ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ «ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Высшая школа бизнеса

**ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА БАЗЫ ДАННЫХ КИНОТЕАТРА**

Курсовая работа студентки

Калякиной Марии Алексеевны

2 курс, направление подготовки: 38.03.05 «Бизнес-информатика»

образовательная программа «Бизнес-информатика»

Научный руководитель

к.т.н., доцент

\_\_\_\_\_\_\_\_\_\_\_\_\_\_

С. М. Ямпольский

Москва 2021

Оглавление

[1. Постановка задачи 3](#_Toc74143432)

[1.1. Использованные методы и инструменты 3](#_Toc74143433)

[1.2. Описание предметной области 3](#_Toc74143434)

[2. Проектирование базы данных 5](#_Toc74143435)

[2.1. Инфологическая модель кинотеатра 5](#_Toc74143436)

[2.1.1. Описание сущностей 5](#_Toc74143437)

[2.2. Даталогическая модель 6](#_Toc74143438)

[2.2.1. Описание таблиц 7](#_Toc74143439)

[2.2.2. Описание связей между таблицами 11](#_Toc74143440)

[3. Реализация базы данных 14](#_Toc74143441)

[3.1. Заполнение начальными данными 15](#_Toc74143442)

[3.2. Функции 16](#_Toc74143443)

[3.3. Триггеры 19](#_Toc74143444)

[3.4. Хранимые процедуры 22](#_Toc74143445)

[3.5. Запросы 27](#_Toc74143446)

[4. Разработка пользовательского приложения 30](#_Toc74143447)

[5. Заключение 41](#_Toc74143448)

[6. Список литературы 42](#_Toc74143449)

# Постановка задачи

Объектом исследования в данной работе служит среднестатистический кинотеатр, а предметом исследования – база данных кинотеатра для систематизации и автоматизации его работы.

Цель работы – создание практичной, удобной для использования базы данных, а также интерфейса с целью предоставления пользователям упрощенного и интуитивно понятного доступа к ней.

В ходе работы передо мной были поставлены следующие задачи:

1. Анализ основных положений, связей и механизмов, присущих кинотеатру.
2. Проектирование базы данных, состоящее из построения инфологической модели предметной области и даталогической моделей данных.
3. Реализация базы данных в СУБД.
4. Создание интерфейса на языке Python для реализованной базы данных.

## Использованные методы и инструменты

Для проектирования инфологической модели предметной области был использован ERD Plus, для даталогической модели данных – Erwin Data Modeler 2020. Реализация базы данных происходила в СУБД MS SQL Server 2019, а разработка интерфейса – в среде разработки PyCharm для языка программирования Python.

## Описание предметной области

Спроектированная для среднестатистического кинотеатра база данных Cinema служит для систематизации обслуживания клиентов кассирами, а также упрощения контроля за бизнесом в целом администратором. База данных позволяет получить доступ ко всем денежным потокам, осуществляемых в течение работы кинотеатра, а также всю необходимую информацию о фильмах в прокате на текущее время.

Одной из задач администратора является составление расписания киносеансов, стоимость билетов на которые определяется как цена за фильм, зависящая от его жанра, умноженная на выраженную в процентных пунктах наценку. Величина наценки зависит от дня недели и времени суток. Другими словами, цена билета на сеанс вечера субботы будет на порядок выше цены на тот же фильм во вторник утром. Также администратор контролирует работу своих подчиненных-кассиров путем составления расписаний смен и закрепления за ними определенного рабочего места в течение фиксированного времени. При продаже билетов также заносится информация о кассире и его смене, что способствует прозрачности в анализе их работы. Администратор также обязан следить за регулярными затратами при ведении своего бизнеса, состоящие из выплат зарплат сотрудникам и уплаты аренды, предоставление систематизированной информации о текущих и будущих обязательствах также поддерживается базой данных Cinema.

# Проектирование базы данных

После того, как была проанализирована и описана предметная область, необходимо спроектировать инфологическую и даталогическую модели, и для лучшего их понимания также описать сущности, таблицы и связи между ними.

## Инфологическая модель кинотеатра

Инфологическая модель позволяет структурированно отобразить основные сущности, принципы и механизмы взаимосвязи между ними, основываясь на описании предметной области. Данная модель была разработана с помощью CASE-средства ERD Plus, базирующегося на методологии Питера Чена.

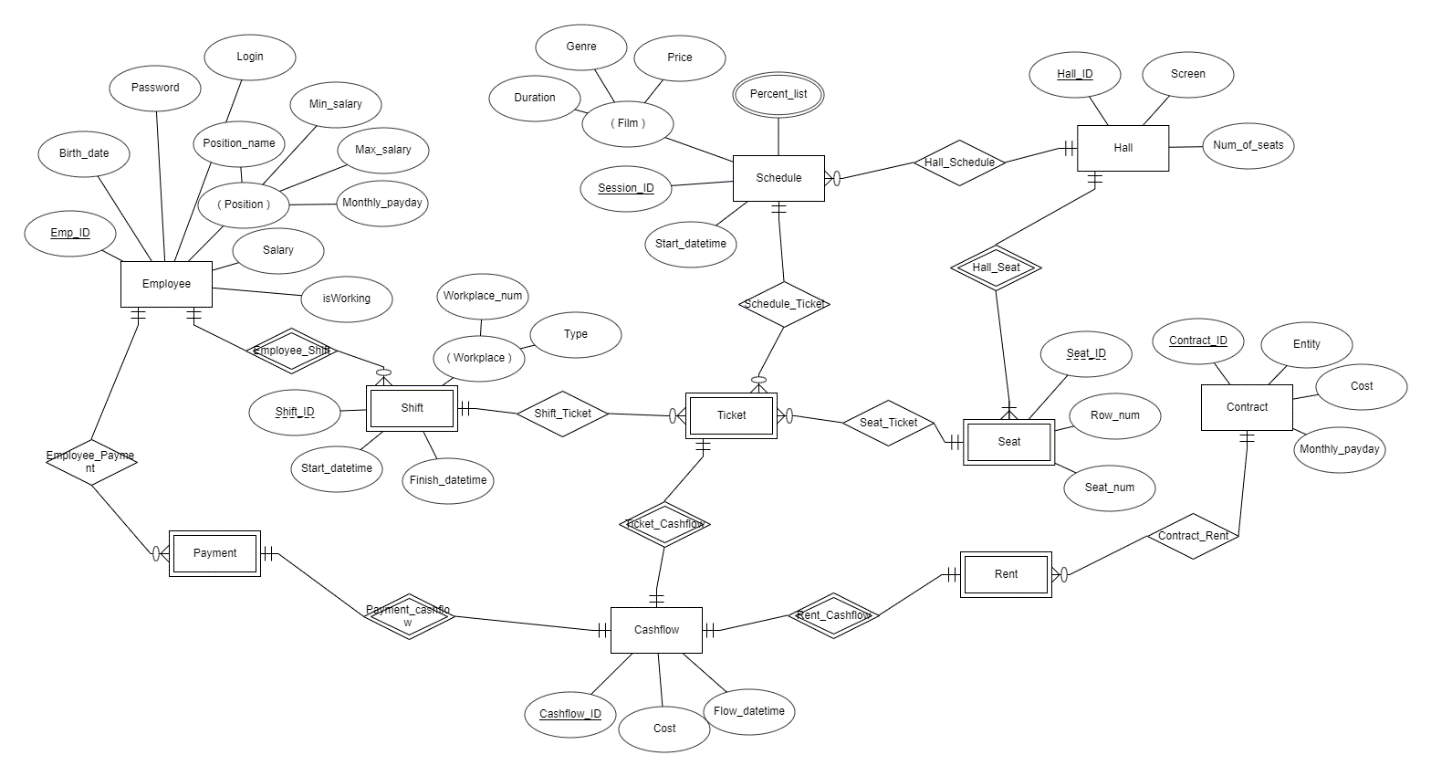


Рисунок 1. Инфологическая модель кинотеатра

### Описание сущностей

При проектировании инфологической модели было создано 10 сущностей, которые делятся на два типа: сущности-справочники и сущности-документы.

Сущности-справочники:

1. Employee – справочные данные о сотрудниках и занимаемых ими должностях.
2. Contract - справочные данные о всех заключенных контрактах об аренде.
3. Hall – справочные данные о залах кинотеатра.
4. Seat - справочные данные о местах в кинотеатре.

Сущности-документы:

1. Shift - данные о сменах сотрудников и прикрепленных к ним рабочим местам.
2. Schedule – данные о киносеансах которые состоят, в частости, из справочных данных о фильмах, жанрах, а также списка наценок на сеансы.
3. Cashflow – данные о всех проведенных на протяжении работы кинотеатра денежных транзакциях.
4. Payment – хранит данные о том, какому сотруднику была выплачена та или иная денежная транзакция (или по-другому – зарплата).
5. Ticket – содержит данные о билетах, а также смене, в которую был продан тот или иной билет, на какой сеанс он был забронирован, а также номер места зала.
6. Rent - хранит данные о том, по какому контракту была осуществлена та или иная денежная транзакция (или по-другому –уплата аренды).

## Даталогическая модель

При создании данной модели необходимо было сопоставить сущностям из инфологической модели таблицы из даталогической модели, а также соблюсти требования, соответствующие третьей нормальной форме базы данных. В частности, сущность Schedule стала совокупностью таких таблиц, как schedule, film, genre, percent\_list. Атрибут Position сущности Employee стал самостоятельной таблицей, хранящей данные о должностях сотрудников, аналогично стал отдельной таблицей атрибут Workplace сущности Shift. Остальные таблицы соответствуют одноименным сущностям из инфологической модели. Данная схема была создана с помощью CASE-средства от Erwin Data Modeler.

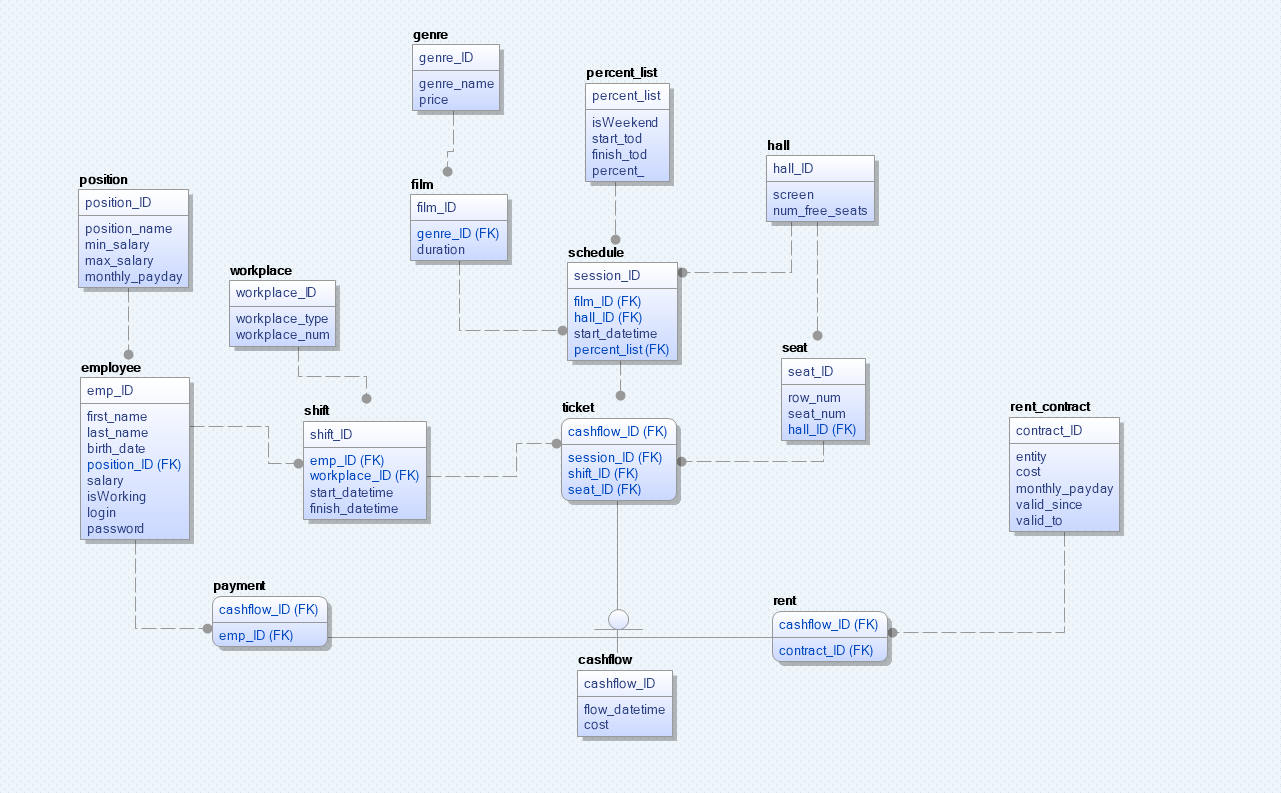


Рисунок 2. Даталогическая модель

### Описание таблиц

Таблица 1. Описание таблиц

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Таблица | Атрибут | Тип данных | Описание | PK/FK |
| genre | genre\_ID | int | Идентификатор жанра | Primary key |
| genre\_name | nvarchar(30) | Название жанра |  |
| price | money | Цена на фильм определенного жанра |  |
| film | film\_ID | int | Идентификатор фильма | Primary key |
| film\_name | nvarchar(50) | Название фильма |  |
| genre\_ID | int | Идентификатор жанра | Foreign key |
| duration | time(7) | Продолжительность фильма |  |
| hall | hall\_ID | int | Номер зала | Primary key |
| screen | nchar(2) | Тип экрана в зале |  |
| num\_of\_seats | int | Количество мест в зале |  |
| seat | seat\_ID | int | Идентификатор места в зале | Primary key |
| row\_num | int | Номер ряда |  |
| seat\_num | int | Номер места в ряду |  |
| hall\_ID | int | Идентификатор зала | Foreign Key |
| percent\_list | percent\_ID | int | Идентификатор наценки | Primary key |
| isWeekend | bit | Является ли день сеанса выходным днем |  |
| start\_tod | time(7) | Время дня, с которого начинается конкретная наценка |  |
| finish\_tod | time(7) | Время дня, в котором заканчивается конкретная наценка |  |
| percent\_ | decimal(3,2) | Значение наценки на стоимость проката фильма |  |
| schedule | session\_ID | int | Идентификатор сеанса в расписании | Primary key |
| film\_ID | int | Идентификатор фильма | Foreign key |
| hall\_ID | int | Идентификатор зала | Foreign Key |
| percent\_ID | int | Идентификатор наценки | Foreign Key |
| start\_datetime | datetime | Дата и время начала сеанса |  |
| position | position\_ID | int | Идентификатор должности | Primary key |
| position\_name | nvarchar(20) | Название должности |  |
| min\_salary | money | Минимальная зарплата сотрудника, занимающего конкретную должность |  |
| max\_salary | money | Максимальная зарплата сотрудника, занимающего конкретную должность |  |
| monthly\_payday | int | День месяца выплаты зарплаты |  |
| employee | emp\_ID | int | Идентификатор сотрудника | Primary key |
| first\_name | nvarchar(50) | Имя сотрудника |  |
| last\_name | nvarchar(50) | Фамилия сотрудника |  |
| birth\_date | date | Дата рождения сотрудника |  |
| position\_ID | int | Идентификатор должности сотрудника | Foreign Key |
| salary | money | Зарплата сотрудника |  |
| login\_ | nvarchar(20) | Логин сотрудника для входа в приложение |  |
| password\_ | nchar(5) | Пароль сотрудника для входа в приложение |  |
| isWorking | bit | Работает ли сотрудник или же уволен |  |
| workplace | workplace\_ID | int | Идентификатор рабочего места | Primary key |
| workplace\_type | nvarchar(20) | Тип рабочего места |  |
| workplace\_num | int | Номер рабочего места |  |
| shift | shift\_ID | int | Идентификатор смены сотрудника | Primary key |
| workplace\_ID | int | Идентификатор рабочего места | Foreign key |
| emp\_ID | int | Идентификатор сотрудника | Foreign key |
| start\_datetime | datetime | Дата и время начала смены |  |
| finish\_datetime | datetime | Дата и время окончания смены |  |
| rent\_contract | contract\_ID | int | Идентификатор контракта на аренду | Primary key |
| entity | nvarchar(30) | Название сущности, по которой выплачивается аренда |  |
| cost | money | Месячная стоимость аренды |  |
| monthly\_payday | int | День месяца уплаты аренды |  |
| cashflow | cashflow\_ID | int | Идентификатор денежной транзакции | Primary key |
| cf\_type | nvarchar(20) | Тип денежной транзакции |  |
| flow\_datetime | datetime | Дата и время денежной транзакции |  |
| cost | money | Сумма денежной транзакции |  |
| ticket | cashflow\_ID | int | Идентификатор денежной транзакции, связанной с билетом | Primary Key & Foreign Key |
| session\_ID | int | Идентификатор сеанса | Foreign Key |
| seat\_ID | int | Идентификатор места в зале | Foreign Key |
| shift\_ID | int | Идентификатор смены, на которой был продан билет | Foreign Key |
| payment | cashflow\_ID | int | Идентификатор денежной транзакции, связанной с выплатой зарплаты | Primary Key & Foreign Key |
| emp\_ID | int | Идентификатор сотрудника, которому выплачивают зарплату | Foreign Key |
| rent | cashflow\_ID | int | Идентификатор денежной транзакции, связанной с оплатой аренды | Primary Key & Foreign Key |
| contract\_ID | int | Идентификатор контракта, по которому оплачивается аренда | Foreign Key |

### Описание связей между таблицами

Таблица 2. Описание связей

|  |  |  |  |
| --- | --- | --- | --- |
| Связь | Отношение | Родительская таблица | Дочерняя таблица |
| cashflow\_payment | один к одному | cashflow | payment |
| cashflow\_rent | один к одному | cashflow | rent |
| cashflow\_ticket | один к одному | cashflow | ticket |
| contract\_rent | один ко многим | rent\_contract | rent |
| employee\_payment | один ко многим | employee | payment |
| employee\_shift | один ко многим | employee | shift |
| film\_schedule | один ко многим | film | schedule |
| genre\_film | один ко многим | genre | film |
| hall\_schedule | один ко многим | hall | schedule |
| hall\_seat | один ко многим | hall | seat |
| percent\_schedule | один ко многим | percent\_list | schedule |
| position\_employee | один ко многим | position | employee |
| schedule\_ticket | один ко многим | schedule | ticket |
| seat\_ticket | один ко многим | seat | ticket |
| shift\_ticket | один ко многим | shift | ticket |
| workplace\_shift | один ко многим | workplace | shift |

1. cashflow\_payment – каждая транзакция соответствует определенной выплате зарплаты. Идентифицирующая связь.
2. cashflow\_rent - каждая транзакция соответствует определенной уплате аренды. Идентифицирующая связь.
3. cashflow\_ticket - каждая транзакция соответствует определенному проданному билету. Идентифицирующая связь.
4. contract\_rent – по каждому контракту могут быть выплачены деньги неоднократно. Не идентифицирующая связь.
5. employee\_payment – каждому сотруднику может быть выплачена зарплата неоднократно, зарплата выплачивается конкретному сотруднику. Не идентифицирующая связь.
6. employee\_shift - каждому сотруднику может быть назначена смена неоднократно, одна смена предназначена конкретному сотруднику. Не идентифицирующая связь.
7. film\_schedule – фильм может быть показан неоднократно, на одном киносеансе показывают один фильм. Не идентифицирующая связь.
8. genre\_film – фильм имеет определенный жанр, может быть много фильмов одного жанра. Не идентифицирующая связь.
9. hall\_schedule – киносеанс проходит в конкретном зале, в каждом зале прокат может быть неоднократно. Не идентифицирующая связь.
10. hall\_seat – место находится в определенном зале, в зале множество мест. Не идентифицирующая связь.
11. percent\_schedule – у киносеанса одна наценка за прокат, одна и та же наценка может быть у множества сеансов. Не идентифицирующая связь.
12. position\_employee – у сотрудника конкретная должность, может быть множество сотрудников одной должности. Не идентифицирующая связь.
13. schedule\_ticket – билет продается на конкретный сеанс, на один сеанс можно продать множество билетов. Не идентифицирующая связь.
14. seat\_ticket - билет продается на конкретное место, на одно место можно продать множество билетов (в разные сеансы). Не идентифицирующая связь.
15. shift\_ticket – билет продается на определенной смене, за одну смену можно продать множество билетов. Не идентифицирующая связь.
16. workplace\_shift – смена проходит на конкретном рабочем месте, которое может быть задействовано на множестве смен. Не идентифицирующая связь.

# Реализация базы данных

При построении диаграммы базы данных в MS SQL Server была использована функция «Schema Generation» из Erwin Data Modeler, генерирующая скрипт для ее создания. Содержимое скрипта было перенесено в запрос к конкретной базе данных Cinema, также были вручную проставлены ограничения IDENTITY (1, 1) для первичных ключей (кроме таблиц ticket, rent и payment т. к. они являются подтипами таблицы cashflow), чтобы снять с пользователей задачу в их мануальном заполнении.



