

***Кафедра информационных систем***

***Рейтинговая работа "Расчетно-аналитическое задание"***

***по дисциплине* «Базы данных»**

***Задание/вариант № 4***

***Выполнена обучающимся группы з.ИЗДт 23.1/Б3-22***

**Герасимовой Анастасией Сергеевной**

(фамилия, имя, отчество)

***Преподаватель Камолова Анастасия Николаевна***

(фамилия, имя, отчество)

Москва – 2025 г.

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 5](#_ubs6nzeuhim0)

[ОСНОВНАЯ ЧАСТЬ 6](#_yzovbv1fifgj)

[1. Анализ предметной области 6](#_c2kd8zgpiac3)

[1.1. Нормативные документы, регламентирующие деятельность предметной области «Гостиница» 6](#_kgi52fkb7t6f)

[1.2. Анализ входных и выходных данных 6](#_qvjc2g9blqsz)

[1.3. Вывод анализа 9](#_jnxwua3y18os)

[2. Проектирование базы данных 10](#_23uern8fnrky)

[2.1. Проектирование таблиц 10](#_cgmwdz4w93vq)

[2.2. Проектирование связей> между таблицами 11](#_m4s7ub3uztea)

[3. SQL – код 13](#_t2qns1m98ms9)

[3.1. Создание базы данных 13](#_yihzzeg3r5ql)

[3.2. Написание триггеров 20](#_idyu0wpk52kx)

[3.3. Заполнение таблиц 33](#_w5xexibhavx8)

[4. SQL-запросы 47](#_6orda4omhfzg)

[4.1. Запрос на выбор всех данных по двум полям таблицы 47](#_z7f114h2mv6e)

[4.2. Запрос на выбор всех неповторяющихся данных по одному полю таблицы 48](#_5ebyhhdsupki)

[4.3. Запрос на выбор всех полей и записей таблицы,сгруппированных по значению одного поля, с использованием условия на группу (секции GROUP BY, HAVING) и с заголовками колонок, заданными в запросе 49](#_u3yeqe7if6f)

[4.4. Запрос на выбор всех неповторяющихся записей по одному полю таблицы с колонкой, образованной агрегирующей функцией SUM и озаглавленной в соответствии со смыслом 50](#_y3nmhffcjko2)

[4.5. Выбор нескольких (не всех) полей таблицы, отсортированных по убыванию 51](#_exaj5yupf2iy)

[4.6. Выбор произвольного количества полей таблицы с добавлением поля, являющегося результатом арифметического выражения, в котором участвуют значения поля таблицы 52](#_aielcr6c0vmu)

[4.7. Запрос на выбор всех записей по одному полю таблицы с колонкой, образованной агрегирующей функцией SUM и озаглавленной в соответствии со смыслом; выбор записей с использованием условия диапазона (between) 54](#_wzug6d4ctygo)

[4.8. Запрос на выбор всех записей по произвольному количеству полей таблицы с использованием агрегирующей функции AVG и условием на отбор записей, заданном в секции WHERE 55](#_apw6ktzeb7o6)

[4.9. Запрос на выбор двух полей таблицы с вычислением третьего поля по данным таблицы и сортировкой по убыванию по первому полю, по возрастанию по второму и по убыванию по третьему 56](#_749odpqq6s7h)

[4.10. Запрос на выборку данных таблицы с условием сравнения по шаблону LIKE 58](#_regmslygx9gy)

[4.11. Запрос с отбором по условию и сортировкой по убыванию одного из полей, а также добавлением поля, содержащего для всех записей константу, определенную при конструировании запроса 59](#_m6q10lxxve2k)

[4.12. Запрос с использованием агрегирующих функций MIN и MAX 60](#_rx9s6ffkdup7)

[4.13. Запрос с использованием сложного условия с логическими операторами AND, OR и сортировкой 61](#_n5d12s52usmo)

[ЗАКЛЮЧЕНИЕ 63](#_m8q2a1zfhe6)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 65](#_rsa0jbke4mms)

[ЛИСТИНГ 66](#_btvovja415vr)

# 

# ВВЕДЕНИЕ

Объектом проектирования является информационная система гостиницы, предназначенная для автоматизации процессов бронирования номеров, управления услугами, учета гостей и обработки платежей. Система обеспечивает эффективное управление ресурсами гостиницы, упрощает процесс бронирования для клиентов и повышает качество обслуживания. Информационная система гостиницы включает в себя базу данных, которая хранит информацию о номерах, гостях, бронированиях, услугах и платежах. Основная цель системы — обеспечить прозрачность и удобство управления данными, а также минимизировать ошибки, связанные с ручным вводом информации.

# 

# ОСНОВНАЯ ЧАСТЬ

## Анализ предметной области

### Нормативные документы, регламентирующие деятельность предметной области «Гостиница»

Нормативные документы, регламентирующие деятельность предметной области «Гостиница»:

а) федеральный закон "О защите прав потребителей": регламентирует права гостей на получение качественных услуг и защиту их интересов;

б) правила предоставления гостиничных услуг: устанавливают стандарты обслуживания, порядок бронирования, оплаты и проживания;

в) СанПиН 2.1.2.2645-10 "Санитарно-эпидемиологические требования к размещению, устройству, оборудованию, содержанию и режиму работы гостиниц": определяет санитарные нормы и правила для гостиниц;

г) ГОСТ Р 53423-2009 "Услуги гостиниц. Общие требования": устанавливает стандарты качества услуг, предоставляемых гостиницами;

д) положение о защите персональных данных (ФЗ-152): регламентирует обработку и хранение персональных данных гостей;

е) постановление Правительства РФ от 18.11.2020 N 1853 (ред. от 27.12.2024) "Об утверждении Правил предоставления гостиничных услуг в Российской Федерации"

### Анализ входных и выходных данных

Входные данные представляют собой информацию, необходимую для управления гостиничным комплексом. Они включают сведения о номерах, гостях, бронированиях, услугах, заказанных услугах и платежах.

Входные данные:

* информация о номерах - позволяет администраторам эффективно

управлять номерным фондом, а гостям — выбрать наиболее подходящий вариант:

а) номер комнаты,

б) количество кроватей (как двуспальных, так и односпальных),

в) тип номера,

г) площадь,

д) цена за ночь;

* информация о гостях - сбор контактных данных обеспечивает

возможность взаимодействия с клиентами и формирования персонализированных предложений:

а) имя,

б) фамилия,

в) отчество,

г) паспорт,

д) email,

е) номер телефона;

* информация о бронированиях - подробная информация о каждом

заказе позволяет контролировать занятость номеров и планировать работу персонала:

а) идентификатор гостя,

б) идентификатор номера,

в) дата заезда,

г) дата выезда;

* информация об услугах - может увеличить доходы гостиницы:

а) название,

б) описание,

в) тип цены (фиксированная или почасовая),

г) цена;

* информация о заказанных услугах - позволяет корректно

рассчитывать общую стоимость бронирования:

а) идентификатор бронирования,

б) идентификатор услуги,

в) количество;

* информация о платежах - детальный учет платежей обеспечивает

прозрачность финансовой деятельности и контроль над денежными потоками:

а) идентификатор бронирования,

б) дата оплаты,

в) способ оплаты,

г) сумма оплаты.

Выходные данные представляют собой отчеты и статистику, которые используются для анализа работы гостиницы и принятия решений.

Выходные данные:

* отчеты о бронированиях - ключевой показатель о загруженности

гостиницы:

а) список активных бронирований,

б) общая стоимость,

в) сколько гость оплатил и сколько осталось оплатить;

* отчеты о доходах - позволяют оценить финансовое состояние

гостиницы и определить наиболее прибыльные направления:

а) общая выручка за определенный период,

б) распределение доходов по номерам и услугам;

* статистика по гостям - позволяет создавать персонализированные

предложения и повышать удовлетворенность клиентов:

а) частота бронирований,

б) предпочтения по номерам и услугам;

* доступность номеров - важно для предотвращения конфликтов при

бронировании:

а) информация о свободных и занятых номерах на определенную дату;

* финансовые отчеты - помогают выявлять проблемные

бронирования и оперативно решать вопросы с гостями:

а) сумма оплаченных и неоплаченных счетов, остатки к оплате.

### Вывод анализа

Анализ предметной области показал, что для эффективного управления гостиницей необходимо разработать информационную систему, которая будет охватывать все ключевые процессы: от бронирования номеров до учета платежей. Основные требования к системе включают:

* удобство и прозрачность обработки данных;
* возможность автоматизации рутинных операций;
* соответствие нормативным документам, регулирующим деятельность гостиниц;
* обеспечение защиты персональных данных гостей.

Разработанная база данных и информационная система позволят гостинице:

* упростить процесс бронирования для клиентов;
* оптимизировать управление номерами и услугами;
* повысить качество обслуживания за счет оперативного доступа к актуальной информации;
* обеспечить прозрачность финансовых операций и отчетности.

Проектирование информационной системы гостиницы является важным этапом в автоматизации процессов управления и повышении качества обслуживания. Анализ предметной области, входных и выходных данных позволил определить ключевые требования к системе и разработать структуру базы данных, которая обеспечит эффективное хранение и обработку информации. Внедрение такой системы позволит гостинице повысить конкурентоспособность, улучшить качество услуг и обеспечить удовлетворенность клиентов.

## 

## Проектирование базы данных

### Проектирование таблиц

Проанализировав предметную область, были выявлены следующие таблицы (таб. 1)

Таблица 1 – Описание таблиц

| Таблица | Описание |
| --- | --- |
| Номера | Имеет поля: Код номера, Номер комнаты, Количество двуспальных кроватей, Количество односпальных кроватей, Класс номера, Площадь, Цена за ночь |
| Гости | Имеет поля: Код гостя, Имя, Фамилия, Отчество, Паспорт, Email, Номер телефона |
| Бронирования | Имеет поля: Код бронирования, Код гостя (внешний ключ), Код номера (внешний ключ), Дата заезда, Дата выезда, Общая стоимость номера, Общая стоимость услуг, Внесено, Остаток к оплате |
| Услуги | Имеет поля: Код услуги, Название услуги, Описание, Тип цены (фиксированная/почасовая), Цена |
| Заказанные услуги | Имеет поля: Код заказанной услуги, Код бронирования (внешний ключ), Код услуги (внешний ключ), Количество, Общая стоимость |
| Платежи | Имеет поля: Код платежа, Код бронирования (внешний ключ), Дата оплаты, Способ оплаты, Сумма оплаты |

Стоит сразу оговориться, что в рамках данной работы не будет реализована логика возврата средств гостю. Это значит, что в ходе разработки базы данных будут установлены следующие запреты:

1. запрет на изменение дат пребывания и смену комнаты, если новая

итоговая стоимость будет меньше старой;

1. запрет на обновление и удаление данных в таблице ЗАКАЗАННЫЕ

УСЛУГИ - обновление или удаление заказанных услуг может привести к необходимости возврата средств гостю;

1. запрет на обновление и удаление данных в таблице ПЛАТЕЖИ -

обновление или удаление платежей может привести к нарушению финансовой отчетности и необходимости возврата средств гостю.

Эти ограничения введены для обеспечения целостности данных, предотвращения ошибок и упрощения управления бронированиями и финансовыми операциями.

### Проектирование связей> между таблицами

Установим связи между таблицами посредством совпадающих полей в связываемых таблицах (Рисунок 1):

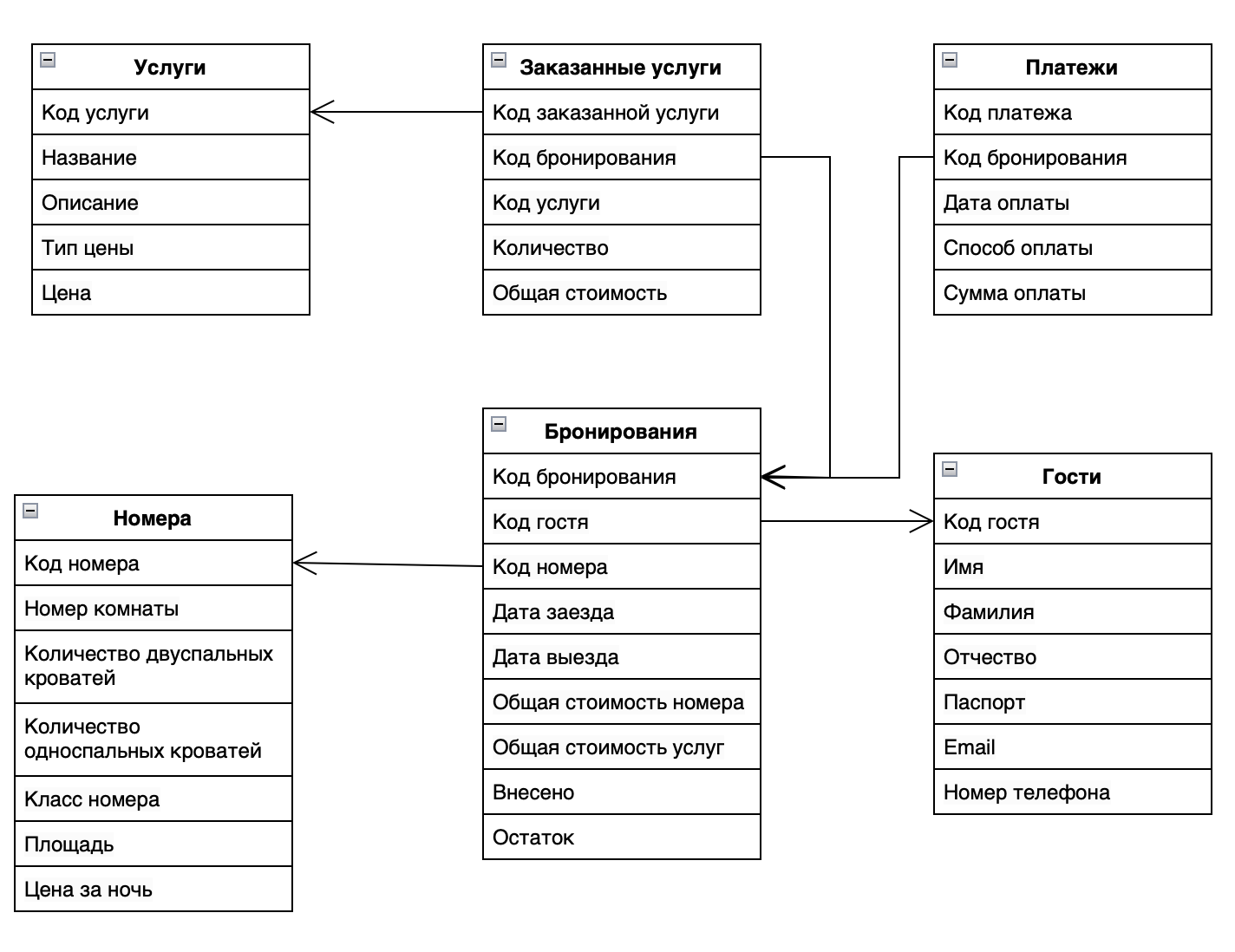


Рисунок 1 – Модель базы данных

Для определения типа связи между таблицами в базе данных гостиницы рассмотрим их взаимодействие.

Таблицы ГОСТИ и БРОНИРОВАНИЯ имеют между собой отношение типа «один-ко-многим», так как один гость может иметь несколько бронирований, но каждое бронирование связано только с одним гостем, что позволяет учитывать все заказы конкретного клиента и обеспечивает удобство управления данными.

Между таблицами НОМЕРА и БРОНИРОВАНИЯ так же установлена связь «один-ко-многим», поскольку один номер может быть забронирован несколько раз в разные периоды, но каждое бронирование связано только с одним номером, что помогает отслеживать занятость номеров и планировать их использование.

Связь между таблицами БРОНИРОВАНИЯ и ЗАКАЗАННЫЕ УСЛУГИ также представляет собой тип «один-ко-многим», так как одно бронирование может включать несколько заказанных услуг, но каждая заказанная услуга связана только с одним бронированием, что позволяет учитывать дополнительные услуги, заказанные клиентом, и корректно рассчитывать общую стоимость.

Аналогичным образом таблицы УСЛУГИ и ЗАКАЗАННЫЕ УСЛУГИ имеют связь «один-ко-многим», поскольку одна услуга может быть заказана несколько раз в разных бронированиях, но каждая заказанная услуга связана только с одной услугой, что обеспечивает гибкость в предоставлении дополнительных услуг и их учете.

Наконец, таблицы БРОНИРОВАНИЯ и ПЛАТЕЖИ также имеют связь «один-ко-многим», так как одно бронирование может включать несколько платежей (например, частичную оплату), но каждый платеж связан только с одним бронированием, что позволяет отслеживать финансовые операции и контролировать оплату бронирований.

Таким образом, все связи в базе данных гостиницы организованы по принципу «один-ко-многим», что характерно для реляционных баз данных, где одна запись в одной таблице может быть связана с несколькими записями в другой таблице.

