Лабораторная работа № 12. Создание БД заданной предметной области

Задание:

1. Выполнить нормализацию даталогической модели предметной области вашего варианта индивидального задания до 3НФ.

Какой результат я имею на данный момент

Table Movie {

id int [pk, increment]

description varchar(255)

rating int

age\_limit int

number\_of\_views int

duration int

release\_year int

country\_id int

website\_id int

publisher\_producer\_id int

director\_id int

genre\_id int

}

ref: Publisher\_Producer.id <> Movie.publisher\_producer\_id

ref : Director.id <> Movie.director\_id

ref : Website.id - Movie.website\_id

ref : Genre.id <> Movie.genre\_id

table Genre {

id int [pk, increment]

name varchar(255)

}

table Publisher\_Producer {

id int [pk, increment]

publisher\_id int

address\_id int

name varchar(255)

}

ref : Publisher\_Producer.publisher\_id > Publisher.id

ref : Address.id - Publisher\_Producer.address\_id

table Publisher {

id int [pk, increment]

name varchar(255)

website\_id int

}

ref : Publisher.website\_id - Website.id

table Director {

id int [pk, increment]

country\_id int UNIQUE

name varchar(255)

}

table Country {

id int [pk, increment]

name varchar(255)

}

ref : Director.country\_id > Country.id

ref : Movie.country\_id > Country.id

table Collection {

id int [pk, increment]

name varchar(255)

description text

}

table MovieCollection {

collection\_id int

movie\_id int

}

ref : Collection.id < MovieCollection.collection\_id

ref : Movie.id < MovieCollection.movie\_id

table Website {

id int [pk, increment]

name varchar(255)

url text

}

table Address {

id int [pk, increment]

country\_id int

street varchar(255)

house int

placement int

}

ref : Country.id < Address.country\_id

table Rentaler {

id int [pk, increment]

name varchar(255)

website\_id int

}

ref : Website.id - Rentaler.website\_id

table Rentaler\_Concrete {

id int [pk, increment]

rentaler\_id int

address\_id int

phone\_number varchar(255)

}

ref : Rentaler\_Concrete.rentaler\_id > Rentaler.id

ref : Rentaler\_Concrete.address\_id > Address.id

table User {

id int [pk, increment]

age int

name varchar(255)

}

table UserFavoriteMovies {

movie\_id int

user\_id int

}

ref : User.id > UserFavoriteMovies.user\_id

ref : Movie.id > UserFavoriteMovies.movie\_id

table UserFavoriteGenre {

genre\_id int

user\_id int

}

ref : User.id > UserFavoriteGenre.user\_id

ref : Genre.id > UserFavoriteGenre.genre\_id

table Carrier { // носитель

id int [pk, increment]

size int

name varchar(255)

type\_id int

}

ref : Carrier.type\_id > Type.id

table Type {

id int [pk, increment]

name varchar(255)

}

table Recording\_Format {

id int [pk, increment]

name varchar(255)

}

table RentedCarriers {

id int [pk, increment]

start\_rent time

end\_rent time

carrier\_id int

rentaler\_concrete\_id int

}

ref : Carrier.id - RentedCarriers.carrier\_id

ref : Rentaler\_Concrete.id < RentedCarriers.rentaler\_concrete\_id

table MovieCarrier {

movie\_id int

carrier\_id int

recording\_format\_id int

}

// один фильм может быть как копия на разных носителях

// как и один носитель может хранить много фильмов

ref : Carrier.id < MovieCarrier.carrier\_id

ref : Movie.id < MovieCarrier.movie\_id

ref : Recording\_Format.id < MovieCarrier.recording\_format\_id

проверим на НФ-ы

1НФ)

Она требует

1. Атомарность данных – все значения в столбцах должны быть неделимыми (в моем результате она выполняется — любое значение в моих столбцах является неделимым)

2. Уникальность строк – каждая строка должна иметь уникальный идентификатор (первичный ключ). (да — во всех таблицах он есть (у кого-то это ID, у кого-то он составной))

3. Отсутствие повторяющихся групп – не должно быть столбцов с одинаковыми данными. (таких у меня не наблюдается)

2НФ)

Она требует

1. Должна выполняться 1НФ (как показано выше — она выполняется).

2. Отсутствие частичной зависимости – все неключевые атрибуты должны зависеть от всего первичного ключа, а не от его части (актуально для составных ключей). (для тех, у кого PK простой(просто id) – все выполняется. для тех, у кого PK составной – все также выполняется)

3НФ)

Она требует

1. Должна выполняться 2НФ (выполняется)

2. Отсутствие транзитивных зависимостей – неключевые атрибуты не должны зависеть от других неключевых атрибутов.

На втором пункте остановимся поподробнее:

Какие потенциально могут быть проблемы и как я их вижу:

table Address {

id int [pk, increment] // Ключевое

country\_id int [ref: > Country.id] // Неключевое

street varchar(255) // Неключевое

house int // Неключевое

placement int // Неключевое

}

Теоретически, Улица может зависеть от Страны, но – не совсем так, так как в мире есть примеры того, как улицы называются одинаково (улица Карла Маркса, к примеру)

table Publisher\_Producer {

id int [pk, increment] // Ключевое

publisher\_id int [ref: > Publisher.id] // Неключевое

address\_id int [ref: > Address.id] // Неключевое

name varchar(255) // Неключевое

}

Я предполагал, что поле name будет как-то сильно отличаться, но я посчитал, что в этом нет нужды и принял решение удалить его

Далее я все таки еще раз пересмотрел свои сущности и решил добавить следующее:

- было учтено, что Адрес зависит от Города. Кроме это – не обязательно, что у нас Город

table LocalityType {

id int [pk, increment] // Ключевое (первичный ключ)

name varchar(50) // Неключевое

code varchar(20) // Неключевое

}

table Locality {

id int [pk, increment] // Ключевое (первичный ключ)

country\_id int [ref: > Country.id] // Неключевое (внешний ключ)

locality\_type\_id int [ref: > LocalityType.id] // Неключевое (внешний ключ)

name varchar(255) // Неключевое

}

table Address {

id int [pk, increment] // Ключевое (первичный ключ)

locality\_id int [ref: > Locality.id] // Неключевое (внешний ключ)

street varchar(255) // Неключевое

house int // Неключевое

}

- для своей задачи я посчитал избыточным хранить столько информации о Издателе (Publisher) (делить издателя на его филиалы) и перенес все в одну сущность Издатель

- добавлена таблица ИсторияПросмотра. Так как БД рассчитана для Домашней медиатеки, а т.е. до 50 пользователей, то хранение в обычной таблице без дополнительных оптимизаций является неплохим решением.

в результате мы имеем

// генерация запроса https://dbdiagram.io/d

// Вопросы:

// - стоит ли уменьшить сущность Фильм?

Table Movie {

id int [pk, increment] // Ключевое (первичный ключ)

description varchar(255) // Неключевое

rating int // Неключевое

age\_limit int // Неключевое

number\_of\_views int // Неключевое

duration int // Неключевое

release\_year int // Неключевое

country\_id int [ref: > Country.id] // Неключевое (внешний ключ)

website\_id int [ref: > Website.id] // Неключевое (внешний ключ)

publisher\_id int [ref: > Publisher.id] // Неключевое (внешний ключ)

director\_id int [ref: > Director.id] // Неключевое (внешний ключ)

genre\_id int [ref: > Genre.id] // Неключевое (внешний ключ)

Note: '''

Таблица сущности Фильм.

Хранит информацию о фильмах:

- количество просмотров

- описание

- рейтинг

- страна производства

- и другие метаданные

'''

}

Table Genre {

id int [pk, increment] // Ключевое (первичный ключ)

name varchar(255) // Неключевое

Note: "Таблица сущности Жанр, хранит название Жанра"

}

Table Publisher {

id int [pk, increment] // Ключевое (первичный ключ)

name varchar(255) // Неключевое

website\_id int [ref: > Website.id] // Неключевое (внешний ключ)

Note: '''

Сущность Издатель (Дисней, Пиксар и т.п.).

'''

}

Table Director {

id int [pk, increment] // Ключевое (первичный ключ)

country\_id int [ref: > Country.id] // Неключевое (внешний ключ)

name varchar(255) // Неключевое

description text // Неключевое

Note: '''

Таблица сущности Режиссер:

хранит в себе имя, страну происхождения

и описание режиссера

'''

}

Table Country {

id int [pk, increment] // Ключевое (первичный ключ)

name varchar(255) // Неключевое

Note: "Таблица сущности Страна: хранит название страны"

}

Table Collection {

id int [pk, increment] // Ключевое (первичный ключ)

name varchar(255) // Неключевое

description text // Неключевое

Note: "Таблица сущности Коллекция: хранит название и описание коллекции"

}

Table MovieCollection {

collection\_id int [ref: > Collection.id] // Ключевое (часть составного PK)

movie\_id int [ref: > Movie.id] // Ключевое (часть составного PK)

Note: "Таблица, описывающая содержимое коллекций"

}

Table Website {

id int [pk, increment] // Ключевое (первичный ключ)

name varchar(255) // Неключевое

url text // Неключевое

Note: "Таблица, описывающая Веб-сайт"

}

Table LocalityType {

id int [pk, increment] // Ключевое (первичный ключ)

name varchar(50) // Неключевое

code varchar(20) // Неключевое

Note: "Хранит виды населенных мест (город, деревня, поселок и т.п.)"

}

Table Locality {

id int [pk, increment] // Ключевое (первичный ключ)

country\_id int [ref: > Country.id] // Неключевое (внешний ключ)

locality\_type\_id int [ref: > LocalityType.id] // Неключевое (внешний ключ)

name varchar(255) // Неключевое

Note: "Связывает Страну и населенный пункт (с указанием типа)"

}

Table Address {

id int [pk, increment] // Ключевое (первичный ключ)

locality\_id int [ref: > Locality.id] // Неключевое (внешний ключ)

street varchar(255) // Неключевое

house int // Неключевое

Note: "Хранит адрес объекта"

}

Table Rentaler {

id int [pk, increment] // Ключевое (первичный ключ)

name varchar(255) // Неключевое

website\_id int [ref: > Website.id] // Неключевое (внешний ключ)

Note: '''

Компания, предоставляющая носители с фильмами в аренду.

Может иметь несколько точек по миру

'''

}

Table Rentaler\_Concrete {

id int [pk, increment] // Ключевое (первичный ключ)

rentaler\_id int [ref: > Rentaler.id] // Неключевое (внешний ключ)

address\_id int [ref: > Address.id] // Неключевое (внешний ключ)

phone\_number varchar(255) // Неключевое

Note: "Конкретная точка аренды с адресом и контактами"

}

Table User {

id int [pk, increment] // Ключевое (первичный ключ)

age int // Неключевое

name varchar(255) // Неключевое

Note: "Пользователь системы: имя и возраст"

}

Table UserFavoriteMovies {

user\_id int [ref: > User.id] // Ключевое (часть составного PK)

movie\_id int [ref: > Movie.id] // Ключевое (часть составного PK)

Note: "Связь пользователя и понравившихся фильмов"

}

Table UserFavoriteGenre {

genre\_id int [ref: > Genre.id] // Ключевое (часть составного PK)

user\_id int [ref: > User.id] // Ключевое (часть составного PK)

Note: "Связь пользователя и предпочитаемых жанров"

}

Table Carrier {

id int [pk, increment] // Ключевое (первичный ключ)

size int // Неключевое

name varchar(255) // Неключевое

type\_id int [ref: > Type.id] // Неключевое (внешний ключ)

Note: "Физический носитель фильма (диск, кассета и т.д.)"

}

Table Type {

id int [pk, increment] // Ключевое (первичный ключ)

name varchar(255) // Неключевое

Note: "Типы носителей (DVD, Blu-ray, VHS и т.д.)"

}

Table RentedCarriers {

id int [pk, increment] // Ключевое (первичный ключ)

start\_rent time // Неключевое

end\_rent time // Неключевое

carrier\_id int [ref: > Carrier.id] // Неключевое (внешний ключ)

rentaler\_concrete\_id int [ref: > Rentaler\_Concrete.id] // Неключевое (внешний ключ)

Note: "История аренды носителей"

}

Table Recording\_Format {

id int [pk, increment] // Ключевое (первичный ключ)

name varchar(255) // Неключевое

Note: "Форматы записи (PAL, NTSC и т.д.)"