Рисунок 3. Создание запроса на основе скрипта

Также была создана диаграмма в MS SQL Server для проверки результата выполнения скрипта.

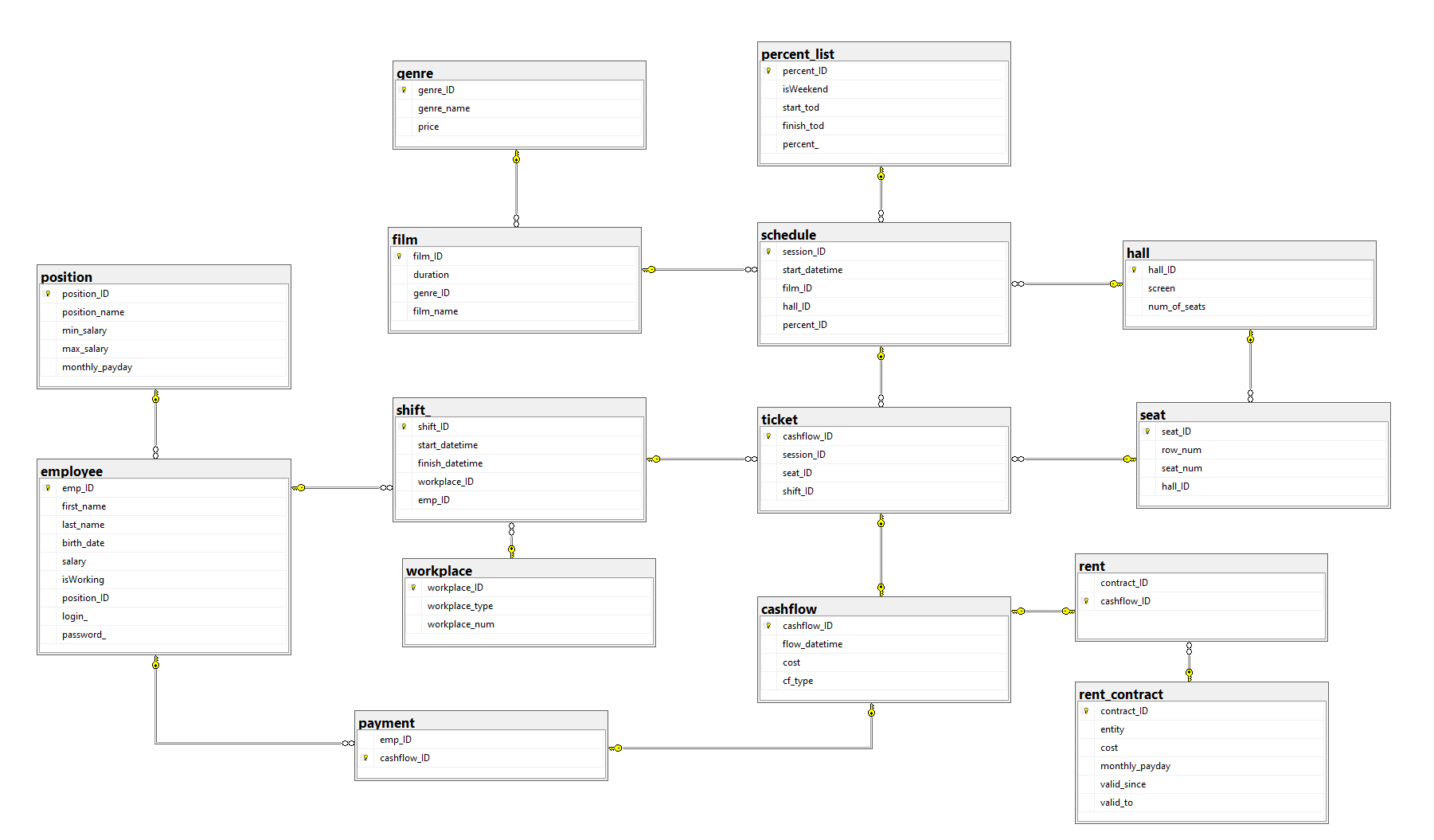


Рисунок 4. Диаграмма базы данных, созданная в MS SQL Server

## Заполнение начальными данными

На данном этапе работы таблицы были заполнены значениями с помощью оператора INSERT. Таблицы schedule, shift, ticket, payment, а также rent остались без изменений, поскольку для их заполнения желательно пользоваться триггерами и хранимыми процедурами для обеспечения корректности ввода данных и автоматического определения ряда атрибутов. Результаты их выполнения будут продемонстрированы позже.

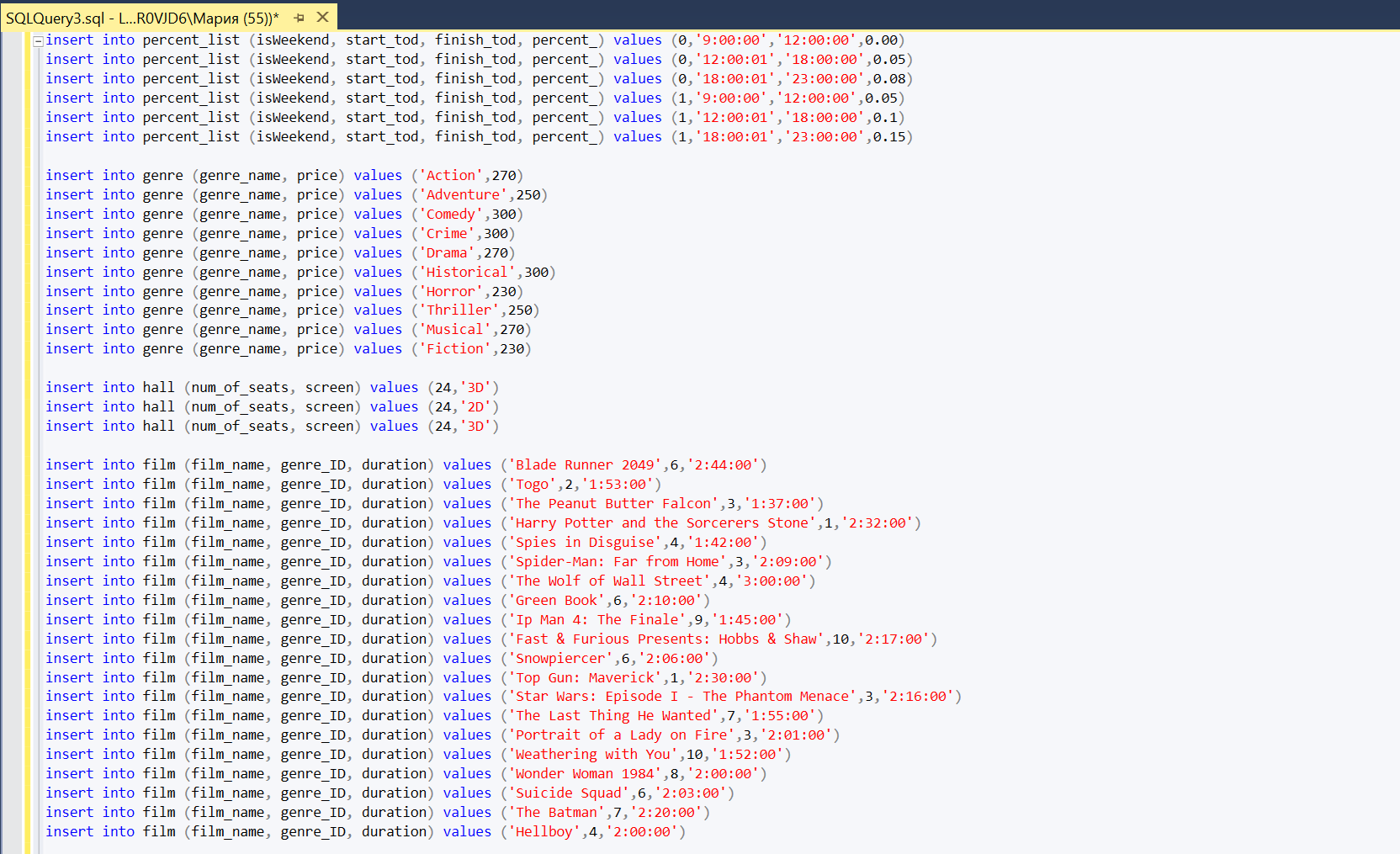


Рисунок 5. Пример заполнения таблиц.

Результат запроса был проверен при рассмотрении строк в таблице film.



Рисунок 6. Проверка вставки данных в таблицу film.

## Функции

Ниже приведены функции, которые призваны ускорить и упростить работу пользователя с базой данных.

1. При обслуживании клиента кассиру необходимо знать, какие места на желаемый сеанс доступны для продажи билетов. Оптимизировать этот процесс ему поможет функция check\_free\_seats. Позже в интерфейсе приложения данная функция будет подкрепляться визуальной составляющей и ее не потребуется вызывать мануально.

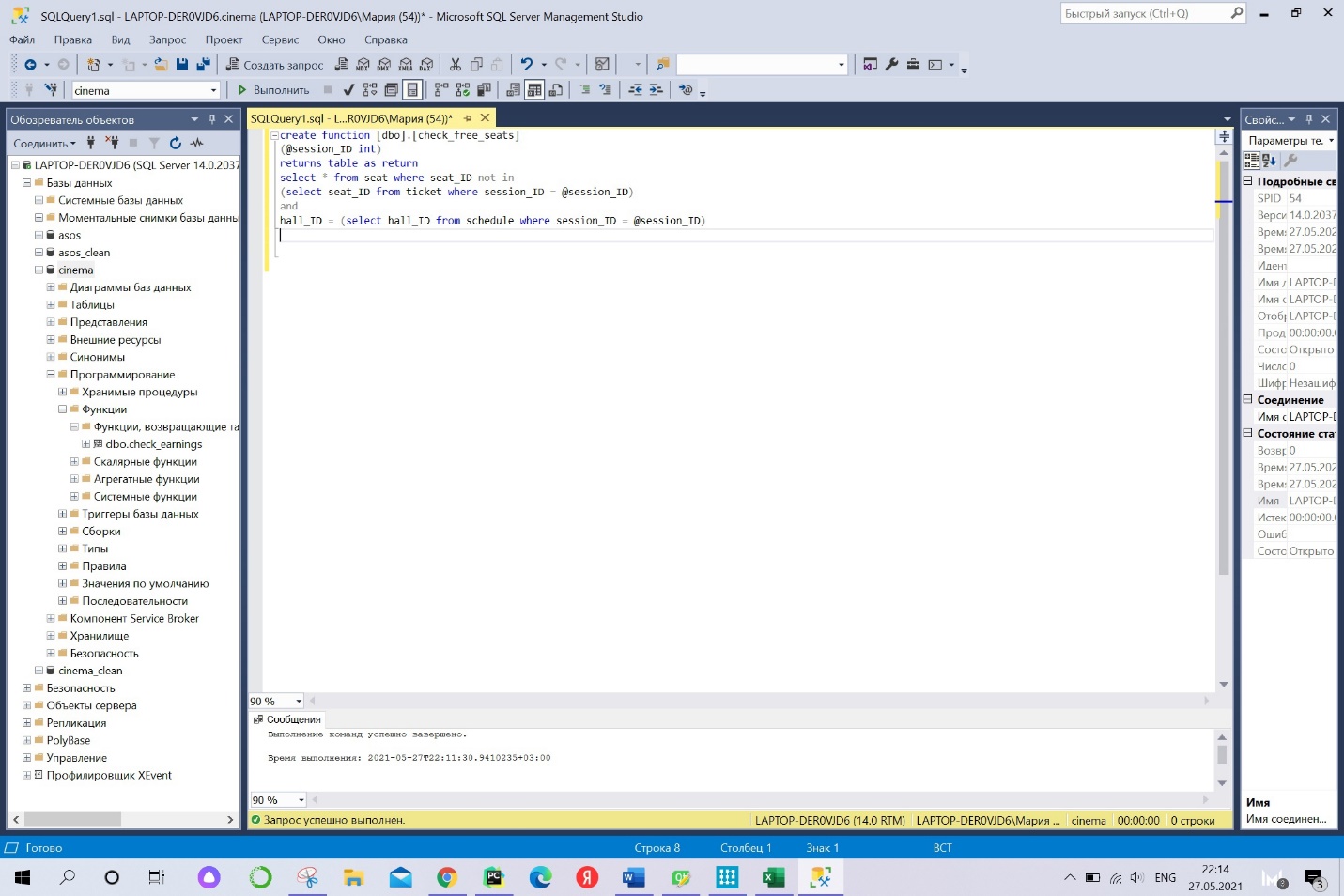


Рисунок 7. Код функции check\_free\_seats

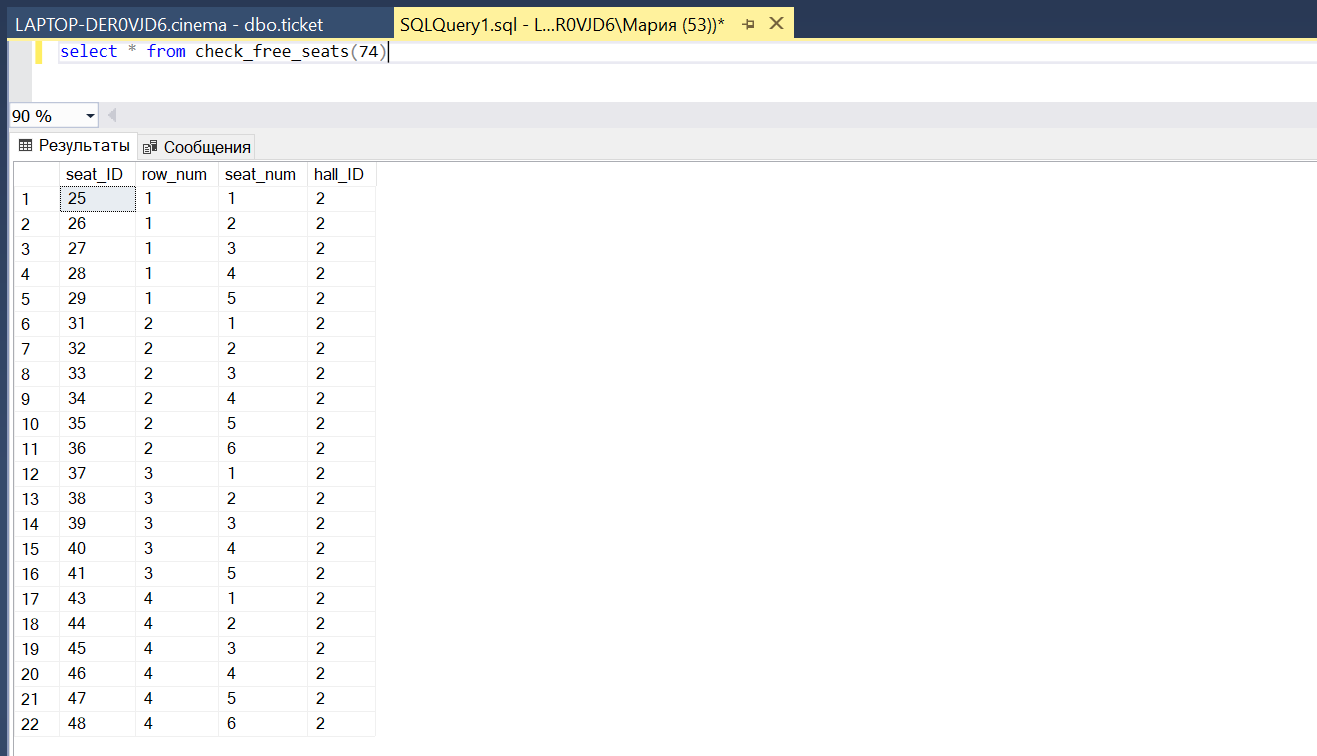


Рисунок 7.1. Результат выполнения функции check\_free\_seats

1. Для администратора кинотеатра необходимо всегда контролировать прибыльность бизнеса. Ему важно понимать, окупает ли продажа билетов издержки на содержание кинотеатра, состоящие из аренды за помещение и оборудование, а также выплаты зарплаты сотрудникам.

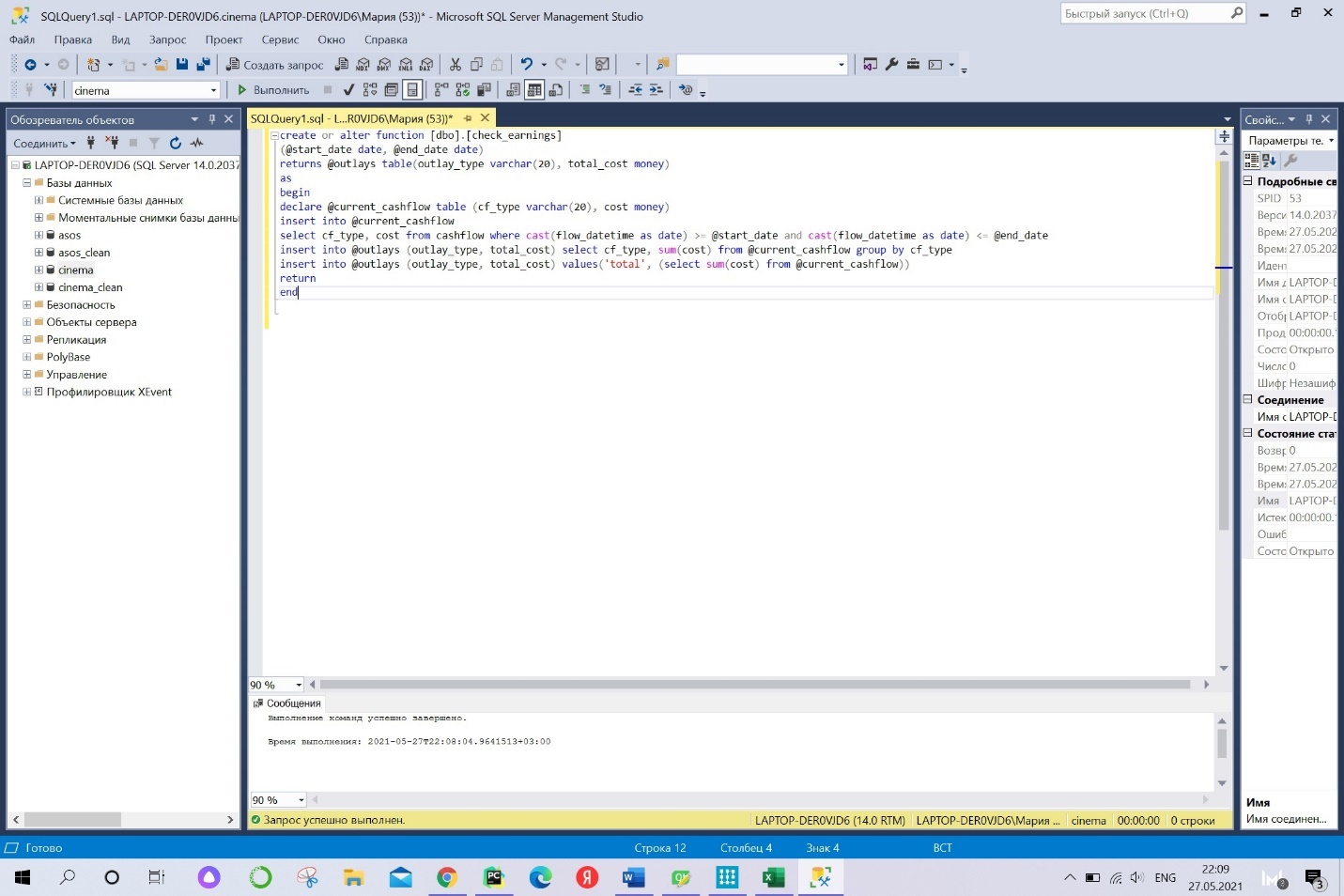


Рисунок 8. Код функции check\_earnings

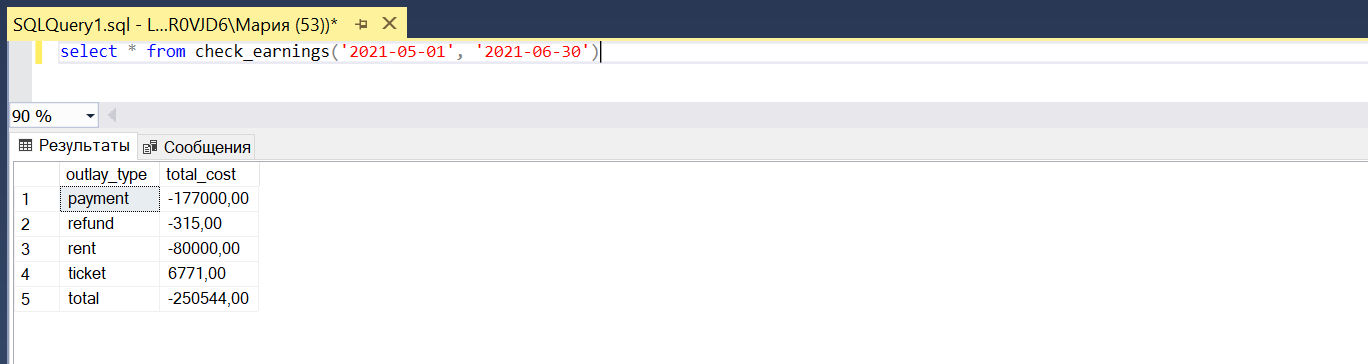


Рисунок 8.1. Результат выполнения функции check\_earnings

1. Для своевременной оплаты труда сотрудников администратором была создана функция is\_paid показывающая дату и факт выплаты (либо же его отсутствие) на текущий месяц.