## 

## SQL – код

### 3.1. Создание базы данных

При создании таблицы rooms были выбраны типы данных, которые обеспечивают точность, оптимальное использование памяти и корректность данных (Рисунок 2):

**CREATE TABLE rooms (**

**room\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**number SMALLINT UNSIGNED NOT NULL UNIQUE,**

**double\_bed TINYINT UNSIGNED NOT NULL,**

**single\_bed TINYINT UNSIGNED NOT NULL,**

**class ENUM('Стандарт', 'Полулюкс', 'Люкс') NOT NULL,**

**area DECIMAL(5, 2) UNSIGNED NOT NULL,**

**price\_per\_night DECIMAL(10, 2) UNSIGNED NOT NULL**

**);**

Рисунок 2 - Структура таблицы rooms

Поле room\_id имеет тип INT AUTO\_INCREMENT PRIMARY KEY, что позволяет уникально идентифицировать каждую комнату и автоматически генерировать значения для новых записей.

Номер комнаты (number) представлен типом SMALLINT UNSIGNED NOT NULL UNIQUE, так как он должен быть уникальным, положительным и обязательно заполненным. SMALLINT позволяет хранить числа от 0 до 65535, что более чем достаточно для нумерации комнат в гостинице.

Для количества двуспальных (double\_bed) и односпальных кроватей (single\_bed) используется тип TINYINT UNSIGNED NOT NULL, поскольку их значение ограничено небольшим диапазоном и не может быть отрицательным. TINYINT позволяет хранить числа от 0 до 255, что предостаточно для количества кроватей в одном номере. NOT NULL гарантирует, что поле заполнено - не может быть номера без хотя бы одной кровати.

Класс номера (class) описан типом ENUM('Стандарт', 'Полулюкс', 'Люкс') NOT NULL. Это гарантирует, что в поле будет указано только одно из допустимых значений.

Площадь номера (area) хранится в формате DECIMAL(5, 2) UNSIGNED NOT NULL для точного представления чисел с плавающей точкой, а цена за ночь (price\_per\_night) — в формате DECIMAL(10, 2) UNSIGNED NOT NULL для точного учета денежных значений. Все эти решения обеспечивают целостность данных, минимизируют ошибки и соответствуют требованиям предметной области.

При создании таблицы guests были выбраны типы данных, которые обеспечивают точность, полноту и корректность хранения информации о гостях (Рисунок 3):

**CREATE TABLE guests (**

**guest\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**first\_name VARCHAR(100) NOT NULL,**

**last\_name VARCHAR(100) NOT NULL,**

**patronymic VARCHAR(100),**

**passport VARCHAR(20) UNIQUE NOT NULL,**

**email VARCHAR(100) UNIQUE NOT NULL,**

**phone\_number VARCHAR(20) UNIQUE NOT NULL**

**);**

Рисунок 3 - Структура таблицы guests

Поле guest\_id имеет тип INT AUTO\_INCREMENT PRIMARY KEY, что позволяет уникально идентифицировать каждого гостя и автоматически генерировать значения для новых записей.

Имя (first\_name) и фамилия (last\_name) представлены типом VARCHAR(100) NOT NULL, так как они являются обязательными полями для идентификации клиента. Для отчества (patronymic) используется также тип VARCHAR(100), но оно не является обязательным (NULL допустимо), поскольку не все клиенты могут иметь отчество (например, иностранцы). Длины в 100 символов должно быть более чем достаточно для хранения для каждого поля.

Поле passport имеет тип данных VARCHAR(20) с ограничениями UNIQUE и NOT NULL. Выбор типа VARCHAR(20) обусловлен тем, что паспортные данные обычно состоят из серии и номера документа, которые в совокупности занимают не более 20 символов. Ограничение UNIQUE обеспечивает уникальность каждого значения в поле, что критически важно для правильной идентификации гостей, так как два человека не могут иметь одинаковые паспортные данные. Ограничение NOT NULL делает это поле обязательным для заполнения, поскольку паспортные данные являются обязательным требованием для регистрации гостя в гостинице в соответствии с действующим законодательством. Таким образом, такая структура поля гарантирует точность и законность хранения информации о клиентах гостиницы.

Электронная почта (email) и номер телефона (phone\_number) представлены типами VARCHAR(100) и VARCHAR(20) соответственно, с ограничением UNIQUE NOT NULL, чтобы обеспечить уникальность контактной информации и возможность оперативной связи с гостем.

Такая структура таблицы гарантирует, что данные о гостях будут полными, корректными, без дублирований и соответствующими требованиям предметной области. Все ограничения (NOT NULL, UNIQUE) добавлены для предотвращения ошибок при заполнении и обеспечения целостности данных.

При создании таблицы reservations были выбраны типы данных, обеспечивающие точность и корректность хранения информации о бронированиях (Рисунок 4):

**CREATE TABLE reservations (**

**reservation\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**guest\_id INT NOT NULL,**

**room\_id INT NOT NULL,**

**check\_in\_date DATE NOT NULL,**

**check\_out\_date DATE NOT NULL,**

**total\_room\_price DECIMAL(10, 2) NOT NULL DEFAULT 0,**

**total\_service\_price DECIMAL(10, 2) NOT NULL DEFAULT 0,**

**amount\_paid DECIMAL(10, 2) UNSIGNED NOT NULL DEFAULT 0,**

**amount\_remaining DECIMAL(10, 2) UNSIGNED NOT NULL DEFAULT 0,**

**FOREIGN KEY (guest\_id) REFERENCES guests(guest\_id) ON DELETE CASCADE,**

**FOREIGN KEY (room\_id) REFERENCES rooms(room\_id) ON DELETE CASCADE**

**);**

Рисунок 4 - Структура таблицы reservations

Поле reservation\_id типа INT AUTO\_INCREMENT PRIMARY KEY уникально идентифицирует каждое бронирование, автоматически генерируя новые значения при добавлении записей.

Поля guest\_id и room\_id типа INT NOT NULL с ограничением FOREIGN KEY связывают таблицу с гостями (guests) и номерами (rooms), обеспечивая корректную привязку бронирования к конкретному гостю и номеру. При удалении записи из таблицы guests или rooms, все связанные бронирования удаляются автоматически благодаря параметру ON DELETE CASCADE.

Даты заезда (`check\_in\_date) и выезда (check\_out\_date) представлены типом DATE NOT NULL, что гарантирует обязательное указание этих параметров для каждого бронирования. Для хранения финансовых данных используются поля типа DECIMAL(10, 2) UNSIGNED NOT NULL DEFAULT 0: total\_room\_price (стоимость проживания в номере), total\_service\_price (стоимость дополнительных услуг), amount\_paid (оплаченная сумма) и amount\_remaining (остаток к оплате). Выбор типа DECIMAL обеспечивает точное представление денежных значений, а модификатор UNSIGNED исключает отрицательные числа. Все эти поля имеют значение по умолчанию 0, что позволяет обрабатывать ситуации, когда расчеты еще не завершены.

Такая структура таблицы гарантирует точность данных, соблюдение бизнес-логики и возможность эффективного управления бронированиями, платежами и связанными данными. Внешние ключи и ограничения (`NOT NULL`, `DEFAULT`, `UNSIGNED`) обеспечивают надежность и корректность работы системы.

Таблица services создана для хранения информации о доступных услугах гостиницы (Рисунок 5):

**CREATE TABLE services (**

**service\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**service\_name VARCHAR(100) NOT NULL UNIQUE,**

**description TEXT,**

**is\_price\_fixed BOOLEAN NOT NULL DEFAULT TRUE,**

**price DECIMAL(10, 2) UNSIGNED NOT NULL**

**);**

Рисунок 5 - Структура таблицы services

Аналогично предыдущим таблицам, поле service\_id имеет тип INT AUTO\_INCREMENT PRIMARY KEY, что обеспечивает уникальную идентификацию каждой услуги.

Название услуги (service\_name) представлено типом VARCHAR(100) NOT NULL UNIQUE, так как название должно быть уникальным и обязательно заполненным.

Поле description имеет тип `TEXT`, что позволяет хранить подробное описание услуги без ограничений по длине.

Тип цены (фиксированная или зависящая от параметров) представлен полем is\_price\_fixed типа BOOLEAN NOT NULL DEFAULT TRUE, которое по умолчанию устанавливается в значение `TRUE` (фиксированная цена). Такая структура таблицы обеспечивает гибкость при добавлении новых услуг и точность их описания.

Цена услуги (price) хранится в формате DECIMAL(10, 2) UNSIGNED NOT NULL, чтобы точно представлять денежные значения.

Такой таблицы вполне достаточно для описания информации о дополнительных услугах в гостинице. Она обеспечивает гибкость при добавлении новых услуг, точность в хранении их стоимости и удобство в управлении данными, что позволяет эффективно учитывать и предоставлять услуги гостям.

Таблица reservation\_services связывает бронирования с заказанными услугами (Рисунок 6):

**CREATE TABLE reservation\_services (**

**reservation\_service\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**reservation\_id INT NOT NULL,**

**service\_id INT NOT NULL,**

**quantity TINYINT UNSIGNED NOT NULL DEFAULT 1,**

**total\_price DECIMAL(10, 2) UNSIGNED NOT NULL DEFAULT 0,**

**FOREIGN KEY (reservation\_id) REFERENCES reservations(reservation\_id) ON DELETE CASCADE,**

**FOREIGN KEY (service\_id) REFERENCES services(service\_id) ON DELETE CASCADE**

**);**

Рисунок 6- Структура таблицы reservation\_services

Поле reservation\_service\_id имеет тип INT AUTO\_INCREMENT PRIMARY KEY, что позволяет уникально идентифицировать каждую запись.

Поля reservation\_id и service\_id представлены типом INT NOT NULL с ограничением внешнего ключа (`FOREIGN KEY`), связывающего их с таблицами reservations и services соответственно.

Количество/объем услуги (quantity) представлено типом TINYINT UNSIGNED NOT NULL DEFAULT 1, что позволяет указывать объем услуги (например, количество завтраков).

Общая стоимость услуги (total\_price) хранится в формате DECIMAL(10, 2) UNSIGNED NOT NULL DEFAULT 0.

Все эти ограничения обеспечивают корректность данных и возможность точного расчета стоимости бронирования.

Таблица payments предназначена для учета финансовых операций, связанных с бронированием (Рисунок 7):

**CREATE TABLE payments (**

**payment\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**reservation\_id INT NOT NULL,**

**payment\_date DATE NOT NULL,**

**payment\_method ENUM('Наличные', 'Карта', 'Онлайн') NOT NULL,**

**total DECIMAL(10, 2) UNSIGNED NOT NULL DEFAULT 0,**

**FOREIGN KEY (reservation\_id) REFERENCES reservations(reservation\_id) ON DELETE CASCADE**

**);**

Рисунок 7 - Структура таблицы payments

Поле payment\_id типа INT AUTO\_INCREMENT PRIMARY KEY уникально идентифицирует каждый платеж, автоматически генерируя новые значения при добавлении записей.

Поле reservation\_id типа INT NOT NULL с ограничением FOREIGN KEY связывает таблицу с бронированием (reservations), обеспечивая целостность данных и автоматически удаляя платежи при удалении связанного бронирования благодаря параметру ON DELETE CASCADE.

Дата платежа (payment\_date) представлена типом DATE NOT NULL, что гарантирует обязательное указание точной даты совершения операции.

Способ оплаты (payment\_method) описан типом ENUM('Наличные', 'Карта', 'Онлайн') NOT NULL, позволяющим использовать только предопределенные варианты, что исключает ошибки при вводе данных.

Для хранения суммы платежа используется поле total типа DECIMAL(10, 2) UNSIGNED NOT NULL DEFAULT 0, обеспечивающее точное представление денежных значений без возможности отрицательных чисел.

Такая структура таблицы гарантирует корректность и безопасность данных, обеспечивает четкую связь с бронированиями и позволяет эффективно отслеживать все финансовые операции, связанные с проживанием гостей. Ограничения (NOT NULL, DEFAULT, UNSIGNED) и использование внешних ключей делают систему надежной и защищенной от ошибок.

Все таблицы созданы с учетом требований предметной области и обеспечивают корректное хранение данных, соблюдение целостности и возможность выполнения необходимых операций.

### 3.2. Написание триггеров

Для обеспечения целостности данных и автоматизации ключевых процессов в базе данных были разработаны и внедрены триггеры. Триггеры — это специальные процедуры, которые автоматически выполняются при возникновении определенных событий, таких как вставка, обновление или удаление записей в таблицах. Они позволяют минимизировать ручные операции, предотвратить ошибки и обеспечить корректность данных.

Перед созданием триггеров используется команда DELIMITER $$, которая временно изменяет стандартный разделитель команд (;) на символ $$. Это необходимо для того, чтобы избежать конфликтов при использовании точки с запятой внутри тела триггера, так как она могла бы быть интерпретирована как завершение всей команды создания триггера.

После определения триггера разделитель снова возвращается к стандартному значению (;) с помощью команды DELIMITER (Рисунок 8):

**DELIMITER $$ -- Временно меняем разделитель на $$**

**CREATE TRIGGER ... -- Определяем триггер, используя ; внутри для разделения операторов**

**... -- Тело триггера**

**END$$ -- Завершаем определение триггера с новым разделителем**

**DELIMITER ; -- Возвращаем разделитель обратно к ;**

Рисунок 8 - Структура использования разделителя

Эта конструкция обеспечивает корректную интерпретацию кода триггера и позволяет использовать точку с запятой внутри его тела без риска преждевременного завершения команды. Такая структура является стандартной практикой при создании не только триггеров, но и других сложных объектов базы данных, таких как хранимые процедуры или функции.

Для автоматизации расчета стоимости бронирования и контроля корректности данных в базе данных были реализованы два ключевых триггера (Рисунок 9 и Рисунок 10)

**CREATE TRIGGER calculate\_total\_room\_price**

**BEFORE INSERT ON reservations**

**FOR EACH ROW**

**BEGIN**

**DECLARE price DECIMAL(10, 2);**

**IF NEW.check\_in\_date < CURDATE() THEN**

**SIGNAL SQLSTATE '45000'**

**SET MESSAGE\_TEXT = 'Дата въезда не может быть раньше текущей даты.';**

**END IF;**

**IF NEW.check\_out\_date <= NEW.check\_in\_date THEN**

**SET @errorMsg = CONCAT('Дата выезда: ',**

**DATE\_FORMAT(NEW.check\_out\_date, '%d-%m-%Y'),**

**' должна быть хотя бы на один день позже даты въезда: ',**

**DATE\_FORMAT(NEW.check\_in\_date, '%d-%m-%Y'));**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = @errorMsg;**

**END IF;**

**SELECT price\_per\_night INTO price**

**FROM rooms**

**WHERE room\_id = NEW.room\_id;**

**SET NEW.total\_room\_price = price \* DATEDIFF(NEW.check\_out\_date, NEW.check\_in\_date);**

**SET NEW.amount\_remaining = NEW.total\_room\_price;**

**END$$**

Рисунок 9 - Триггер, считающий стоимость бронирования комнаты

**CREATE TRIGGER update\_total\_price**

**BEFORE UPDATE ON reservations**

**FOR EACH ROW**

**BEGIN**

**DECLARE price DECIMAL(10, 2);**

**DECLARE old\_duration INT;**

**DECLARE new\_duration INT;**

**IF NEW.check\_in\_date < CURDATE() THEN**

**SIGNAL SQLSTATE '45000'**

**SET MESSAGE\_TEXT = 'Дата въезда не может быть раньше текущей даты.';**

**END IF;**

**IF NEW.check\_out\_date <= NEW.check\_in\_date THEN**

**SET @errorMsg = CONCAT('Дата выезда: ',**

**DATE\_FORMAT(NEW.check\_out\_date, '%d-%m-%Y'),**

**' должна быть хотя бы на один день позже даты въезда: ',**

**DATE\_FORMAT(NEW.check\_in\_date, '%d-%m-%Y'));**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = @errorMsg;**

**END IF;**

**SET old\_duration = DATEDIFF(OLD.check\_out\_date, OLD.check\_in\_date);**

**SET new\_duration = DATEDIFF(NEW.check\_out\_date, NEW.check\_in\_date);**

**IF OLD.check\_in\_date <= CURDATE() AND CURDATE() <= OLD.check\_out\_date THEN**

**IF NEW.check\_in\_date <> OLD.check\_in\_date THEN**

**SET @errorMsg = CONCAT('Дата въезда: ',**

**DATE\_FORMAT(OLD.check\_in\_date, '%d-%m-%Y'),**

**' не может быть изменена: ',**

**DATE\_FORMAT(NEW.check\_in\_date, '%d-%m-%Y'),**

**', так как бронирование уже активно. ID: ',**

**NEW.reservation\_id);**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = @errorMsg;**

**END IF;**

**IF NEW.check\_out\_date < OLD.check\_out\_date THEN**

**SET @errorMsg = CONCAT('Дата выезда: ',**

**DATE\_FORMAT(OLD.check\_out\_date, '%d-%m-%Y'),**

**' не может быть уменьшена: ',**

**DATE\_FORMAT(NEW.check\_out\_date, '%d-%m-%Y'),**

**', так как бронирование уже активно. ID: ',**

**NEW.reservation\_id);**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = @errorMsg;**

**END IF;**

**END IF;**

**SELECT price\_per\_night INTO price**

**FROM rooms**

**WHERE room\_id = NEW.room\_id;**

**IF price \* new\_duration < OLD.total\_room\_price THEN**

**SET @errorMsg = CONCAT('Новая стоимость: ',**

**price \* new\_duration,**

**' не может быть меньше старой: ',**

**OLD.total\_room\_price,**

**', так как возврат средств не предусмотрен. ID: ',**

**NEW.reservation\_id);**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = @errorMsg;**

**END IF;**

**SET NEW.total\_room\_price = price \* new\_duration;**

**SET NEW.amount\_remaining = NEW.total\_room\_price + NEW.total\_service\_price - NEW.amount\_paid;**

**END$$**

Рисунок 10 - Триггер, обновляющий стоимость бронирования

Эти триггеры обеспечивают автоматический расчет стоимости проживания и предотвращают ошибки при вводе или изменении данных.