}

Table MovieCarrier {

movie\_id int [ref: > Movie.id] // Ключевое (часть составного PK)

carrier\_id int [ref: > Carrier.id] // Ключевое (часть составного PK)

recording\_format\_id int [ref: > Recording\_Format.id] // Ключевое (часть составного PK)

Note: "Связь между фильмом и его носителями"

}

Table WatchHistory {

id int [pk, increment] // Ключевое (первичный ключ)

movie\_id int [ref: > Movie.id] // Неключевое (внешний ключ)

user\_id int [ref: > User.id, null] // Неключевое (внешний ключ, может быть null)

watched\_at datetime [default: `now()`] // Неключевое

Note: '''

хранит информацию о просмотре фильма

'''

}

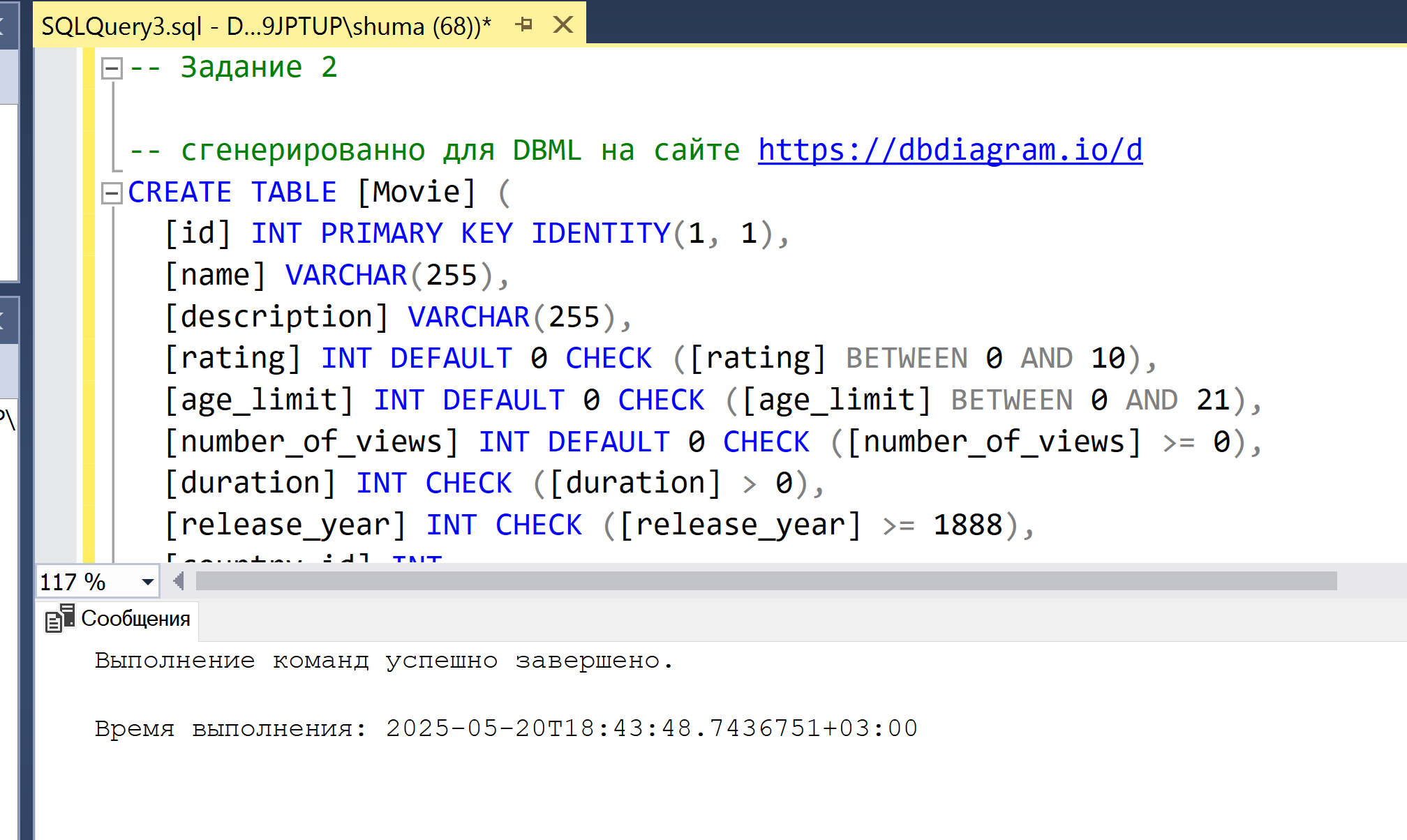
2. Построить структуру БД для предметной области вашего варианта индивидуального задания в выбранной СУБД (создать таблицы, ключи, ограничения в соответствии с разработанной в ЛР-6 даталогической моделью БД). Для этого использовать любой из доступных вариантов:

1. создать структуру БД с использованием экспорта из используемого в ЛР-3 CASE-средства для разработки моделей БД.

2. создать структуру БД с использованием интерфейсных элементов клиентского приложения для работы с выбранной СУБД (например, MS SQL Management Studio. 3. создать структуру БД с использованием операторов DDL и клиентского приложения для работы с выбранной СУБД.

для описания структуры БД использовался язык разметки DBML, который можно конвертировать в необходимый нам запрос.

запрос:



-- Задание 2

-- сгенерированно для DBML на сайте https://dbdiagram.io/d

CREATE TABLE [Movie] (

[id] INT PRIMARY KEY IDENTITY(1, 1),

[name] VARCHAR(255),

[description] VARCHAR(255),

[rating] INT DEFAULT 0 CHECK ([rating] BETWEEN 0 AND 10),

[age\_limit] INT DEFAULT 0 CHECK ([age\_limit] BETWEEN 0 AND 21),

[number\_of\_views] INT DEFAULT 0 CHECK ([number\_of\_views] >= 0),

[duration] INT CHECK ([duration] > 0),

[release\_year] INT CHECK ([release\_year] >= 1888),

[country\_id] INT,

[website\_id] INT,

[publisher\_id] INT,

[director\_id] INT,

[genre\_id] INT

)

GO

CREATE TABLE [Genre] (

[id] INT PRIMARY KEY IDENTITY(1, 1),

[name] VARCHAR(255) NOT NULL

)

GO

CREATE TABLE [Publisher] (

[id] INT PRIMARY KEY IDENTITY(1, 1),

[name] VARCHAR(255) NOT NULL,

[website\_id] INT

)

GO

CREATE TABLE [Director] (

[id] INT PRIMARY KEY IDENTITY(1, 1),

[country\_id] INT,

[name] VARCHAR(255) NOT NULL,

[description] TEXT

)

GO

CREATE TABLE [Country] (

[id] INT PRIMARY KEY IDENTITY(1, 1),

[name] VARCHAR(255) NOT NULL

)

GO

CREATE TABLE [Collection] (

[id] INT PRIMARY KEY IDENTITY(1, 1),

[name] VARCHAR(255) NOT NULL,

[description] TEXT

)

GO

CREATE TABLE [MovieCollection] (

[collection\_id] INT,

[movie\_id] INT,

PRIMARY KEY ([collection\_id], [movie\_id])

)

GO

CREATE TABLE [Website] (

[id] INT PRIMARY KEY IDENTITY(1, 1),

[name] VARCHAR(255) NOT NULL,

[url] TEXT

)

GO

CREATE TABLE [LocalityType] (

[id] INT PRIMARY KEY IDENTITY(1, 1),

[name] VARCHAR(50) NOT NULL,

[code] VARCHAR(20) NOT NULL

)

GO

CREATE TABLE [Locality] (

[id] INT PRIMARY KEY IDENTITY(1, 1),

[country\_id] INT,

[locality\_type\_id] INT,

[name] VARCHAR(255) NOT NULL

)

GO

CREATE TABLE [Address] (

[id] INT PRIMARY KEY IDENTITY(1, 1),

[locality\_id] INT,

[street] VARCHAR(255) NOT NULL,

[house] INT CHECK ([house] > 0)

)

GO

CREATE TABLE [Rentaler] (

[id] INT PRIMARY KEY IDENTITY(1, 1),

[name] VARCHAR(255) NOT NULL,

[website\_id] INT

)

GO

CREATE TABLE [Rentaler\_Concrete] (

[id] INT PRIMARY KEY IDENTITY(1, 1),

[rentaler\_id] INT,

[address\_id] INT,

[phone\_number] VARCHAR(255) NOT NULL

)

GO

CREATE TABLE [User] (

[id] INT PRIMARY KEY IDENTITY(1, 1),

[age] INT CHECK ([age] >= 0 AND [age] <= 120),

[name] VARCHAR(255) NOT NULL

)

GO

CREATE TABLE [UserFavoriteMovies] (

[user\_id] INT,

[movie\_id] INT,

PRIMARY KEY ([user\_id], [movie\_id])

)

GO

CREATE TABLE [UserFavoriteGenre] (

[genre\_id] INT,

[user\_id] INT,

PRIMARY KEY ([genre\_id], [user\_id])

)

GO

CREATE TABLE [Carrier] (

[id] INT PRIMARY KEY IDENTITY(1, 1),

[size] INT CHECK ([size] >= 0), -- 0 для облачных сервисов

[name] VARCHAR(255) NOT NULL,

[type\_id] INT

)

GO

CREATE TABLE [Type] (

[id] INT PRIMARY KEY IDENTITY(1, 1),

[name] VARCHAR(255) NOT NULL

)

GO

CREATE TABLE [RentedCarriers] (

[id] INT PRIMARY KEY IDENTITY(1, 1),

[start\_rent] DATETIME NOT NULL,

[end\_rent] DATETIME NOT NULL,

[carrier\_id] INT,

[rentaler\_concrete\_id] INT

)

GO

CREATE TABLE [Recording\_Format] (

[id] INT PRIMARY KEY IDENTITY(1, 1),

[name] VARCHAR(255) NOT NULL

)

GO

CREATE TABLE [MovieCarrier] (

[movie\_id] INT,

[carrier\_id] INT,

[recording\_format\_id] INT,

PRIMARY KEY ([movie\_id], [carrier\_id], [recording\_format\_id])

)

GO

CREATE TABLE [WatchHistory] (

[id] INT PRIMARY KEY IDENTITY(1, 1),

[movie\_id] INT,

[user\_id] INT NULL,

[watched\_at] DATETIME DEFAULT GETDATE()

)

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'Таблица сущности Фильм.

Хранит информацию о фильмах:

- количество просмотров

- описание

- рейтинг

- страна производства

- и другие метаданные

',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'Movie';

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'Таблица сущности Жанр, хранит название Жанра',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'Genre';

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'Сущность Издатель (Дисней, Пиксар и т.п.).

',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'Publisher';

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'Таблица сущности Режиссер:

хранит в себе имя, страну происхождения

и описание режиссера

',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'Director';

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'Таблица сущности Страна: хранит название страны',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'Country';

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'Таблица сущности Коллекция: хранит название и описание коллекции',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'Collection';

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'Таблица, описывающая содержимое коллекций',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'MovieCollection';

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'Таблица, описывающая Веб-сайт',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'Website';

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'Хранит виды населенных мест (город, деревня, поселок и т.п.)',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'LocalityType';

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'Связывает Страну и населенный пункт (с указанием типа)',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'Locality';

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'Хранит адрес объекта',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'Address';

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'Компания, предоставляющая носители с фильмами в аренду.