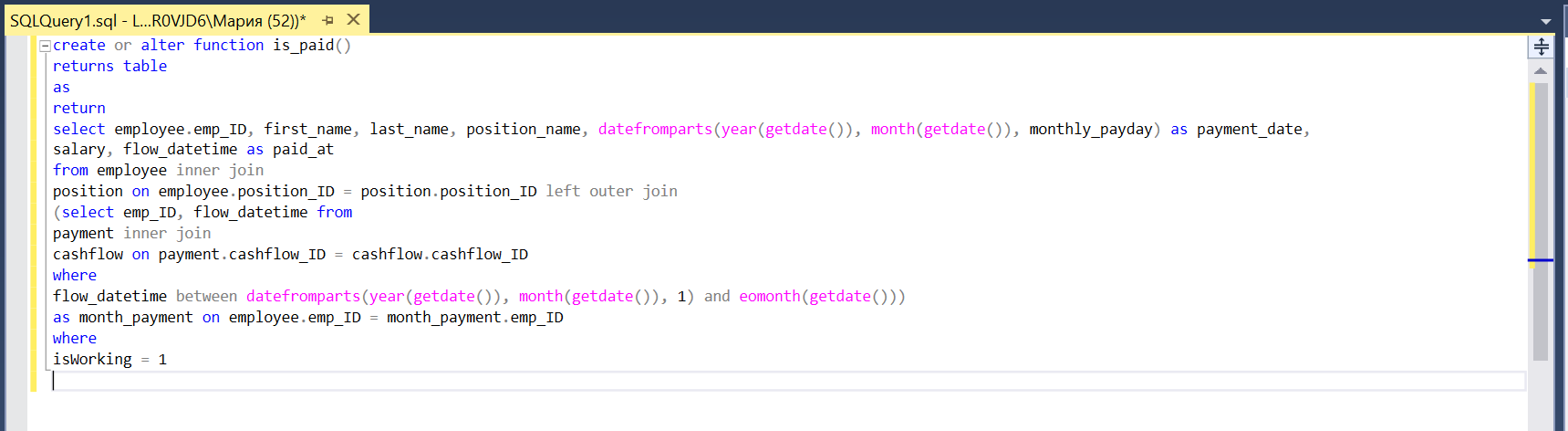


Рисунок 9. Код функции is\_paid

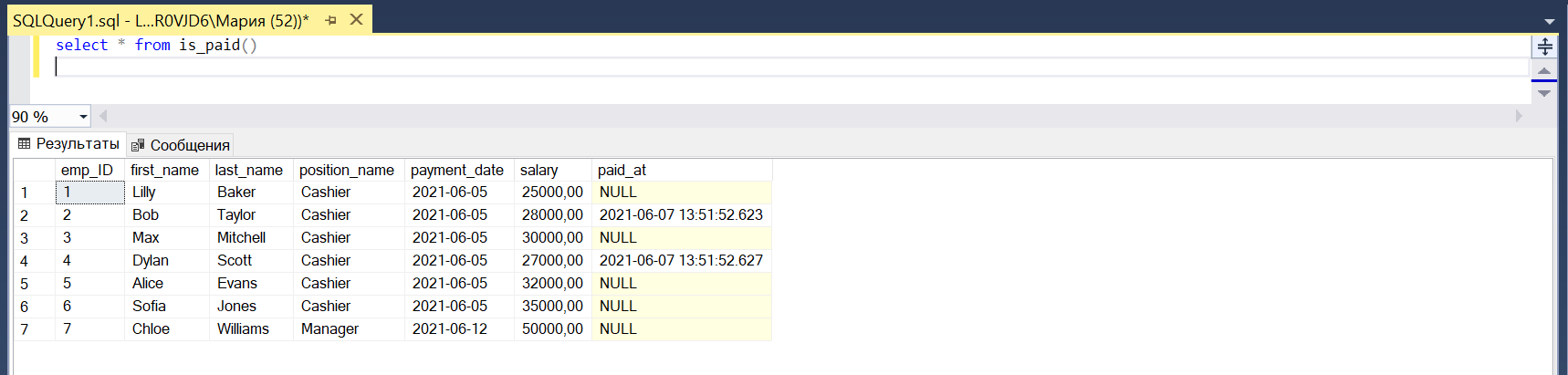


Рисунок 9.1. Результат выполнения функции is\_paid

## Триггеры

В данной работе триггеры служат для двух вещей. Первая из них – фиксирование (или логирование) каких-либо денежных транзакций, вторая – проверка корректности ввода данных.

1. Логирование выплат зарплат. В базе данных сохраняется информация о дате и времени проведения транзакции, а также ее стоимость, что повышает прозрачность ведения бизнеса для администратора.

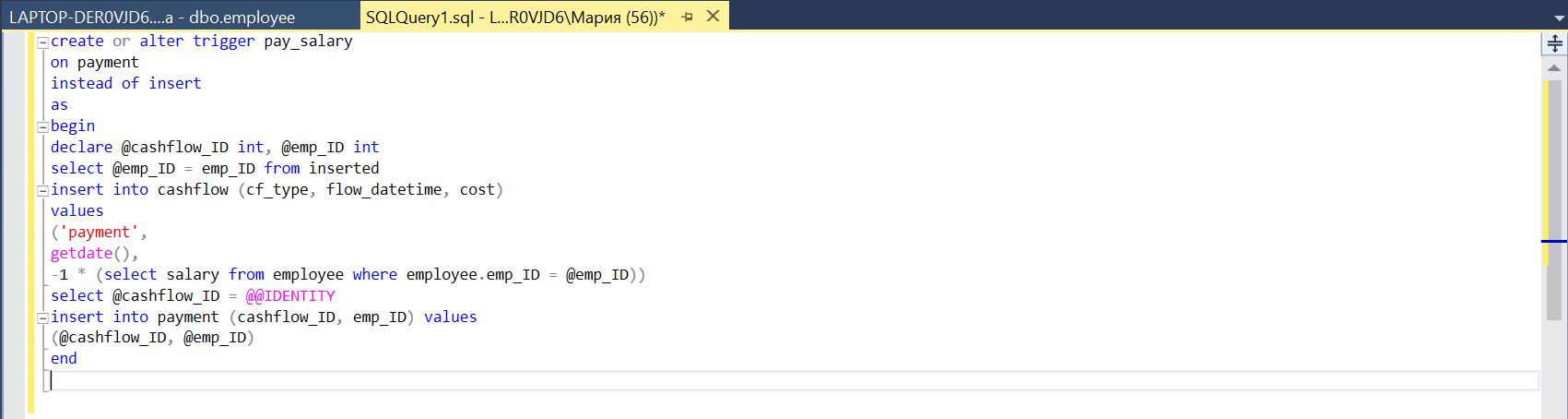


Рисунок 10. Код триггера pay\_salary

Проверка корректности кода триггера была осуществлена с помощью вставки строки в таблицу payment.

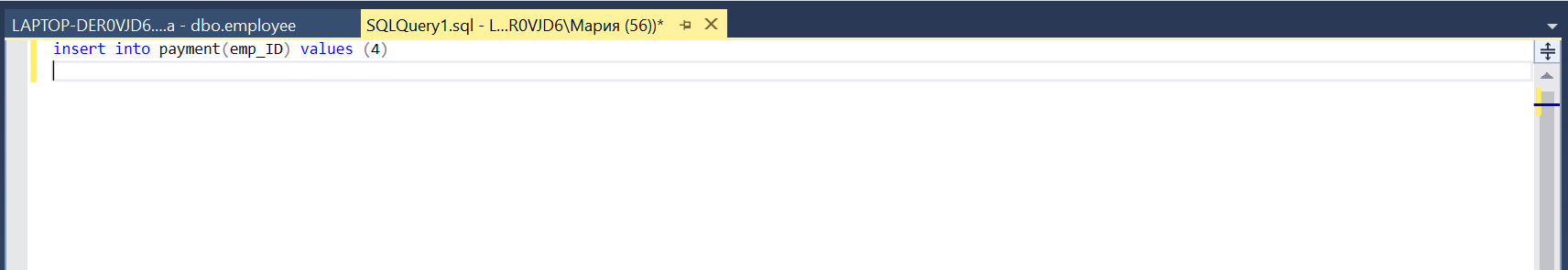


Рисунок 10.1. Запрос с использованием триггера pay\_salary

После исполнения запроса были зафиксированы изменения в таблицах cashflow и payment, а также результат вызова функции is\_paid.

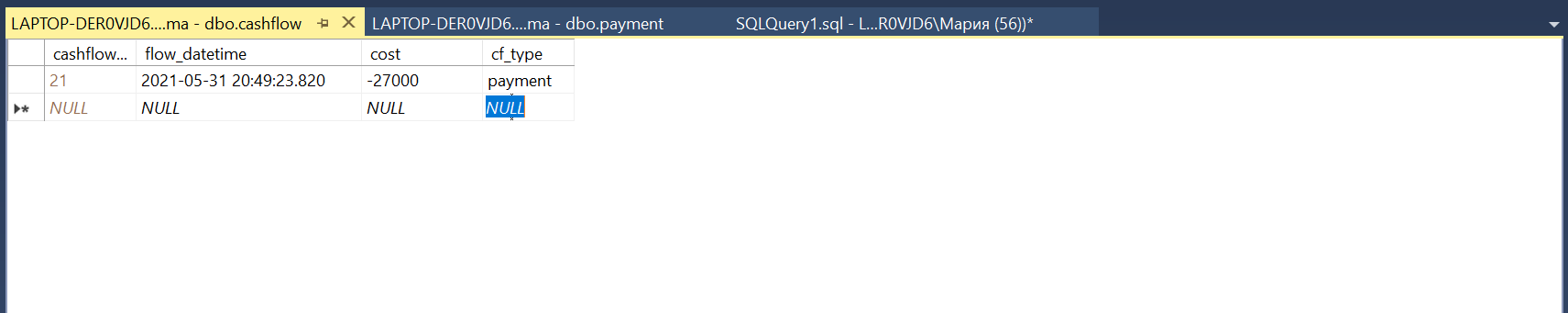


Рисунок 10.2. Результат выполнения запроса в таблице cashflow

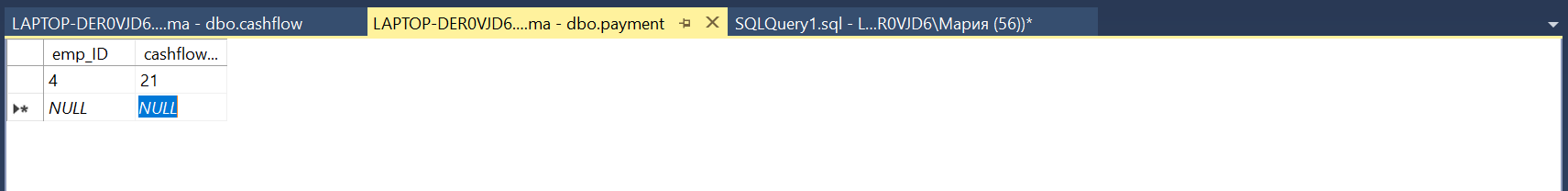


Рисунок 10.3. Результат выполнения запроса в таблице payment

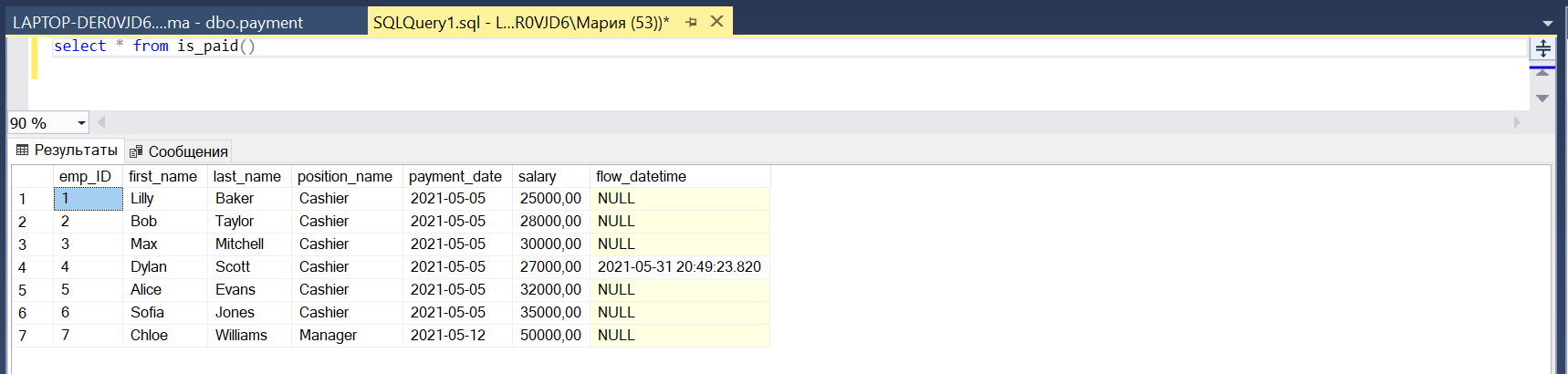


Рисунок 10.4. Результат выполнения функции is\_paid

1. Логирование уплаты аренды повторяет логику предыдущего триггера.

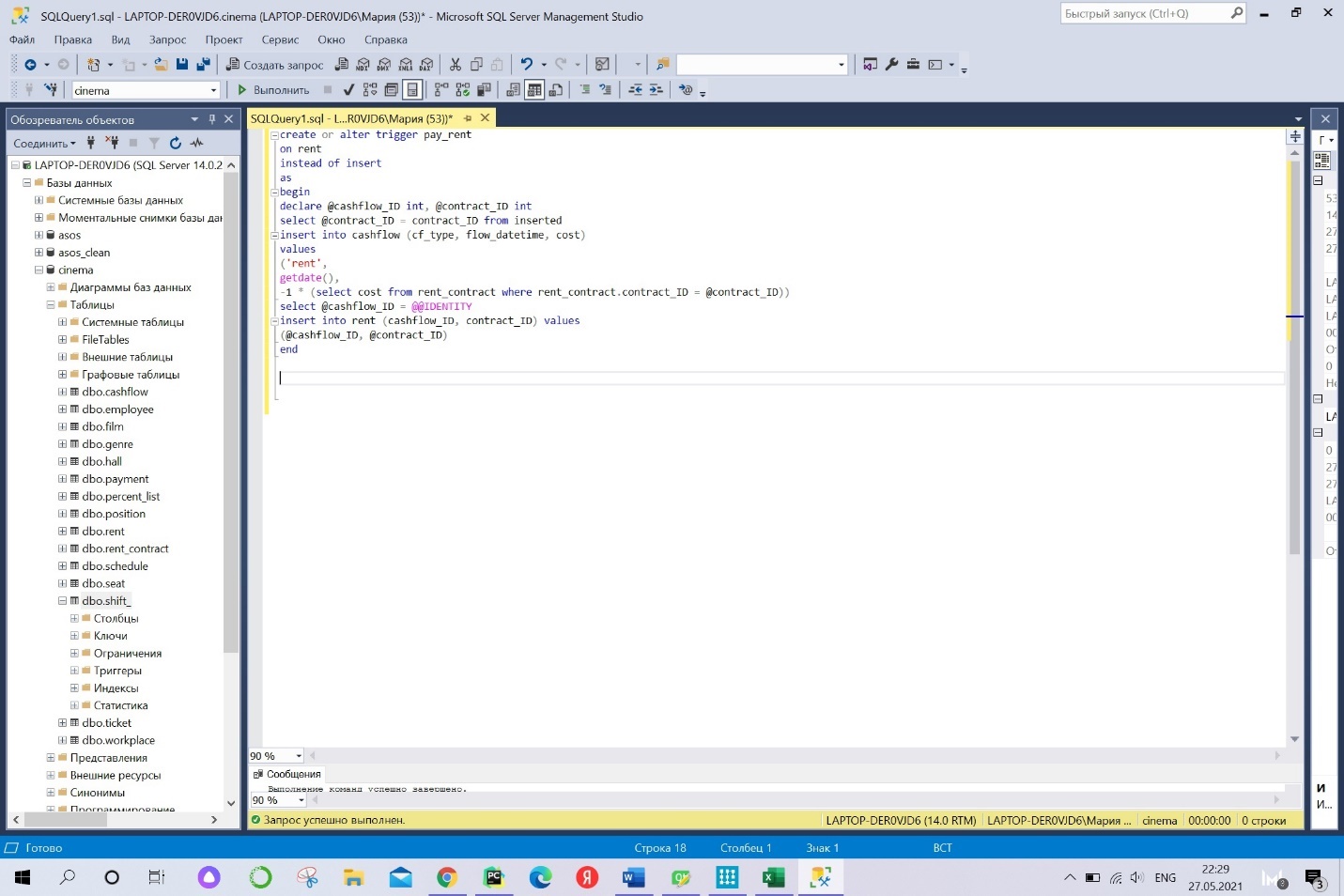


Рисунок 11. Код триггера pay\_rent

Проверка корректности кода триггера была осуществлена с помощью вставки строки в таблицу rent.

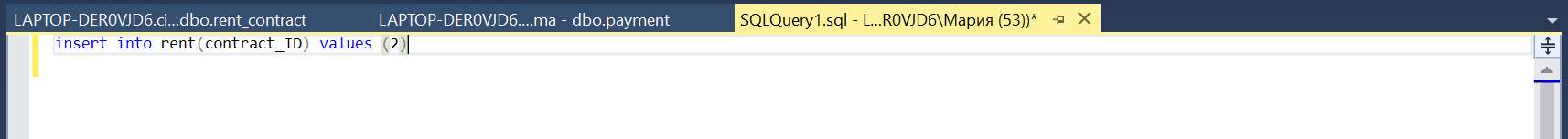


Рисунок 11.1. Запрос с использованием триггера pay\_rent

После исполнения запроса были зафиксированы изменения в таблицах cashflow и rent.



Рисунок 11.2. Результат выполнения запроса в таблице rent

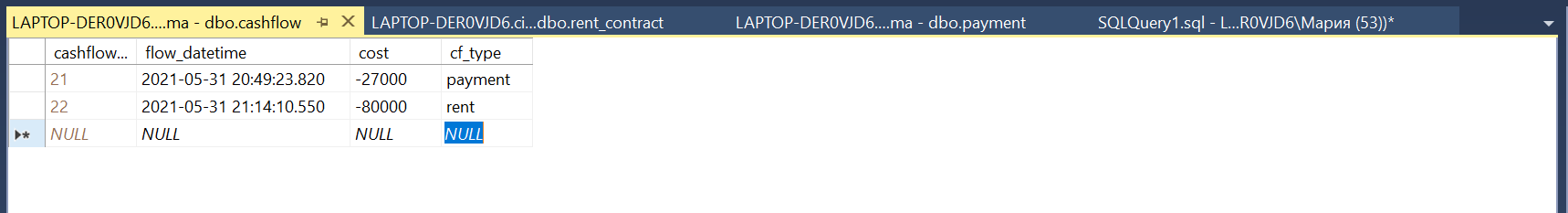


Рисунок 11.3. Результат выполнения запроса в таблице cashflow

1. В обязанности администратора также входит составление расписаний для кассиров. Данный триггер запускается при попытке выполнения оператора INSERT для новой строки, однако не позволит ее вставить пока вносимые данные не удовлетворят всем для этого необходимым условиям. А именно: касса не может быть занята двумя кассирами одновременно, один сотрудник не может иметь две и более смен в одно время, а также конец смены должен быть позднее ее начала.