Триггер calculate\_total\_room\_price выполняется перед вставкой новой записи в таблицу reservations. Он рассчитывает общую стоимость проживания (total\_room\_price) на основе цены за ночь выбранного номера и количества дней бронирования, а также устанавливает начальную сумму к оплате (amount\_remaining). Дополнительно триггер проверяет корректность дат бронирования.

Логика работы:

1. проверка даты въезда: если дата въезда (check\_in\_date) указана

раньше текущей даты, генерируется ошибка;

1. проверка даты выезда: если дата выезда (check\_out\_date) указана

earlier или равна дате въезда, генерируется ошибка с детальным сообщением;

1. расчет стоимости проживания: стоимость проживания

(total\_room\_price) рассчитывается как произведение цены за ночь (price\_per\_night, извлеченная из rooms ) и количества ночей (DATEDIFF(NEW.check\_out\_date, NEW.check\_in\_date));

1. установка суммы к оплате: первоначально вся стоимость

проживания считается неоплаченной, поэтому поле amount\_remaining устанавливается равным total\_room\_price.  
 Триггер update\_total\_price выполняется перед обновлением существующей записи в таблице reservations. Он пересчитывает стоимость бронирования при изменении дат въезда/выезда и предотвращает некорректные изменения дат уже активных бронирований. Логика работа схожа с предыдущим триггером, но накладываются дополнительные ограничения и amount\_remaining рассчитывается, учитывая уже оплаченную сумму (amount\_paid).

Триггеры calculate\_total\_room\_price и update\_total\_price обеспечивают автоматический расчет стоимости бронирования и контроль корректности данных. Такая реализация гарантирует точность финансовых расчетов, предотвращает ошибки и соблюдает бизнес-правила гостиницы.

Для предотвращения конфликтов при бронировании номеров в гостинице были реализованы два триггера: check\_room\_availability и check\_room\_availability\_on\_update (Рисунок 11 и Рисунок 12):

**CREATE TRIGGER check\_room\_availability**

**BEFORE INSERT ON reservations**

**FOR EACH ROW**

**BEGIN**

**DECLARE conflict\_reservation\_id INT;**

**DECLARE conflict\_room\_id INT;**

**DECLARE conflict\_check\_in DATE;**

**DECLARE conflict\_check\_out DATE;**

**SELECT reservation\_id, room\_id, check\_in\_date, check\_out\_date**

**INTO conflict\_reservation\_id, conflict\_room\_id, conflict\_check\_in, conflict\_check\_out**

**FROM reservations**

**WHERE room\_id = NEW.room\_id**

**AND (**

**(NEW.check\_in\_date < check\_out\_date) AND**

**(NEW.check\_out\_date > check\_in\_date)**

**)**

**LIMIT 1;**

**IF conflict\_room\_id IS NOT NULL THEN**

**SET @errorMsg = CONCAT(**

**'Комната ', conflict\_room\_id,**

**' уже забронирована с ', DATE\_FORMAT(conflict\_check\_in, '%d-%m-%Y'),**

**' по ', DATE\_FORMAT(conflict\_check\_out, '%d-%m-%Y'), ', id: ', conflict\_reservation\_id,**

**'. Невозможно забронировать с ', DATE\_FORMAT(NEW.check\_in\_date, '%d-%m-%Y'),**

**' по ', DATE\_FORMAT(NEW.check\_out\_date, '%d-%m-%Y'));**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = @errorMsg;**

**END IF;**

**END$$**

Рисунок 11 - Триггер, проверяющий, что номер не занят на определенную дату

**CREATE TRIGGER check\_room\_availability\_on\_update**

**BEFORE UPDATE ON reservations**

**FOR EACH ROW**

**BEGIN**

**DECLARE conflict\_reservation\_id INT;**

**DECLARE conflict\_room\_id INT;**

**DECLARE conflict\_check\_in DATE;**

**DECLARE conflict\_check\_out DATE;**

**SELECT reservation\_id, room\_id, check\_in\_date, check\_out\_date**

**INTO conflict\_reservation\_id, conflict\_room\_id, conflict\_check\_in, conflict\_check\_out**

**FROM reservations**

**WHERE room\_id = NEW.room\_id AND reservation\_id <> NEW.reservation\_id**

**AND (**

**(NEW.check\_in\_date < check\_out\_date) AND**

**(NEW.check\_out\_date > check\_in\_date)**

**)**

**LIMIT 1;**

**IF conflict\_room\_id IS NOT NULL THEN**

**SET @errorMsg = CONCAT(**

**'Комната ', conflict\_room\_id,**

**' уже забронирована с ', DATE\_FORMAT(conflict\_check\_in, '%d-%m-%Y'),**

**' по ', DATE\_FORMAT(conflict\_check\_out, '%d-%m-%Y'),**

**', id: ', conflict\_reservation\_id,**

**'. Невозможно оформить бронирование ', NEW.reservation\_id, '.');**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = @errorMsg;**

**END IF;**

**END$$**

Рисунок 12 - Триггер, проверяющий, что номер не занят на определенную дату (во время обновления таблицы)

Триггер check\_room\_availability выполняется перед вставкой новой записи в таблицу reservations. Он проверяет, что выбранный номер (room\_id) не занят на указанные даты заезда (check\_in\_date) и выезда (check\_out\_date). Если номер уже забронирован на пересекающийся период, триггер генерирует ошибку.

Аналогично работает check\_room\_availability\_on\_update. Триггер выполняется перед обновлением существующей записи в таблице reservations. Он проверяет, что выбранный номер не занят на новые даты заезда и выезда, исключая текущее бронирование из проверки. Если номер уже забронирован на пересекающийся период, триггер генерирует ошибку.

Эти триггеры предотвращают двойное бронирование одного номера на пересекающиеся периоды, что является критически важным для эффективного управления гостиницей.

Триггер calculate\_service\_price обеспечивает автоматический расчет стоимости заказанных услуг и добавление этой суммы к общей стоимости бронирования (Рисунок 13):

**CREATE TRIGGER calculate\_service\_price**

**BEFORE INSERT ON reservation\_services**

**FOR EACH ROW**

**BEGIN**

**DECLARE service\_price DECIMAL(10, 2);**

**DECLARE is\_service\_price\_fixed BOOLEAN;**

**DECLARE existing\_fixed\_service INT;**

**SELECT price, is\_price\_fixed INTO service\_price, is\_service\_price\_fixed**

**FROM services**

**WHERE service\_id = NEW.service\_id;**

**IF NOT is\_service\_price\_fixed THEN**

**SET NEW.total\_price = service\_price \* NEW.quantity;**

**ELSE**

**SET NEW.total\_price = service\_price;**

**END IF;**

**IF is\_service\_price\_fixed THEN**

**SELECT COUNT(\*) INTO existing\_fixed\_service**

**FROM reservation\_services**

**WHERE reservation\_id = NEW.reservation\_id AND service\_id = NEW.service\_id;**

**IF existing\_fixed\_service > 0 THEN**

**SET @errorMsg = CONCAT('Услуга ', NEW.service\_id, ' уже была добавлена в эту (', NEW.reservation\_id, ') бронь.');**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = @errorMsg;**

**END IF;**

**END IF;**

**UPDATE reservations**

**SET total\_service\_price = total\_service\_price + NEW.total\_price,**

**amount\_remaining = amount\_remaining + NEW.total\_price**

**WHERE reservation\_id = NEW.reservation\_id;**

**END$$**

Рисунок 13 - Триггер, считающий стоимость услуг

Он выполняется перед вставкой новой записи в таблицу reservation\_services. Его основная задача — рассчитать общую стоимость заказанной услуги (total\_price) на основе данных из таблицы services и обновить соответствующие поля в таблице reservations.

Логика работы:

1. из таблицы services извлекаются цена (price) и тип цены

(is\_price\_fixed) для указанной услуги (service\_id = NEW.service\_id);

1. расчет стоимости услуги:

2.1.) если услуга имеет переменную цену (не фиксированная), стоимость рассчитывается как произведение цены за единицу (service\_price) и количества/объема (NEW.quantity);

2.2.) если услуга имеет фиксированную цену, ее стоимость равна указанной цене (service\_price);

1. проверка дублирования фиксированных услуг:

3.1.) для услуг с фиксированной ценой проверяется, не была ли она

уже добавлена к данному бронированию (reservation\_id = NEW.reservation\_id AND service\_id = NEW.service\_id);

3.2.) если такая запись уже существует, генерируется ошибка;

1. в таблице reservations увеличивается общая стоимость

дополнительных услуг (total\_service\_price) и сумма к оплате (amount\_remaining) на значение NEW.total\_price.

Также, как уже было сказано раннее, вводится запрет на обновление и удаление заказанных услуг, так как в рамках данной работы не будет разрабатываться логика, отвечающая за возврат средств гостю (Рисунок 14 и Рисунок 15):

**CREATE TRIGGER prevent\_update\_reservation\_services**

**BEFORE UPDATE ON reservation\_services**

**FOR EACH ROW**

**BEGIN**

**SIGNAL SQLSTATE '45000'**

**SET MESSAGE\_TEXT = 'Обновление данных о заказанных услугах запрещено';**

**END$$**

Рисунок 14 - Запрет на обновление заказанных услуг

**CREATE TRIGGER prevent\_delete\_reservation\_services**

**BEFORE DELETE ON reservation\_services**

**FOR EACH ROW**

**BEGIN**

**SIGNAL SQLSTATE '45000'**

**SET MESSAGE\_TEXT = 'Удаление данных о заказанных услугах запрещено';**

**END$$**

Рисунок 15 - Запрет на удаление заказанных услуг

Триггер set\_total\_payment отвечает за установку значения в поле total таблицы payments (Рисунок 16):

**CREATE TRIGGER set\_total\_payment**

**BEFORE INSERT ON payments**

**FOR EACH ROW**

**BEGIN**

**DECLARE remaining DECIMAL(10, 2);**

**SELECT amount\_remaining INTO remaining**

**FROM reservations**

**WHERE reservation\_id = NEW.reservation\_id;**

**IF remaining = 0 THEN**

**SET @errorMsg = CONCAT('Нет неоплаченных счетов для: ', NEW.reservation\_id);**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = @errorMsg;**

**END IF;**

**IF NEW.total = 0 THEN**

**SET NEW.total = remaining;**

**ELSEIF NEW.total > remaining THEN**

**SET @errorMsg = CONCAT('Сумма платежа (', NEW.total,**

**') превышает оставшуюся сумму к оплате (', remaining,**

**') для бронирования с ID: ', NEW.reservation\_id, '.');**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = @errorMsg;**

**END IF;**

**END$$**

Рисунок 16 - Установка значения в поле total

Триггер set\_total\_payment обеспечивает корректность платежей при внесении данных в таблицу `payments`. Он проверяет оставшуюся сумму к оплате (amount\_remaining) для бронирования: если она равна нулю, генерирует ошибку (оплата невозможна для полностью оплаченных бронирований). При отсутствии явно указанной суммы платежа (`total = 0`) автоматически устанавливает её равной остатку. Если указанная сумма превышает остаток, триггер блокирует операцию, предотвращая переплату. Таким образом, триггер защищает финансовую целостность данных, оставляя возможность оплаты по частям, автоматизирует процесс оплаты и формализует бизнес-логику системы.

Триггер update\_amounts\_on\_payment обновляет значение полей таблицы reservations после проведения оплаты (Рисунок 17):

**CREATE TRIGGER update\_amounts\_on\_payment**

**AFTER INSERT ON payments**

**FOR EACH ROW**

**BEGIN**

**UPDATE reservations**

**SET amount\_paid = amount\_paid + NEW.total,**

**amount\_remaining = amount\_remaining - NEW.total**

**WHERE reservation\_id = NEW.reservation\_id;**

**END$$**

Рисунок 17 - Обновление значений amount\_paid и amount\_remaining

Триггеры prevent\_update\_payments и prevent\_delete\_payments запрещают обновлять и удалять данные из таблицы payments (Рисунок 18 и Рисунок 19):

**CREATE TRIGGER prevent\_update\_payments**

**BEFORE UPDATE ON payments**

**FOR EACH ROW**

**BEGIN**

**SIGNAL SQLSTATE '45000'**

**SET MESSAGE\_TEXT = 'Обновление данных об оплате запрещено';**

**END$$**

Рисунок 18 - Запрет на обновление данных в таблице payments

**CREATE TRIGGER prevent\_delete\_payments**

**BEFORE DELETE ON payments**

**FOR EACH ROW**

**BEGIN**

**SIGNAL SQLSTATE '45000'**

**SET MESSAGE\_TEXT = 'Удаление данных об оплате запрещено';**

**END$$**

Рисунок 19 - Запрет на удаление данных из таблицы payments

Реализованные триггеры создают надежную систему контроля и автоматизации ключевых бизнес-процессов гостиницы. При последующем заполнении таблиц все триггеры будут протестированы на практике, что позволит убедиться в их работоспособности и корректности.

### 3.3. Заполнение таблиц

Начнем по порядку, а именно с таблицы rooms. Добавим 18 номеров, передав в запросе необходимые данные (Рисунок 20):

**INSERT INTO rooms (number, double\_bed, single\_bed, class, area, price\_per\_night) VALUES**

**(1, 1, 0, 'Стандарт', 15, 3000),**

**(2, 1, 0, 'Стандарт', 15, 3000),**

**(3, 0, 2, 'Стандарт', 18, 3500),**

**(4, 0, 2, 'Стандарт', 18, 3500),**

**(5, 1, 2, 'Стандарт', 20, 4000),**

**(6, 1, 2, 'Стандарт', 20, 4000),**

**(7, 1, 0, 'Полулюкс', 35, 6000),**

**(8, 1, 0, 'Полулюкс', 35, 6000),**

**(9, 0, 2, 'Полулюкс', 38, 7500),**

**(10, 0, 2, 'Полулюкс', 38, 7500),**

**(11, 1, 2, 'Полулюкс', 40, 9000),**

**(12, 1, 2, 'Полулюкс', 40, 9000),**

**(13, 1, 0, 'Люкс', 55, 15000),**

**(14, 1, 0, 'Люкс', 55, 15000),**

**(15, 0, 2, 'Люкс', 58, 17500),**

**(16, 0, 2, 'Люкс', 58, 17500),**

**(17, 1, 2, 'Люкс', 60, 19000),**

**(18, 1, 2, 'Люкс', 60, 19000);**

Рисунок 20 - Заполнение таблицы rooms

Получившаяся таблица (Рисунок 21):