Может иметь несколько точек по миру

',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'Rentaler';

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'Конкретная точка аренды с адресом и контактами',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'Rentaler\_Concrete';

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'Пользователь системы: имя и возраст',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'User';

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'Связь пользователя и понравившихся фильмов',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'UserFavoriteMovies';

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'Связь пользователя и предпочитаемых жанров',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'UserFavoriteGenre';

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'Физический носитель фильма (диск, кассета и т.д.)',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'Carrier';

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'Типы носителей (DVD, Blu-ray, VHS и т.д.)',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'Type';

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'История аренды носителей',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'RentedCarriers';

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'Форматы записи (PAL, NTSC и т.д.)',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'Recording\_Format';

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'Связь между фильмом и его носителями',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'MovieCarrier';

GO

EXEC sp\_addextendedproperty

@name = N'Table\_Description',

@value = 'хранит информацию о просмотре фильма

',

@level0type = N'Schema', @level0name = 'dbo',

@level1type = N'Table', @level1name = 'WatchHistory';

GO

ALTER TABLE [Movie] ADD FOREIGN KEY ([country\_id]) REFERENCES [Country] ([id])

GO

ALTER TABLE [Movie] ADD FOREIGN KEY ([website\_id]) REFERENCES [Website] ([id])

GO

ALTER TABLE [Movie] ADD FOREIGN KEY ([publisher\_id]) REFERENCES [Publisher] ([id])

GO

ALTER TABLE [Movie] ADD FOREIGN KEY ([director\_id]) REFERENCES [Director] ([id])

GO

ALTER TABLE [Movie] ADD FOREIGN KEY ([genre\_id]) REFERENCES [Genre] ([id])

GO

ALTER TABLE [Publisher] ADD FOREIGN KEY ([website\_id]) REFERENCES [Website] ([id])

GO

ALTER TABLE [Director] ADD FOREIGN KEY ([country\_id]) REFERENCES [Country] ([id])

GO

ALTER TABLE [MovieCollection] ADD FOREIGN KEY ([collection\_id]) REFERENCES [Collection] ([id])

GO

ALTER TABLE [MovieCollection] ADD FOREIGN KEY ([movie\_id]) REFERENCES [Movie] ([id])

GO

ALTER TABLE [Locality] ADD FOREIGN KEY ([country\_id]) REFERENCES [Country] ([id])

GO

ALTER TABLE [Locality] ADD FOREIGN KEY ([locality\_type\_id]) REFERENCES [LocalityType] ([id])

GO

ALTER TABLE [Address] ADD FOREIGN KEY ([locality\_id]) REFERENCES [Locality] ([id])

GO

ALTER TABLE [Rentaler] ADD FOREIGN KEY ([website\_id]) REFERENCES [Website] ([id])

GO

ALTER TABLE [Rentaler\_Concrete] ADD FOREIGN KEY ([rentaler\_id]) REFERENCES [Rentaler] ([id])

GO

ALTER TABLE [Rentaler\_Concrete] ADD FOREIGN KEY ([address\_id]) REFERENCES [Address] ([id])

GO

ALTER TABLE [UserFavoriteMovies] ADD FOREIGN KEY ([user\_id]) REFERENCES [User] ([id])

GO

ALTER TABLE [UserFavoriteMovies] ADD FOREIGN KEY ([movie\_id]) REFERENCES [Movie] ([id])

GO

ALTER TABLE [UserFavoriteGenre] ADD FOREIGN KEY ([genre\_id]) REFERENCES [Genre] ([id])

GO

ALTER TABLE [UserFavoriteGenre] ADD FOREIGN KEY ([user\_id]) REFERENCES [User] ([id])

GO

ALTER TABLE [Carrier] ADD FOREIGN KEY ([type\_id]) REFERENCES [Type] ([id])

GO

ALTER TABLE [RentedCarriers] ADD FOREIGN KEY ([carrier\_id]) REFERENCES [Carrier] ([id])

GO

ALTER TABLE [RentedCarriers] ADD FOREIGN KEY ([rentaler\_concrete\_id]) REFERENCES [Rentaler\_Concrete] ([id])

GO

ALTER TABLE [MovieCarrier] ADD FOREIGN KEY ([movie\_id]) REFERENCES [Movie] ([id])

GO

ALTER TABLE [MovieCarrier] ADD FOREIGN KEY ([carrier\_id]) REFERENCES [Carrier] ([id])

GO

ALTER TABLE [MovieCarrier] ADD FOREIGN KEY ([recording\_format\_id]) REFERENCES [Recording\_Format] ([id])

GO

ALTER TABLE [WatchHistory] ADD FOREIGN KEY ([movie\_id]) REFERENCES [Movie] ([id])

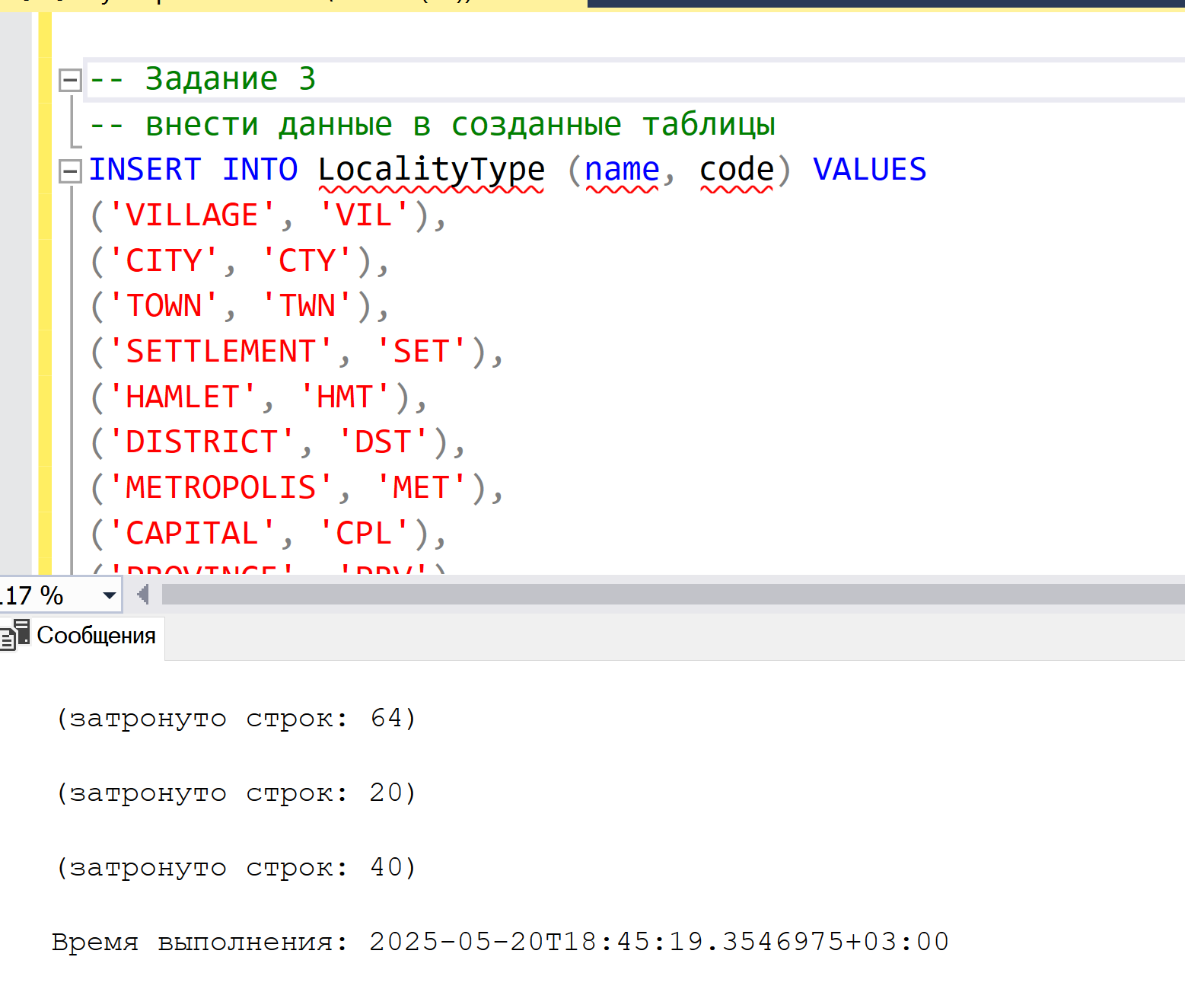
GO

ALTER TABLE [WatchHistory] ADD FOREIGN KEY ([user\_id]) REFERENCES [User] ([id])

GO

- добавлена некоторые проверки на условия данных (возраст положительное целое число, различные поля не должны быть пустыми и так далее)

3. Внести в созданные таблицы БД информацию, соответствующую предметной области вашего варианта индивидального задания (не менее 10- 15 записей в каждой таблице, используя операторы DML. Тексты запросов сохранить в файле.



INSERT INTO LocalityType (name, code) VALUES

('VILLAGE', 'VIL'),

('CITY', 'CTY'),

('TOWN', 'TWN'),

('SETTLEMENT', 'SET'),

('HAMLET', 'HMT'),

('DISTRICT', 'DST'),

('METROPOLIS', 'MET'),

('CAPITAL', 'CPL'),

('PROVINCE', 'PRV'),

('SUBURB', 'SUB');

INSERT INTO Country (name) VALUES

('USA'),

('Canada'),

('UK'),

('France'),

('Germany'),

('Japan'),

('Australia'),

('Russia'),

('Brazil'),

('India'),

('China'),

('Mexico'),

('South Korea'),

('Italy'),

('Spain');

INSERT INTO Website (name, url) VALUES

('Netflix', 'https://www.netflix.com'),

('Amazon Prime', 'https://www.primevideo.com'),

('Disney+', 'https://www.disneyplus.com'),

('Hulu', 'https://www.hulu.com'),

('HBO Max', 'https://www.hbomax.com'),

('YouTube Movies', 'https://www.youtube.com/movies'),

('Apple TV+', 'https://tv.apple.com'),

('Google Play Movies', 'https://play.google.com/store/movies'),

('Vudu', 'https://www.vudu.com'),

('Peacock', 'https://www.peacocktv.com'),

('Paramount+', 'https://www.paramountplus.com'),

('Tubi', 'https://tubitv.com'),

('Crunchyroll', 'https://www.crunchyroll.com'),

('Mubi', 'https://mubi.com'),

('Shudder', 'https://www.shudder.com');

INSERT INTO Rentaler (name, website\_id) VALUES

('Netflix', 1),

('Amazon Rentals', 2),

('Disney Rentals', 3),

('Hulu Rentals', 4),

('HBO Rentals', 5),

('YouTube Rentals', 6),

('Apple Rentals', 7),

('Google Rentals', 8),

('Vudu Rentals', 9),

('Peacock Rentals', 10),

('Paramount Rentals', 11),

('Tubi Rentals', 12),

('Crunchyroll Rentals', 13),

('Mubi Rentals', 14),

('Shudder Rentals', 15);

INSERT INTO Locality (country\_id, locality\_type\_id, name) VALUES

(1, 2, 'New York'),

(1, 2, 'Los Angeles'),

(1, 2, 'Chicago'),

(1, 1, 'Greenwich Village'),

(1, 1, 'Sleepy Hollow'),

(2, 2, 'Toronto'),

(2, 1, 'Niagara-on-the-Lake'),

(3, 2, 'London'),

(3, 1, 'Little Venice'),

(4, 2, 'Paris'),

(4, 1, 'Giverny'),

(5, 2, 'Berlin'),

(5, 1, 'Neuhardenberg'),

(6, 2, 'Tokyo'),

(6, 1, 'Mitaka');

INSERT INTO Address (locality\_id, street, house) VALUES

(1, 'Broadway', 123),

(2, 'Sunset Blvd', 45),

(3, 'Michigan Ave', 100),

(4, 'Washington Square', 12),

(5, 'Main St', 77),

(6, 'Queen St', 555),

(7, 'King St', 101),

(8, 'Baker St', 221),

(9, 'Rose St', 5),

(10, 'Champs Elysees', 10),

(11, 'Rue de Paris', 1),

(12, 'Unter den Linden', 8),

(13, 'Hauptstrasse', 15),

(14, 'Shinjuku', 3),

(15, 'Mitaka St', 7);

INSERT INTO Rentaler\_Concrete (rentaler\_id, address\_id, phone\_number) VALUES

(1, 1, '+1-212-555-1234'),

(1, 4, '+1-212-555-5678'),

(2, 6, '+1-416-555-7890'),

(2, 7, '+1-416-555-4567'),

(3, 10, '+33-1-555-2345'),

(3, 11, '+33-1-555-6789'),

(4, 8, '+44-20-555-3456'),

(4, 9, '+44-20-555-7890'),

(5, 12, '+49-30-555-4567'),

(5, 13, '+49-30-555-7891'),

(6, 14, '+81-3-555-5678'),

(6, 15, '+81-3-555-1234'),

(7, 2, '+1-310-555-9876'),

(8, 3, '+1-312-555-4321'),

(9, 5, '+1-914-555-6543');

INSERT INTO Type (name) VALUES

('DVD'),

('Blu-ray'),

('VHS'),

('Digital'),

('LaserDisc'),

('Betamax'),

('HD DVD'),

('4K UHD'),

('CD'),

('Cassette'),

('Streaming'),

('USB'),

('External HDD'),

('SD Card'),

('Blu-ray 3D');