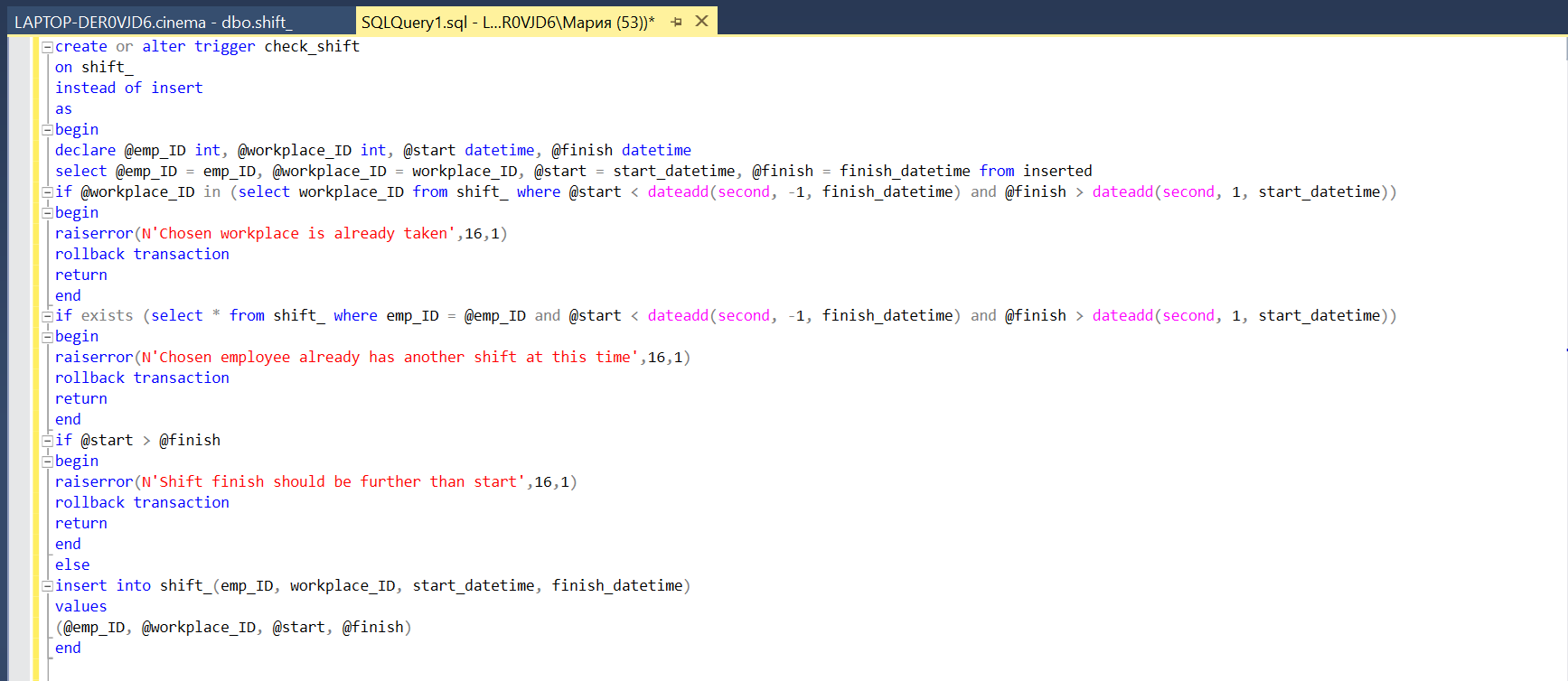


Рисунок 12. Код триггера check\_shift

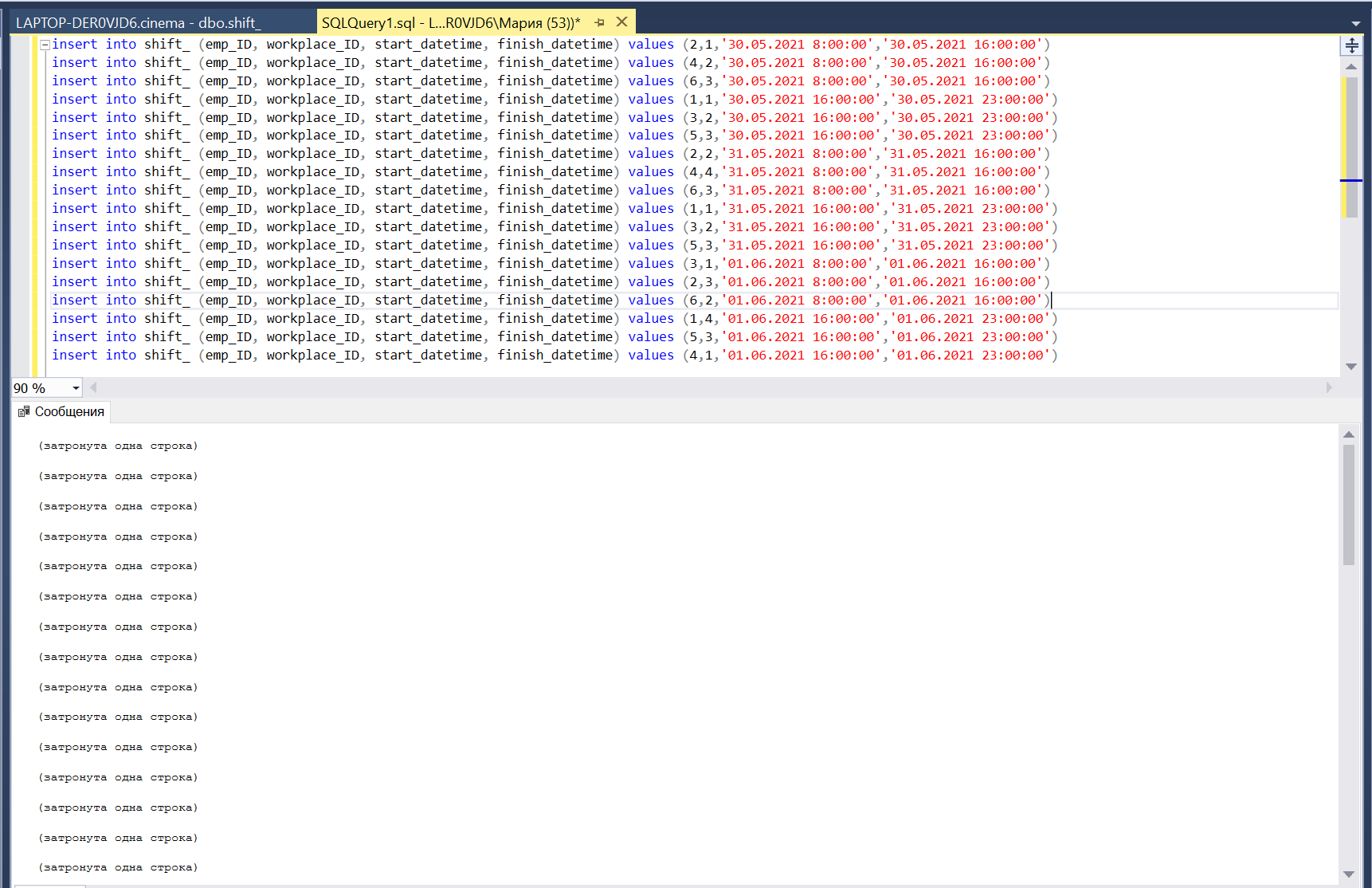
В качестве примера работы данной процедуры были заведены сразу множество строк с корректной информацией о сменах и две специально ошибочные (начало одной из смен стоит раньше конца другой смены на той же кассе, и один человек записан на две пересекающиеся по времени смены)

Рисунок 12.1. Результат успешного выполнения запросов с использованием триггера check\_shift

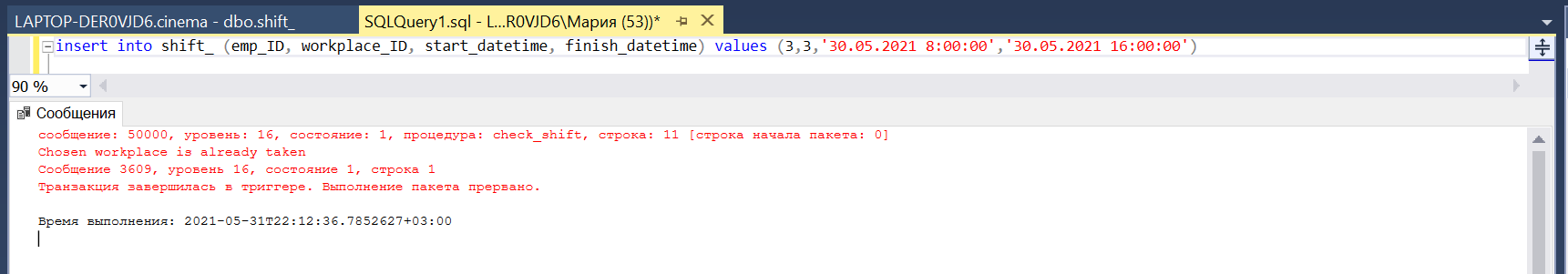


Рисунок 12.2. Пример невыполненного запроса

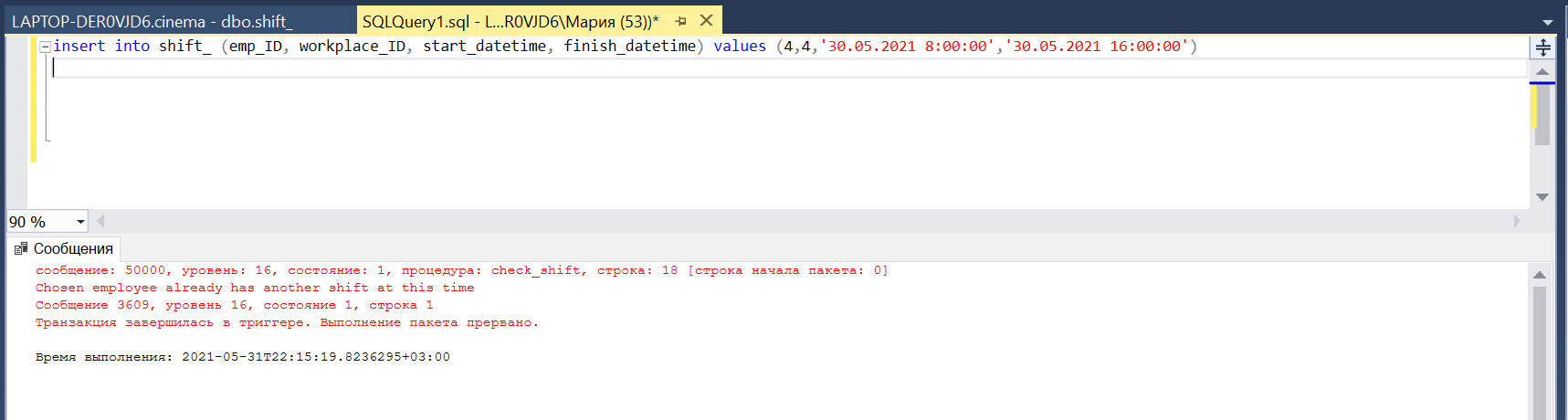


Рисунок 12.3. Пример невыполненного запроса

## Хранимые процедуры

В данной работе хранимые процедуры призваны доопределять некоторые атрибуты при вставке новой строки в ряде таблиц, а также проверять корректность ввода данных.

1. Кроме расписания смен администратору требуется составлять и расписание сеансов. При этом необходимо избежать накладок соседних киносеансов в одном зале. Более того, данная функция сама определяет наценку на тот или иной сеанс исходя из дня недели и времени суток проката, информацию о процентах она берет из таблицы percent\_list.



Рисунок 13. Код процедуры add\_session

В качестве примера работы данной процедуры было заведено сразу множество строк с корректной информацией о сеансах и одна специально ошибочная (начало одного из сеансов стоит раньше конца другого сеанса в том же зале).

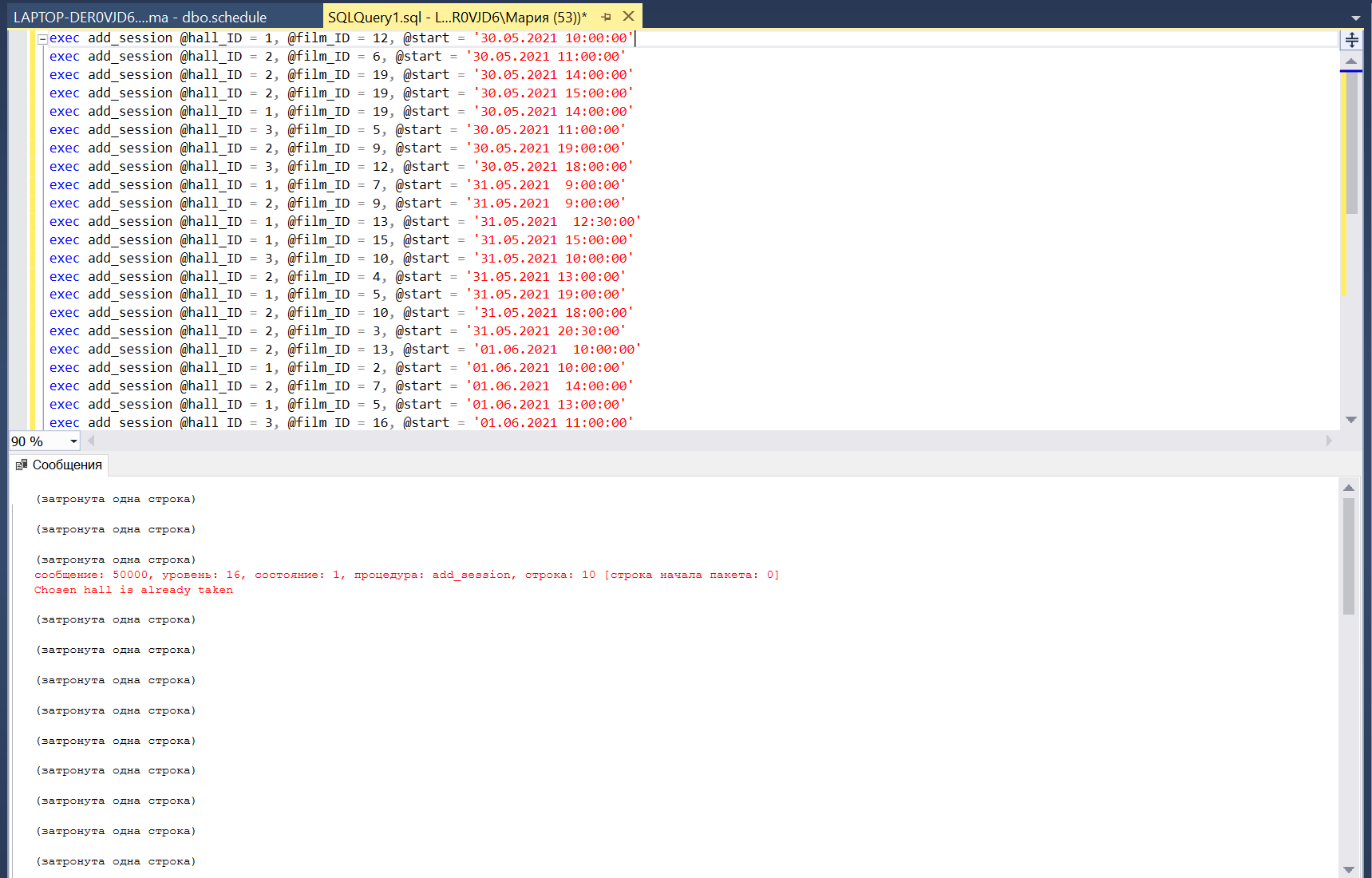


Рисунок 13.1. Результат выполнения запросов с применением процедуры add\_session

1. Процедуру ticket\_buying следует запускать кассирам при оформлении и продаже билета клиентам. При ее создании также использовалась вышеупомянутая функция check\_free\_seats. Данная процедура уберегает кассира от ввода уже ранее проданного на текущий сеанс места, а также не позволяет продать билет на уже начавшийся сеанс. Продажа билетов, как и остальные денежные потоки, фиксируется в таблице cashflow.



Рисунок 14. Код процедуры ticket\_buying

На данном этапе все места в зале свободны, в качестве параметра в процедуру было заведено 3-е место 3-го ряда (с идентификатором 15)

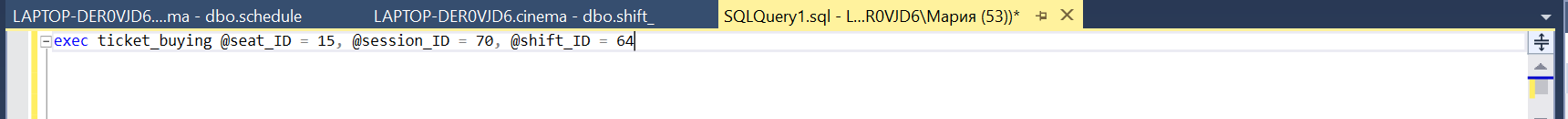


Рисунок 14.1. Запрос с использованием процедуры ticket\_buying

После исполнения запроса были зафиксированы изменения в таблицах ticket и cashflow.

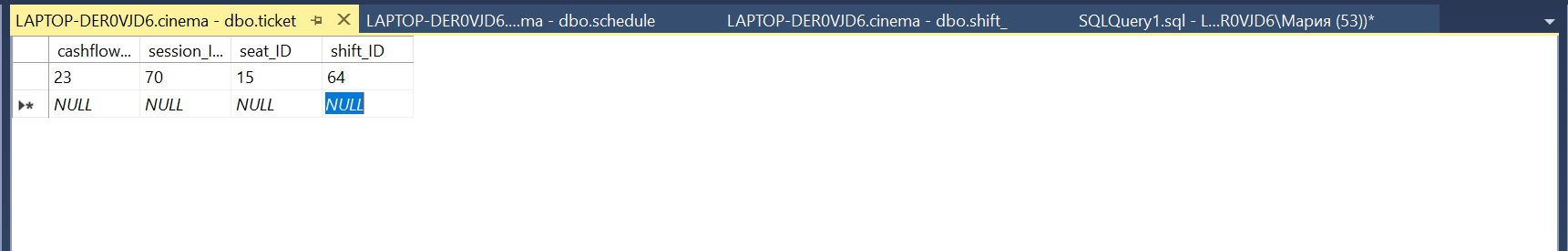


Рисунок 14.2. Результат выполнения запроса в таблице ticket

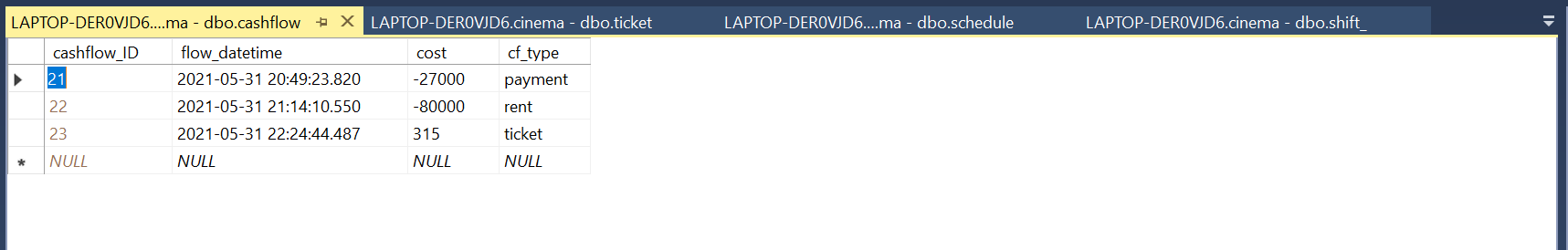


Рисунок 14.3. Результат выполнения запроса в таблице cashflow

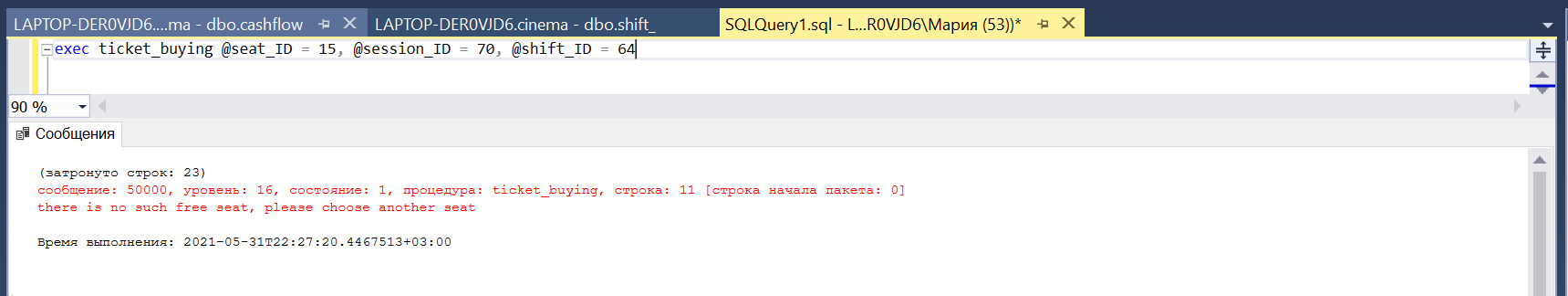
Также была произведена попытка продажи билета с указанием уже занятого места.

Рисунок 14.4. Пример невыполненного запроса

После вышеупомянутых манипуляций запрос с функцией check\_free\_seats выдает список всех мест кроме 3-го места 3-го ряда.