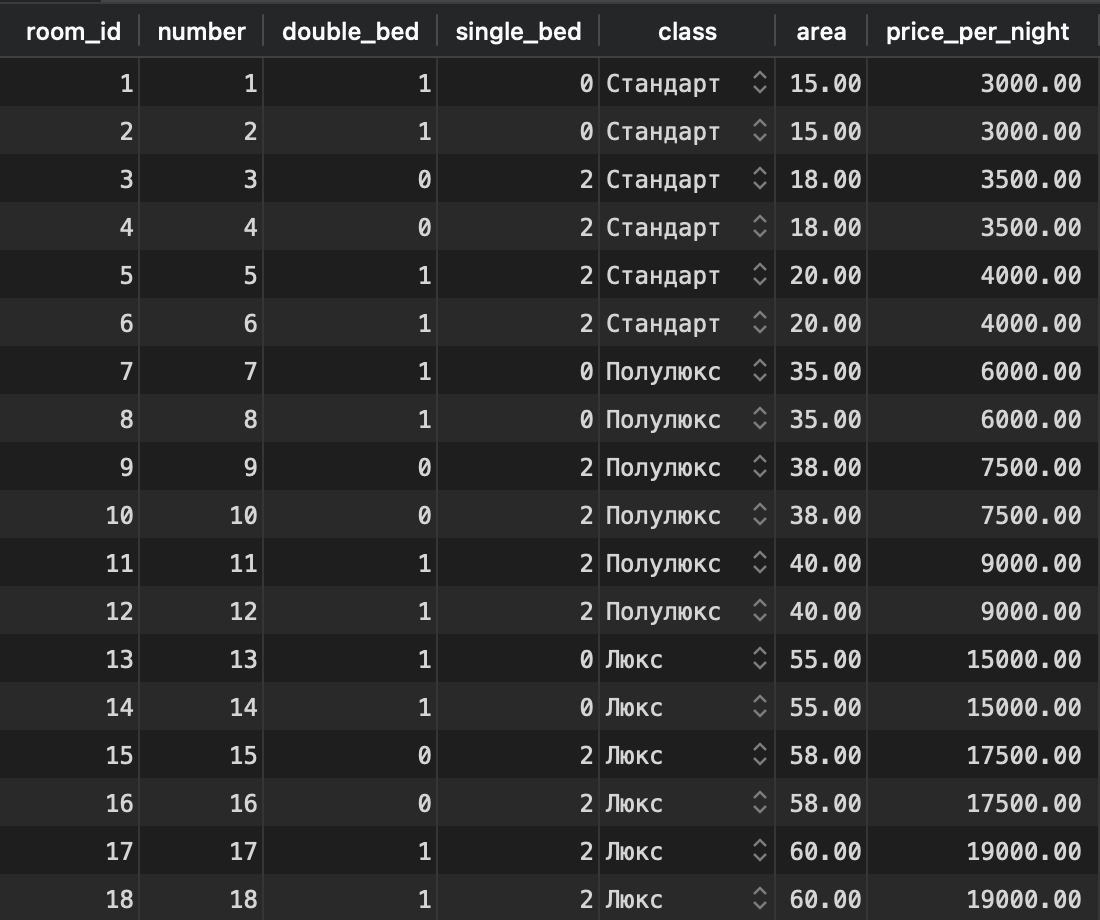


Рисунок 21- Заполненная таблица rooms

В таблицу guests добавим 20 данных (Рисунок 22):

**INSERT INTO guests (first\_name, last\_name, patronymic, passport, email, phone\_number) VALUES**

**('Иван', 'Иванов', 'Иванович', '1234 567890', 'ivan.ivanov@some\_mail.com', '+79001234567'),**

**('Мария', 'Петрова', 'Сергеевна', '2345 678901', 'maria.petrova@some\_gmail.com', '+79007654321'),**

**('Алексей', 'Сидоров', 'Алексеевич', '3456 789012', 'alexei.sidorov@some\_mail.com', '+79001112233'),**

**('Елена', 'Кузнецова', 'Дмитриевна', '4567 890123', 'elena.kuznetsova@some\_gmail.com', '+79009876543'),**

**('Дмитрий', 'Николаев', 'Андреевич', '5678 901234', 'dmitry.nikolaev@some\_mail.com', '+79003216549'),**

**('Ольга', 'Васильева', 'Викторовна', '6789 012345', 'olga.vasileva@some\_gmail.com', '+79004567890'),**

**('Андрей', 'Смирнов', 'Павлович', '7890 123456', 'andrey.smirnov@some\_mail.com', '+79007894561'),**

**('Татьяна', 'Морозова', 'Игоревна', '8901 234567', 'tatyana.morozova@some\_gmail.com', '+79002345678'),**

**('Сергей', 'Лебедев', 'Николаевич', '9012 345678', 'sergey.lebedev@some\_mail.com', '+79008765432'),**

**('Анна', 'Романова', 'Александровна', '0123 456789', 'anna.romanova@some\_gmail.com', '+79005551234'),**

**('Александр', 'Попов', 'Владимирович', '1122 334455', 'alexander.popov@some\_mail.com', '+79001111111'),**

**('Екатерина', 'Соколова', 'Анатольевна', '2233 445566', 'ekaterina.sokolova@some\_gmail.com', '+79002222222'),**

**('Максим', 'Кузнецов', 'Сергеевич', '3344 556677', 'maksim.kuznetsov@some\_mail.com', '+79003333333'),**

**('Олег', 'Федоров', 'Дмитриевич', '4455 667788', 'oleg.fedorov@some\_gmail.com', '+79004444444'),**

**('Наталья', 'Волкова', 'Олеговна', '5566 778899', 'natalya.volkova@some\_mail.com', '+79005555555'),**

**('Дарья', 'Иванова', 'Андреевна', '6677 889900', 'darya.ivanova@some\_gmail.com', '+79006666666'),**

**('Антон', 'Смирнов', 'Иванович', '7788 990011', 'anton.smirnov@some\_gmail.com', '+79007777777'),**

**('Юлия', 'Ковалева', 'Валерьевна', '8899 001122', 'yulia.kovaleva@some\_mail.com', '+79008888888'),**

**('Артем', 'Романов', 'Алексеевич', '9900 112233', 'artem.romanov@some\_gmail.com', '+79009999999'),**

**('Виктория', 'Лебедева', 'Сергеевна', '0011 223344', 'viktoria.lebedeva@some\_gmail.com', '+79000000000');**

Рисунок 22 - Заполнение таблицы guests

Результат (Рисунок 23):

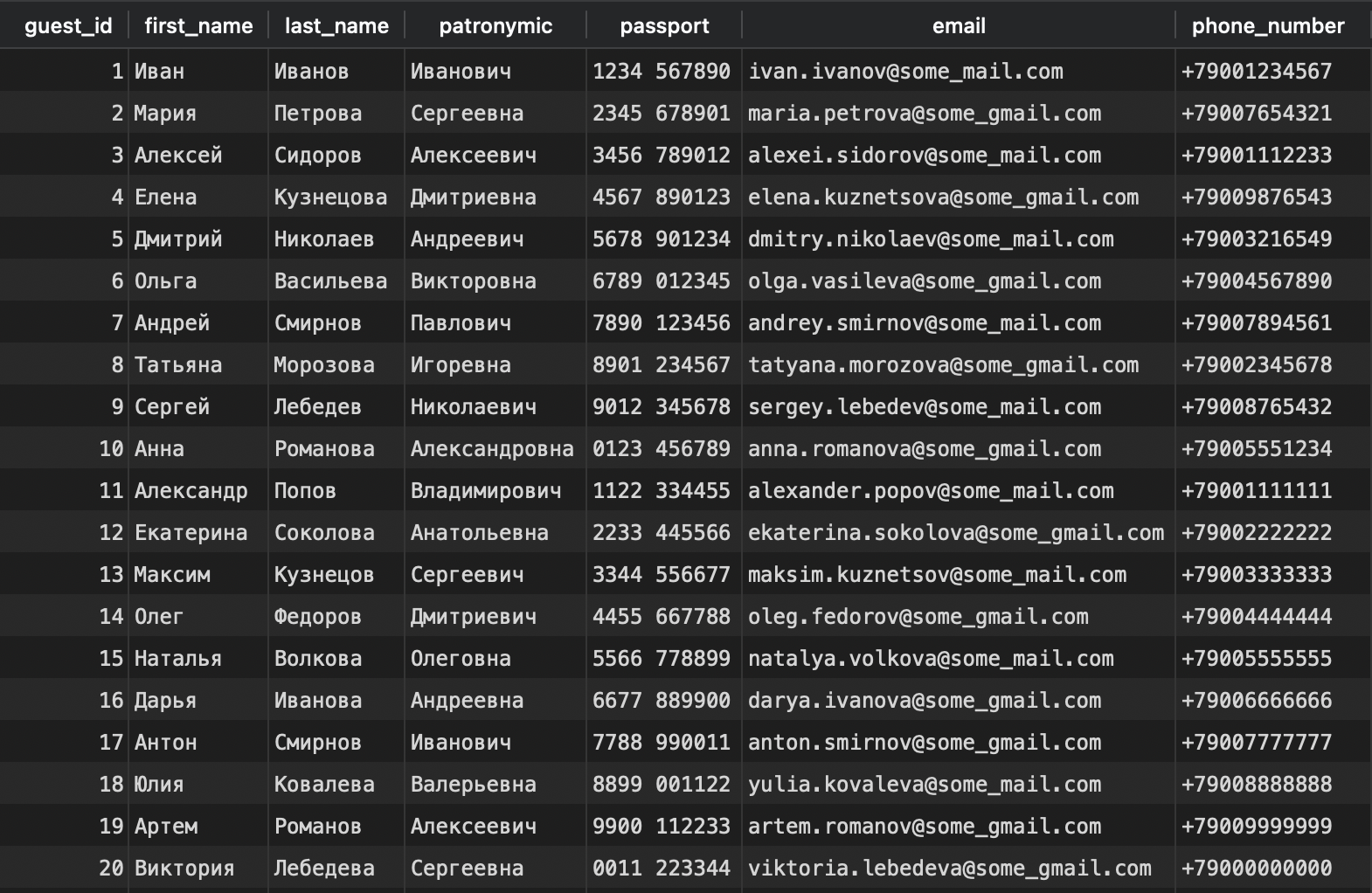


Рисунок 23 - Заполненная таблица guests

Для reservations заполнение будет выглядеть следующим образом (Рисунок 24):

**INSERT INTO reservations (guest\_id, room\_id, check\_in\_date, check\_out\_date) VALUES**

**(1, 1, '2025-03-10', '2025-04-15'),**

**(2, 2, '2025-03-02', '2025-03-11'),**

**(3, 3, '2025-03-01', '2025-03-02'),**

**(4, 4, '2025-03-04', '2025-04-10'),**

**(5, 5, '2025-03-04', '2025-04-09'),**

**(6, 6, '2025-03-02', '2025-03-10'),**

**(7, 7, '2025-03-01', '2025-03-11'),**

**(8, 8, '2025-03-08', '2025-03-12'),**

**(9, 9, '2025-03-09', '2025-03-13'),**

**(10, 10, '2025-03-10', '2025-03-19'),**

**(11, 11, '2025-04-15', '2025-04-18'),**

**(12, 12, '2025-03-13', '2025-03-20'),**

**(13, 13, '2025-03-17', '2025-03-21'),**

**(14, 14, '2025-04-10', '2025-04-11'),**

**(15, 15, '2025-04-10', '2025-04-11'),**

**(16, 16, '2025-04-15', '2025-04-18'),**

**(17, 17, '2025-03-13', '2025-03-20'),**

**(18, 18, '2025-03-17', '2025-03-21'),**

**(19, 1, '2025-04-15', '2025-04-19'),**

**(20, 2, '2025-04-09', '2025-04-23');**

Рисунок 24 - Заполнение таблицы reservation

Получим 20 записей (Рисунок 25):

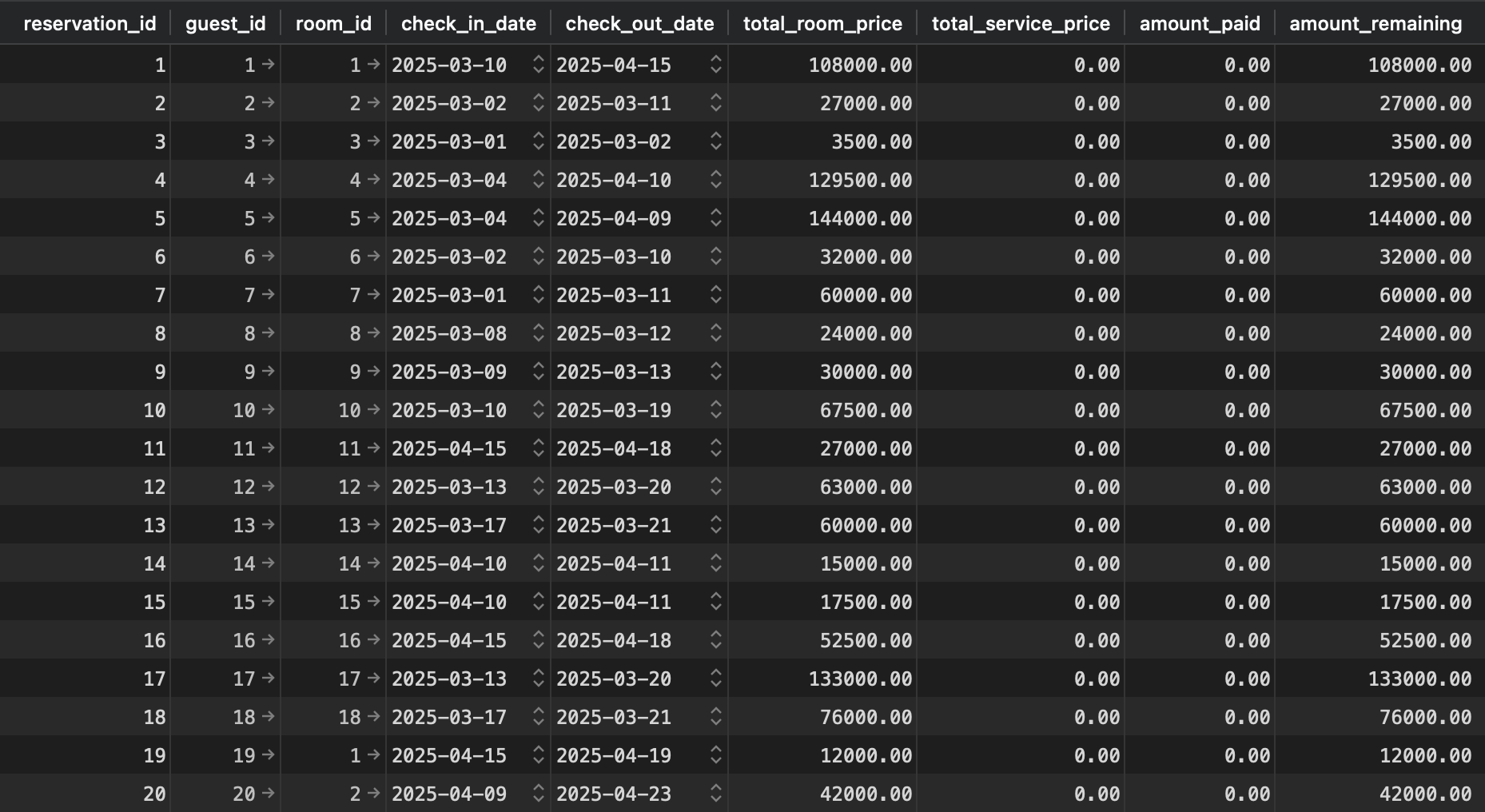


Рисунок 25 - Заполненная таблица reservation

Поле amount-remaining посчиталось самостоятельно благодаря триггеру calculate\_total\_room\_price. На этом этапе мы можем проверить, как отрабатывают ошибки. Попробуем добавить несколько броней, которые пересекаются с уже имеющейся (Рисунок 26):

**INSERT INTO reservations (guest\_id, room\_id, check\_in\_date, check\_out\_date) VALUES**

**(16, 16, '2025-04-13', '2025-04-17');**

**INSERT INTO reservations (guest\_id, room\_id, check\_in\_date, check\_out\_date) VALUES**

**(16, 16, '2025-04-16', '2025-04-17');**

**INSERT INTO reservations (guest\_id, room\_id, check\_in\_date, check\_out\_date) VALUES**

**(16, 16, '2025-04-16', '2025-04-20');**

**INSERT INTO reservations (guest\_id, room\_id, check\_in\_date, check\_out\_date) VALUES**

**(16, 16, '2025-04-14', '2025-04-19');**

Рисунок 26 - Проверка на ошибки

В первом случае происходит попытка добавить бронь, дата выезда которой до уже существующей брони, а дата выезда - после. Получим ошибку (Рисунок 27):



Рисунок 27 - Ошибка

Во втором случае происходит попытка снять номер внутри уже забронированных дат (Рисунок 28):



Рисунок 28 - Ошибка

В третьем случае происходит попытка забронировать номер, где дата въезда во время существующего пребывания, а дата выезда - после (Рисунок 29):



Рисунок 29 - Ошибка

В последнем случае происходит попытка снять номер, где даты пребывания включают в себя уже существующую бронь (Рисунок 30):



Рисунок 30 - Ошибка

Также проверим, что нельзя изменить бронирование, если новая итоговая стоимость меньше. Выполним два запроса (Рисунок 31 и Рисунок 32):

**UPDATE reservations**

**SET room\_id = 1**

**WHERE reservation\_id = 18;**

Рисунок 31 - Запрос на изменение комнаты

**UPDATE reservations**

**SET check\_in\_date = '2025-03-18'**

**WHERE reservation\_id = 18;**

Рисунок 32 - Запросы на изменение даты заезда

В первом случае получим ошибку (Рисунок 33):



Рисунок 33 - Ошибка

Во втором случае так же получим ошибку (Рисунок 34):



Рисунок 34 - Ошибка

Таким образом была протестирована работоспособность триггеров. Можно убедиться, что бронирования защищены от некорректного заполнения и обновления информации.

