнить запросы и убедиться, что они корректно выполняют поставленные задачи.

# **Ход работы.**

# **Подключение к MySQL Server.**

* 1. Создадим MySQL Sever, работающий на localhost:3306



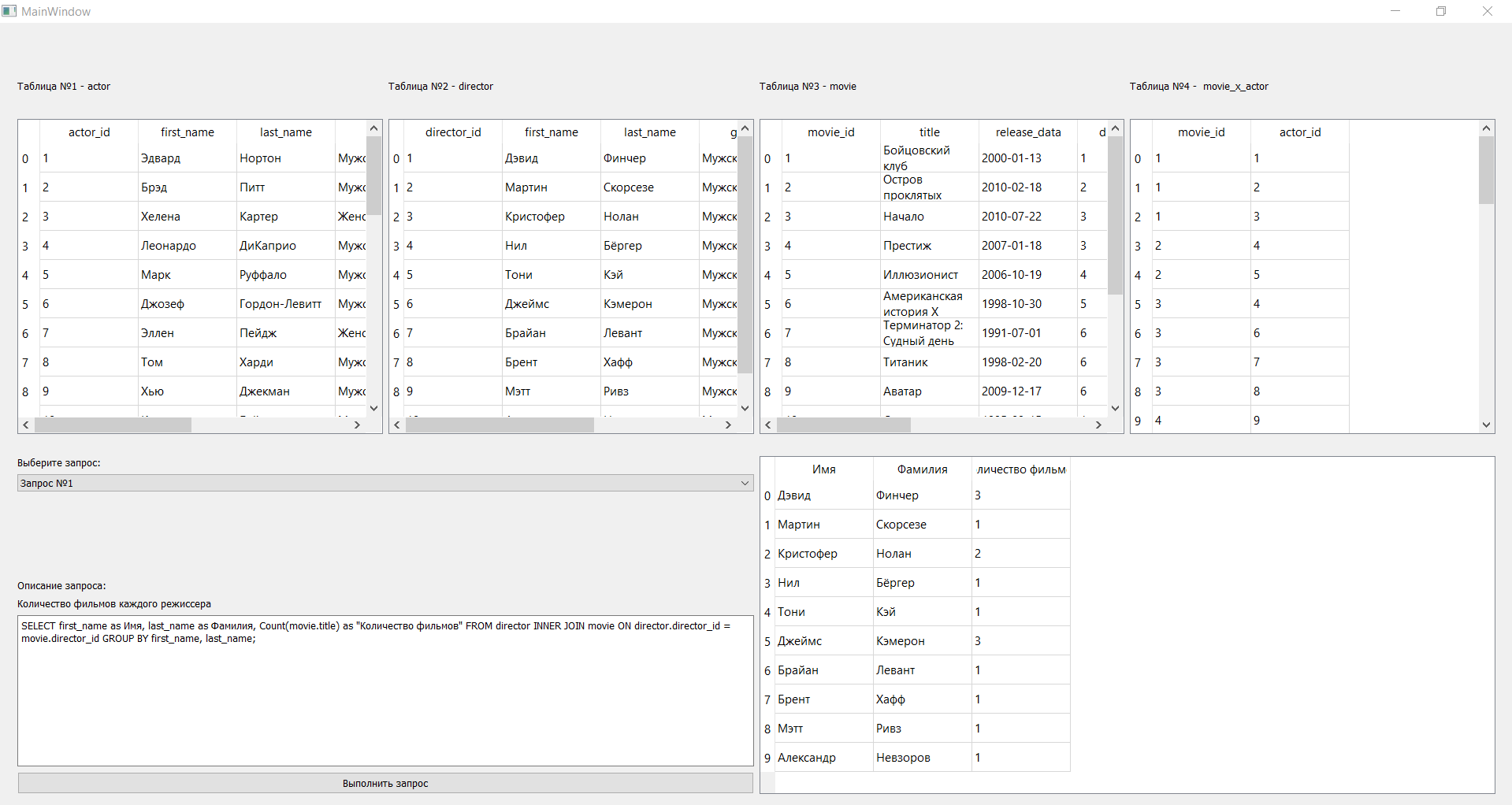
**Рисунок 1 — Сервер**

* 1. Реализуем класс на Python 3.7 для подключения к данному серверу.

class DataBase:  
 def \_\_init\_\_(self, name='sql', user='root',  
 password='1234', host='127.0.0.1'):  
 self.connection = pymysql.connect(host=host,  
 user=user,  
 password=password)

# **Создание приложения**

* 1. Для реализации графической составляющей приложения воспользуемся библиотекой PyQt.