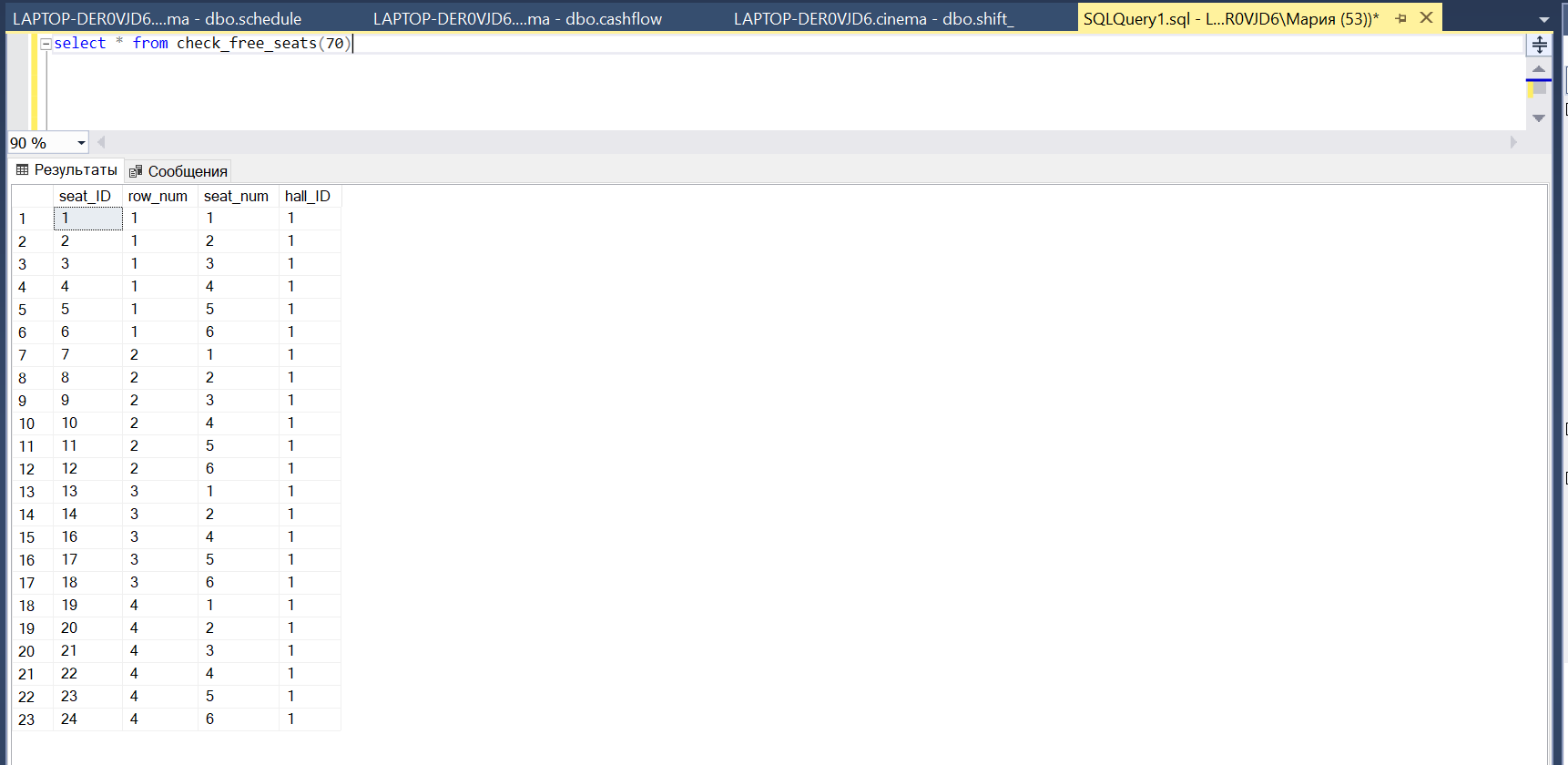


Рисунок 14.5. Результат выполнения запроса с применением функции check\_free\_seats

1. Нередко происходит так, что планы клиента могут поменяться и ему необходимо вернуть билет. Однако, такое возможно только если киносеанс еще не начался. Моделирование данной логики в рамках СУБД продемонстрировано в процедуре ticket\_return. При возврате билета кинотеатр также возвращает деньги, что заносится в таблицу cashflow.

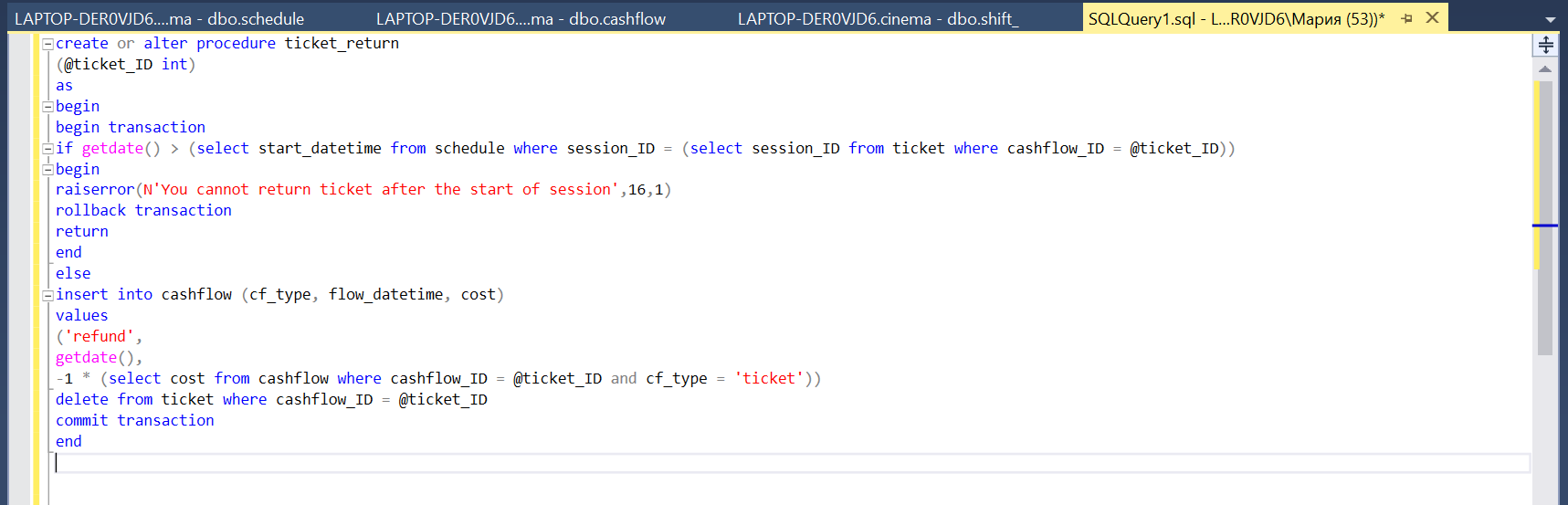


Рисунок 15. Код процедуры ticket\_return

Для показа работы процедуры был создан запрос, входным параметром которого является билет, созданный на прошлом шаге.

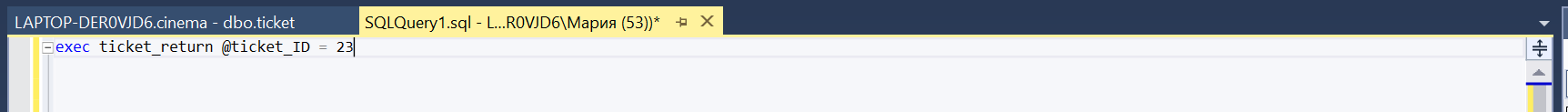


Рисунок 15.1. Запрос с использованием процедуры ticket\_return

После исполнения запроса были зафиксированы изменения в таблицах ticket и cashflow. Появилась новая строка с типом транзакции refund и суммой денег равной стоимости возвращенного билета, записи указанного в процедуре билета в таблице ticket нет.

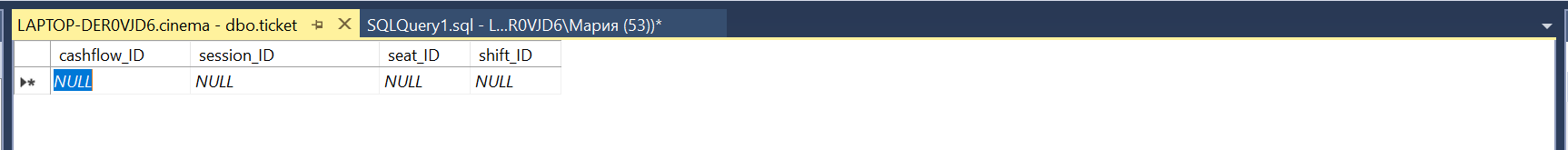


Рисунок 15.2. Результат выполнения запроса в таблице ticket

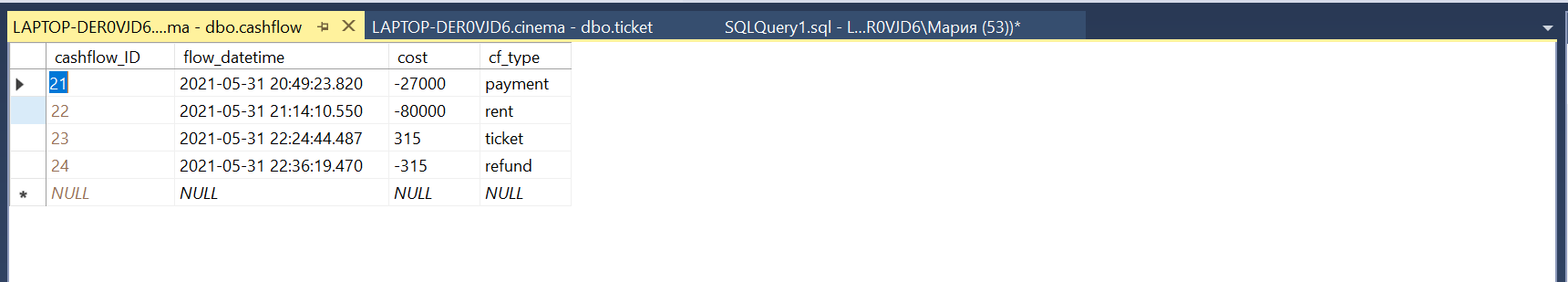


Рисунок 15.3. Результат выполнения запроса в таблице cashflow

Также была произведена попытка возвратить билет на прошедший киносеанс.

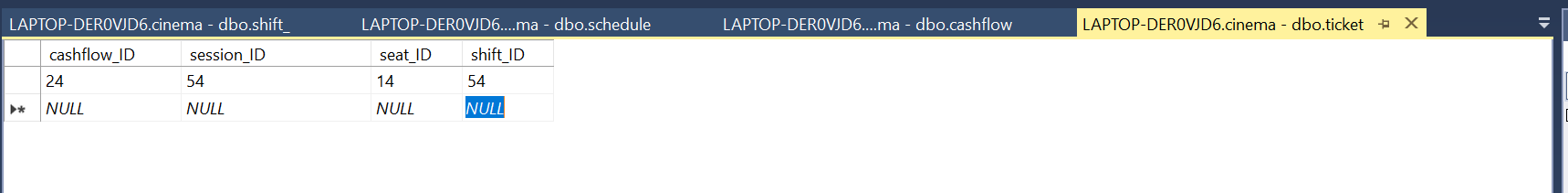


Рисунок 15.4. Строка таблицы ticket\_ID c билетом на прошедший сеанс

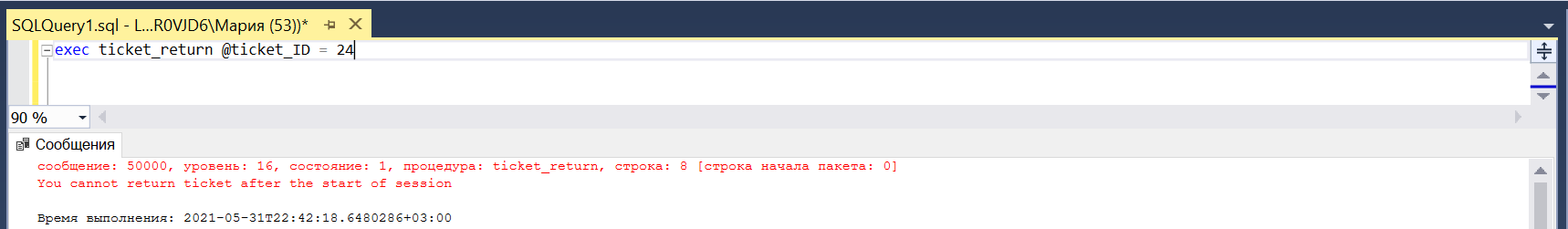


Рисунок 15.5. Пример невыполненного запроса

Для проверки того, как отразились манипуляции на текущем и предыдущих шагах на результате функции check\_earnings, был выполнен запрос с ее использованием.

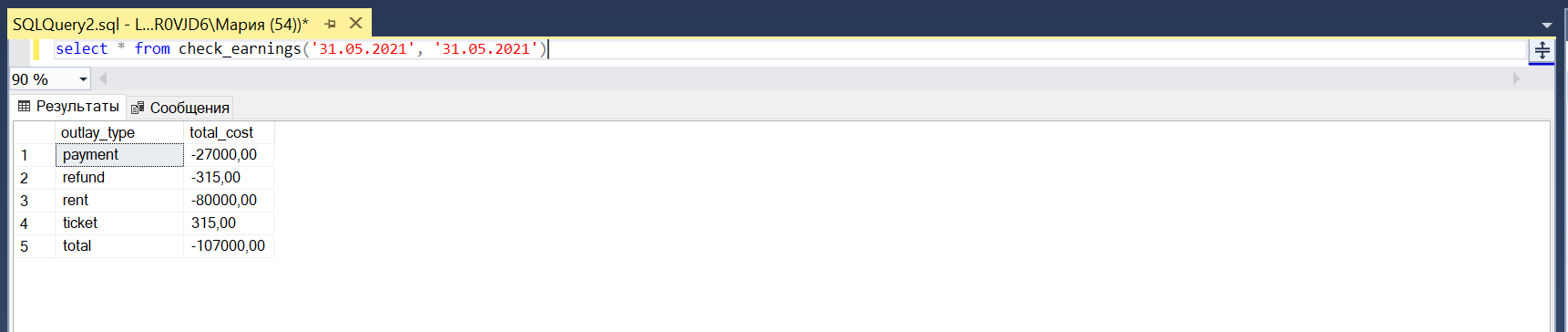


Рисунок 15.6. Результат выполнения запроса с применением функции check\_earnings

Также произошли изменения и в результате запроса с функцией check\_free\_seats: вновь все места стали свободными.

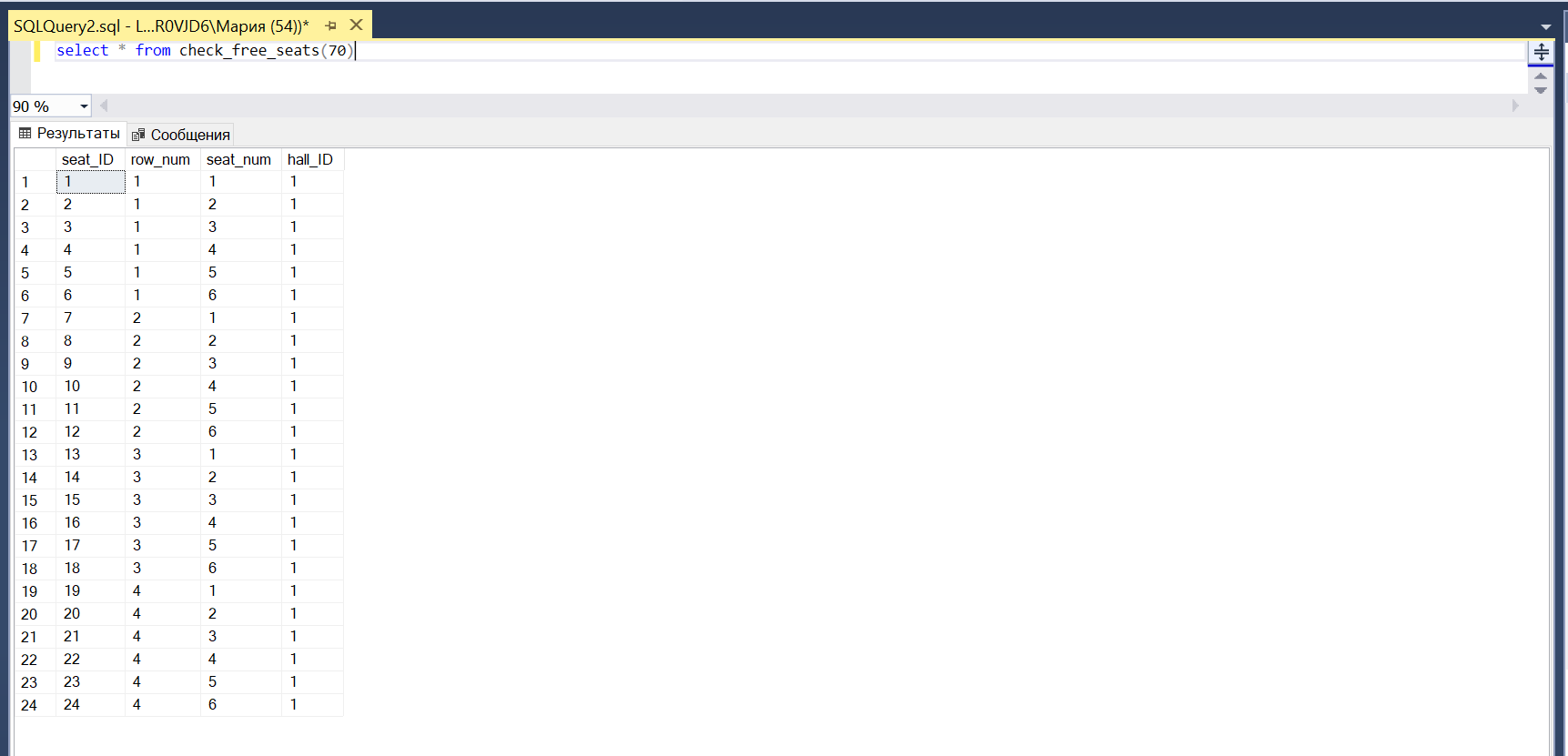


Рисунок 15.7. Результат выполнения запроса с применением функции check\_free\_seats

## Запросы

Итак, когда все данные корректно заполнены, из базы данных можно извлечь необходимую пользователю информацию.

1. Получение кассиром списка всех его будущих смен.

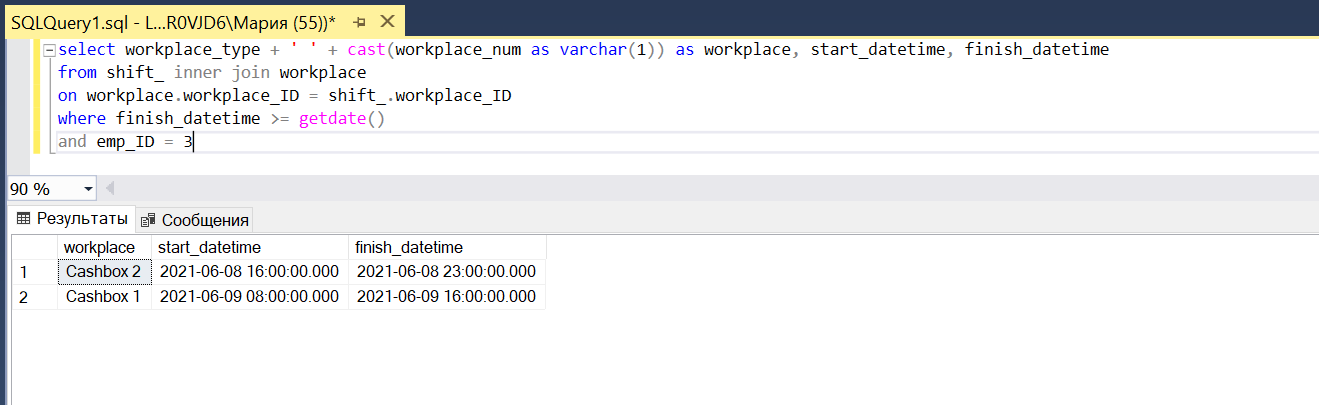


Рисунок 16. Результат выполнения запроса 1

1. Вывод полной информации о сеансах на текущий день.