Заполним информацию о предоставляемых гостиницей услугах. Cначала заполним услуги с фиксированной ценой (Рисунок 35):

**INSERT INTO services (service\_name, description, price) VALUES**

**('Неограниченный доступ к бассейну', 'Наслаждайтесь неограниченным доступом к бассейну, включая шезлонги и бар у бассейна.', 5000.00),**

**('Неограниченный доступ в тренажерный зал', 'Получите доступ к тренажерному залу с возможностью занятий с личным тренером.', 3000.00),**

**('Неограниченный доступ в СПА', 'Полный доступ ко всем услугам СПА, включая массажи, сауну и другие процедуры.', 10000.00),**

**('Детская кроватка', 'Предоставление детской кроватки в номер для вашего малыша.', 1000.00),**

**('Трансфер из/в аэропорт', 'Комфортабельный трансфер на автомобиле бизнес-класса от аэропорта.', 4500.00),**

**('Личный доступ в Интернет', 'Подключим в вашем номере персональный высокоскоростной интернет для комфортного использования.', 5000.00);**

Рисунок 35 - Заполнение услуг с фиксированной ценой

И услуги с нефиксированной ценой (Рисунок 36):

**INSERT INTO services (service\_name, description, price, is\_price\_fixed) VALUES**

**('Завтрак в номер', 'Наслаждайтесь вкусным завтраком, который мы доставим в ваш номер в удобное для вас время (с 8:00 до 11:00).', 1500.00, FALSE),**

**('Обед в номер', 'Закажите обед, и мы принесем его в ваш номер в выбранное вами время (с 13:00 до 16:00).', 1500.00, FALSE),**

**('Ужин в номер', 'Ужин в уютной обстановке вашего номера. Доставка в удобное для вас время (с 18:00 до 21:00).', 1500.00, FALSE),**

**('Организация экскурсий', 'Мы полностью организуем для вас экскурсию в любое выбранное вами место. Подробности уточняйте у администратора.', 15000.00, FALSE),**

**('Стирка и химчистка', 'Наши специалисты позаботятся о том, чтобы ваши вещи были безупречно чистыми и ухоженными. Цена указана за минимальный заказ.', 3000.00, FALSE),**

**('Почасовая аренда конференц-зала', 'Идеальное пространство для проведения мероприятий.', 15000.00, FALSE),**

**('Услуги няни', 'Профессиональная няня позаботится о вашем ребенке в удобное для вас время. Минимальное время заказа — 2 часа.', 4000.00, FALSE),**

**('Почасовая аренда автомобиля', 'Предоставление автомобиля на выбор: эконом, комфорт или премиум класс. Подробности уточняйте у администратора.', 3500.00, FALSE),**

**('Фотосессия в отеле', 'Профессиональный фотограф организует для вас фотосессию в интерьерах отеля или на территории. Включает 10 обработанных фото. Оплата почасовая.', 7500.00, FALSE);**

Рисунок 36 - Заполнение услуг с нефиксированной ценой

Получим 15 записей (Рисунок 37):

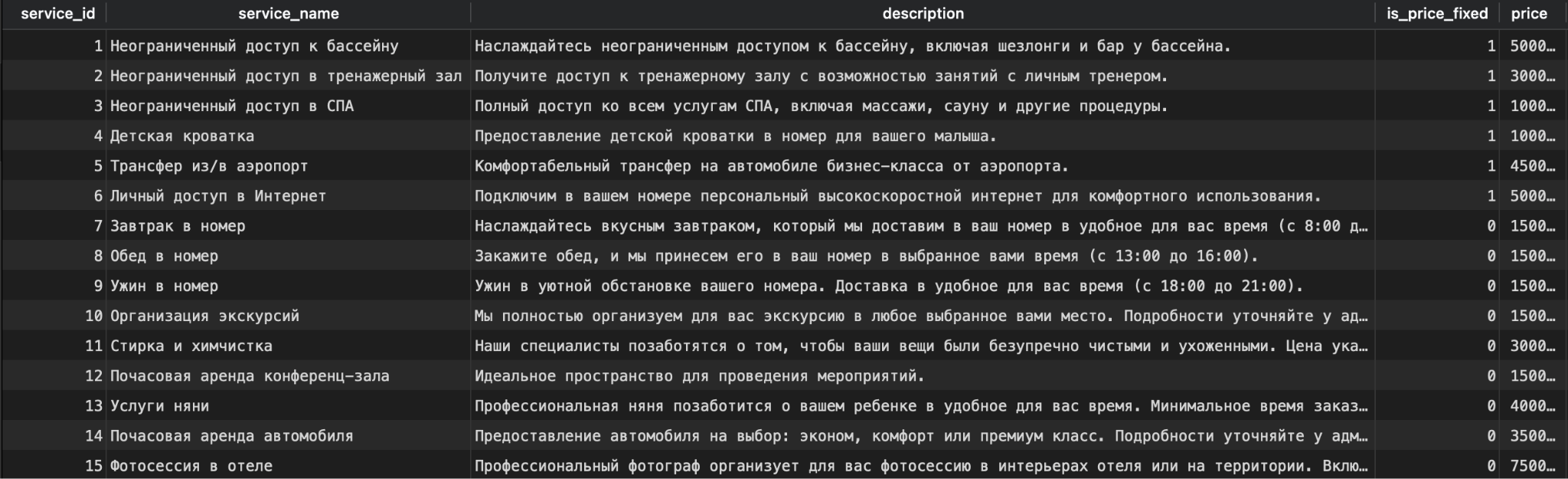


Рисунок 37 - Заполненная таблица services

Добавим услуги к бронированиям. Сперва добавим услуги с фиксированной ценой (Рисунок 38):

**INSERT INTO reservation\_services (reservation\_id, service\_id) VALUES**

**(1, 1),**

**(1, 2),**

**(1, 3),**

**(1, 6),**

**(2, 1),**

**(3, 1),**

**(4, 1),**

**(5, 2),**

**(6, 3),**

**(7, 3),**

**(8, 6),**

**(8, 4),**

**(9, 1),**

**(10, 1);**

Рисунок 38 - Добавление услуг с фиксированной ценой

Также добавим услуги с нефиксированной ценой (Рисунок 39):

**INSERT INTO reservation\_services (reservation\_id, service\_id, quantity) VALUES**

**(1, 7, 2),**

**(1, 8, 1),**

**(1, 9, 1),**

**(2, 14, 5),**

**(2, 13, 6),**

**(2, 9, 2),**

**(3, 7, 1),**

**(3, 11, 1),**

**(3, 12, 4),**

**(4, 8, 2),**

**(4, 11, 1),**

**(4, 15, 3),**

**(5, 7, 2),**

**(5, 10, 2),**

**(5, 8, 2),**

**(6, 7, 2),**

**(6, 8, 2),**

**(6, 10, 2);**

Рисунок 39 - Добавление услуг с нефиксированной ценой

Теперь таблица reservations выглядит следующим образом (Рисунок 40):

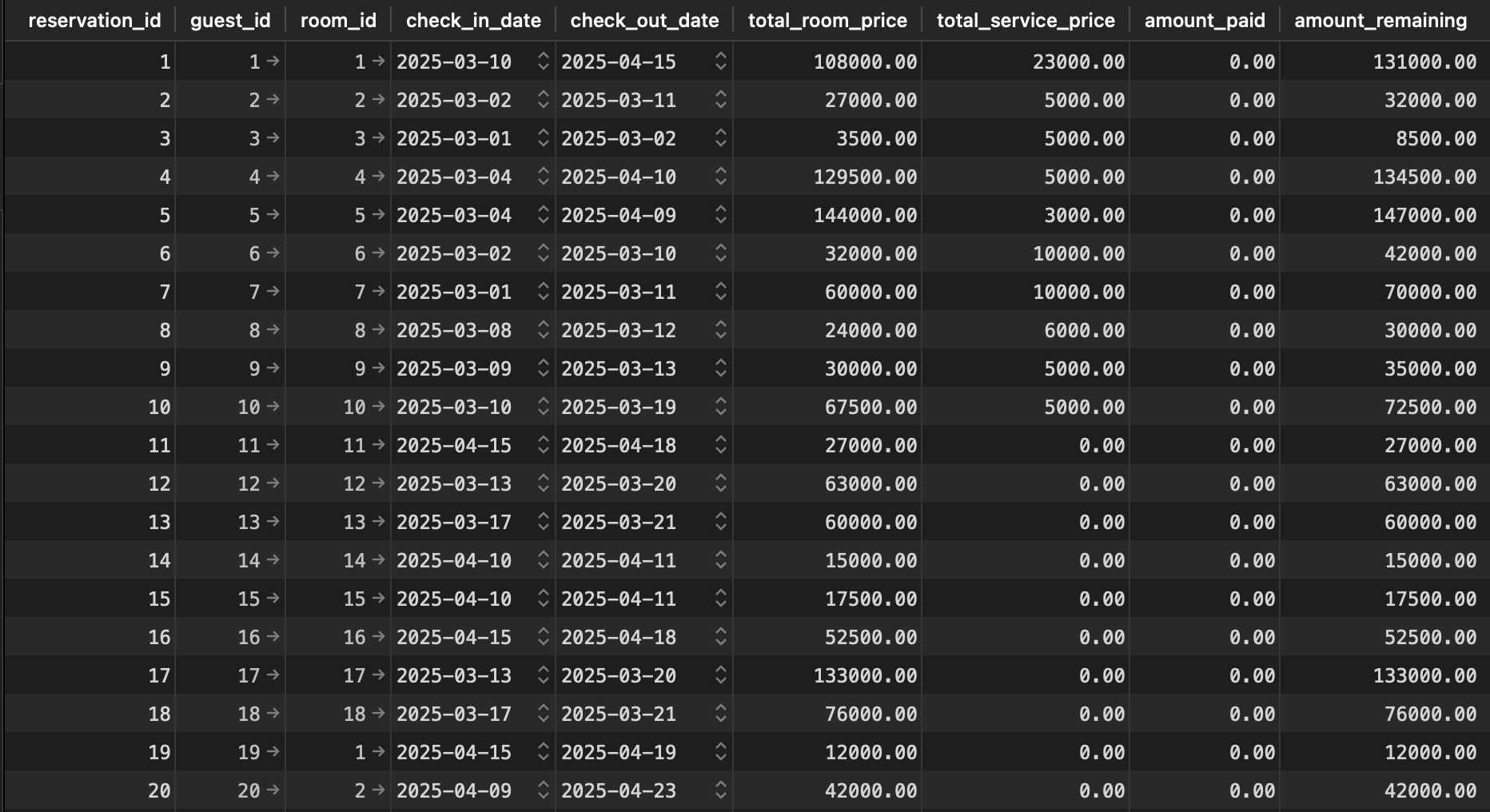


Рисунок 40 - Таблица reservations с добавленными услугами

Поле total\_service\_price автоматически посчиталось, а поле amount\_remaining теперь содержит сумму значений полей total\_room\_price и total\_service\_price.

Проверим, что в одно бронирование нельзя добавить более одной услуги с фиксированной ценой. Попробуем добавить еще один абонемент в бассейн к бронированию 1 (Рисунок 41):

**INSERT INTO reservation\_services (reservation\_id, service\_id) VALUES**

**(1, 1);**

Рисунок 41 - Добавление услуги с фиксированной ценой

Получим ошибку (Рисунок 42):



Рисунок 42 - Ошибка

Осталась последняя таблица - payments. Добавим несколько платежей (Рисунок 43):

**INSERT INTO payments (reservation\_id, payment\_date, payment\_method)** **VALUES**

**(1, '2025-03-10', 'Карта'),**

**(2, '2025-03-01', 'Онлайн'),**

**(3, '2025-03-01', 'Карта'),**

**(4, '2025-03-04', 'Наличные'),**

**(5, '2025-03-04', 'Карта'),**

**(6, '2025-03-02', 'Карта'),**

**(7, '2025-03-01', 'Карта'),**

**(8, '2025-03-08', 'Наличные'),**

**(9, '2025-03-09', 'Карта'),**

**(10, '2025-03-01', 'Онлайн'),**

**(11, '2025-03-01', 'Онлайн'),**

**(12, '2025-03-17', 'Онлайн'),**

**(13, '2025-03-08', 'Наличные'),**

**(14, '2025-04-10', 'Наличные'),**

**(15, '2025-04-10', 'Карта');**

Рисунок 43 - Заполнение таблицы payments

В результате имеем (Рисунок 44):

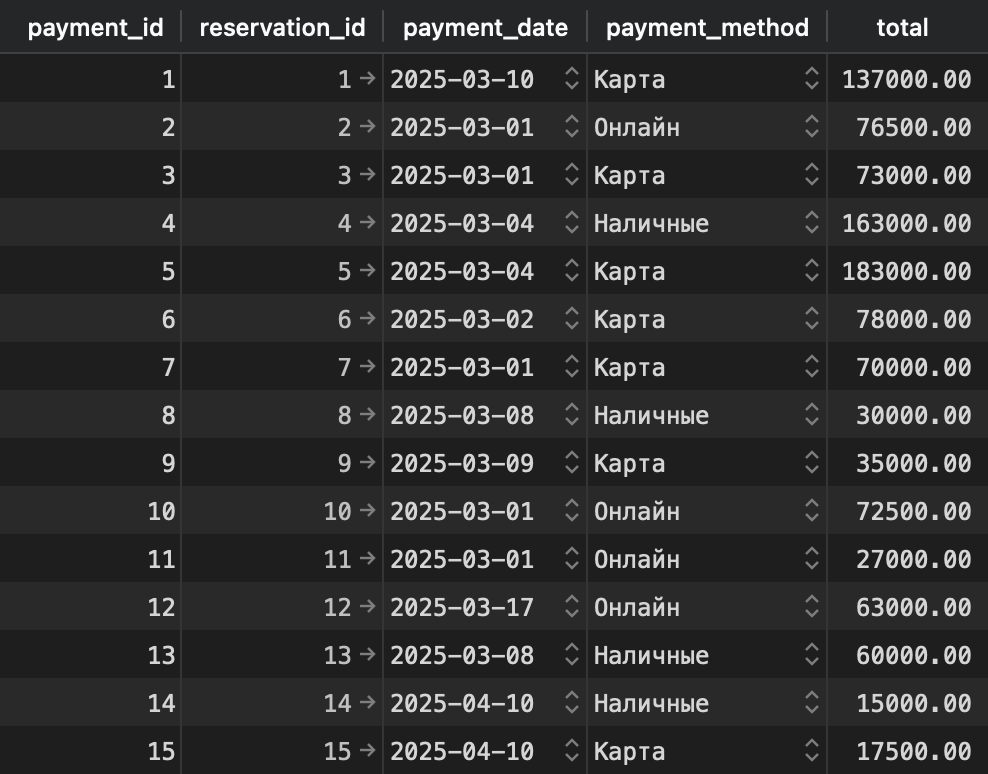


Рисунок 44 - Заполненная таблица payments

Поскольку при добавлении записей в payments не были явно указаны значения для total, благодаря триггеру они приняли соответствующие значение из amount\_remaiming.

Таблица reservations обновилась и выглядит следующим образом (Рисунок 45):

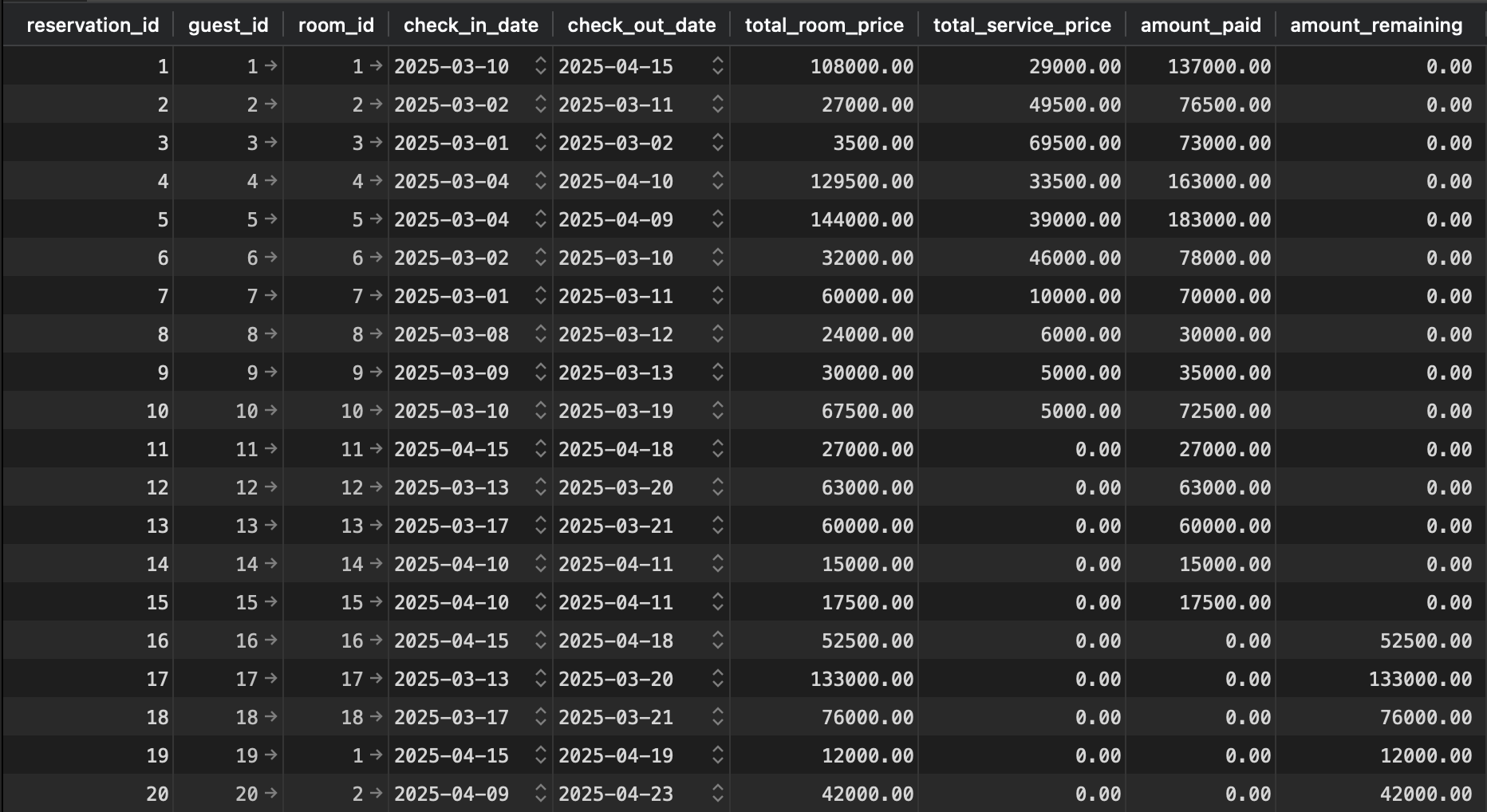


Рисунок 45 - Таблица reservations с некоторыми оплаченными счетами

Так же проведем несколько проверок.