INSERT INTO Carrier (size, name, type\_id) VALUES

(700, 'Carrier A', 1),

(2500, 'Carrier B', 2),

(1200, 'Carrier C', 3),

(0, 'Digital Stream', 4),

(4400, 'LaserDisc 1', 5),

(800, 'Betamax Tape', 6),

(5000, 'HD DVD 1', 7),

(6000, '4K UHD Disk', 8),

(650, 'Audio CD', 9),

(90, 'Music Cassette', 10),

(0, 'Netflix Stream', 11),

(16000, 'USB Stick', 12),

(100000, 'External HDD 1', 13),

(64, 'SD Card 64GB', 14),

(2700, 'Blu-ray 3D Disc', 15),

(2700, 'Blu-ray with Intersteller', 15),

(3700, 'Blu-ray with La La Land', 15),

(2500, 'Blu-ray withThe Shawshank Redemption', 15),

(2900, 'Blu-ray with Spirited Away', 15),

(3000, 'Blu-ray with The Grand Budapest Hotel', 15),

(700, 'Collectors DVD', 6),

(2500, 'Directors Blu-ray', 6),

(700, 'Limited VHS', 2),

(16000, 'HD Stream', 1),

(0, 'LaserClassic', 11),

(700, 'BetaArchive', 15),

(0, 'HD Master Copy', 6),

(700, '4K Collector Edition', 11),

(700, 'Studio CD', 14),

(0, 'Classic Cassette', 9),

(700, 'Netflix Digital', 11),

(700, 'USB Archive', 4),

(5000, 'HDD Vault', 7),

(5000, 'SD Memory', 10),

(5000, 'Blu-ray 3D Master', 13),

(2500, 'Bonus Disk', 1),

(2500, 'Behind the Scenes', 9),

(5000, 'Remastered Copy', 10),

(0, 'Studio Screener', 5),

(0, 'Original Print', 13);

INSERT INTO Recording\_Format (name) VALUES

('MP4'),

('MKV'),

('AVI'),

('MOV'),

('WMV'),

('FLV'),

('MPEG'),

('WEBM'),

('HEVC'),

('ProRes');

INSERT INTO Genre (name) VALUES

('Action'),

('Comedy'),

('Drama'),

('Horror'),

('Thriller'),

('Romance'),

('Sci-Fi'),

('Fantasy'),

('Documentary'),

('Animation'),

('Adventure'),

('Crime'),

('Mystery'),

('Musical'),

('Western');

INSERT INTO Publisher (name, website\_id) VALUES

('Disney', 3),

('Warner Bros', 5),

('Universal Pictures', 6),

('Paramount Pictures', 11),

('Sony Pictures', 8),

('20th Century Fox', 9),

('Lionsgate', 10),

('MGM', 12),

('Columbia Pictures', 7),

('New Line Cinema', 13),

('Pixar', 3),

('DreamWorks', 14),

('Focus Features', 15),

('Blumhouse', 4),

('A24', 2);

INSERT INTO Director (country\_id, name, description) VALUES

(1, 'Стивен Спилберг', 'Знаменитый американский режиссёр'),

(2, 'Джеймс Кэмерон', 'Канадский режиссёр, известный по фильму «Титаник»'),

(3, 'Кристофер Нолан', 'Британско-американский кинорежиссёр'),

(4, 'Люк Бессон', 'Французский режиссёр и продюсер'),

(5, 'Вернер Херцог', 'Немецкий режиссёр'),

(6, 'Хаяо Миядзаки', 'Японский режиссёр анимации'),

(7, 'Баз Лурман', 'Австралийский режиссёр, известный мюзиклами'),

(8, 'Андрей Тарковский', 'Российский кинорежиссёр'),

(9, 'Жозе Падилья', 'Бразильский режиссёр'),

(10, 'Сатьяджит Рей', 'Индийский режиссёр'),

(11, 'Чжан Имоу', 'Китайский режиссёр'),

(12, 'Гильермо дель Торо', 'Мексиканский режиссёр'),

(13, 'Пак Чхан Ук', 'Южнокорейский режиссёр'),

(14, 'Паоло Соррентино', 'Итальянский режиссёр'),

(15, 'Педро Альмодовар', 'Испанский режиссёр');

INSERT INTO Movie ([name], description, rating, age\_limit, number\_of\_views, duration, release\_year, country\_id, website\_id, publisher\_id, director\_id, genre\_id) VALUES

('Звёздные войны: Новая надежда', 'Эпическое космическое приключение', 9, 13, 2, 150, 2010, 1, 1, 1, 3, 7),

('Реальная любовь', 'Романтическая комедия о любви', 7, 12, 3, 120, 2015, 3, 3, 11, 3, 6),

('Тёмный рыцарь', 'Остросюжетная криминальная драма', 8, 16, 1, 130, 2005, 5, 5, 2, 1, 12),

('История игрушек 4', 'Анимационный фильм для детей', 10, 0, 2, 90, 2018, 6, 3, 11, 6, 10),

('История Рима', 'Исторический документальный фильм', 6, 0, 2, 110, 2012, 4, 6, 13, 5, 9),

('Деревня', 'История ужасов в деревне', 5, 18, 1, 100, 2011, 8, 4, 14, 8, 4),

('Властелин колец', 'Фэнтезийная эпическая сага', 9, 13, 6, 140, 2014, 7, 7, 3, 7, 8),

('Джон Уик', 'Остросюжетный боевик', 8, 16, 3, 115, 2013, 1, 2, 1, 2, 1),

('Мальчишник в Вегасе', 'Комедия о повседневной жизни', 7, 0, 6000, 105, 2016, 3, 3, 14, 3, 2),

('Исчезнувшая', 'Детективная история о преступлении', 8, 16, 6, 125, 2009, 5, 9, 2, 1, 13),

('Начало', 'Научно-фантастический триллер', 9, 14, 2, 130, 2017, 1, 1, 3, 3, 7),

('Дневник памяти', 'Романтическая драма', 7, 12, 3, 100, 2018, 3, 3, 11, 3, 6),

('Планета Земля II', 'Документальный фильм о природе', 8, 0, 1, 90, 2014, 4, 6, 13, 5, 9),

('Холодное сердце 2', 'Анимационное приключение', 9, 0, 1, 95, 2019, 6, 3, 11, 6, 10),

('Настоящая grit', 'Ковбойская история в стиле вестерн', 7, 16, 1, 130, 2010, 1, 2, 1, 1, 15),

('Интерстеллар', 'Захватывающая космическая одиссея о любви и времени', 9, 13, 0, 169, 2014, 1, 1, 3, 3, 7),

('Ла-Ла Ленд', 'Современный мюзикл о мечтах и романтике в Голливуде', 8, 12, 0, 128, 2016, 1, 3, 14, 7, 14),

('Побег из Шоушенка', 'Мощная драма о надежде и дружбе в тюрьме', 10, 16, 0, 142, 1994, 1, 5, 2, 1, 3),

('Унесённые призраками', 'Магическое аниме о девочке в мире духов', 9, 0, 0, 125, 2001, 6, 3, 11, 6, 10),

('Отель «Гранд Будапешт»', 'Причудливая комедия о легендарном консьерже отеля', 8, 13, 0, 100, 2014, 1, 7, 3, 15, 2),

('Матрица', 'Хакер обнаруживает, что реальность — это симуляция', 7, 18, 18, 143, 1991, 8, 12, 6, 3, 6),

('Форрест Гамп', 'Малоумный человек становится свидетелем исторических событий', 9, 6, 7, 127, 2022, 11, 4, 8, 3, 14),

('Криминальное чтиво', 'Истории преступлений в Лос-Анджелесе', 9, 18, 19, 113, 2003, 4, 4, 13, 2, 10),

('Крёстный отец', 'Мафиозная семья борется за власть и верность', 9, 16, 7, 155, 2000, 8, 13, 6, 12, 1),

('Молчание ягнят', 'Стажер ФБР просит помощи у каннибала', 8, 18, 0, 117, 2022, 3, 4, 15, 6, 4),

('Выживший', 'Колонист сражается за выживание после предательства', 7, 12, 0, 118, 2018, 13, 11, 13, 15, 13),

('Чёрный лебедь', 'Балерина погружается в безумие', 10, 16, 10, 109, 2022, 7, 2, 9, 8, 8),

('Социальная сеть', 'История создания Facebook', 9, 18, 7, 163, 2019, 10, 10, 5, 12, 6),

('Одержимость', 'Молодой барабанщик сталкивается с требовательным учителем', 8, 6, 7, 97, 2005, 7, 12, 8, 14, 13),

('Она', 'Мужчина влюбляется в систему искусственного интеллекта', 6, 18, 2, 134, 2012, 14, 5, 3, 6, 9),

('Прибытие', 'Лингвист расшифровывает язык пришельцев', 10, 16, 12, 95, 1995, 2, 7, 5, 6, 1),

('Паразиты', 'Бедная семья проникает в богатый дом', 6, 12, 14, 176, 2020, 11, 13, 15, 7, 6),

('Престиж', 'Два мага в смертельном соперничестве', 8, 16, 14, 169, 1996, 3, 3, 10, 9, 5),

('Игра в имитацию', 'Алан Тьюринг взламывает нацистскую Энигму', 6, 18, 19, 144, 2000, 15, 7, 12, 1, 9),

('Амели', 'Причудливая француженка меняет жизни окружающих', 9, 18, 1, 179, 2006, 13, 14, 12, 10, 11),

('Миллионер из трущоб', 'Сирота побеждает в телеигре вопреки всему', 8, 16, 10, 99, 2017, 11, 5, 11, 14, 12),

('Джокер', 'Психически больной человек погружается в безумие', 6, 6, 11, 172, 1991, 12, 1, 4, 9, 6),

('1917', 'Два солдата Первой мировой несут важное сообщение', 9, 6, 2, 133, 2004, 12, 5, 6, 10, 14),

('Тайна Коко', 'Мальчик попадает в мир мёртвых', 9, 16, 2, 96, 2010, 13, 15, 8, 10, 3),

('Головоломка', 'Эмоции внутри сознания девочки', 6, 16, 1, 176, 1994, 5, 1, 6, 15, 11);

INSERT INTO Collection (name, description) VALUES

('Лучшая научная фантастика', 'Лучшие научно-фантастические фильмы'),

('Романтическая классика', 'Лучшие романтические фильмы всех времён'),

('Ночь ужасов', 'Страшные фильмы ужасов'),

('Семейные развлечения', 'Фильмы, подходящие для семейного просмотра'),

('Лучшие комедии', 'Лучшие комедийные фильмы'),

('Экшн-триллеры', 'Лучшие остросюжетные боевики'),

('Документальное кино', 'Образовательные и познавательные фильмы'),

('Любимая анимация', 'Популярные анимационные фильмы'),

('Криминальные истории', 'Лучшие криминальные драмы'),

('Фэнтези-миры', 'Фильмы, действие которых происходит в фантастических мирах'),

('Приключенческие путешествия', 'Захватывающие приключенческие фильмы'),

('Тайны и саспенс', 'Интригующие детективные фильмы'),

('Вестерн-легенды', 'Классические вестерны'),

('Мюзиклы', 'Знаменитые мюзиклы'),

('Мировое кино', 'Лучшие зарубежные фильмы');

INSERT INTO MovieCollection (collection\_id, movie\_id) VALUES

(1,1),

(1, 16),

(1, 21),

(1, 11),

(2,2),

(2, 12),

(2, 17),

(2, 30),

(3,6),

(3, 25),

(3, 10),

(4,4),

(4, 19),

(4, 31),

(4, 14),

(5,9),

(6,8),

(7,5),

(8,14),

(9,3),

(10,7),

(11,11),

(12,10),

(13,15),

(14,2),

(15,13);

INSERT INTO [User] (age, name) VALUES

(45, 'Mom'),

(48, 'Dad'),

(20, 'Daughter'),

(18, 'Son'),

(6, 'Cat');

INSERT INTO UserFavoriteMovies (user\_id, movie\_id) VALUES

-- Mom

(1, 1), (1, 3), (1, 5), (1, 7), (1, 9),

-- Dad

(2, 2), (2, 4), (2, 6), (2, 8), (2, 10),

-- Daughter

(3, 11), (3, 12), (3, 13), (3, 14), (3, 15),

-- Son

(4, 1), (4, 4), (4, 7), (4, 10), (4, 13),

-- Cat

(5, 2), (5, 5), (5, 8), (5, 11), (5, 14);

INSERT INTO UserFavoriteGenre (genre\_id, user\_id) VALUES

-- Mom

(1, 1), (3, 1), (5, 1),

-- Dad

(2, 2), (4, 2), (6, 2),

-- Daughter

(1, 3), (2, 3), (7, 3),

-- Son

(3, 4), (5, 4), (7, 4),

-- Cat

(4, 5), (6, 5), (7, 5);