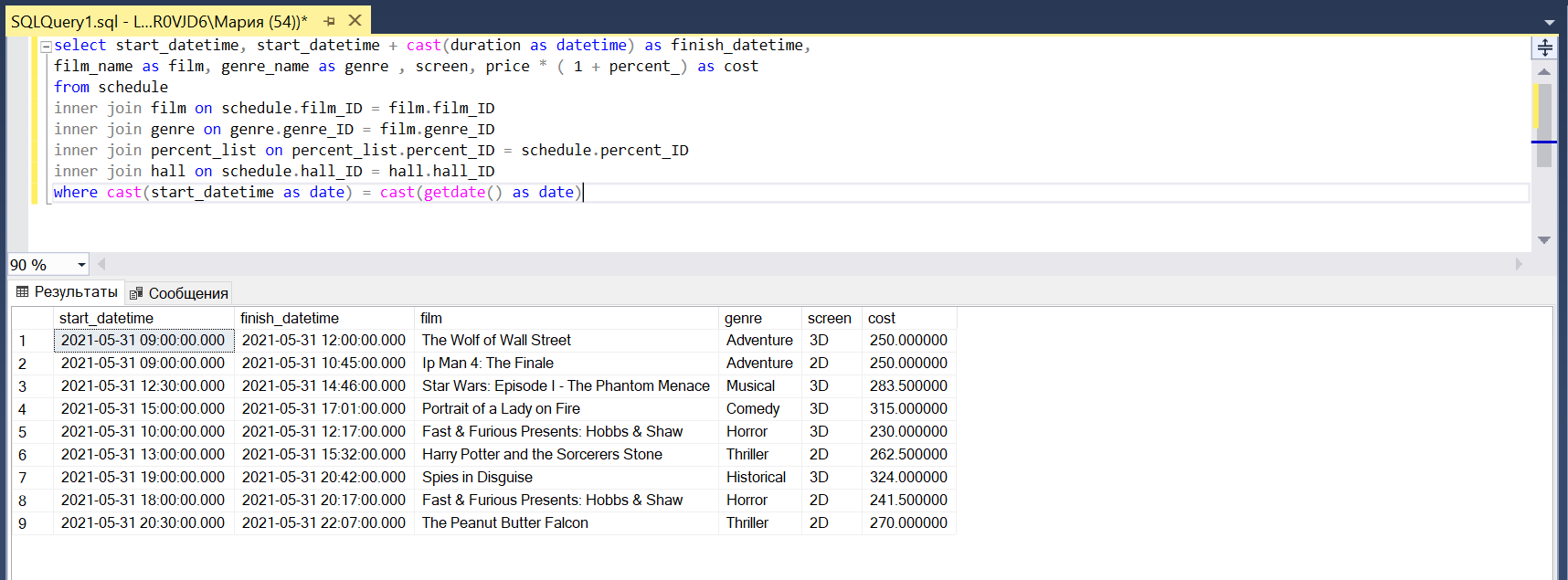


Рисунок 17. Результат выполнения запроса 2

1. Сопоставление издержек на сотрудников и денежной отдачи от их работы.

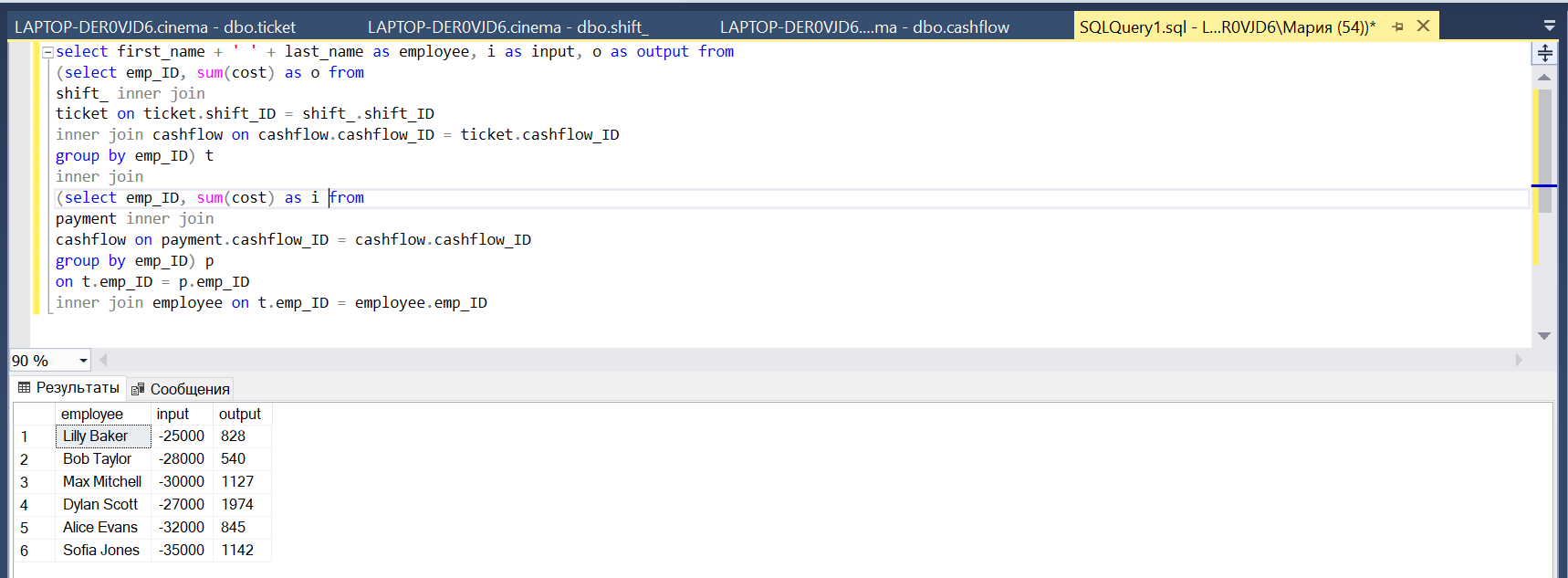


Рисунок 18. Результат выполнения запроса 3

1. Внутренний рейтинг фильмов по величине кассовых сборов.

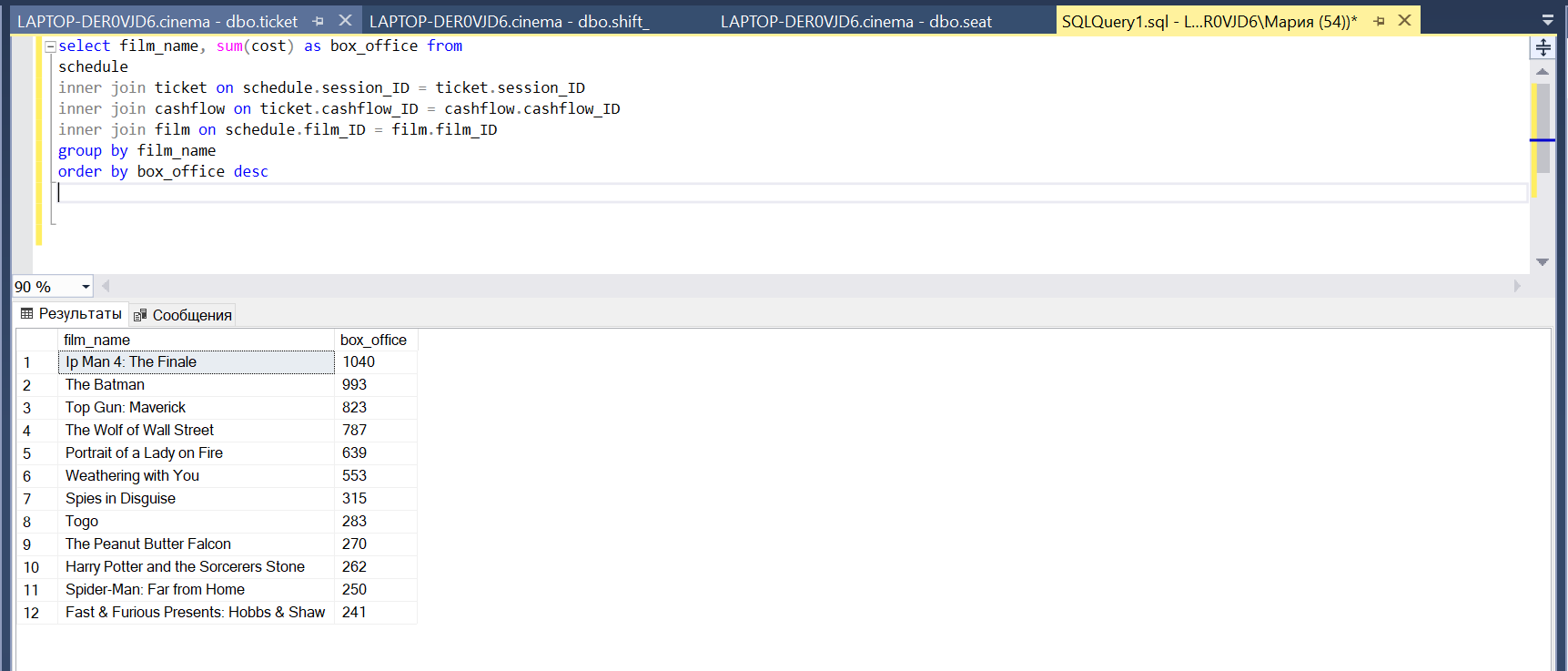


Рисунок 19. Результат выполнения запроса 4

1. Среднее количество проданных билетов в день за год.

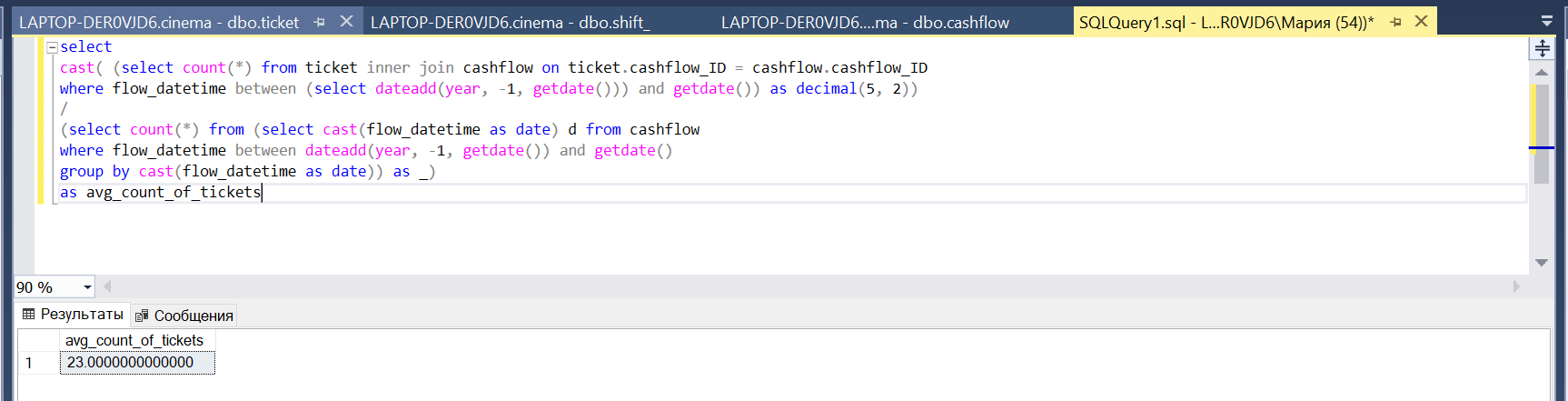


Рисунок 20. Результат выполнения запроса 5

# Разработка пользовательского приложения

Пользовательское приложение разрабатывалось на программном языке Python с использованием библиотек pyodbc для непосредственной работы с базой данных и PyQt5 для создания интерфейса.

Приложение имеет два вида функционала, каждый из которых соответствует типу пользователя (администратор или кассир). Данная программная разработка предоставит возможность администраторам мониторить и вносить новые записи сеансов и смен, а также контролировать сумму денежных потоков и выплачивать зарплаты сотрудникам. Для кассиров предоставлена возможность удобно и прозрачно предоставлять услуги клиентам, в которые входят продажа и возврат билетов, и ознакомляться со своими текущей и будущими сменами.

При запуске приложения всплывает окно аутентификации, оно и обеспечивает разделение функционала на две области.

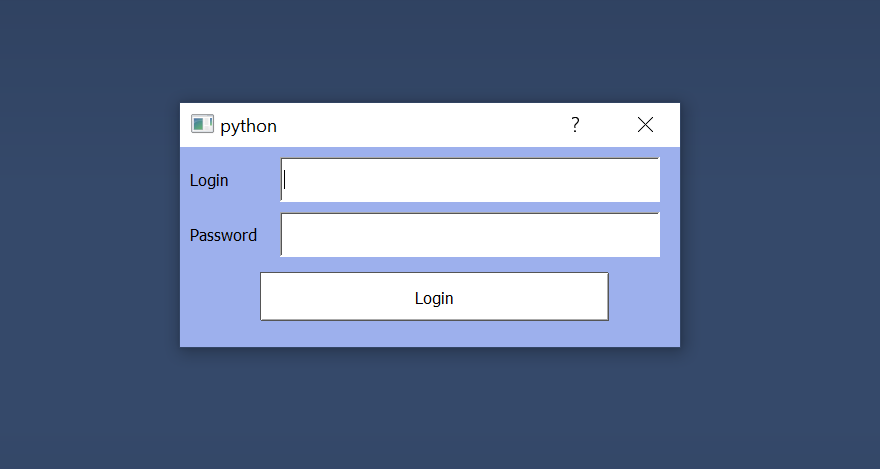


Рисунок 21. Окно аутентификации

При входе в приложение кассира всплывает окно с тремя вкладками: «Ticket Sell», «Ticket Refund», «Shifts».

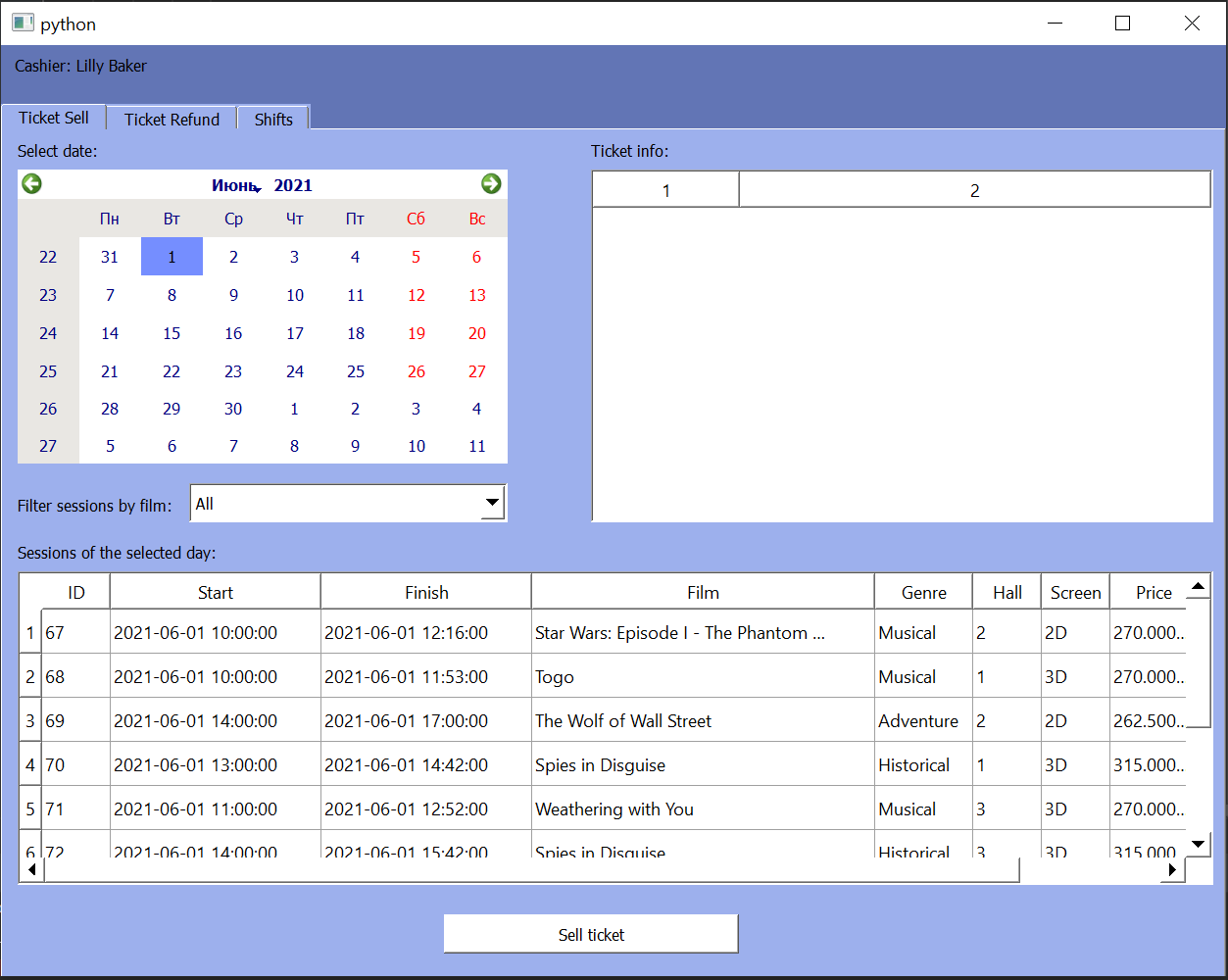


Рисунок 22. Панель управления для кассира

В разделе «Ticket Sell» главным из его атрибутов является таблица с данными о сеансах, прокат которых происходит в выбранный в календаре день. Также есть возможность фильтровать сеансы по фильму с целью обеспечения ускорения поиска желаемого клиентом сеанса.

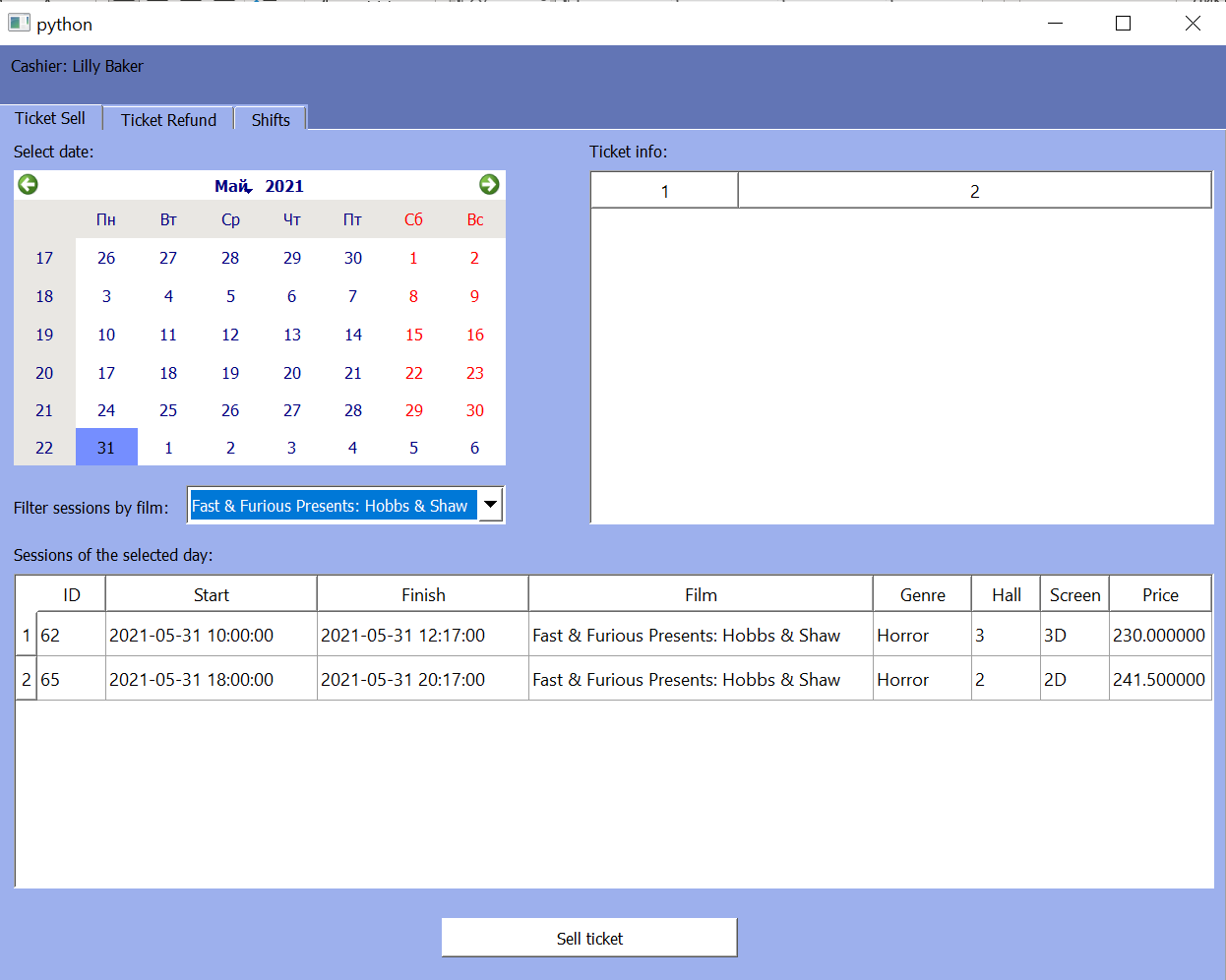


Рисунок 23. Пример фильтрации сеансов

При нажатии на соответствующий сеанс всплывает окно зала, на котором изображены места. Красным помечены занятые, зеленым – свободные.