Попробуем оплатить счет, положив в total значение, превышающее amount\_remainig (Рисунок 46):

**INSERT INTO payments (reservation\_id, payment\_date, payment\_method, total) VALUES**

**(16, '2025-03-10', 'Карта', 55000);**

Рисунок 46 - Добавление платежа к бронированию

Если мы запустим этот скрипт, то получим ошибку (Рисунок 47):



Рисунок 47 - Ошибка

Проведем еще одну проверку. Попробуем оплатить уже оплаченное бронирование (Рисунок 48):

**INSERT INTO payments (reservation\_id, payment\_date, payment\_method) VALUES**

**(1, '2025-03-10', 'Карта');**

Рисунок 48 - Добавление платежа к бронированию

Получим ошибку (Рисунок 49):



Рисунок 49 - Ошибка

## SQL-запросы

### 4.1. Запрос на выбор всех данных по двум полям таблицы

Выведем фамилию гостя и его номер телефона с помощью запроса (Рисунок 50):

**SELECT**

**last\_name AS 'Фамилия гостя',**

**phone\_number AS 'Телефон для связи'**

**FROM guests;**

Рисунок 50 - Запрос на выбор всех данных по двум полям таблицы

Запрос возвращает таблицу, содержащую фамилии гостей и их номера телефонов (Рисунок 51):

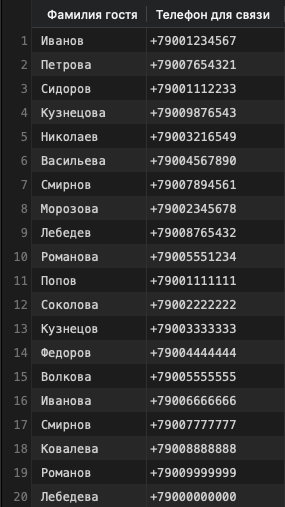


Рисунок 51 - Результат работы запроса на выбор данных по двум полям таблицы

Таким образом, данный запрос позволяет получить список гостей с их контактными данными.

### 4.2. Запрос на выбор всех неповторяющихся данных по одному полю таблицы

Для получения всех уникальных (неповторяющихся) значений по одному полю в таблице MySQL используется ключевое слово DISTINCT. Оно позволяет выбрать только уникальные значения из указанного столбца.

Получим список уникальных цен за ночь для всех номеров в отеле из таблицы rooms (Рисунок 52):

**SELECT DISTINCT**

**price\_per\_night AS 'Цены номеров за ночь'**

**FROM rooms;**

Рисунок 52 - Запрос для получения уникальных значений столбца

Запрос возвращает список уникальных цен за ночь для всех номеров в отеле (Рисунок 53):



Рисунок 53 - Результат работы запроса на выбор всех неповторяющихся данных по одному полю таблицы

Таким образом, данный запрос позволяет получить уникальные значения стоимости номеров за ночь.

### 4.3. Запрос на выбор всех полей и записей таблицы,сгруппированных по значению одного поля, с использованием условия на группу (секции GROUP BY, HAVING) и с заголовками колонок, заданными в запросе

Рассмотрим запрос к таблице reservations. Выберем все бронирования, сгруппированные по room\_id, и выведем только те группы, где итоговая стоимость бронирования комнаты превышает определенное значение (Рисунок 54):

**SELECT**

**room\_id AS 'Номер комнаты',**

**COUNT(reservation\_id) AS 'Количество бронирований',**

**SUM(total\_room\_price) AS 'Общая стоимость бронирований'**

**FROM reservations**

**GROUP BY room\_id**

**HAVING SUM(total\_room\_price) > 50000;**

Рисунок 54 - Запрос с GROUP BY и HAVING

Запрос возвращает таблицу, содержащую номер комнаты, количество бронирований для каждой комнаты и общую стоимость бронирований для каждой комнаты (Рисунок 55):

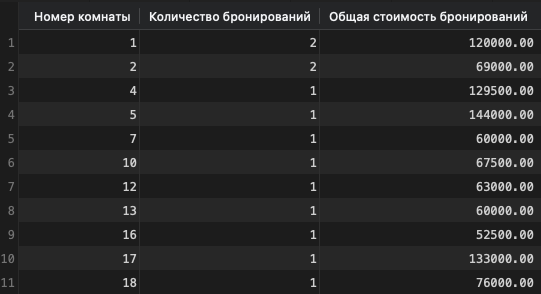


Рисунок 55 - Результат работы запроса с использованием группировки и условием на группу

Таким образом, данный запрос позволяет анализировать данные о бронированиях, выделяя только те комнаты, где общая стоимость бронирований превышает заданное значение. Это может быть полезно для выявления наиболее доходных номеров в отеле.

### 4.4. Запрос на выбор всех неповторяющихся записей по одному полю таблицы с колонкой, образованной агрегирующей функцией SUM и озаглавленной в соответствии со смыслом

Для таблицы reservation\_services составим запрос, который выбирает все уникальные бронирования (reservation\_id) и рассчитывает общую стоимость всех услуг, заказанных в рамках каждого бронирования. Благодаря использованию секции GROUP BY, значения в результирующей таблице будут уникальными для каждого бронирования (Рисунок 56):

**SELECT**

**reservation\_id AS 'ID бронирования',**

**SUM(total\_price) AS 'Итого стоимость услуг'**

**FROM reservation\_services**

**GROUP BY reservation\_id;**

Рисунок 56 - Запрос с колонкой, образованной агрегирующей функцией SUM

Запрос возвращает таблицу, содержащую уникальные идентификаторы бронирований и общую стоимость услуг, заказанных в рамках каждого бронирования (Рисунок 57):

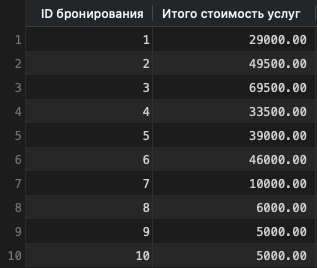


Рисунок 57 - Результат работы запроса

Таким образом, данный запрос позволяет получить информацию о стоимости услуг, заказанных в рамках каждого бронирования. Благодаря использованию GROUP BY, значения в результирующей таблице будут уникальными для каждого reservation\_id, что упрощает анализ данных.

### 4.5. Выбор нескольких (не всех) полей таблицы, отсортированных по убыванию

Для таблицы reservations выведем столбцы reservation\_id, guest\_id и отсортированный по убыванию amount\_remaining (Рисунок 58):

**SELECT**

**reservation\_id AS 'ID бронирования',**

**guest\_id AS 'ID гостя',**

**amount\_remaining AS 'Осталось оплатить'**

**FROM reservations**

**ORDER BY amount\_remaining DESC;**

Рисунок 58 - Запрос с использованием сортировки по убыванию

На выходе получим табличное представление данных, демонстрирующее информацию о неоплаченных остатках по всем бронированиям в удобном для анализа формате (Рисунок 59):

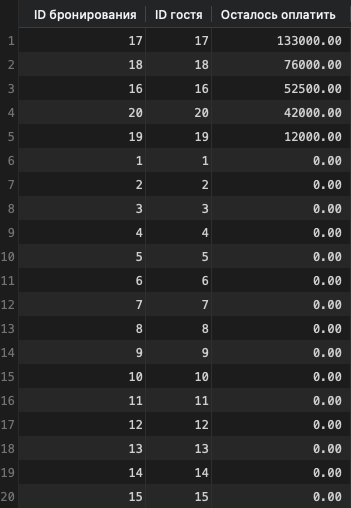


Рисунок 59 - Результат работы запроса с сортировкой

### 4.6. Выбор произвольного количества полей таблицы с добавлением поля, являющегося результатом арифметического выражения, в котором участвуют значения поля таблицы

Для таблицы reservations составим запрос, который выбирает произвольное количество полей, а также добавляет новое поле, представляющее собой результат арифметического выражения. В данном случае вычисляется общая стоимость бронирования номера, включая стоимость дополнительных услуг (Рисунок 60):

**SELECT**

**reservation\_id AS 'ID бронирования',**

**guest\_id AS 'ID гостя',**

**total\_room\_price AS 'Итого за номер',**

**total\_service\_price AS 'Итого за услуги',**

**total\_room\_price + total\_service\_price AS 'Общая стоимость'**

**FROM reservations**

**ORDER BY reservation\_id;**

Рисунок 60 - Запрос с вычислением общей стоимости

Запрос возвращает таблицу, содержащую идентификаторы бронирования, идентификаторы гостей, стоимость номера, стоимость услуг и общую стоимость бронирования (Рисунок 61):

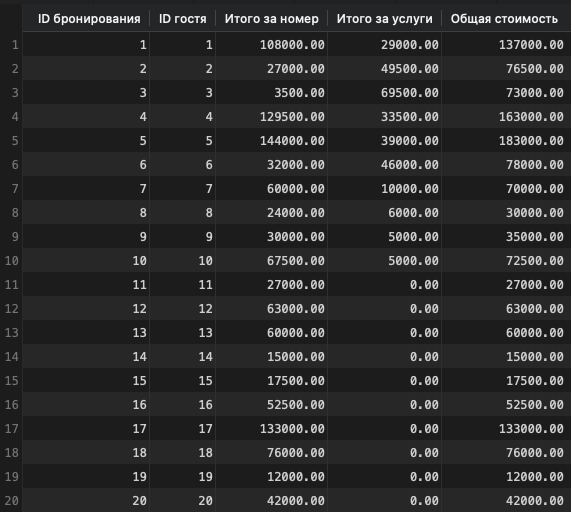


Рисунок 61 - Результат запроса с использованием арифметического выражения для столбца

Таким образом, данный запрос позволяет получить детализированную информацию о бронированиях, включая расчет общей стоимости на основе суммы стоимости номера и дополнительных услуг.

### 4.7. Запрос на выбор всех записей по одному полю таблицы с колонкой, образованной агрегирующей функцией SUM и озаглавленной в соответствии со смыслом; выбор записей с использованием условия диапазона (between)

Для таблицы payments составим запрос, который вычисляет общую сумму оплат за указанный период времени (Рисунок 62):

**SELECT**

**payment\_id AS 'ID платежа',**

**SUM(total) AS 'Общая сумма оплат'**

**FROM payments**

**WHERE payment\_date BETWEEN '2025-03-01' AND '2025-03-31'**

**GROUP BY payment\_id;**

Рисунок 62 - Запрос с вычислением общей суммы оплат за март

Оператор BETWEEN работает включительно, то есть в диапазон попадают все записи, начиная с 1 марта 2025 года и заканчивая 31 марта 2025 года.

Запрос возвращает таблицу, содержащую идентификаторы платежей и соответствующие им общие суммы оплат за указанный период (Рисунок 63):

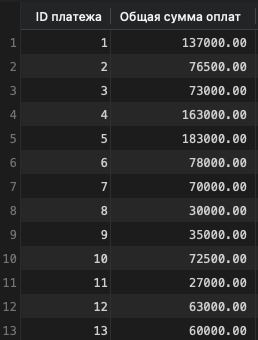


Рисунок 63 - Результат работы запроса с функцией SUM и условием диапозона BETWEEN

Таким образом, данный запрос позволяет анализировать общую сумму оплат за выбранный период времени с группировкой по идентификаторам платежей.

### 4.8. Запрос на выбор всех записей по произвольному количеству полей таблицы с использованием агрегирующей функции AVG и условием на отбор записей, заданном в секции WHERE

Для таблицы rooms составим запрос, который вычисляет среднюю стоимость за ночь для номеров, способных разместить четырех человек (одна двуспальная кровать и две односпальные кровати), с группировкой по классу номера. (Рисунок 64):

**SELECT**

**class,**

**AVG(price\_per\_night) AS 'Средняя стоимость за ночь'**

**FROM rooms**

**WHERE double\_bed = 1 AND single\_bed = 2**

**GROUP BY class;**

Рисунок 64 - Запрос с использованием функции AVG и условием WHERE

Запрос возвращает таблицу, содержащую классы номеров и соответствующую среднюю стоимость за ночь для указанных условий (Рисунок 65):

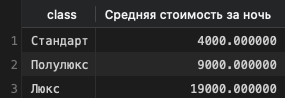


Рисунок 65 - Результат выполнения запроса с функцией AVG и условием WHERE

Таким образом, данный запрос позволяет анализировать среднюю стоимость номеров, удовлетворяющих заданным условиям, с группировкой по классу.

### 4.9. Запрос на выбор двух полей таблицы с вычислением третьего поля по данным таблицы и сортировкой по убыванию по первому полю, по возрастанию по второму и по убыванию по третьему

Для таблицы reservations составим запрос, который вычисляет общую сумму оплат по каждой комнате с учетом услуг, добавленных к бронированию. Запрос включает два поля из таблицы (room\_id, reservation\_id) и вычисляет третье поле как сумму total\_price всех связанных услуг для каждой брони. Запрос сортирует результаты по убыванию общей стоимости, затем по возрастанию ID комнаты и по убыванию ID бронирования (Рисунок 66):

**SELECT**

**room\_id AS 'ID комнаты',**

**reservation\_id AS 'ID бронирования',**

**(total\_room\_price + total\_service\_price) AS 'Общая стоимость бронирования'**

**FROM reservations**

**ORDER BY**

**'Общая стоимость бронирования' DESC,**

**room\_id ASC,**

**reservation\_id DESC;**

Рисунок 66 - Запрос с сортировкой по нескольким полям

Запрос возвращает таблицу, содержащую ID комнаты, ID бронирования и соответствующую общую стоимость бронирования для каждой записи (Рисунок 67):



Рисунок 67 - Результат работы запроса с сортировкой по трем столбцам

### 4.10. Запрос на выборку данных таблицы с условием сравнения по шаблону LIKE

Для таблицы guests составим запрос, который выбирает имена и фамилии гостей, чьи фамилии заканчиваются на букву "а". Это позволяет предположительно отобрать женщин, основываясь на характерной особенности русских фамилий (Рисунок 68):

**SELECT**

**first\_name AS 'Имя',**

**last\_name AS 'Фамилия'**

**FROM guests**

**WHERE last\_name LIKE '%а';**

Рисунок 68 - Запрос с условием сравнения LIKE

Запрос возвращает таблицу, содержащую имена и фамилии гостей, чьи фамилии заканчиваются на "а" (Рисунок 69):

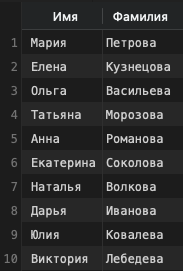


Рисунок 69 - Результат работы запроса с условием сравнения LIKE

Таким образом, данный запрос помогает предварительно определить гостей, вероятно являющихся женщинами, основываясь на характеристике их фамилий. Хотя метод не является абсолютно точным (например, существуют исключения в виде мужских фамилий, оканчивающихся на "а", или женских фамилий, не оканчивающихся на "а"), он может быть полезен для быстрой сегментации клиентов в маркетинговых или аналитических целях.