**Рисунок 2 — Окно приложения**

# **Создание таблиц**

* 1. Для создания базы данных напишем запрос, в котором дополнительно укажем дополнительные опции: кодировка по умолчанию и способ сравнения строковых данных (без учета регистра).

def \_create\_tables(self):  
 self.execute(f"CREATE DATABASE IF NOT EXISTS `{self.name}` "  
 "CHARACTER SET utf8mb4 "  
 "COLLATE utf8mb4\_unicode\_ci")  
 self.execute("USE `sql`")

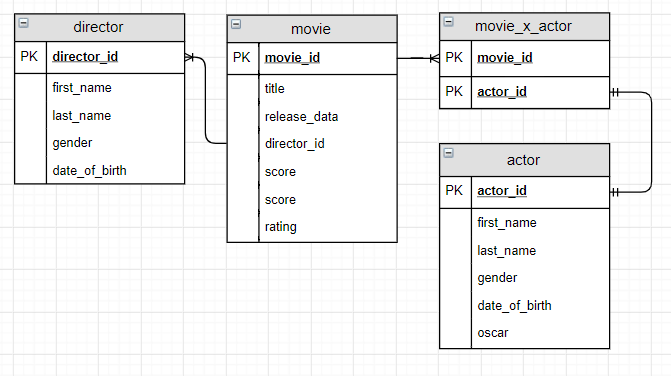
* 1. Напишем запросы на создание таблиц *actor, director, movie, movie\_x\_actor.*

tables = {  
 'actor': """  
CREATE TABLE actor(  
 actor\_id **int** NOT NULL PRIMARY KEY,  
 first\_name **VARCHAR** (50) NOT NULL,  
 last\_name **VARCHAR** (50) NOT NULL,  
 gender **VARCHAR** (10) NOT NULL,  
 date\_of\_birth **Date** NOT NULL,   
 oscar BOOLEAN NOT NULL DEFAULT 0  
);  
""",  
 'director': """  
CREATE TABLE director(  
 director\_id **int** NOT NULL PRIMARY KEY,  
 first\_name **VARCHAR** (50) NOT NULL,  
 last\_name **VARCHAR** (50) NOT NULL,  
 gender **VARCHAR** (10) NOT NULL,  
 date\_of\_birth **Date** NOT NULL  
);  
""",  
 'movie': """  
CREATE TABLE movie(  
 movie\_id **int** NOT NULL PRIMARY KEY,  
 title **varchar** (50) NOT NULL,  
 release\_data **Date** NOT NULL,  
 director\_id **INT** NOT NULL,  
 score **FLOAT** NOT NULL,  
 budget BIGINT NOT NULL,   
 rating **SMALLINT** NOT NULL  
);  
""",  
 'movie\_x\_actor': """  
CREATE TABLE movie\_x\_actor(  
 movie\_id **int** NOT NULL,  
 actor\_id **int** NOT NULL,  
 PRIMARY KEY (movie\_id, actor\_id)  
);  
"""  
}  
  
for name, sql in tables.items():  
 self.execute('DROP TABLE IF EXISTS {}'.format(name))  
 self.execute(sql)