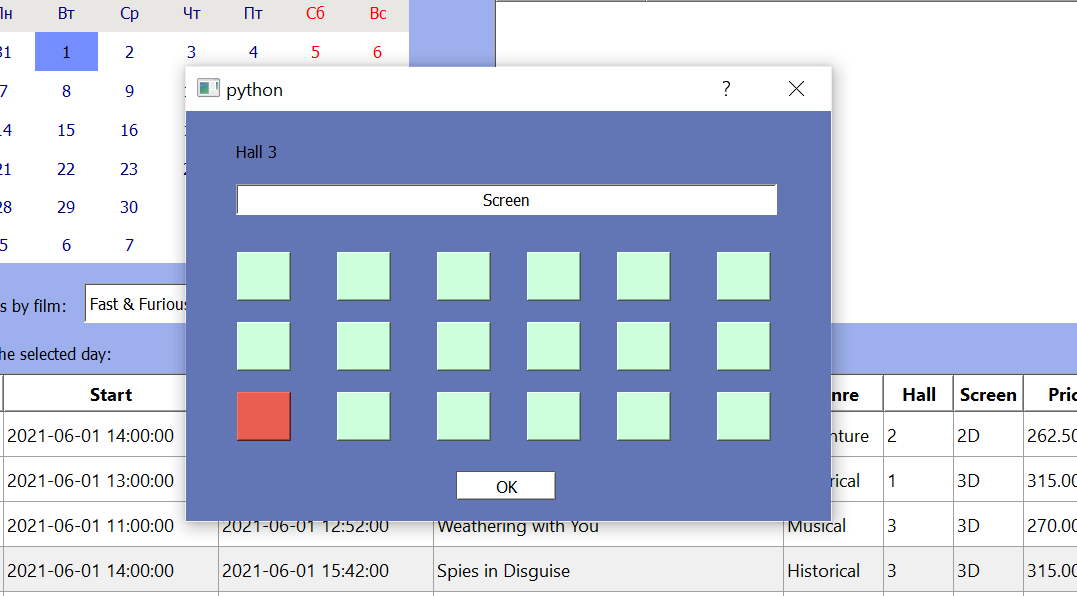


Рисунок 24. Пример изображения мест в кинозале

После выбора места и нажатия кнопки «OK» закрывается окно с залом и в поле «Ticket info» вкладки «Ticket Sell» содержится вся необходимая для клиента и кассира информация о билете, ждущем подтверждения продажи.

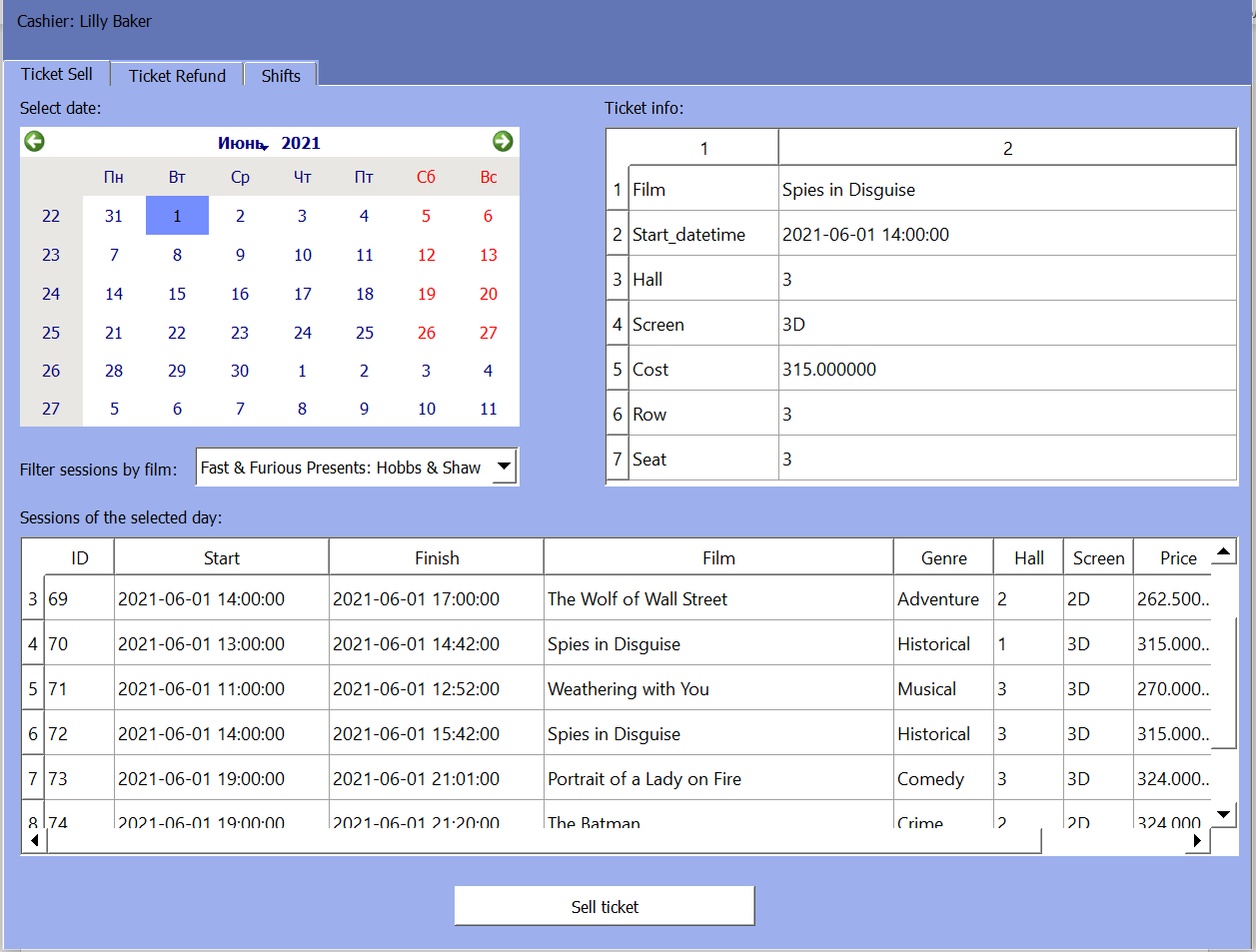


Рисунок 25. Пример вывода информации о билете

Кнопка «Sell ticket» эквивалентна запросу, содержащему процедуру ticket\_buying. Входящий параметр смены в процедуре определяется и заполняется автоматически. Таким образом, если сотрудник решит продать кому-то билет в не свою смену, то у него этого сделать не получится.

Во вкладке «Return Ticket» при вводе в текстовое поле номера билета и последующего нажатия клавиши «Enter» в таблице «Ticket info» высвечивается данные о билете, чтобы кассир убедился в его существовании. Кнопка «Refund Ticket» соответствует запросу к базе данных с применением процедуры ticket\_return.

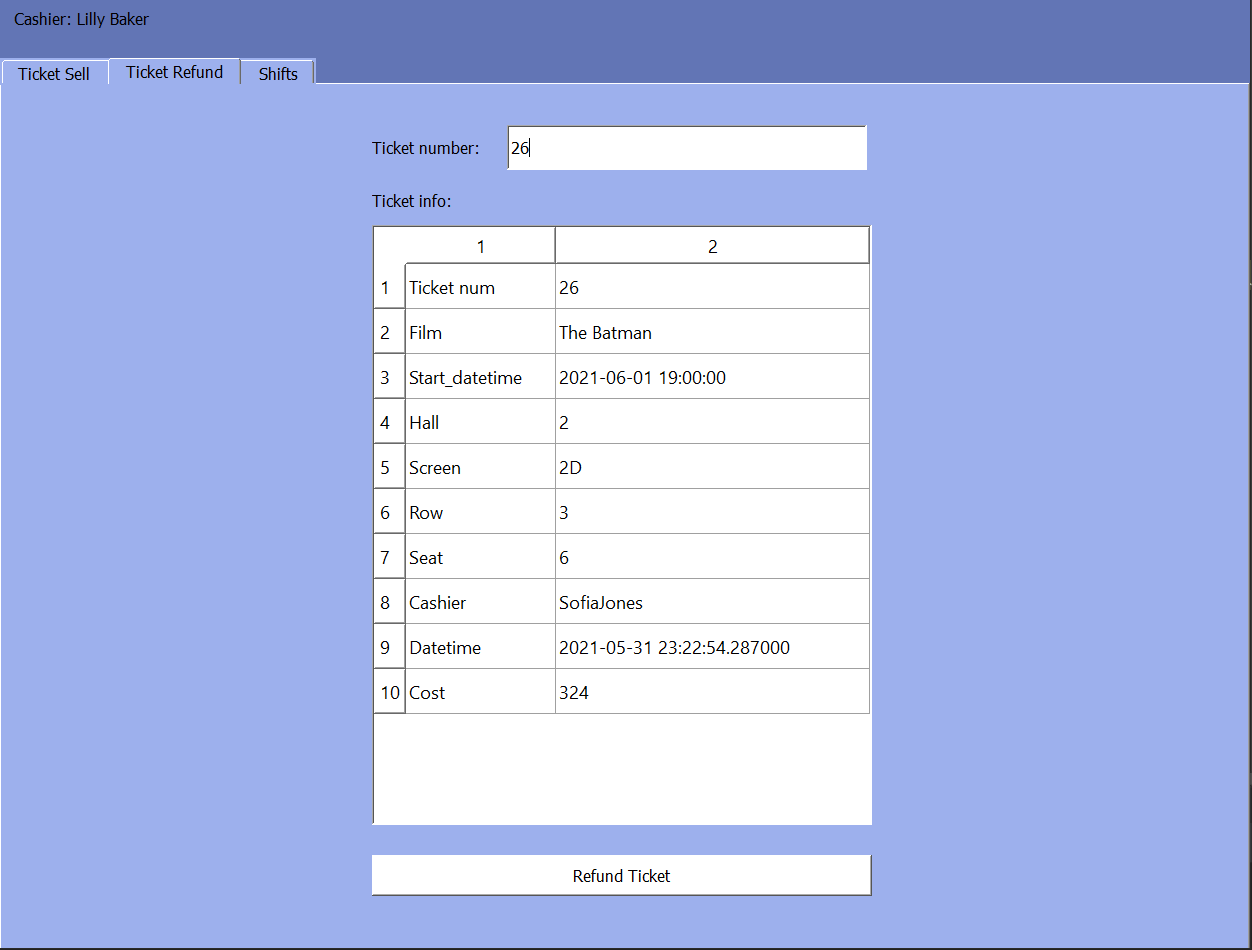


Рисунок 26. Пример вывода информации о возвращаемом билете

Таблица, содержащаяся во вкладке «Shifts», заполняется посредством запроса, созданного ранее (Рисунок 16).

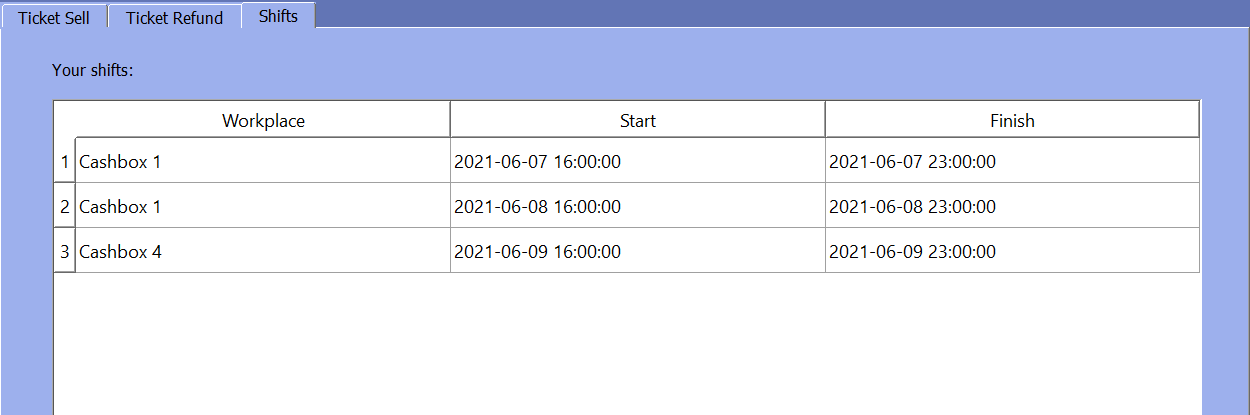


Рисунок 27. Вид вкладки «Shifts»

Если же в приложение зайдет пользователь с аккаунтом администратора, то после входа всплывет окно с вкладками «Shifts», «Sessions», «Check outlays» и «Payments».

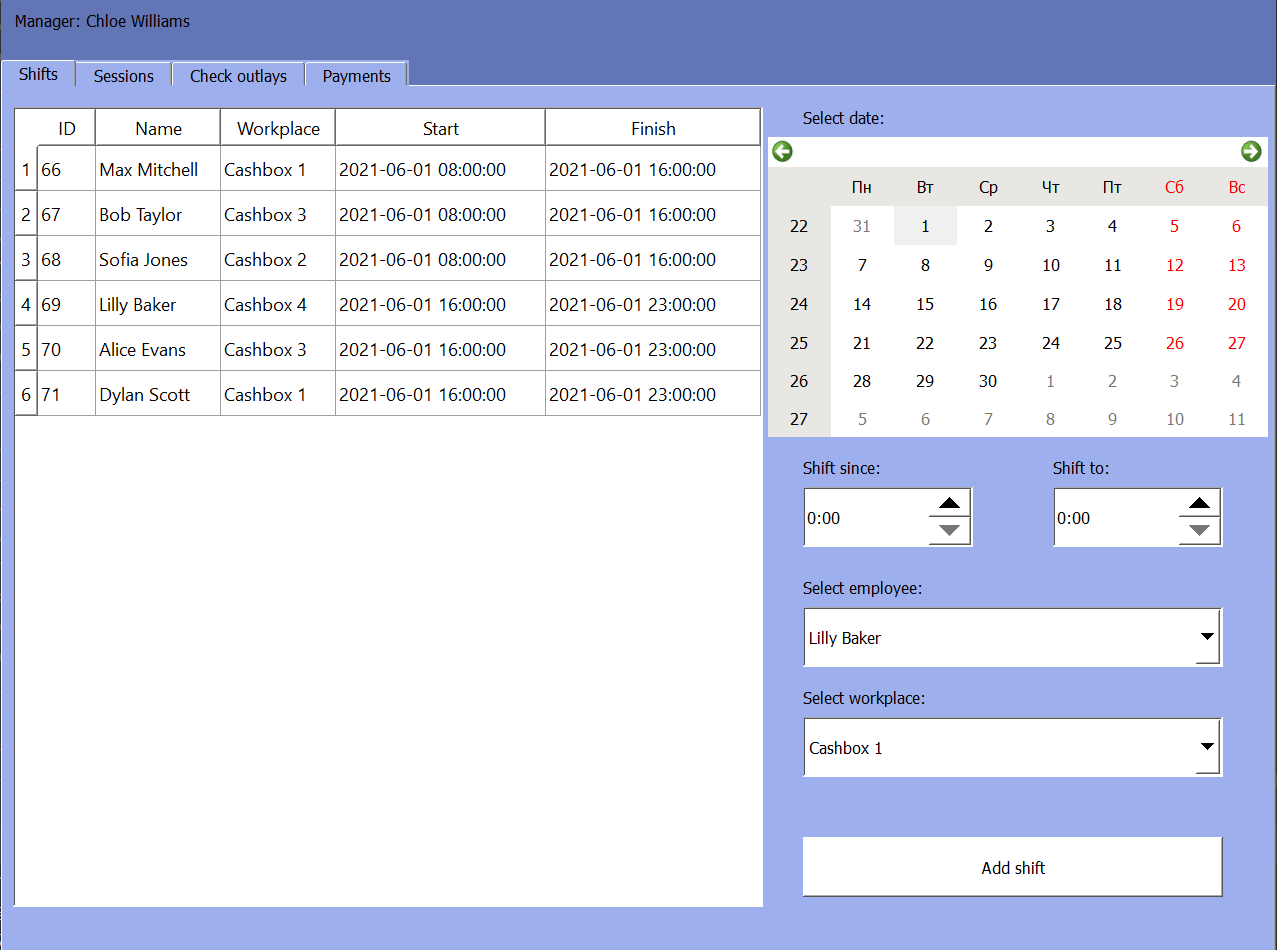


Рисунок 28. Панель управления администратора

Раздел «Sessions» предоставляет информацию обо всех сеансах по выбранной дате, правая часть окна представляет собой интерфейс для ввода данных и последующего занесения их в таблицу sessions.

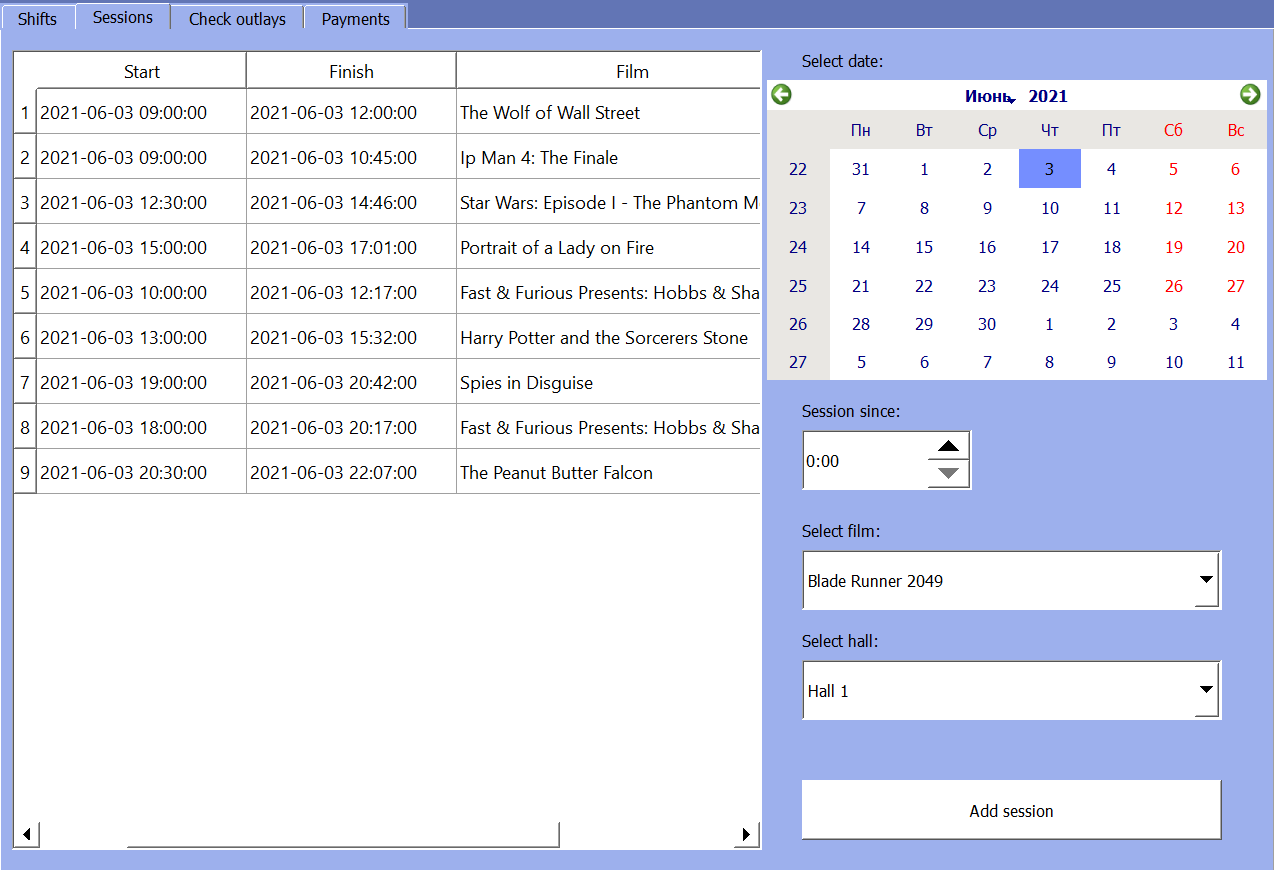


Рисунок 29. Раздел «Sessions»

Логика функционала вкладки «Shifts» аналогична предыдущей вкладке.