INSERT INTO WatchHistory (movie\_id, user\_id, watched\_at) VALUES

-- Mom

(1, 1, '2024-05-01T20:00:00'),

(3, 1, '2024-05-02T21:00:00'),

(5, 1, '2024-05-03T22:00:00'),

(7, 1, '2024-05-04T19:30:00'),

(9, 1, '2024-05-05T20:15:00'),

(27, 1, '2025-04-06T15:34:55'),

(28, 1, '2025-05-09T15:34:55'),

(36, 1, '2025-05-13T15:34:55'),

(38, 1, '2025-05-01T15:34:55'),

(40, 1, '2025-04-06T15:34:55'),

(40, 1, '2025-05-11T15:34:55'),

-- Dad

(2, 2, '2024-04-10T18:45:00'),

(4, 2, '2024-04-12T20:00:00'),

(6, 2, '2024-04-13T21:15:00'),

(8, 2, '2024-04-15T19:00:00'),

(10, 2, '2024-04-17T20:30:00'),

(25, 2, '2025-05-13T15:34:55'),

(27, 2, '2025-05-12T15:34:55'),

(29, 2, '2025-03-29T15:34:55'),

(34, 2, '2025-04-15T15:34:55'),

(35, 2, '2025-04-07T15:34:55'),

(38, 2, '2025-03-25T15:34:55'),

(39, 2, '2025-04-15T15:34:55'),

-- Daughter

(11, 3, '2024-03-20T17:00:00'),

(12, 3, '2024-03-22T18:00:00'),

(13, 3, '2024-03-24T19:15:00'),

(14, 3, '2024-03-26T20:45:00'),

(15, 3, '2024-03-28T21:30:00'),

(25, 3, '2025-05-06T15:34:55'),

(27, 3, '2025-04-12T15:34:55'),

(28, 3, '2025-04-16T15:34:55'),

(31, 3, '2025-04-19T15:34:55'),

(31, 3, '2025-05-04T15:34:55'),

(32, 3, '2025-04-12T15:34:55'),

(33, 3, '2025-04-09T15:34:55'),

(37, 3, '2025-04-07T15:34:55'),

(38, 3, '2025-05-11T15:34:55'),

(39, 3, '2025-04-15T15:34:55'),

-- Son

(1, 4, '2024-02-10T16:30:00'),

(4, 4, '2024-02-12T17:45:00'),

(7, 4, '2024-02-14T19:00:00'),

(10, 4, '2024-02-16T20:15:00'),

(13, 4, '2024-02-18T21:00:00'),

(21, 4, '2025-05-05T15:34:55'),

(21, 4, '2025-03-21T15:34:55'),

(25, 4, '2025-03-28T15:34:55'),

(26, 4, '2025-05-10T15:34:55'),

(29, 4, '2025-04-03T15:34:55'),

(30, 4, '2025-04-23T15:34:55'),

-- Cat

(2, 5, '2024-01-05T15:00:00'),

(5, 5, '2024-01-07T16:30:00'),

(8, 5, '2024-01-09T17:45:00'),

(11, 5, '2024-01-11T19:00:00'),

(14, 5, '2024-01-13T20:30:00'),

(22, 5, '2025-05-02T15:34:55'),

(23, 5, '2025-04-28T15:34:55'),

(24, 5, '2025-05-08T15:34:55'),

(26, 5, '2025-03-24T15:34:55'),

(32, 5, '2025-04-16T15:34:55'),

(32, 5, '2025-04-10T15:34:55'),

(36, 5, '2025-05-06T15:34:55'),

(36, 5, '2025-04-03T15:34:55'),

(39, 5, '2025-04-06T15:34:55'),

(40, 5, '2025-05-11T15:34:55');

INSERT INTO RentedCarriers (start\_rent, end\_rent, carrier\_id, rentaler\_concrete\_id) VALUES

-- просрочил

(DATEADD(DAY, -10, GETDATE()), DATEADD(DAY, -9, GETDATE()), 1, 1),

(DATEADD(DAY, -8, GETDATE()), DATEADD(DAY, -7, GETDATE()), 2, 2),

(DATEADD(DAY, -6, GETDATE()), DATEADD(DAY, -5, GETDATE()), 3, 3),

(DATEADD(DAY, -4, GETDATE()), DATEADD(DAY, -3, GETDATE()), 4, 4),

-- сегодня закончиться

(DATEADD(DAY, -1, GETDATE()), DATEADD(HOUR, 2, GETDATE()), 5, 5),

-- активные

(DATEADD(DAY, -2, GETDATE()), DATEADD(DAY, 1, GETDATE()), 6, 6),

(DATEADD(DAY, -3, GETDATE()), DATEADD(DAY, 2, GETDATE()), 7, 7),

(DATEADD(DAY, -4, GETDATE()), DATEADD(DAY, 3, GETDATE()), 8, 8),

(DATEADD(DAY, -5, GETDATE()), DATEADD(DAY, 4, GETDATE()), 9, 9),

(DATEADD(DAY, -6, GETDATE()), DATEADD(DAY, 5, GETDATE()), 10, 10),

(GETDATE(), DATEADD(DAY, 1, GETDATE()), 21, 11),

(GETDATE(), DATEADD(DAY, 1, GETDATE()), 23, 9),

(GETDATE(), DATEADD(DAY, 5, GETDATE()), 25, 8),

(GETDATE(), DATEADD(DAY, 4, GETDATE()), 27, 11),

(GETDATE(), DATEADD(DAY, 3, GETDATE()), 29, 7),

(GETDATE(), DATEADD(DAY, 2, GETDATE()), 31, 13),

(GETDATE(), DATEADD(DAY, 2, GETDATE()), 33, 6),

(GETDATE(), DATEADD(DAY, 1, GETDATE()), 35, 11),

(GETDATE(), DATEADD(DAY, 5, GETDATE()), 37, 11),

(GETDATE(), DATEADD(DAY, 3, GETDATE()), 39, 7);

INSERT INTO [MovieCarrier] (movie\_id, carrier\_id, recording\_format\_id) VALUES

(1, 1, 1),

(2, 2, 7),

(3, 3, 3),

(4, 4, 10),

(5, 5, 1),

(6, 6, 1),

(7, 7, 1),

(8, 8, 3),

(9, 9, 4),

(10, 10, 5),

(11, 11, 7),

(12, 12, 1),

(13, 13, 1),

(14, 14, 1),

(15, 15, 9),

(16, 16, 6),

(17, 17, 2),

(18, 18, 8),

(19, 19, 1),

(20, 20, 1),

(21, 21, 6),

(22, 22, 7),

(23, 23, 7),

(24, 24, 8),

(25, 25, 6),

(26, 26, 10),

(27, 27, 7),

(28, 28, 7),

(29, 29, 6),

(30, 30, 1),

(31, 31, 3),

(32, 32, 7),

(33, 33, 2),

(34, 34, 4),

(35, 35, 10),

(36, 36, 6),

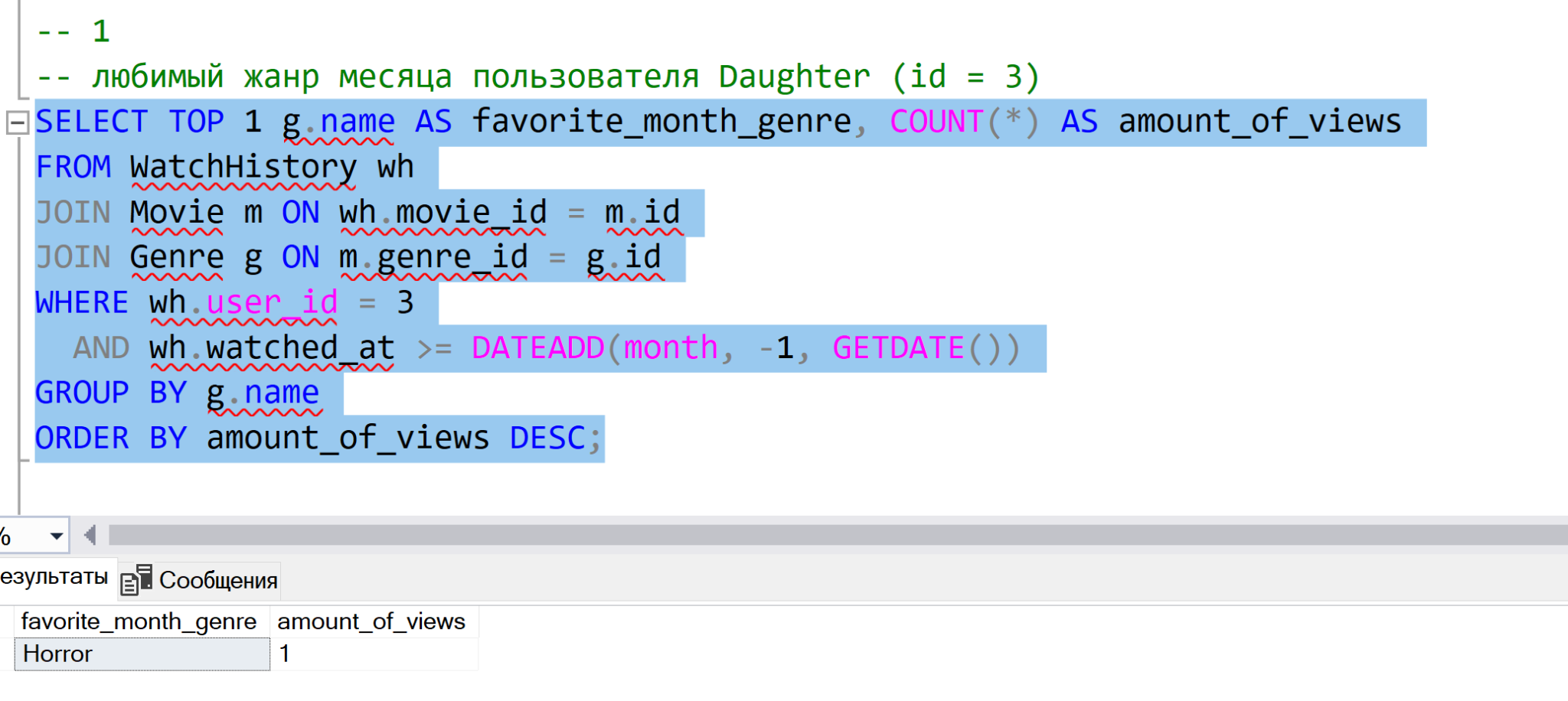
(37, 37, 3),

(38, 38, 8),

(39, 39, 4),

(40, 40, 5);

4. Сформировать не менее 10 запросов на выборку данных, соответствующих специфике предметной области, потребностям предполагаемых пользователей и описанию выходных документов (ЛР-4). Тексты запросов сохранить в файле.