* 1. Напишем запросы на заполнение таблиц

def \_insert\_values(self):  
 values = {  
 'actor':  
 """  
 INSERT INTO actor VALUES   
 (1,'Эдвард','Нортон','Мужской','1969-08-18',0),  
 (2,'Брэд','Питт','Мужской','1963-12-18',1),  
 (3,'Хелена','Картер','Женский','1966-05-26',0),  
 (4,'Леонардо','ДиКаприо','Мужской','1974-11-11',1),  
 (5,'Марк','Руффало','Мужской','1967-11-22',0),  
 (6,'Джозеф','Гордон-Левитт','Мужской','1981-02-17',0),  
 (7,'Эллен','Пейдж','Женский','1987-02-21',0),  
 (8,'Том','Харди','Мужской','1977-09-15',0),  
 (9,'Хью','Джекман','Мужской','1968-10-12',0),  
 (10,'Кристиан','Бэйл','Мужской','1974-01-30',1),  
 (11,'Майкл','Кейн','Мужской','1933-03-14',1),  
 (12,'Пол','Джаматти','Мужской','1967-06-06',0),  
 (13,'Джессика','Бил','Женский','1982-03-03',0),  
 (14,'Эдвард','Ферлонг','Мужской','1977-08-02',0),  
 (15,'Беверли','Д’Анджело','Женский','1951-11-15',0),  
 (16,'Арнольд','Шварценеггер','Мужской','1947-07-30',0),  
 (17,'Линда','Хэмилтон','Женский','1956-09-26',0),  
 (18,'Кейт','Уинслет','Женский','1975-10-05',1),  
 (19,'Сэм','Уортингтон','Мужской','1967-08-02',0),  
 (20,'Зои','Салдана','Женский','1978-06-19',0),  
 (21,'Морган','Фриман','Мужской','1937-06-01',1),  
 (22,'Гвинет','Пэлтроу','Женский','1972-09-27',1),  
 (23,'Джейк','Джилленхол','Мужской','1980-12-19',0),  
 (24,'Роберт','Дауни','Мужской','1965-04-04',0),  
 (25,'Чарльз','Гродин','Мужской','1935-04-21',0),  
 (26,'Бонни','Хант','Женский','1961-09-22',0),  
 (27,'Александр','Невский','Мужской','1971-07-17',0),  
 (28,'Лиззи','Каплан','Женский','1982-06-30',0),  
 (29,'Джессика','Лукас','Женский','1985-09-24',0),  
 (30,'Дмитрий','Нагиев','Мужской','1967-04-04',0);  
 """,  
 'director':  
 """  
 INSERT INTO director VALUES  
 (1, 'Дэвид', 'Финчер', 'Мужской', '1962-08-28'),  
 (2, 'Мартин', 'Скорсезе', 'Мужской', '1942-11-17'),  
 (3, 'Кристофер', 'Нолан', 'Мужской', '1970-07-30'),  
 (4, 'Нил', 'Бёргер', 'Мужской', '1964-01-01'),  
 (5, 'Тони', 'Кэй', 'Мужской', '1952-06-08'),  
 (6, 'Джеймс', 'Кэмерон', 'Мужской', '1954-08-16'),  
 (7, 'Брайан', 'Левант', 'Мужской', '1952-08-06'),  
 (8, 'Брент', 'Хафф', 'Мужской', '1961-03-11'),  
 (9, 'Мэтт', 'Ривз', 'Мужской', '1966-04-27'),  
 (10, 'Александр', 'Невзоров', 'Мужской', '1958-08-03');  
 """,  
 'movie':  
 """  
 INSERT INTO movie VALUES   
 (1, 'Бойцовский клуб', '2000-01-13', 1, 8.648, 63000000, 18),  
 (2, 'Остров проклятых', '2010-02-18', 2, 8.492, 80000000, 16),  
 (3, 'Начало', '2010-07-22', 3, 8.665, 160000000, 12),  
 (4, 'Престиж', '2007-01-18', 3, 8.524, 40000000, 12),  
 (5, 'Иллюзионист', '2006-10-19', 4, 8.015, 16500000, 12),  
 (6, 'Американская история X', '1998-10-30', 5, 8.297, 20000000, 16),  
 (7, 'Терминатор 2: Судный день', '1991-07-01', 6, 8.305, 102000000, 18),  
 (8, 'Титаник', '1998-02-20', 6, 8.369, 200000000, 12),  
 (9, 'Аватар', '2009-12-17', 6, 7.943, 237000000, 12),  
 (10, 'Семь', '1995-09-15', 1, 8.295, 33000000, 18),  
 (11, 'Зодиак', '2007-08-02', 1, 7.337, 65000000, 18),  
 (12, 'Бетховен', '1993-09-17', 7, 7.277, 40000000, 6),  
 (13, 'Форсаж да Винчи', '2008-04-03', 8, 1.317, 1000000, 16),  
 (14, 'Монстро', '2008-01-18', 9, 6.926, 25000000, 16),  
 (15, 'Чистилище', '1998-03-23', 10, 7.763, 1000000, 16);  
 """,  
 'movie\_x\_actor':  
 """  
 INSERT INTO movie\_x\_actor VALUES   
 (1, 1),  
 (1, 2),  
 (1, 3),  
 (2, 4),  
 (2, 5),  
 (3, 4),  
 (3, 6),  
 (3, 7),  
 (3, 8),  
 (4, 9),  
 (4, 10),  
 (4, 11),  
 (5, 1),  
 (5, 12),  
 (5, 13),  
 (6, 1),  
 (6, 14),  
 (6, 15),  
 (7, 14),  
 (7, 16),  
 (7, 17),  
 (8, 4),  
 (8, 18),  
 (9, 19),  
 (9, 20),  
 (10, 2),  
 (10, 21),  
 (10, 22),  
 (11, 5),  
 (11, 23),  
 (11, 24),  
 (12, 25),  
 (12, 26),  
 (13, 27),  
 (14, 28),  
 (14, 29),  
 (15, 30);  
 """  
 }  
  
 for name, sql in values.items():  
 self.execute(sql)

* 1. Реляционная модель представлена на рис. 3. Все таблицы находятся в 3НФ.



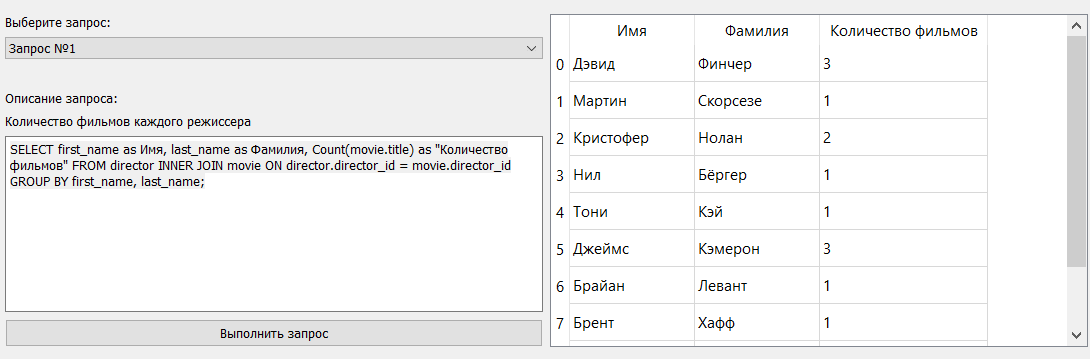
**Рисунок 3 — Реляционная модель базы данных**

1. **Запросы**
   1. Запрос №1

Описание: найти количество фильмов каждого режиссера

Запрос: SELECT first\_name as Имя, last\_name as Фамилия, Count(movie.title) as "Количество фильмов" FROM director INNER JOIN movie ON director.director\_id = movie.director\_id GROUP BY first\_name, last\_name;