### 4.11. Запрос с отбором по условию и сортировкой по убыванию одного из полей, а также добавлением поля, содержащего для всех записей константу, определенную при конструировании запроса

Составим запрос, который выбирает все активные бронирования (т.е. те, у которых дата выезда еще не наступила или равна текущей дате) и рассчитывает общую стоимость бронирования как сумму стоимости номера (total\_room\_price) и стоимости услуг (total\_service\_price). Для каждой записи добавляется поле Статус с константным значением "Активное бронирование". Результаты сортируются по убыванию общей стоимости бронирования (Рисунок 70):

**SELECT**

**reservation\_id AS 'ID бронирования',**

**check\_in\_date AS 'Дата заезда',**

**check\_out\_date AS 'Дата выезда',**

**total\_service\_price + total\_room\_price AS 'Общая стоимость бронирования',**

**'Активное бронирование' AS 'Статус'**

**FROM reservations**

**WHERE check\_out\_date >= CURRENT\_DATE()**

**ORDER BY total\_service\_price + total\_room\_price DESC;**

Рисунок 70 - Запрос с отбором по условию, сортировкой и добавлением константы

Получим следующий результат на текущую дату 23 февраля 2025 года (Рисунок 71):

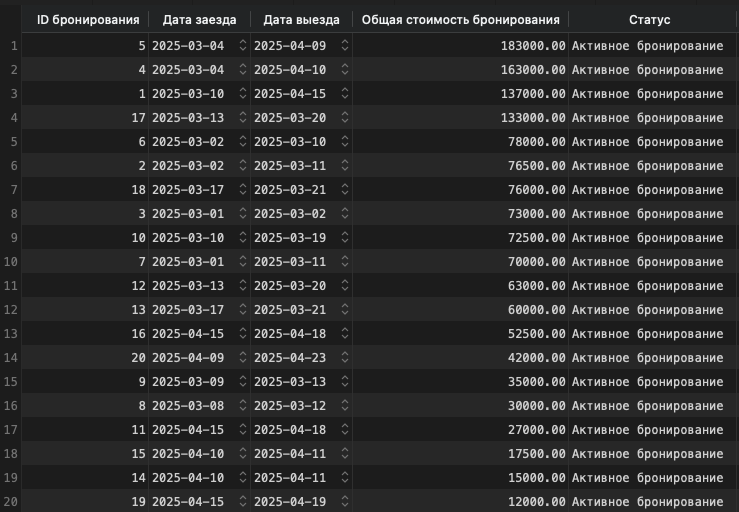


Рисунок 71 - Результат выполнения запроса

Такой запрос позволяет быстро определить наиболее дорогие активные бронирования, что может быть полезно для анализа доходов отеля или планирования ресурсов. Добавление статуса "Активное бронирование" делает результат более информативным для пользователей.

### 4.12. Запрос с использованием агрегирующих функций MIN и MAX

Напишем запрос, который использует агрегирующие функции MIN и MAX для определения ближайшего (MIN) и наиболее позднего (MAX) бронирования по дате заезда (check\_in\_date). Также добавим дополнительное условие, что рассматриваемые даты не должны быть раньше текущего дня (Рисунок 72):

**SELECT**

**MIN(check\_in\_date) AS 'Ближайшее бронирование',**

**MAX(check\_in\_date) AS 'Самое позднее бронирование'**

**FROM reservations**

**WHERE check\_in\_date >= CURRENT\_DATE();**

Рисунок 72 - Запрос с использованием агрегирующих функций MIN и MAX

В качестве результата выводятся две записи: первая — самая ранняя ближайшая дата заезда, вторая — самая поздняя дата заезда (Рисунок 73):



Рисунок 73 - Результат выполнения запроса

Результат запроса зависит от текущей даты, так что стоит отметить, что запрос был выполнен 23 февраля 2025 года.

### 4.13. Запрос с использованием сложного условия с логическими операторами AND, OR и сортировкой

Напишем такой запрос, который выбирает все бронирования, соответствующие следующим условиям:

1. продолжительность бронирования составляет ровно 1 день, и

общая стоимость бронирования превышает или равна 50 000 рублей;

1. или выбираются бронирования, где средняя дневная стоимость

бронирования (включая услуги) составляет минимум 15 000 рублей.

Результаты сортируются по возрастанию даты заезда и по убыванию общей стоимости бронирования. Запрос будет выглядеть следующим образом (Рисунок 74):

**SELECT**

**reservation\_id AS 'ID бронирования',**

**guest\_id AS 'ID гостя',**

**check\_in\_date AS 'Дата заезда',**

**check\_out\_date AS 'Дата выезда',**

**total\_room\_price + total\_service\_price AS 'Общая стоимость бронирования'**

**FROM reservations**

**WHERE**

**(DATEDIFF(check\_out\_date, check\_in\_date) = 1 AND (total\_room\_price + total\_service\_price) >= 50000)**

**OR ((total\_room\_price + total\_service\_price) >= DATEDIFF(check\_out\_date, check\_in\_date) \* 15000)**

**ORDER BY**

**check\_in\_date ASC,**

**total\_room\_price + total\_service\_price DESC;**

Рисунок 74 - Запрос с логическими операторами OR, AND и сортировкой

На выводе получим (Рисунок 75):

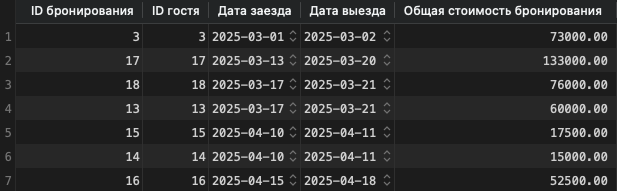


Рисунок 75 - Результат работы запроса

Таким образом, данный запрос позволяет анализировать бронирования, которые либо имеют высокую стоимость за короткий период (1 день, более 50 000 рублей), либо обеспечивают высокую дневную доходность (минимум 15 000 рублей за ночь).

# ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы была разработана и реализована информационная система для управления гостиничным комплексом. Основной целью проекта было создание базы данных, которая обеспечивает автоматизацию процессов бронирования номеров, учета гостей, управления услугами и обработки платежей. В результате проведенного анализа предметной области были определены ключевые требования к системе, а также разработана структура базы данных, которая включает в себя таблицы для хранения информации о номерах, гостях, бронированиях, услугах и платежах.

Были спроектированы таблицы, отражающие основные сущности предметной области, такие как номера, гости, бронирования, услуги и платежи. Для каждой таблицы были определены типы данных, обеспечивающие точность и целостность информации. Также были установлены связи между таблицами по принципу «один-ко-многим», что позволяет эффективно управлять данными и избегать дублирования информации.

Для автоматизации ключевых процессов и обеспечения целостности данных были разработаны триггеры, которые автоматически рассчитывают стоимость бронирования, проверяют доступность номеров, контролируют корректность ввода данных и предотвращают ошибки, связанные с двойным бронированием или некорректными датами. Триггеры также обеспечивают автоматическое обновление финансовых данных при внесении изменений в бронирования или платежи.

Таблицы были заполнены тестовыми данными, что позволило проверить работоспособность триггеров и корректность работы базы данных. В процессе заполнения были протестированы различные сценарии, включая попытки двойного бронирования, добавление услуг с фиксированной и переменной ценой, а также обработку платежей.

Были написаны SQL-запросы, соответствующие заданию.

Разработанная информационная система гостиницы позволяет:

* упростить процесс бронирования для клиентов за счет

автоматизации рутинных операций;

* оптимизировать управление номерами и услугами, что повышает

эффективность работы гостиницы;

* обеспечить прозрачность финансовых операций и отчетности, что

способствует улучшению контроля над доходами и расходами;

* повысить качество обслуживания за счет оперативного доступа к

актуальной информации о гостях, бронированиях и услугах.

# 

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Руководство по стилю SQL [Электронный ресурс]. – URL: <https://www.sqlstyle.guide/ru/> (дата обращения: 23.02.2025).
2. Романова, И. П. Базы данных: работа с PostgreSQL: учебное пособие / И. П. Романова, П. С. Романов; Московский университет имени С. Ю. Витте, кафедра информационных систем [Электронное издание]. – Москва: изд. ЧОУВО «МУ им. С. Ю. Витте», 2023. – 3,09 Мб.
3. Новиков, Б. А. Основы технологий баз данных: учебное пособие / Б. А. Новиков, Е. А. Горшкова, Н. Г. Графеева; под ред. Е. В. Рогова. – 2-е изд. – М.: ДМК Пресс, 2020. – 582 с.
4. Триггеры в MySQL [Электронный ресурс]. – URL: <https://www.rldp.ru/mysql/mysqlpro/triggers.htm> (дата обращения: 23.02.2025).
5. Хабр: SQL-запросы и оптимизация [Электронный ресурс]. – URL: <https://habr.com/ru/articles/514364/> (дата обращения: 23.02.2025).

# 

# ЛИСТИНГ

**-- Создание таблиц**

**CREATE TABLE rooms (**

**room\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**number SMALLINT UNSIGNED NOT NULL UNIQUE,**

**double\_bed TINYINT UNSIGNED NOT NULL,**

**single\_bed TINYINT UNSIGNED NOT NULL,**

**class ENUM('Стандарт', 'Полулюкс', 'Люкс') NOT NULL,**

**area DECIMAL(5, 2) UNSIGNED NOT NULL,**

**price\_per\_night DECIMAL(10, 2) UNSIGNED NOT NULL**

**);**

**CREATE TABLE guests (**

**guest\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**first\_name VARCHAR(100) NOT NULL,**

**last\_name VARCHAR(100) NOT NULL,**

**patronymic VARCHAR(100),**

**passport VARCHAR(20) UNIQUE NOT NULL,**

**email VARCHAR(100) UNIQUE NOT NULL,**

**phone\_number VARCHAR(20) UNIQUE NOT NULL**

**);**

**CREATE TABLE reservations (**

**reservation\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**guest\_id INT NOT NULL,**

**room\_id INT NOT NULL,**

**check\_in\_date DATE NOT NULL,**

**check\_out\_date DATE NOT NULL,**

**total\_room\_price DECIMAL(10, 2) UNSIGNED NOT NULL DEFAULT 0,**

**total\_service\_price DECIMAL(10, 2) UNSIGNED NOT NULL DEFAULT 0,**

**amount\_paid DECIMAL(10, 2) UNSIGNED NOT NULL DEFAULT 0,**

**amount\_remaining DECIMAL(10, 2) UNSIGNED NOT NULL DEFAULT 0,**

**FOREIGN KEY (guest\_id) REFERENCES guests(guest\_id) ON DELETE CASCADE,**

**FOREIGN KEY (room\_id) REFERENCES rooms(room\_id) ON DELETE CASCADE**

**);**

**CREATE TABLE services (**

**service\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**service\_name VARCHAR(100) NOT NULL UNIQUE,**

**description TEXT,**

**is\_price\_fixed BOOLEAN NOT NULL DEFAULT TRUE,**

**price DECIMAL(10, 2) UNSIGNED NOT NULL**

**);**

**CREATE TABLE reservation\_services (**

**reservation\_service\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**reservation\_id INT NOT NULL,**

**service\_id INT NOT NULL,**

**quantity TINYINT UNSIGNED NOT NULL DEFAULT 1,**

**total\_price DECIMAL(10, 2) UNSIGNED NOT NULL DEFAULT 0,**

**FOREIGN KEY (reservation\_id) REFERENCES reservations(reservation\_id) ON DELETE CASCADE,**

**FOREIGN KEY (service\_id) REFERENCES services(service\_id) ON DELETE CASCADE**

**);**

**CREATE TABLE payments (**

**payment\_id INT AUTO\_INCREMENT PRIMARY KEY,**

**reservation\_id INT NOT NULL,**

**payment\_date DATE NOT NULL,**

**payment\_method ENUM('Наличные', 'Карта', 'Онлайн') NOT NULL,**

**total DECIMAL(10, 2) UNSIGNED NOT NULL DEFAULT 0,**

**FOREIGN KEY (reservation\_id) REFERENCES reservations(reservation\_id) ON DELETE CASCADE**

**);**

**-- Триггеры**

**-- Триггер, считающий стоимость бронирования комнаты**

**DELIMITER $$**

**CREATE TRIGGER calculate\_total\_room\_price**

**BEFORE INSERT ON reservations**

**FOR EACH ROW**

**BEGIN**

**DECLARE price DECIMAL(10, 2);**

**IF NEW.check\_in\_date < CURDATE() THEN**

**SIGNAL SQLSTATE '45000'**

**SET MESSAGE\_TEXT = 'Дата въезда не может быть раньше текущей даты.';**

**END IF;**

**IF NEW.check\_out\_date <= NEW.check\_in\_date THEN**

**SET @errorMsg = CONCAT('Дата выезда: ',**

**DATE\_FORMAT(NEW.check\_out\_date, '%d-%m-%Y'),**

**' должна быть хотя бы на один день позже даты въезда: ',**

**DATE\_FORMAT(NEW.check\_in\_date, '%d-%m-%Y'));**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = @errorMsg;**

**END IF;**

**SELECT price\_per\_night INTO price**

**FROM rooms**

**WHERE room\_id = NEW.room\_id;**

**SET NEW.total\_room\_price = price \* DATEDIFF(NEW.check\_out\_date, NEW.check\_in\_date);**

**SET NEW.amount\_remaining = NEW.total\_room\_price;**

**END$$**

**DELIMITER ;**

**-- Триггер, обновляющий стоимость бронирования, если изменились даты или комната**

**DELIMITER $$**

**CREATE TRIGGER update\_total\_price**

**BEFORE UPDATE ON reservations**

**FOR EACH ROW**

**BEGIN**

**DECLARE price DECIMAL(10, 2);**

**DECLARE old\_duration INT;**

**DECLARE new\_duration INT;**

**IF NEW.check\_in\_date < CURDATE() THEN**

**SIGNAL SQLSTATE '45000'**

**SET MESSAGE\_TEXT = 'Дата въезда не может быть раньше текущей даты.';**

**END IF;**

**IF NEW.check\_out\_date <= NEW.check\_in\_date THEN**

**SET @errorMsg = CONCAT('Дата выезда: ',**

**DATE\_FORMAT(NEW.check\_out\_date, '%d-%m-%Y'),**

**' должна быть хотя бы на один день позже даты въезда: ',**

**DATE\_FORMAT(NEW.check\_in\_date, '%d-%m-%Y'));**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = @errorMsg;**

**END IF;**

**SET old\_duration = DATEDIFF(OLD.check\_out\_date, OLD.check\_in\_date);**

**SET new\_duration = DATEDIFF(NEW.check\_out\_date, NEW.check\_in\_date);**

**IF OLD.check\_in\_date <= CURDATE() AND CURDATE() <= OLD.check\_out\_date THEN**

**IF NEW.check\_in\_date <> OLD.check\_in\_date THEN**

**SET @errorMsg = CONCAT('Дата въезда: ',**

**DATE\_FORMAT(OLD.check\_in\_date, '%d-%m-%Y'),**

**' не может быть изменена: ',**

**DATE\_FORMAT(NEW.check\_in\_date, '%d-%m-%Y'),**

**', так как бронирование уже активно. ID: ',**

**NEW.reservation\_id);**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = @errorMsg;**

**END IF;**

**IF NEW.check\_out\_date < OLD.check\_out\_date THEN**

**SET @errorMsg = CONCAT('Дата выезда: ',**

**DATE\_FORMAT(OLD.check\_out\_date, '%d-%m-%Y'),**

**' не может быть уменьшена: ',**

**DATE\_FORMAT(NEW.check\_out\_date, '%d-%m-%Y'),**

**', так как бронирование уже активно. ID: ',**

**NEW.reservation\_id);**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = @errorMsg;**

**END IF;**

**END IF;**

**SELECT price\_per\_night INTO price**

**FROM rooms**

**WHERE room\_id = NEW.room\_id;**

**IF price \* new\_duration < OLD.total\_room\_price THEN**

**SET @errorMsg = CONCAT('Новая стоимость: ',**

**price \* new\_duration,**

**' не может быть меньше старой: ',**

**OLD.total\_room\_price,**

**', так как возврат средств не предусмотрен. ID: ',**

**NEW.reservation\_id);**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = @errorMsg;**

**END IF;**

**SET NEW.total\_room\_price = price \* new\_duration;**

**SET NEW.amount\_remaining = NEW.total\_room\_price + NEW.total\_service\_price - NEW.amount\_paid;**

**END$$**

**DELIMITER ;**

**-- Триггер, проверяющий, что номер не занят на определенную дату**

**DELIMITER $$**

**CREATE TRIGGER check\_room\_availability**

**BEFORE INSERT ON reservations**

**FOR EACH ROW**

**BEGIN**

**DECLARE conflict\_reservation\_id INT;**

**DECLARE conflict\_room\_id INT;**

**DECLARE conflict\_check\_in DATE;**

**DECLARE conflict\_check\_out DATE;**

**SELECT reservation\_id, room\_id, check\_in\_date, check\_out\_date**

**INTO conflict\_reservation\_id, conflict\_room\_id, conflict\_check\_in, conflict\_check\_out**

**FROM reservations**

**WHERE room\_id = NEW.room\_id**

**AND (**

**(NEW.check\_in\_date < check\_out\_date) AND**

**(NEW.check\_out\_date > check\_in\_date)**

**)**

**LIMIT 1;**

**IF conflict\_room\_id IS NOT NULL THEN**

**SET @errorMsg = CONCAT(**

**'Комната ', conflict\_room\_id,**

**' уже забронирована с ', DATE\_FORMAT(conflict\_check\_in, '%d-%m-%Y'),**

**' по ', DATE\_FORMAT(conflict\_check\_out, '%d-%m-%Y'), ', id: ', conflict\_reservation\_id,**

**'. Невозможно забронировать с ', DATE\_FORMAT(NEW.check\_in\_date, '%d-%m-%Y'),**

**' по ', DATE\_FORMAT(NEW.check\_out\_date, '%d-%m-%Y'));**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = @errorMsg;**