SELECT TOP 1 g.name AS favorite\_month\_genre, COUNT(\*) AS amount\_of\_views

FROM WatchHistory wh

JOIN Movie m ON wh.movie\_id = m.id

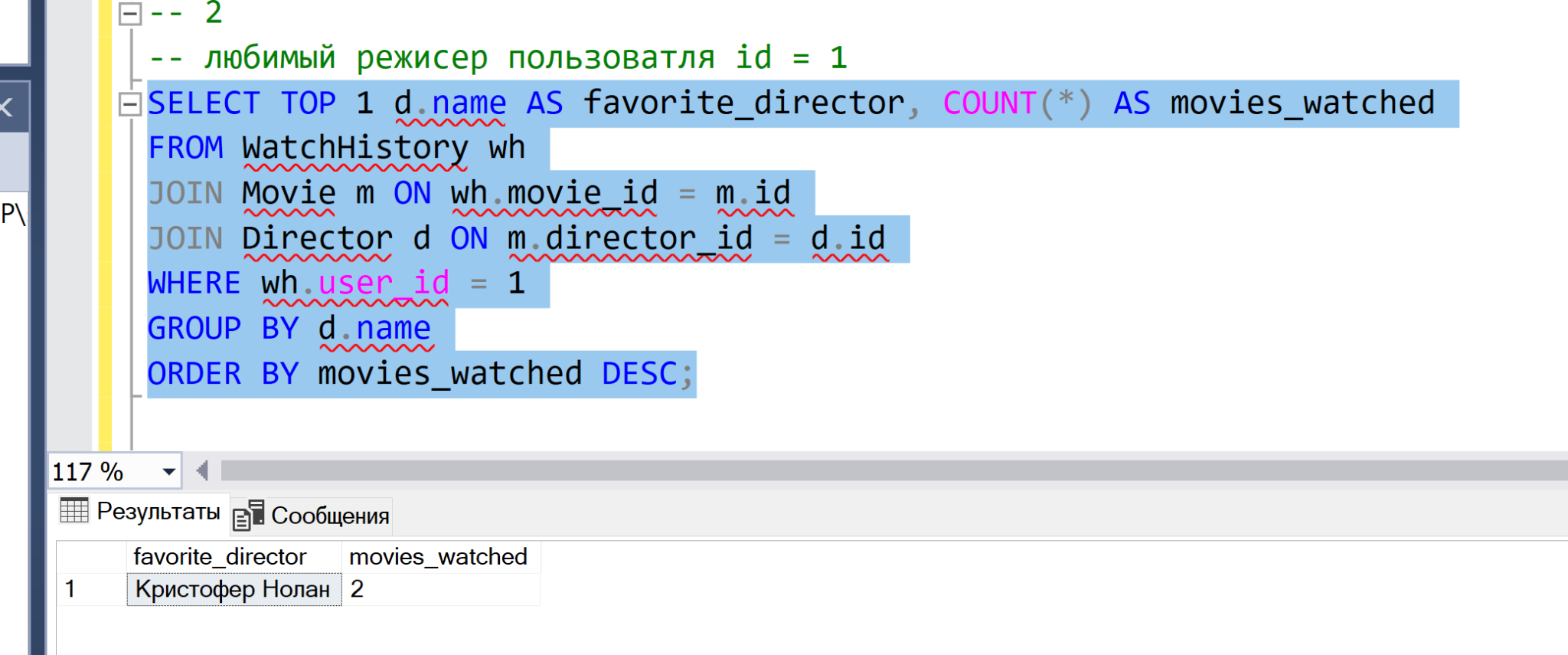
JOIN Genre g ON m.genre\_id = g.id

WHERE wh.user\_id = 3

AND wh.watched\_at >= DATEADD(month, -1, GETDATE())

GROUP BY g.name

ORDER BY amount\_of\_views DESC;



SELECT TOP 1 d.name AS favorite\_director, COUNT(\*) AS movies\_watched

FROM WatchHistory wh

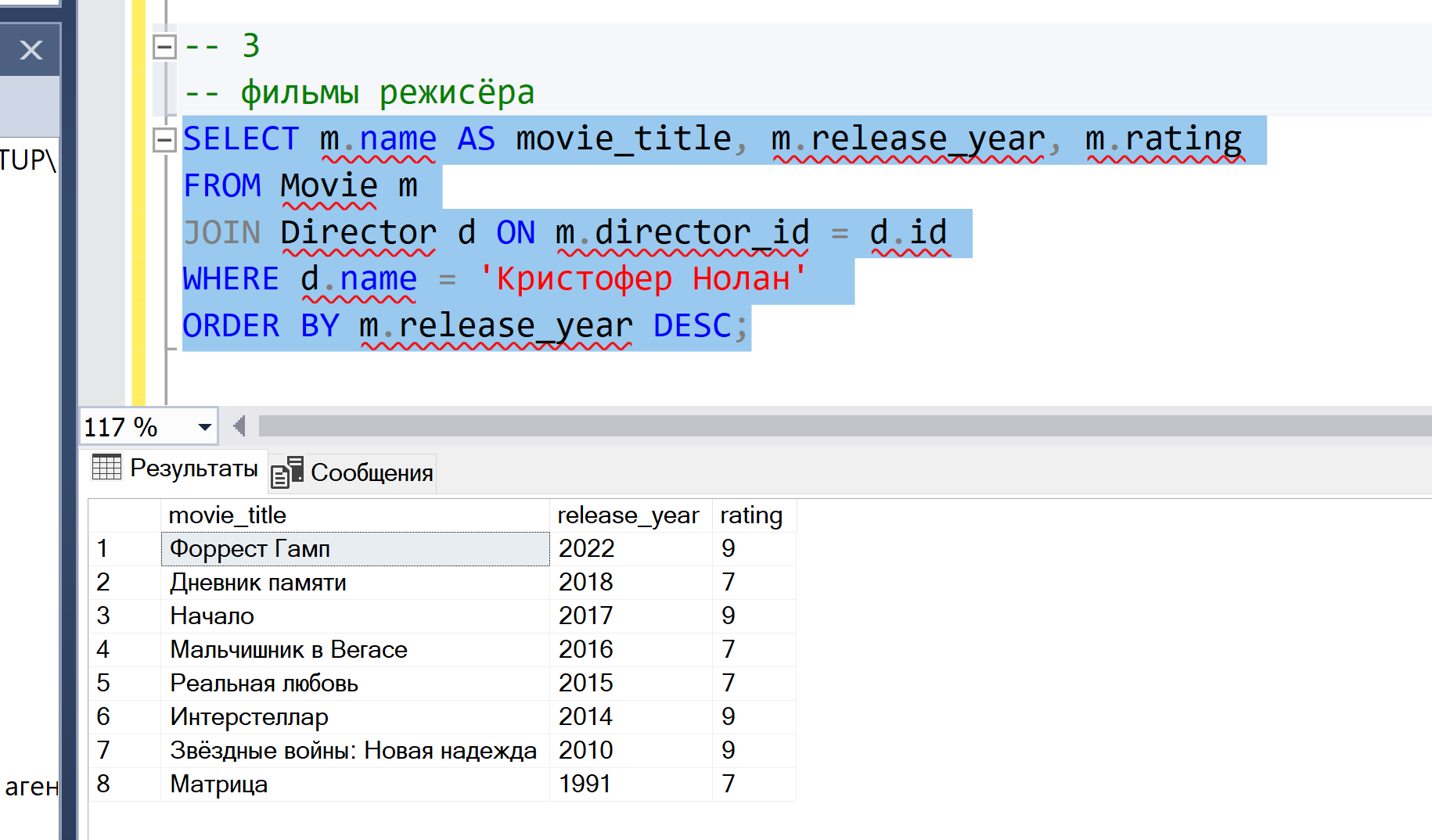
JOIN Movie m ON wh.movie\_id = m.id

JOIN Director d ON m.director\_id = d.id

WHERE wh.user\_id = 1

GROUP BY d.name

ORDER BY movies\_watched DESC;



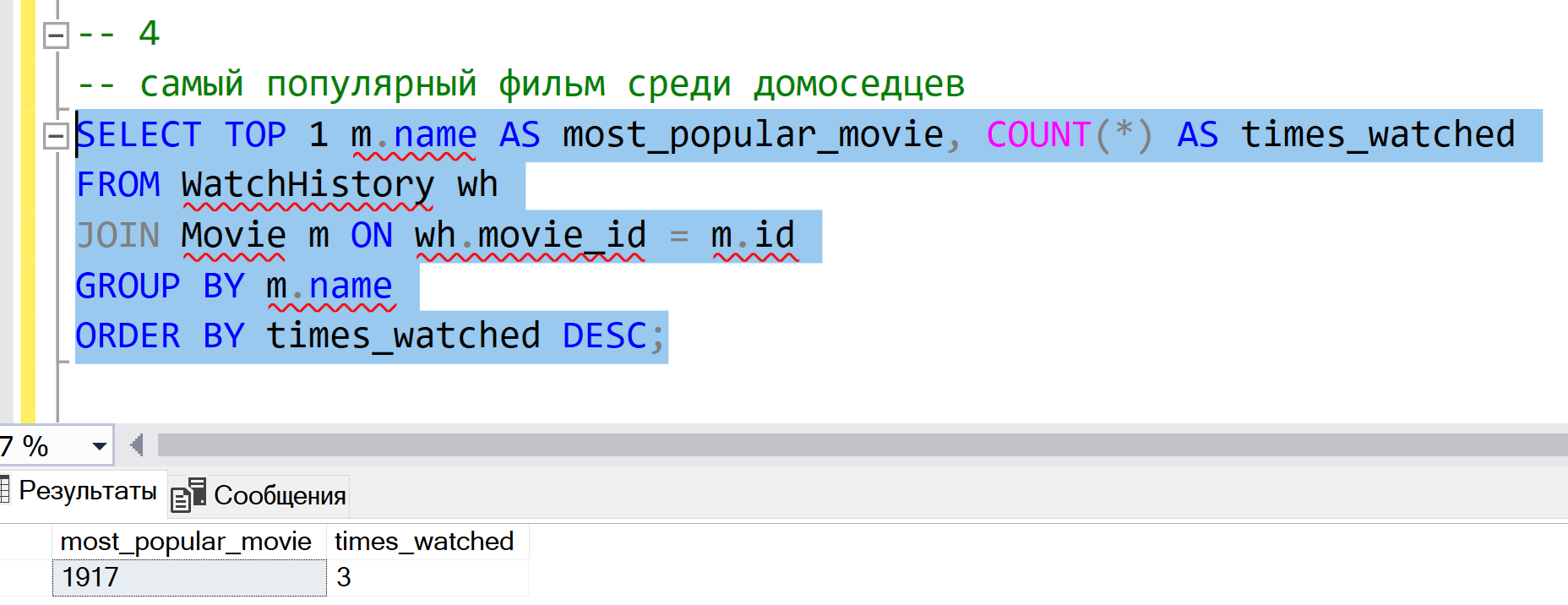
SELECT m.name AS movie\_title, m.release\_year, m.rating

FROM Movie m

JOIN Director d ON m.director\_id = d.id

WHERE d.name = 'Кристофер Нолан'

ORDER BY m.release\_year DESC;



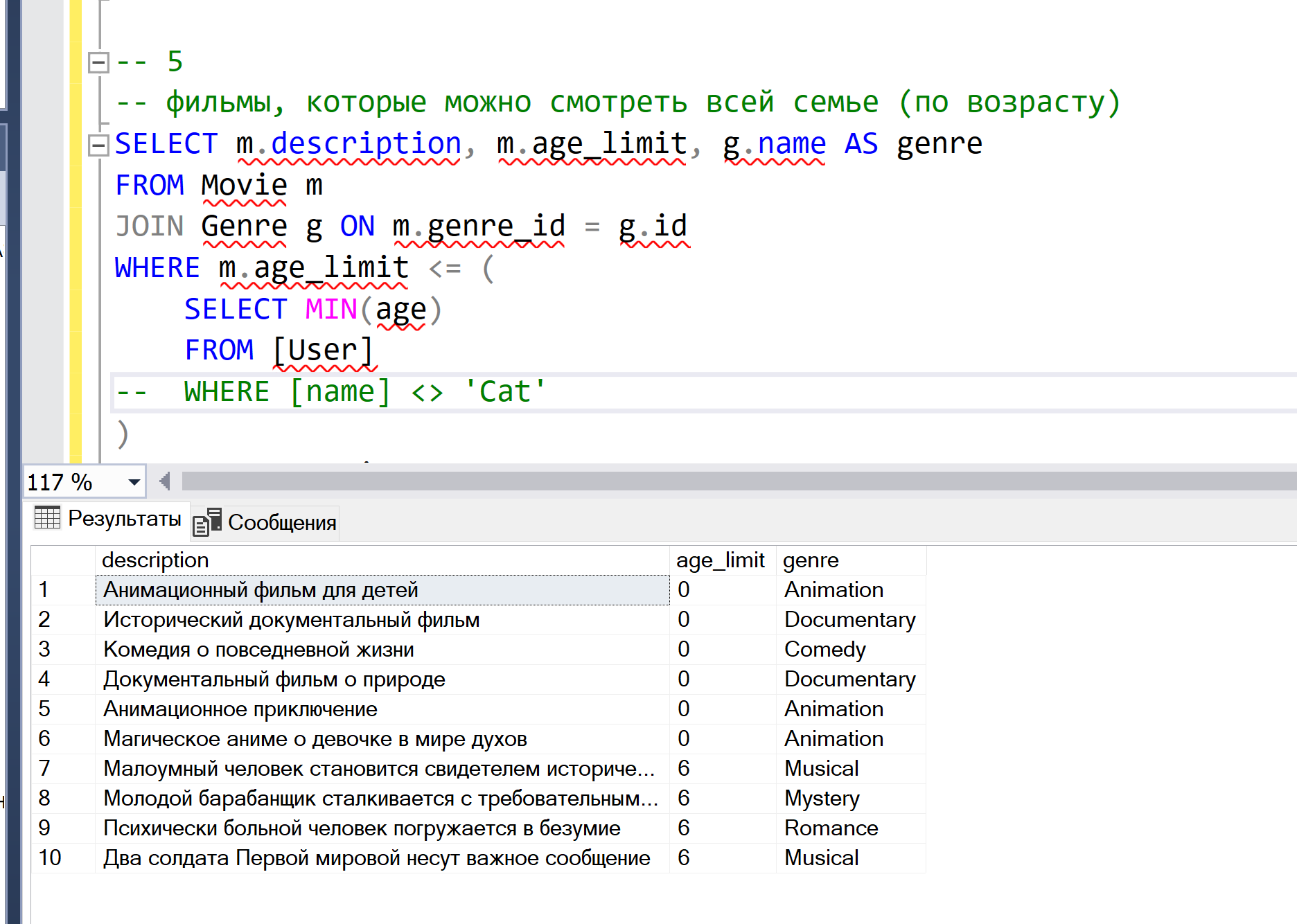
SELECT TOP 1 m.name AS most\_popular\_movie, COUNT(\*) AS times\_watched