Результат представлен на рис. 4



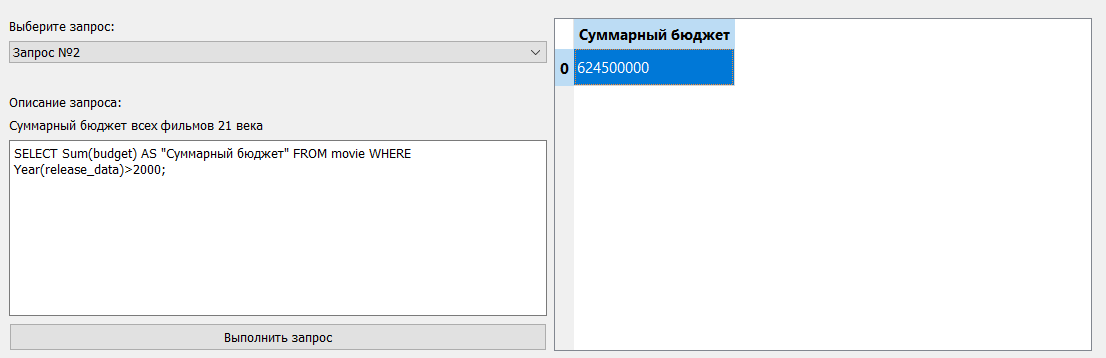
**Рисунок 4 — Запрос №1**

* 1. Запрос №2

Описание: суммарный бюджет всех фильмов 21 века

Запрос: SELECT Sum(budget) AS "Суммарный бюджет" FROM movie WHERE Year(release\_data)>2000;

Результат представлен на рис. 5



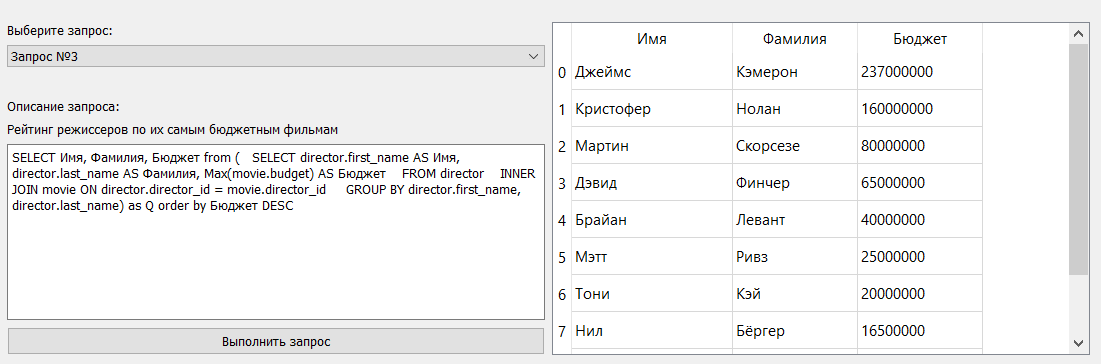
**Рисунок 5 — Запрос №2**

* 1. Запрос №3

Описание: рейтинг режиссеров по их самым бюджетным фильмам

Запрос: SELECT Имя, Фамилия, Бюджет from ( SELECT director.first\_name AS Имя, director.last\_name AS Фамилия, Max(movie.budget) AS Бюджет FROM director INNER JOIN movie ON director.director\_id = movie.director\_id GROUP BY director.first\_name, director.last\_name) as Q order by Бюджет DESC

Результат представлен на рис. 6



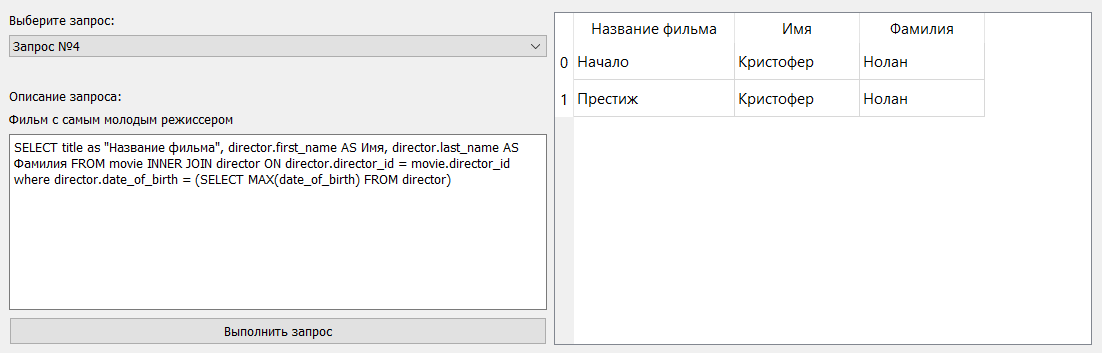
**Рисунок 6 — Запрос №3**

* 1. Запрос №4

Описание: фильм с самым молодым режиссером

Запрос: SELECT title as "Название фильма", director.first\_name AS Имя, director.last\_name AS Фамилия FROM movie INNER JOIN director ON director.director\_id = movie.director\_id where director.date\_of\_birth = (SELECT MAX(date\_of\_birth) FROM director)