**END IF;**

**END$$**

**DELIMITER ;**

**DELIMITER $$**

**CREATE TRIGGER check\_room\_availability\_on\_update**

**BEFORE UPDATE ON reservations**

**FOR EACH ROW**

**BEGIN**

**DECLARE conflict\_reservation\_id INT;**

**DECLARE conflict\_room\_id INT;**

**DECLARE conflict\_check\_in DATE;**

**DECLARE conflict\_check\_out DATE;**

**SELECT reservation\_id, room\_id, check\_in\_date, check\_out\_date**

**INTO conflict\_reservation\_id, conflict\_room\_id, conflict\_check\_in, conflict\_check\_out**

**FROM reservations**

**WHERE room\_id = NEW.room\_id AND reservation\_id <> NEW.reservation\_id**

**AND (**

**(NEW.check\_in\_date < check\_out\_date) AND**

**(NEW.check\_out\_date > check\_in\_date)**

**)**

**LIMIT 1;**

**IF conflict\_room\_id IS NOT NULL THEN**

**SET @errorMsg = CONCAT(**

**'Комната ', conflict\_room\_id,**

**' уже забронирована с ', DATE\_FORMAT(conflict\_check\_in, '%d-%m-%Y'),**

**' по ', DATE\_FORMAT(conflict\_check\_out, '%d-%m-%Y'),**

**', id: ', conflict\_reservation\_id,**

**'. Невозможно оформить бронирование ', NEW.reservation\_id, '.');**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = @errorMsg;**

**END IF;**

**END$$**

**DELIMITER ;**

**-- Триггер, считающий стоимость услуг**

**DELIMITER $$**

**CREATE TRIGGER calculate\_service\_price**

**BEFORE INSERT ON reservation\_services**

**FOR EACH ROW**

**BEGIN**

**DECLARE service\_price DECIMAL(10, 2);**

**DECLARE is\_service\_price\_fixed BOOLEAN;**

**DECLARE existing\_fixed\_service INT;**

**SELECT price, is\_price\_fixed INTO service\_price, is\_service\_price\_fixed**

**FROM services**

**WHERE service\_id = NEW.service\_id;**

**IF NOT is\_service\_price\_fixed THEN**

**SET NEW.total\_price = service\_price \* NEW.quantity;**

**ELSE**

**SET NEW.total\_price = service\_price;**

**END IF;**

**IF is\_service\_price\_fixed THEN**

**SELECT COUNT(\*) INTO existing\_fixed\_service**

**FROM reservation\_services**

**WHERE reservation\_id = NEW.reservation\_id AND service\_id = NEW.service\_id;**

**IF existing\_fixed\_service > 0 THEN**

**SET @errorMsg = CONCAT('Услуга ', NEW.service\_id, ' уже была добавлена в эту (', NEW.reservation\_id, ') бронь.');**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = @errorMsg;**

**END IF;**

**END IF;**

**UPDATE reservations**

**SET total\_service\_price = total\_service\_price + NEW.total\_price,**

**amount\_remaining = amount\_remaining + NEW.total\_price**

**WHERE reservation\_id = NEW.reservation\_id;**

**END$$**

**DELIMITER ;**

**-- Запрет на обновление заказанных услуг**

**DELIMITER $$**

**CREATE TRIGGER prevent\_update\_reservation\_services**

**BEFORE UPDATE ON reservation\_services**

**FOR EACH ROW**

**BEGIN**

**SIGNAL SQLSTATE '45000'**

**SET MESSAGE\_TEXT = 'Обновление данных о заказанных услугах запрещено';**

**END$$**

**DELIMITER ;**

**-- Запрет на удаление заказанных услуг**

**DELIMITER $$**

**CREATE TRIGGER prevent\_delete\_reservation\_services**

**BEFORE DELETE ON reservation\_services**

**FOR EACH ROW**

**BEGIN**

**SIGNAL SQLSTATE '45000'**

**SET MESSAGE\_TEXT = 'Удаление данных о заказанных услугах запрещено';**

**END$$**

**DELIMITER ;**

**-- Триггеры для оплаты**

**DELIMITER $$**

**CREATE TRIGGER set\_total\_payment**

**BEFORE INSERT ON payments**

**FOR EACH ROW**

**BEGIN**

**DECLARE remaining DECIMAL(10, 2);**

**SELECT amount\_remaining INTO remaining**

**FROM reservations**

**WHERE reservation\_id = NEW.reservation\_id;**

**IF remaining = 0 THEN**

**SET @errorMsg = CONCAT('Нет неоплаченных счетов для: ', NEW.reservation\_id);**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = @errorMsg;**

**END IF;**

**IF NEW.total = 0 THEN**

**SET NEW.total = remaining;**

**ELSEIF NEW.total > remaining THEN**

**SET @errorMsg = CONCAT('Сумма платежа (', NEW.total,**

**') превышает оставшуюся сумму к оплате (', remaining,**

**') для бронирования с ID: ', NEW.reservation\_id, '.');**

**SIGNAL SQLSTATE '45000' SET MESSAGE\_TEXT = @errorMsg;**

**END IF;**

**END$$**

**DELIMITER ;**

**DELIMITER $$**

**CREATE TRIGGER update\_amounts\_on\_payment**

**AFTER INSERT ON payments**

**FOR EACH ROW**

**BEGIN**

**UPDATE reservations**

**SET amount\_paid = amount\_paid + NEW.total,**

**amount\_remaining = amount\_remaining - NEW.total**

**WHERE reservation\_id = NEW.reservation\_id;**

**END$$**

**DELIMITER ;**

**-- Запрет на обновление оплаты**

**DELIMITER $$**

**CREATE TRIGGER prevent\_update\_payments**

**BEFORE UPDATE ON payments**

**FOR EACH ROW**

**BEGIN**

**SIGNAL SQLSTATE '45000'**

**SET MESSAGE\_TEXT = 'Обновление данных об оплате запрещено';**

**END$$**

**DELIMITER ;**

**-- Запрет на удаление оплаты**

**DELIMITER $$**

**CREATE TRIGGER prevent\_delete\_payments**

**BEFORE DELETE ON payments**

**FOR EACH ROW**

**BEGIN**

**SIGNAL SQLSTATE '45000'**

**SET MESSAGE\_TEXT = 'Удаление данных об оплате запрещено';**

**END$$**

**DELIMITER ;**

**INSERT INTO rooms (number, double\_bed, single\_bed, class, area, price\_per\_night) VALUES**

**(1, 1, 0, 'Стандарт', 15, 3000),**

**(2, 1, 0, 'Стандарт', 15, 3000),**

**(3, 0, 2, 'Стандарт', 18, 3500),**

**(4, 0, 2, 'Стандарт', 18, 3500),**

**(5, 1, 2, 'Стандарт', 20, 4000),**

**(6, 1, 2, 'Стандарт', 20, 4000),**

**(7, 1, 0, 'Полулюкс', 35, 6000),**

**(8, 1, 0, 'Полулюкс', 35, 6000),**

**(9, 0, 2, 'Полулюкс', 38, 7500),**

**(10, 0, 2, 'Полулюкс', 38, 7500),**

**(11, 1, 2, 'Полулюкс', 40, 9000),**

**(12, 1, 2, 'Полулюкс', 40, 9000),**

**(13, 1, 0, 'Люкс', 55, 15000),**

**(14, 1, 0, 'Люкс', 55, 15000),**

**(15, 0, 2, 'Люкс', 58, 17500),**

**(16, 0, 2, 'Люкс', 58, 17500),**

**(17, 1, 2, 'Люкс', 60, 19000),**

**(18, 1, 2, 'Люкс', 60, 19000);**

**INSERT INTO guests (first\_name, last\_name, patronymic, passport, email, phone\_number) VALUES**

**('Иван', 'Иванов', 'Иванович', '1234 567890', 'ivan.ivanov@some\_mail.com', '+79001234567'),**

**('Мария', 'Петрова', 'Сергеевна', '2345 678901', 'maria.petrova@some\_gmail.com', '+79007654321'),**

**('Алексей', 'Сидоров', 'Алексеевич', '3456 789012', 'alexei.sidorov@some\_mail.com', '+79001112233'),**

**('Елена', 'Кузнецова', 'Дмитриевна', '4567 890123', 'elena.kuznetsova@some\_gmail.com', '+79009876543'),**

**('Дмитрий', 'Николаев', 'Андреевич', '5678 901234', 'dmitry.nikolaev@some\_mail.com', '+79003216549'),**

**('Ольга', 'Васильева', 'Викторовна', '6789 012345', 'olga.vasileva@some\_gmail.com', '+79004567890'),**

**('Андрей', 'Смирнов', 'Павлович', '7890 123456', 'andrey.smirnov@some\_mail.com', '+79007894561'),**

**('Татьяна', 'Морозова', 'Игоревна', '8901 234567', 'tatyana.morozova@some\_gmail.com', '+79002345678'),**

**('Сергей', 'Лебедев', 'Николаевич', '9012 345678', 'sergey.lebedev@some\_mail.com', '+79008765432'),**

**('Анна', 'Романова', 'Александровна', '0123 456789', 'anna.romanova@some\_gmail.com', '+79005551234'),**

**('Александр', 'Попов', 'Владимирович', '1122 334455', 'alexander.popov@some\_mail.com', '+79001111111'),**

**('Екатерина', 'Соколова', 'Анатольевна', '2233 445566', 'ekaterina.sokolova@some\_gmail.com', '+79002222222'),**

**('Максим', 'Кузнецов', 'Сергеевич', '3344 556677', 'maksim.kuznetsov@some\_mail.com', '+79003333333'),**

**('Олег', 'Федоров', 'Дмитриевич', '4455 667788', 'oleg.fedorov@some\_gmail.com', '+79004444444'),**

**('Наталья', 'Волкова', 'Олеговна', '5566 778899', 'natalya.volkova@some\_mail.com', '+79005555555'),**

**('Дарья', 'Иванова', 'Андреевна', '6677 889900', 'darya.ivanova@some\_gmail.com', '+79006666666'),**

**('Антон', 'Смирнов', 'Иванович', '7788 990011', 'anton.smirnov@some\_gmail.com', '+79007777777'),**

**('Юлия', 'Ковалева', 'Валерьевна', '8899 001122', 'yulia.kovaleva@some\_mail.com', '+79008888888'),**

**('Артем', 'Романов', 'Алексеевич', '9900 112233', 'artem.romanov@some\_gmail.com', '+79009999999'),**

**('Виктория', 'Лебедева', 'Сергеевна', '0011 223344', 'viktoria.lebedeva@some\_gmail.com', '+79000000000');**

**INSERT INTO reservations (guest\_id, room\_id, check\_in\_date, check\_out\_date) VALUES**

**(1, 1, '2025-03-10', '2025-04-15'),**

**(2, 2, '2025-03-02', '2025-03-11'),**

**(3, 3, '2025-03-01', '2025-03-02'),**

**(4, 4, '2025-03-04', '2025-04-10'),**

**(5, 5, '2025-03-04', '2025-04-09'),**

**(6, 6, '2025-03-02', '2025-03-10'),**

**(7, 7, '2025-03-01', '2025-03-11'),**

**(8, 8, '2025-03-08', '2025-03-12'),**

**(9, 9, '2025-03-09', '2025-03-13'),**

**(10, 10, '2025-03-10', '2025-03-19'),**

**(11, 11, '2025-04-15', '2025-04-18'),**

**(12, 12, '2025-03-13', '2025-03-20'),**

**(13, 13, '2025-03-17', '2025-03-21'),**

**(14, 14, '2025-04-10', '2025-04-11'),**

**(15, 15, '2025-04-10', '2025-04-11'),**

**(16, 16, '2025-04-15', '2025-04-18'),**

**(17, 17, '2025-03-13', '2025-03-20'),**

**(18, 18, '2025-03-17', '2025-03-21'),**

**(19, 1, '2025-04-15', '2025-04-19'),**

**(20, 2, '2025-04-09', '2025-04-23');**

**-- Добавление услуг с фиксированной ценой**

**INSERT INTO services (service\_name, description, price) VALUES**

**('Неограниченный доступ к бассейну', 'Наслаждайтесь неограниченным доступом к бассейну, включая шезлонги и бар у бассейна.', 5000.00),**

**('Неограниченный доступ в тренажерный зал', 'Получите доступ к тренажерному залу с возможностью занятий с личным тренером.', 3000.00),**

**('Неограниченный доступ в СПА', 'Полный доступ ко всем услугам СПА, включая массажи, сауну и другие процедуры.', 10000.00),**

**('Детская кроватка', 'Предоставление детской кроватки в номер для вашего малыша.', 1000.00),**

**('Трансфер из/в аэропорт', 'Комфортабельный трансфер на автомобиле бизнес-класса от аэропорта.', 4500.00),**

**('Личный доступ в Интернет', 'Подключим в вашем номере персональный высокоскоростной интернет для комфортного использования.', 5000.00);**

**-- Добавление услуг с нефиксированной ценой**

**INSERT INTO services (service\_name, description, price, is\_price\_fixed) VALUES**

**('Завтрак в номер', 'Наслаждайтесь вкусным завтраком, который мы доставим в ваш номер в удобное для вас время (с 8:00 до 11:00).', 1500.00, FALSE),**

**('Обед в номер', 'Закажите обед, и мы принесем его в ваш номер в выбранное вами время (с 13:00 до 16:00).', 1500.00, FALSE),**

**('Ужин в номер', 'Ужин в уютной обстановке вашего номера. Доставка в удобное для вас время (с 18:00 до 21:00).', 1500.00, FALSE),**

**('Организация экскурсий', 'Мы полностью организуем для вас экскурсию в любое выбранное вами место. Подробности уточняйте у администратора.', 15000.00, FALSE),**

**('Стирка и химчистка', 'Наши специалисты позаботятся о том, чтобы ваши вещи были безупречно чистыми и ухоженными. Цена указана за минимальный заказ.', 3000.00, FALSE),**

**('Почасовая аренда конференц-зала', 'Идеальное пространство для проведения мероприятий.', 15000.00, FALSE),**

**('Услуги няни', 'Профессиональная няня позаботится о вашем ребенке в удобное для вас время. Минимальное время заказа — 2 часа.', 4000.00, FALSE),**

**('Почасовая аренда автомобиля', 'Предоставление автомобиля на выбор: эконом, комфорт или премиум класс. Подробности уточняйте у администратора.', 3500.00, FALSE),**

**('Фотосессия в отеле', 'Профессиональный фотограф организует для вас фотосессию в интерьерах отеля или на территории. Включает 10 обработанных фото. Оплата почасовая.', 7500.00, FALSE);**

**-- Заказ услуг с фиксированной ценой**

**INSERT INTO reservation\_services (reservation\_id, service\_id) VALUES**

**(1, 1),**

**(1, 2),**

**(1, 3),**

**(1, 6),**

**(2, 1),**

**(3, 1),**

**(4, 1),**

**(5, 2),**

**(6, 3),**

**(7, 3),**

**(8, 6),**

**(8, 4),**

**(9, 1),**

**(10, 1);**

**-- Заказ услуг с нефиксированной ценой**

**INSERT INTO reservation\_services (reservation\_id, service\_id, quantity) VALUES**

**(1, 7, 2),**

**(1, 8, 1),**

**(1, 9, 1),**

**(2, 14, 5),**

**(2, 13, 6),**

**(2, 9, 2),**

**(3, 7, 1),**

**(3, 11, 1),**

**(3, 12, 4),**

**(4, 8, 2),**

**(4, 11, 1),**

**(4, 15, 3),**

**(5, 7, 2),**

**(5, 10, 2),**

**(5, 8, 2),**

**(6, 7, 2),**

**(6, 8, 2),**

**(6, 10, 2);**

**-- Оплата**

**INSERT INTO payments (reservation\_id, payment\_date, payment\_method) VALUES**

**(1, '2025-03-10', 'Карта'),**

**(2, '2025-03-01', 'Онлайн'),**

**(3, '2025-03-01', 'Карта'),**

**(4, '2025-03-04', 'Наличные'),**

**(5, '2025-03-04', 'Карта'),**

**(6, '2025-03-02', 'Карта'),**

**(7, '2025-03-01', 'Карта'),**

**(8, '2025-03-08', 'Наличные'),**

**(9, '2025-03-09', 'Карта'),**

**(10, '2025-03-01', 'Онлайн'),**

**(11, '2025-03-01', 'Онлайн'),**

**(12, '2025-03-17', 'Онлайн'),**

**(13, '2025-03-08', 'Наличные'),**

**(14, '2025-04-10', 'Наличные'),**

**(15, '2025-04-10', 'Карта');**