FROM WatchHistory wh

JOIN Movie m ON wh.movie\_id = m.id

GROUP BY m.name

ORDER BY times\_watched DESC;



SELECT m.description, m.age\_limit, g.name AS genre

FROM Movie m

JOIN Genre g ON m.genre\_id = g.id

WHERE m.age\_limit <= (

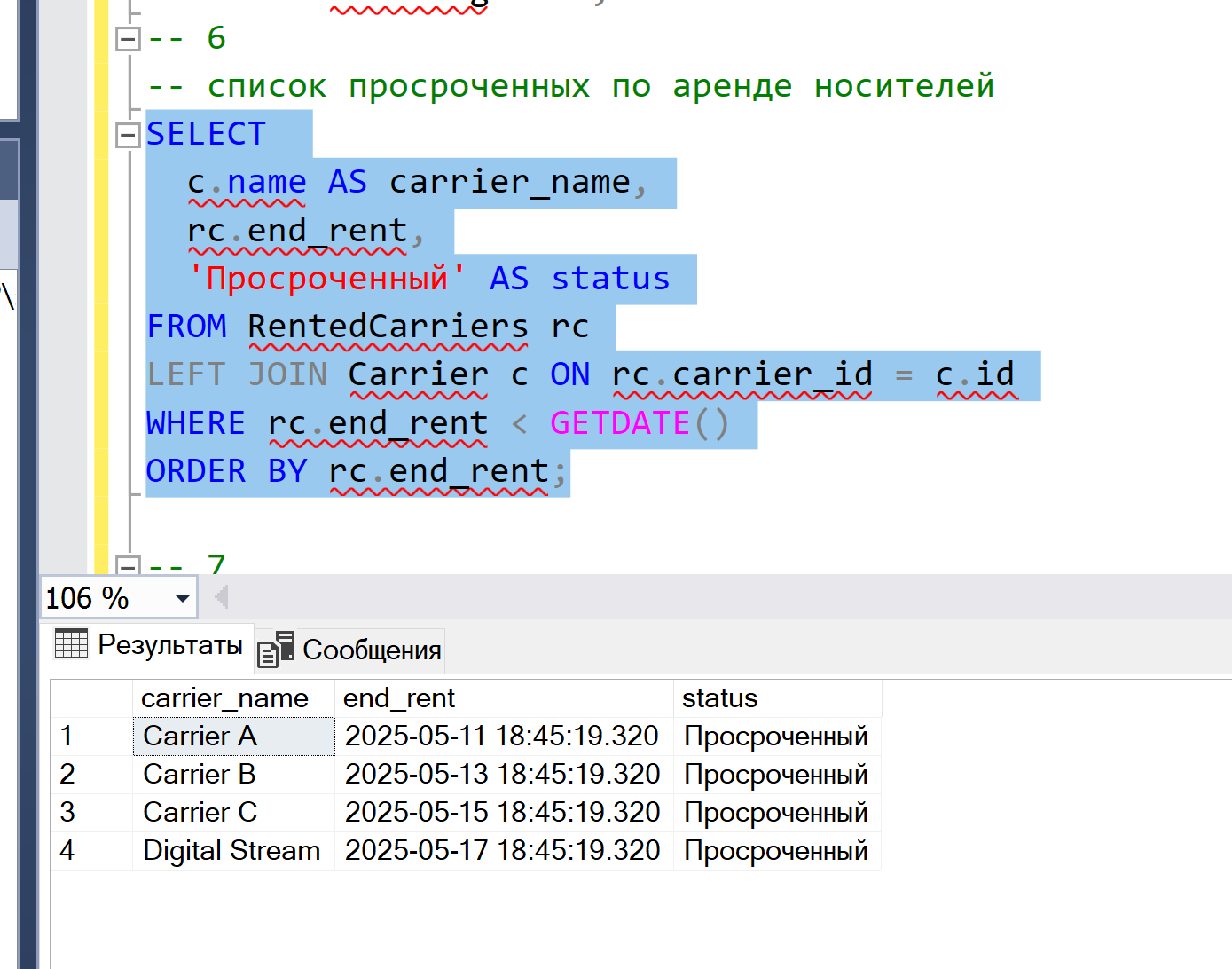
SELECT MIN(age)

FROM [User]

-- WHERE [name] <> 'Cat'

)

ORDER BY m.rating DESC;



SELECT

c.name AS carrier\_name,

rc.end\_rent,

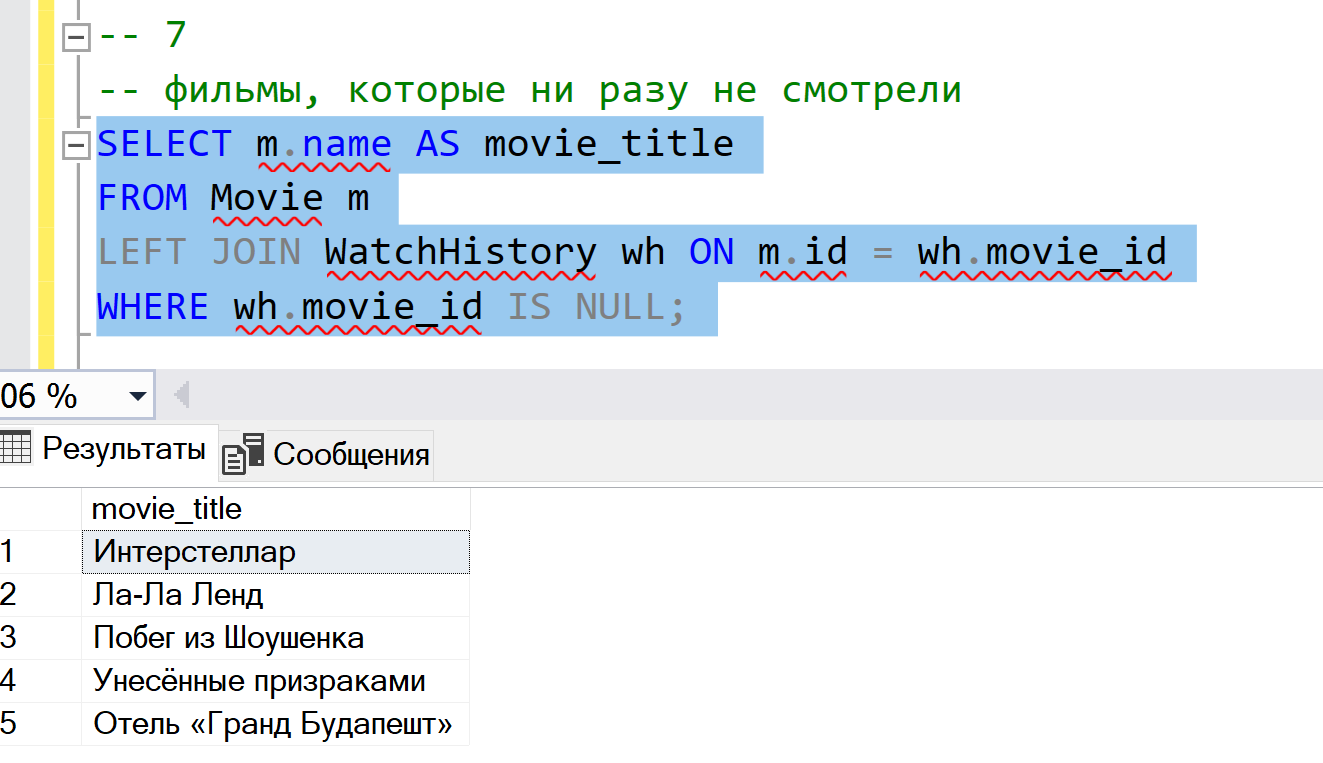
'Просроченный' AS status

FROM RentedCarriers rc

LEFT JOIN Carrier c ON rc.carrier\_id = c.id

WHERE rc.end\_rent < GETDATE()

ORDER BY rc.end\_rent;

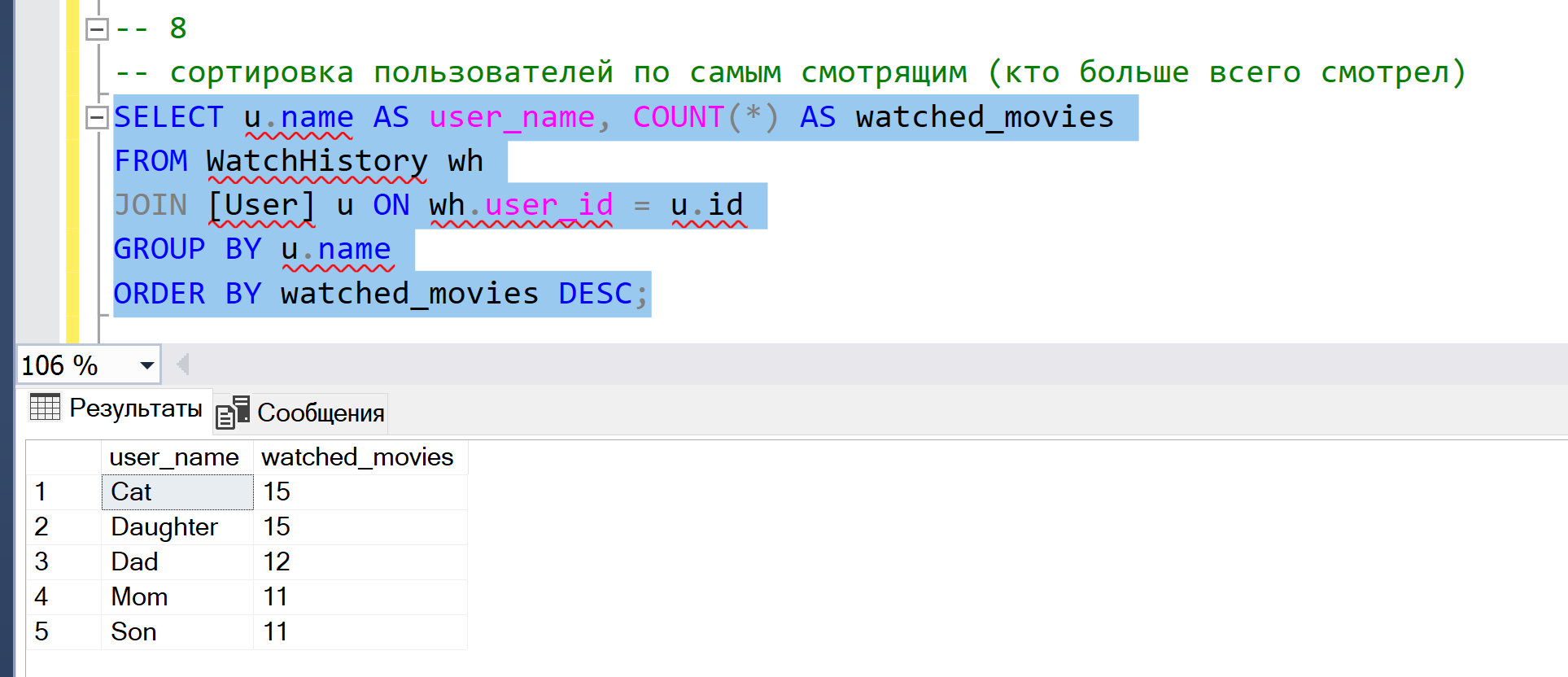


SELECT m.name AS movie\_title

FROM Movie m

LEFT JOIN WatchHistory wh ON m.id = wh.movie\_id

WHERE wh.movie\_id IS NULL;