Результат представлен на рис. 7



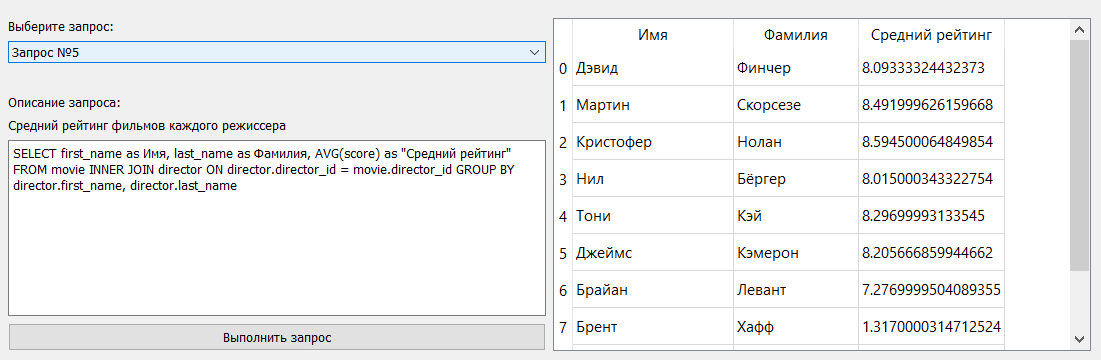
**Рисунок 7 — Запрос №4**

* 1. Запрос №5

Описание: средний рейтинг фильмов каждого режиссера

Запрос: SELECT first\_name as Имя, last\_name as Фамилия, AVG(score) as "Средний рейтинг" FROM movie INNER JOIN director ON director.director\_id = movie.director\_id GROUP BY director.first\_name, director.last\_name

Результат представлен на рис. 8



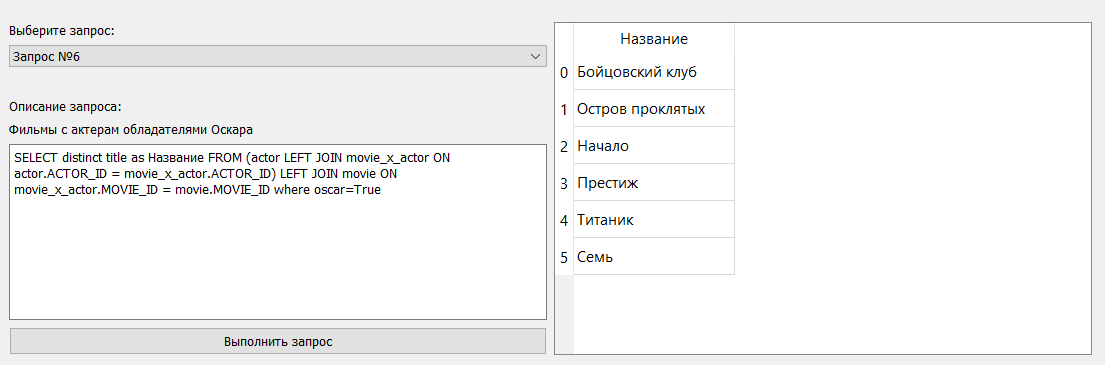
**Рисунок 8 — Запрос №5**

* 1. Запрос №6

Описание: фильмы с актерами обладателями Оскара

Запрос: SELECT distinct title as Название FROM (actor LEFT JOIN movie\_x\_actor ON actor.ACTOR\_ID = movie\_x\_actor.ACTOR\_ID) LEFT JOIN movie ON movie\_x\_actor.MOVIE\_ID = movie.MOVIE\_ID where oscar=True

Результат представлен на рис. 9



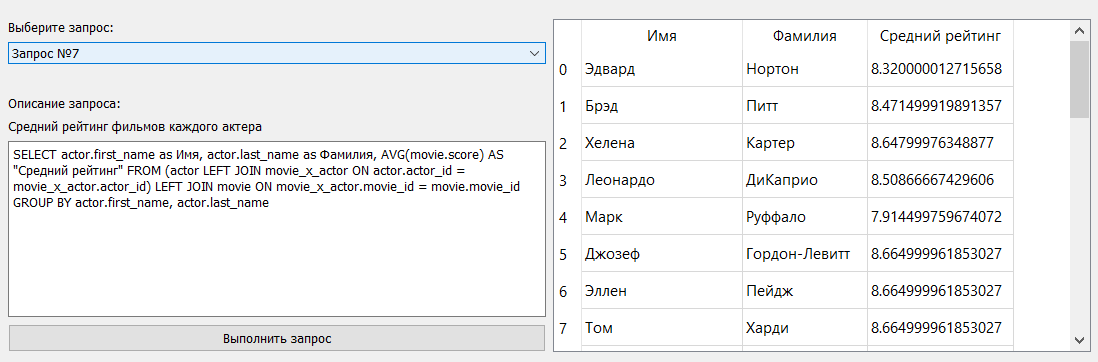
**Рисунок 9 — Запрос №6**

* 1. Запрос №7

Описание: средний рейтинг фильмов каждого актера

Запрос: SELECT actor.first\_name as Имя, actor.last\_name as Фамилия, AVG(movie.score) AS "Средний рейтинг" FROM (actor LEFT JOIN movie\_x\_actor ON actor.actor\_id = movie\_x\_actor.actor\_id) LEFT JOIN movie ON movie\_x\_actor.movie\_id = movie.movie\_id GROUP BY actor.first\_name, actor.last\_name