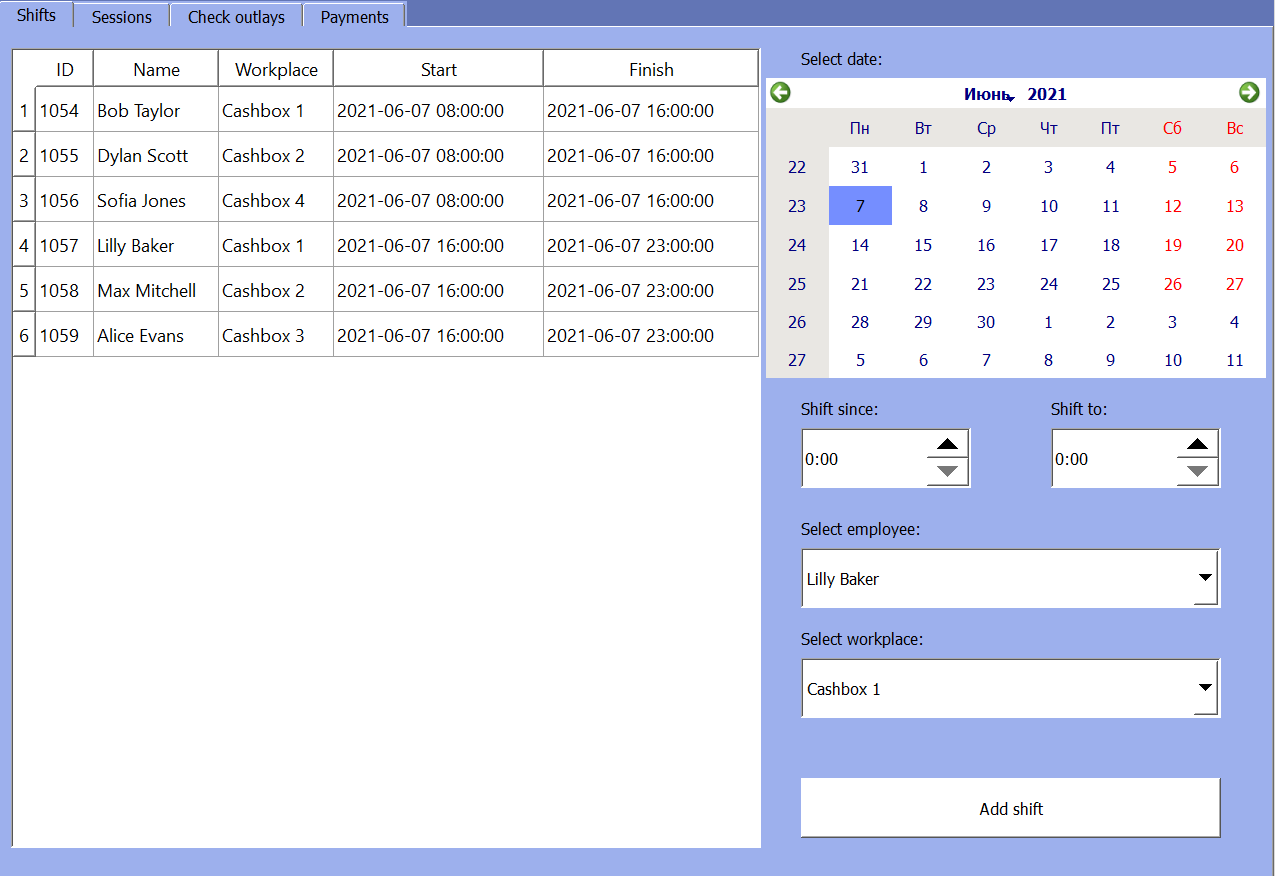


Рисунок 30. Раздел «Shifts»

В разделе «Check outlays» администратору представлена возможность ввести с какую по какую дату он хочет вывести смету о денежных транзакциях. Нажатие кнопки «Show outlays» соответствует вызову функции check\_earnings.



Рисунок 31. Пример вывода сметы

В последнем разделе «Payments» таблица заполняется при помощи функции is\_paid. При выборе одной и более строчек в данной таблице, соответствующие работникам кинотеатра, и нажатия на кнопку «Pay Salary» данные заносятся в таблицу payment.



Рисунок 32. Вид вкладки «Payments»

Также стоит отметить, что все предупреждения о некорректном вводе/удалении строчек в базе данных выводятся и в приложении.



Рисунок 33. Пример уведомления об ошибке

Для примера работы запросов в приложении было выполнено добавление сеанса администратором. Чтобы это осуществить необходимо в правой части экрана выбрать дату и время начала сеанса, а также указать фильм и зал для киносеанса.

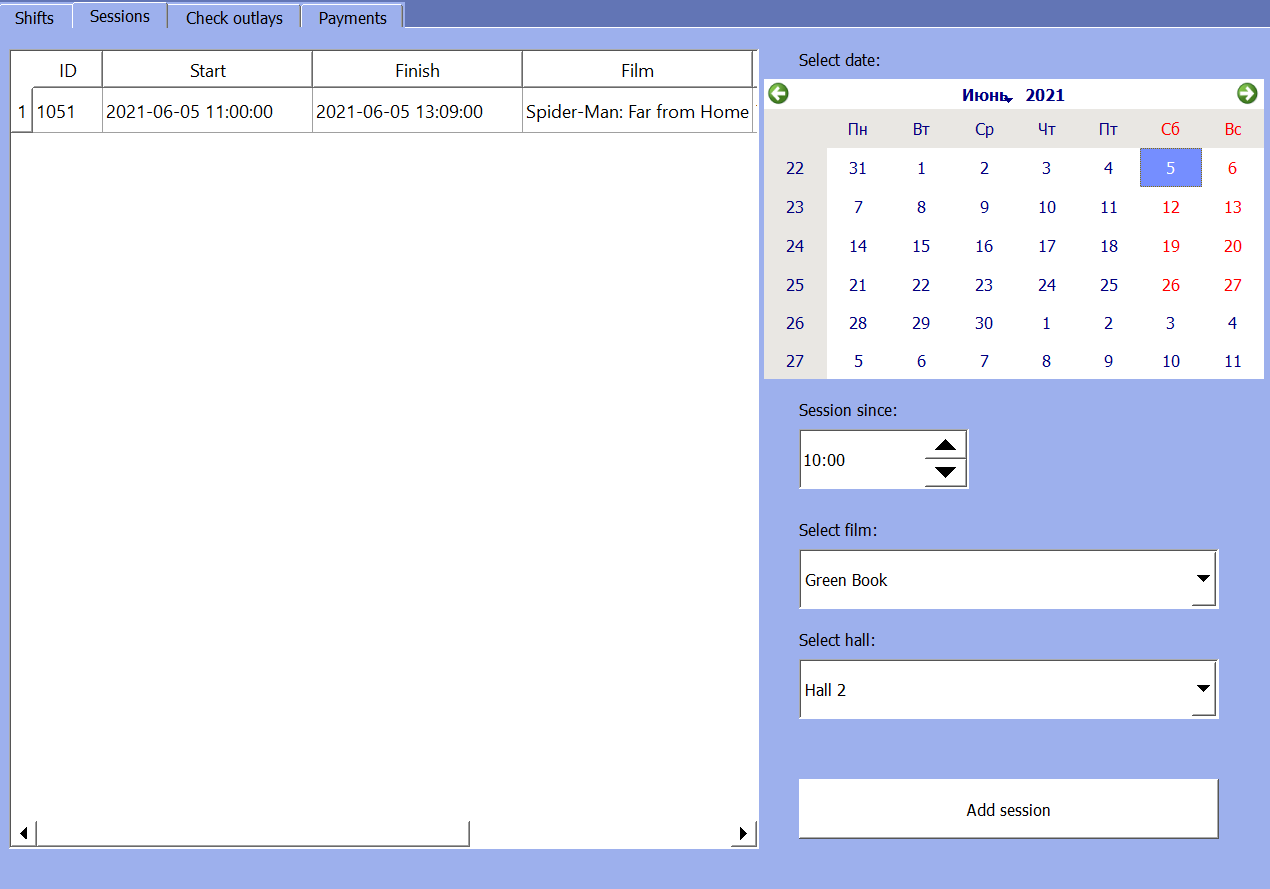


Рисунок 34. Указание данных для добавления сеанса

После нажатия кнопки «Add Session», обновления страницы и выбора соответствующей даты появляется строка с ранее заведенными параметрами.

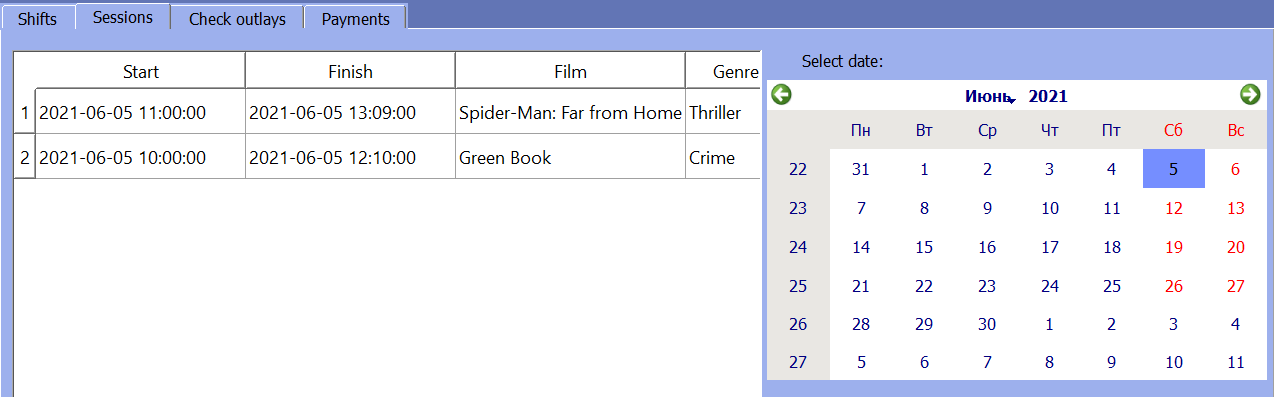


Рисунок 35. Изменения в расписании после добавления сеанса

Код программы представлен ниже. В его алгоритме содержится считывание заведенных данных и использование их в качестве входных параметров для вызова процедуры из базы данных add\_session.

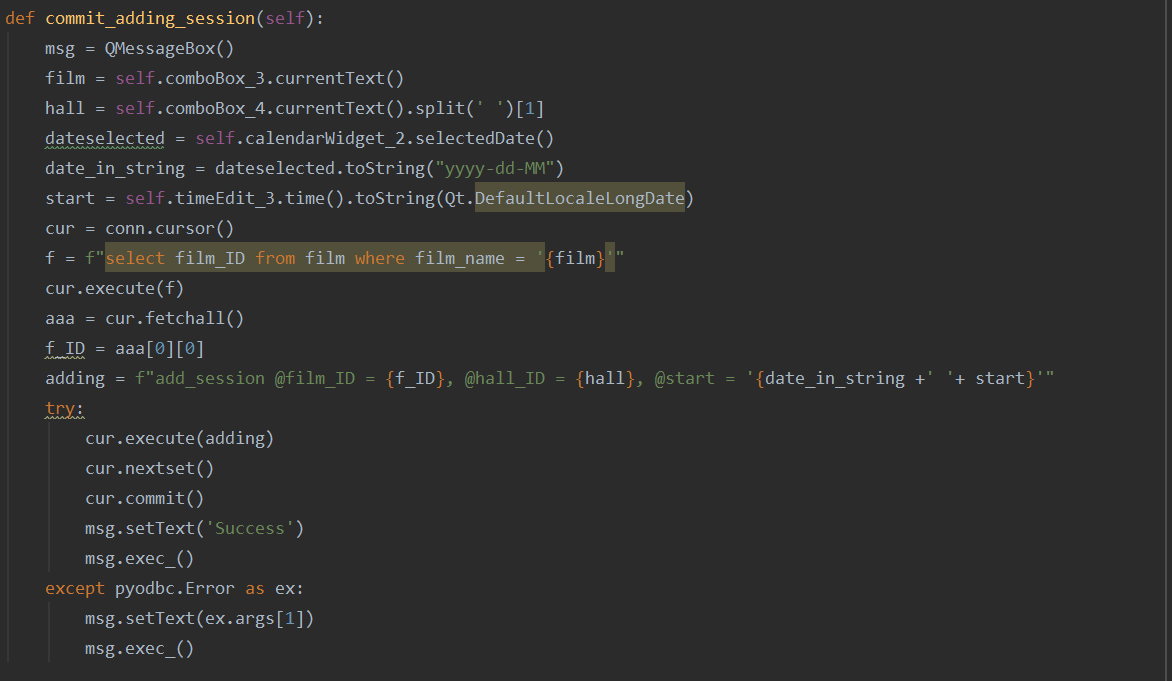


Рисунок 36. Программный код, выполняющий добавление сеанса в базу данных

Также продемонстрировано выполнение запроса на выплату зарплаты. Были выбраны три строки, соответствующие сотрудникам, которым необходимо выплатить зарплату.

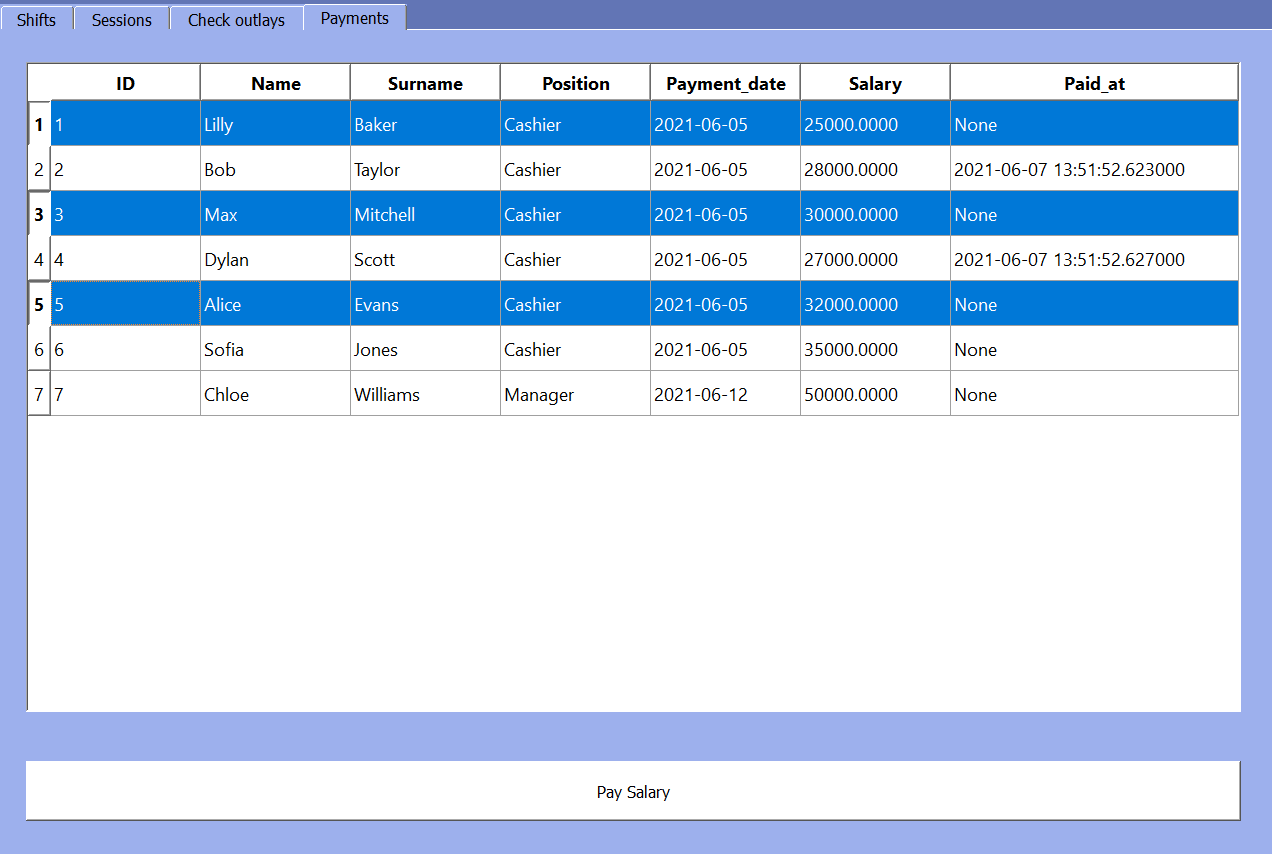


Рисунок 37. Выбор строк с сотрудниками для выплаты им зарплаты

После нажатия кнопки «Pay Salary» и обновления вкладки «Payments» были зафиксированы изменения в ранее выбранных строках.

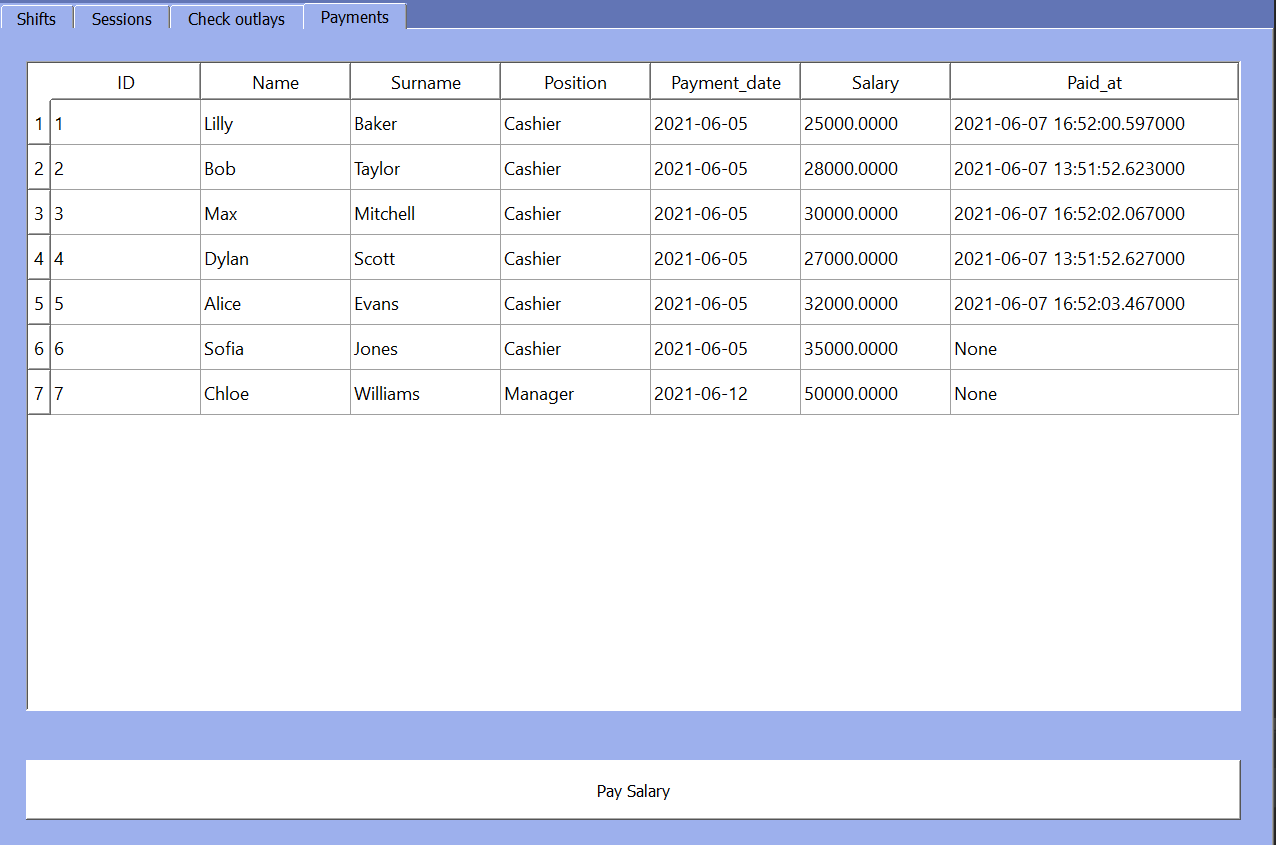


Рисунок 38. Изменение в отчетности после выплаты зарплат

Код программы для осуществления данных действий показан ниже. Функция show\_unpayed отвечает за наполнение таблицы данными с помощью вызова функции из базы данных is\_paid. Алгоритм для выплаты зарплаты содержится в функции pay, где для каждой выбранной строчки создается запрос с оператором INSERT в таблицу payment.



Рисунок 39. Программный код, выполняющий заполнение таблицы с отчетностью и выплату зарплаты

# Заключение

В ходе выполнения курсовой работы по теме «Проектирование и разработка базы данных кинотеатра» были выполнены следующие задачи:

1. Описание предметной области кинотеатра.
2. Проектирование инфологической и даталогической моделей, описание сущностей и таблиц с их связями соответственно.
3. Реализация базы данных в СУБД, заполнение ее данными.
4. Создание функций, триггеров и хранимых процедур, формирование SQL-запросов.
5. Разработка пользовательского приложения для доступа к информации и взаимодействия с базой данных.

При желании данную базу можно наполнять информацией при помощи добавления новых сущностей (например, расширение бизнеса приведет к появлению новых источников денежных потоков), расширять функционал за счет новых хранимых процедур, функций и триггеров, а также добавлять новые возможности в пользовательское приложение.

# Список литературы

1. Техническая документация по MS SQL Server (Дата обращения 12.05.2021) – URL: <https://docs.microsoft.com/ru-ru/sql/sql-server/?view=sql-server-ver15>
2. Erwin Data Modeler Documentation (дата обращения 01.05.2021) – URL: <https://erwin.com/bookshelf/public_html/2020R1/Content/Release%20Notes/Data%20Modeler%20Release%20Notes/6940.html>
3. *Маркин А. В.* Построение запросов и программирование на SQL. Учебное пособие. Рязань: РГРТУ, 2008. – 312 с.