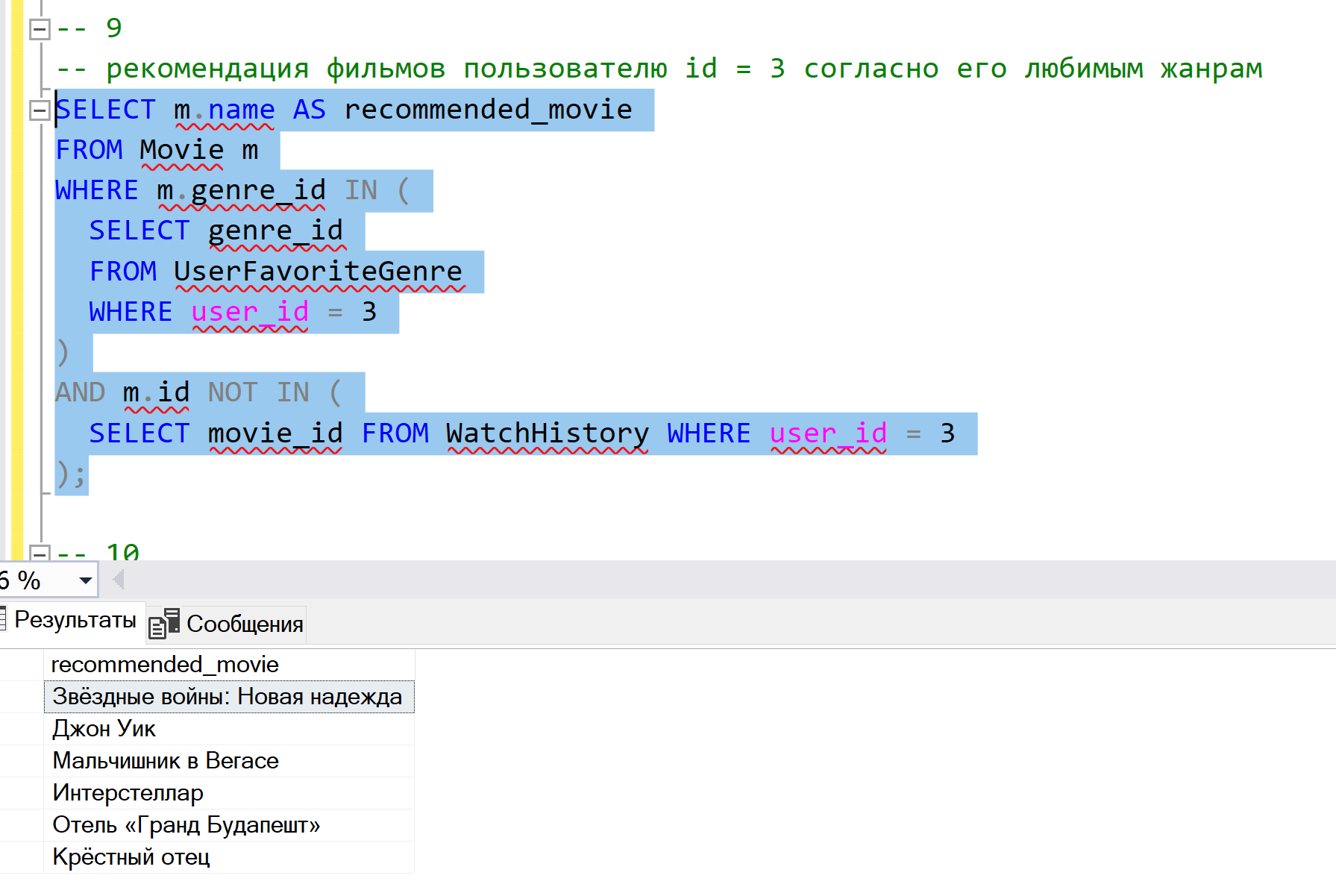
SELECT u.name AS user\_name, COUNT(\*) AS watched\_movies

FROM WatchHistory wh

JOIN [User] u ON wh.user\_id = u.id

GROUP BY u.name

ORDER BY watched\_movies DESC;



SELECT m.name AS recommended\_movie

FROM Movie m

WHERE m.genre\_id IN (

SELECT genre\_id

FROM UserFavoriteGenre

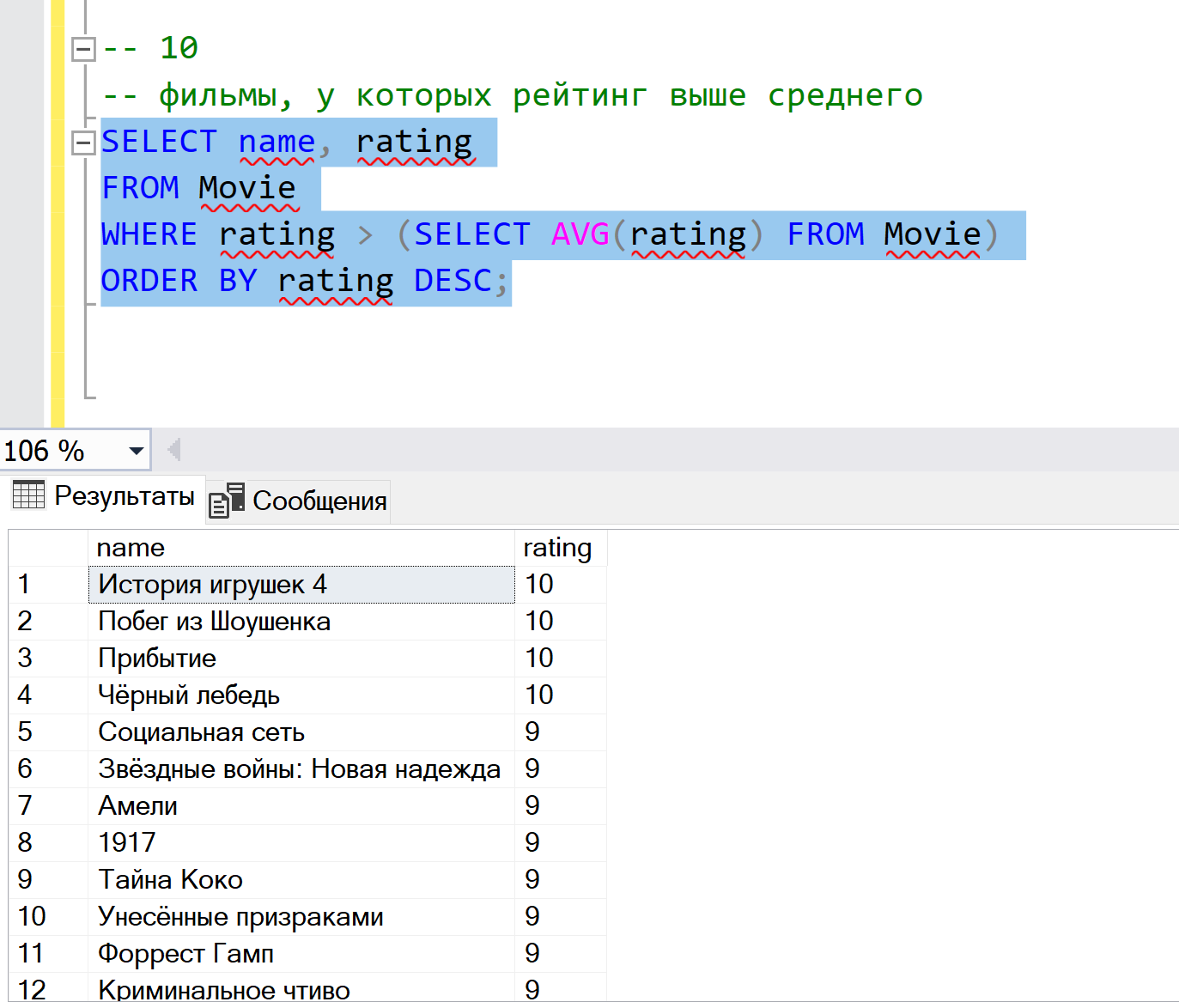
WHERE user\_id = 3

)

AND m.id NOT IN (

SELECT movie\_id FROM WatchHistory WHERE user\_id = 3

);

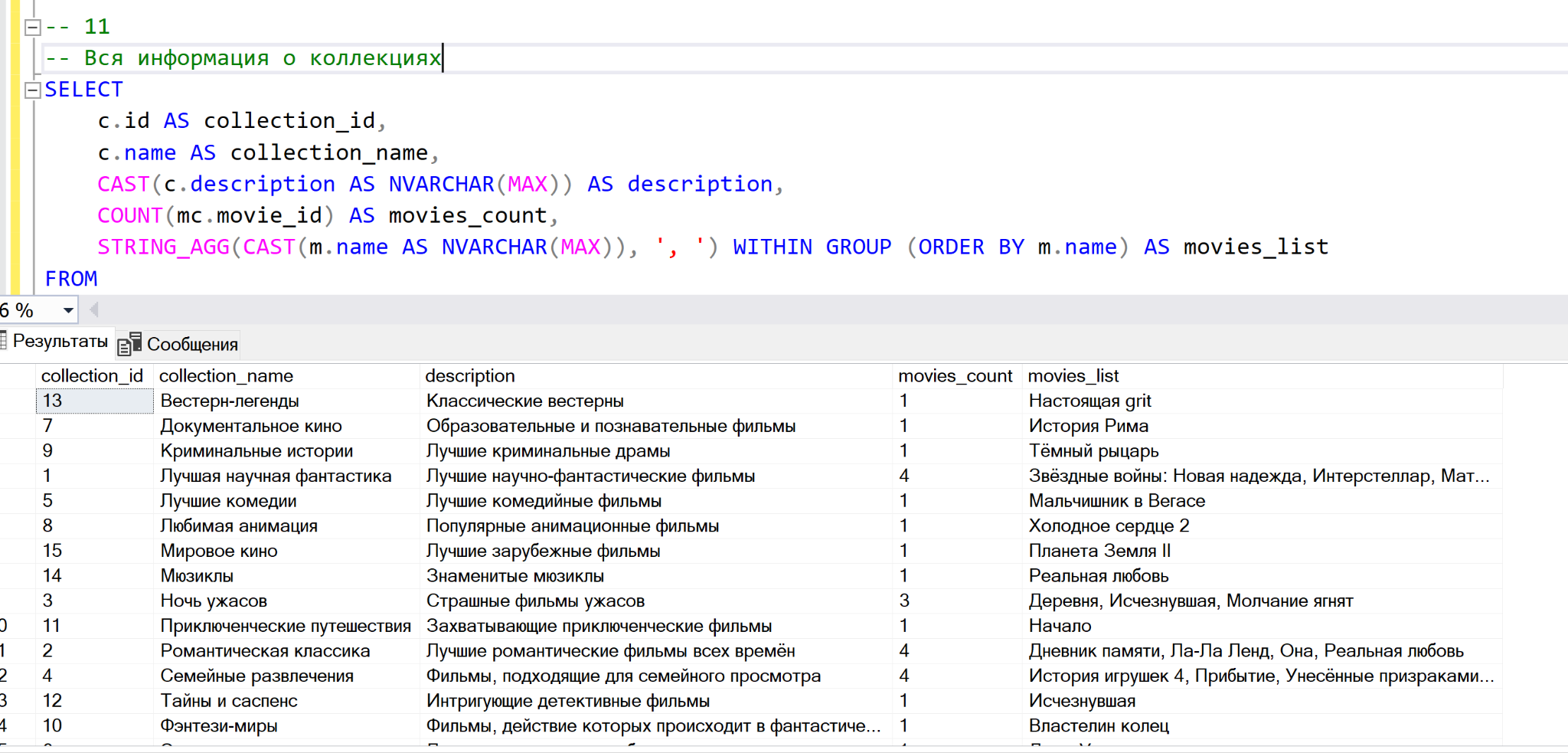


SELECT name, rating

FROM Movie

WHERE rating > (SELECT AVG(rating) FROM Movie)

ORDER BY rating DESC;



SELECT

c.id AS collection\_id,

c.name AS collection\_name,

CAST(c.description AS NVARCHAR(MAX)) AS description,

COUNT(mc.movie\_id) AS movies\_count,

STRING\_AGG(CAST(m.name AS NVARCHAR(MAX)), ', ') WITHIN GROUP (ORDER BY m.name) AS movies\_list

FROM

[Collection] c

LEFT JOIN

[MovieCollection] mc ON c.id = mc.collection\_id

LEFT JOIN

[Movie] m ON mc.movie\_id = m.id

GROUP BY

c.id, c.name, CAST(c.description AS NVARCHAR(MAX))

ORDER BY

c.name;