Результат представлен на рис. 10



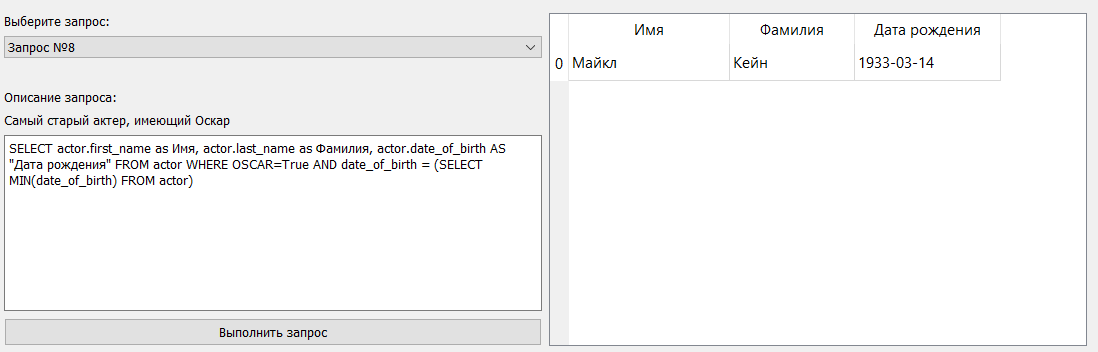
**Рисунок 10 — Запрос №7**

* 1. Запрос №8

Описание: самый старый актер, имеющий Оскар

Запрос: SELECT actor.first\_name as Имя, actor.last\_name as Фамилия, actor.date\_of\_birth AS "Дата рождения" FROM actor WHERE OSCAR=True AND date\_of\_birth = (SELECT MIN(date\_of\_birth) FROM actor)

Результат представлен на рис. 11



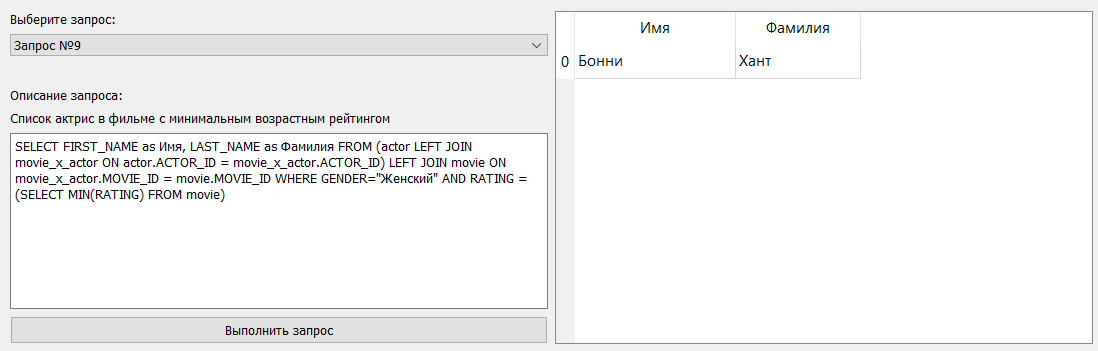
**Рисунок 11 — Запрос №8**

* 1. Запрос №9

Описание: список актрис в фильме с минимальным возрастным рейтингом

Запрос: SELECT FIRST\_NAME as Имя, LAST\_NAME as Фамилия FROM (actor LEFT JOIN movie\_x\_actor ON actor.ACTOR\_ID = movie\_x\_actor.ACTOR\_ID) LEFT JOIN movie ON movie\_x\_actor.MOVIE\_ID = movie.MOVIE\_ID WHERE GENDER="Женский" AND RATING = (SELECT MIN(RATING) FROM movie)

Результат представлен на рис. 12



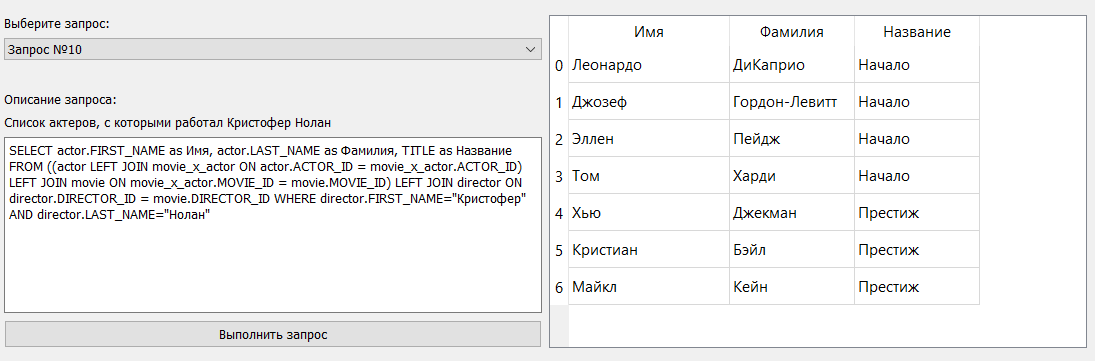
**Рисунок 12 — Запрос №9**

* 1. Запрос №10

Описание: список актеров, с которыми работал Кристофер Нолан

Запрос: SELECT actor.FIRST\_NAME as Имя, actor.LAST\_NAME as Фамилия, TITLE as Название FROM ((actor LEFT JOIN movie\_x\_actor ON actor.ACTOR\_ID = movie\_x\_actor.ACTOR\_ID) LEFT JOIN movie ON movie\_x\_actor.MOVIE\_ID = movie.MOVIE\_ID) LEFT JOIN director ON director.DIRECTOR\_ID = movie.DIRECTOR\_ID WHERE director.FIRST\_NAME="Кристофер" AND director.LAST\_NAME="Нолан"

Результат представлен на рис. 13



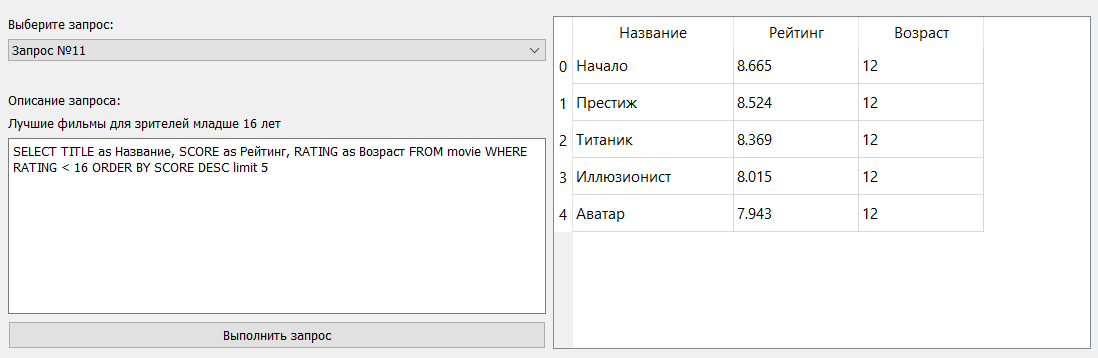
**Рисунок 13 — Запрос №10**

* 1. Запрос №11

Описание: лучшие фильмы для зрителей младше 16 лет

Запрос: SELECT TITLE as Название, SCORE as Рейтинг, RATING as Возраст FROM movie WHERE RATING < 16 ORDER BY SCORE DESC limit 5

Результат представлен на рис. 14



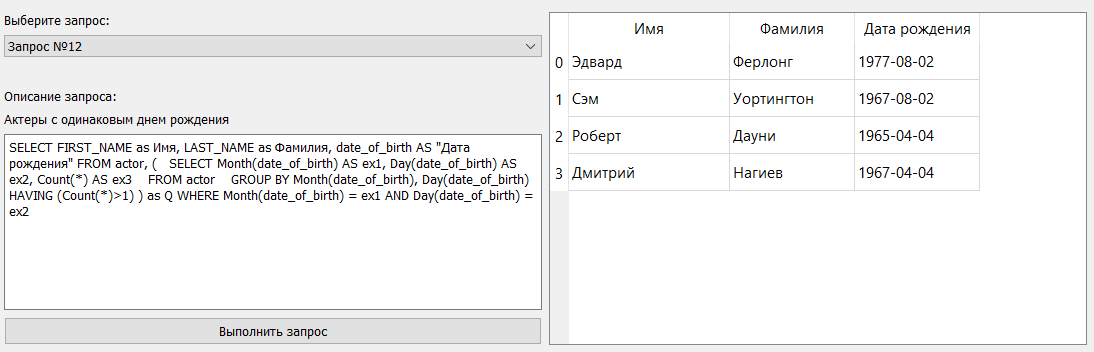
**Рисунок 14 — Запрос №11**

* 1. Запрос №12

Описание: актеры с одинаковым днем рождения

Запрос: SELECT FIRST\_NAME as Имя, LAST\_NAME as Фамилия, date\_of\_birth AS "Дата рождения" FROM actor, ( SELECT Month(date\_of\_birth) AS ex1, Day(date\_of\_birth) AS ex2, Count(\*) AS ex3 FROM actor GROUP BY Month(date\_of\_birth), Day(date\_of\_birth) HAVING (Count(\*)>1) ) as Q WHERE Month(date\_of\_birth) = ex1 AND Day(date\_of\_birth) = ex2

Результат представлен на рис. 15



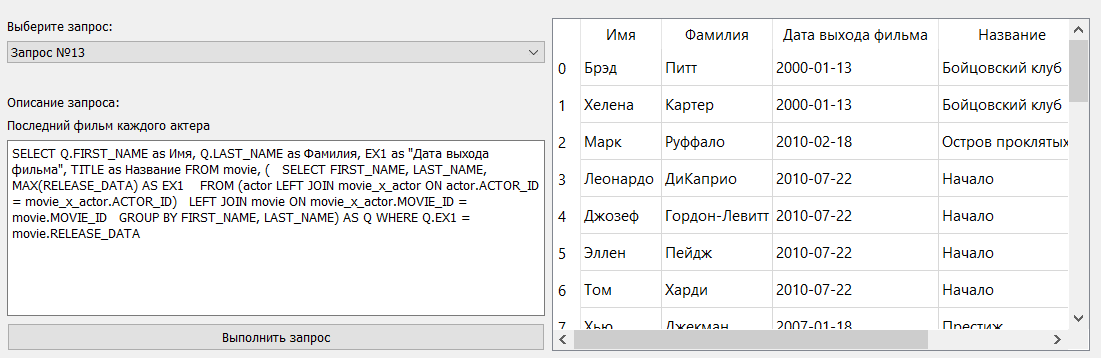
**Рисунок 15 — Запрос №12**

* 1. Запрос №13

Описание: последний фильм каждого актера

Запрос: SELECT Q.FIRST\_NAME as Имя, Q.LAST\_NAME as Фамилия, EX1 as "Дата выхода фильма", TITLE as Название FROM movie, ( SELECT FIRST\_NAME, LAST\_NAME, MAX(RELEASE\_DATA) AS EX1 FROM (actor LEFT JOIN movie\_x\_actor ON actor.ACTOR\_ID = movie\_x\_actor.ACTOR\_ID) LEFT JOIN movie ON movie\_x\_actor.MOVIE\_ID = movie.MOVIE\_ID GROUP BY FIRST\_NAME, LAST\_NAME) AS Q WHERE Q.EX1 = movie.RELEASE\_DATA

Результат представлен на рис. 16



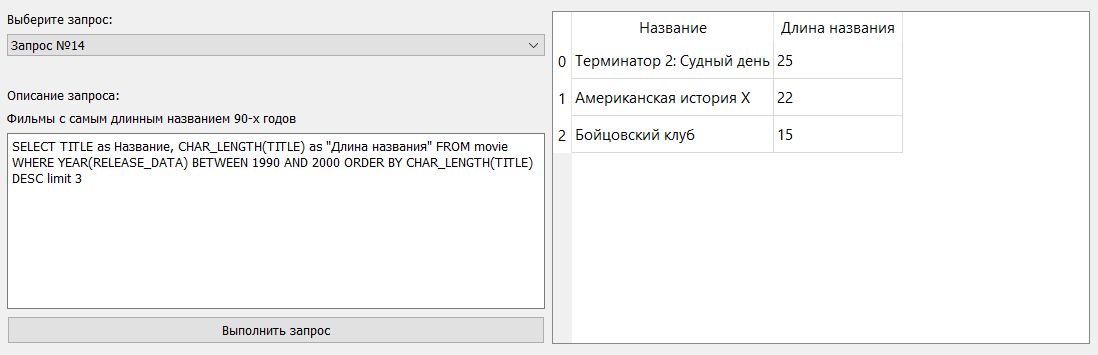
**Рисунок 16 — Запрос №13**

* 1. Запрос №14

Описание: фильмы с самым длинным названием 90-х годов

Запрос: SELECT TITLE as Название, CHAR\_LENGTH(TITLE) as "Длина названия" FROM movie WHERE YEAR(RELEASE\_DATA) BETWEEN 1990 AND 2000 ORDER BY CHAR\_LENGTH(TITLE) DESC limit 3

Результат представлен на рис. 17



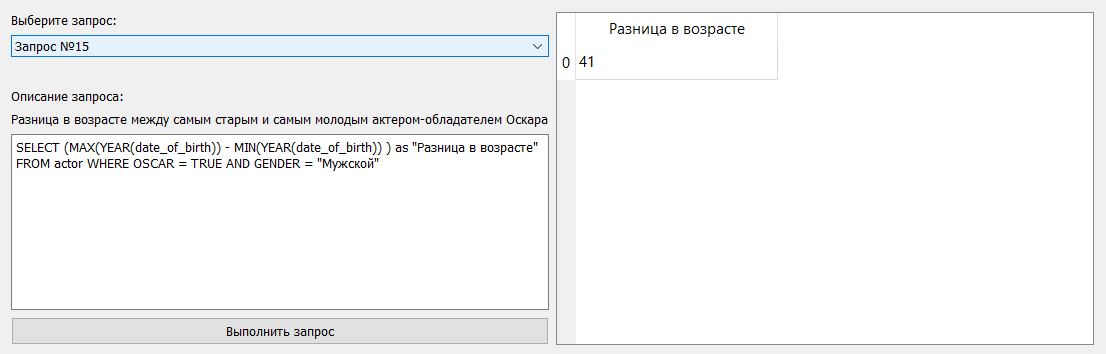
**Рисунок 17 — Запрос №14**

* 1. Запрос №15

Описание: разница в возрасте между самым старым и самым молодым актером-обладателем Оскара

Запрос: SELECT (MAX(YEAR(date\_of\_birth)) - MIN(YEAR(date\_of\_birth)) ) as "Разница в возрасте" FROM actor WHERE OSCAR = TRUE AND GENDER = "Мужской"

Результат представлен на рис. 18



**Рисунок 18 — Запрос №15**

**Вывод.**

В процессе выполнения данной практической работы разработана БД в MySQL, а также реализовано определенное количество запросов, в следствие чего получены навыки работы с языком структурированных запросов SQL в СУБД MySQL.