**Министерство науки и высшего образования Российской Федерации**

**федеральное государственное бюджетное образовательное учреждение**

**высшего образования**

**«Российский экономический университет имени Г.В. Плеханова»**

**Воронежский филиал**

| Кафедра | Информационных технологии в экономике |
| --- | --- |
| Направление | Прикладная информатика |
| Профиль | Прикладная информатика в экономике |

# КУРСОВОЙ ПРОЕКТ

**по дисциплине** Базы данных

**тема:** Разработка базы данных “Интернет магазин”

### Выполнил(а) \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Плотников И.А.

(подпись) (ФИО)

Студент(ка) гр. ПРд121

### Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ проф. Дылевский А.В.

(подпись) (ФИО)

### Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

### Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

#### Воронеж 2023

**СОДЕРЖАНИЕ**

1. Введение
2. Анализ требований к базе данных интернет-магазина
3. Проектирование базы данных
4. Выбор программного обеспечения
5. Разработка схемы базы данных на SQL
6. Реализация функциональности базы данных
7. Заключение
8. Список использованных источников

# Введение

В современном мире интернет-магазины играют важную роль в торговле, обеспечивая покупателям возможность удобно и быстро приобрести различные товары и услуги. С развитием электронной коммерции становится все более важным создание и поддержка эффективной базы данных для интернет-магазина. База данных является неотъемлемой частью функционирования интернет-магазина, обеспечивая хранение, управление и доступ к информации о товарах, заказах, клиентах и других сущностях, необходимых для его работы.

Целью данной работы является разработка базы данных для интернет-магазина с использованием языка SQL. Основными задачами являются анализ требований к базе данных, проектирование ее структуры, реализация необходимой функциональности и оптимизация производительности.

Для достижения поставленных целей мы будем проводить анализ требований к базе данных интернет-магазина. Мы определим основные сущности, такие как товары, категории, клиенты, заказы и другие, и проанализируем их атрибуты и связи между ними. На основе этого анализа мы приступим к проектированию структуры базы данных, включая определение таблиц, их полей и связей. Мы также обсудим вопросы нормализации базы данных и определения ограничений целостности.

Далее мы перейдем к реализации схемы базы данных на SQL. Мы создадим таблицы с использованием соответствующего синтаксиса SQL и определим первичные и внешние ключи для обеспечения связей между таблицами. Мы также реализуем необходимые хранимые процедуры и триггеры, которые позволят нам обрабатывать различные операции, такие как добавление товаров в корзину или оформление заказов.

Важным аспектом разработки базы данных интернет-магазина является ее тестирование и оптимизация. Мы проведем тестирование базы данных на различных наборах данных, чтобы убедиться в ее корректности и работоспособности. При необходимости мы внесем коррективы и исправления, чтобы устранить ошибки и неэффективные запросы. Мы также обсудим возможности оптимизации структуры базы данных и запросов для повышения ее производительности и эффективности.

В заключении работы мы подведем итоги проделанной работы и оценим достигнутые результаты в ходе разработки базы данных интернет-магазина. Мы также обсудим возможные пути дальнейшего развития базы данных и ее интеграции с другими системами интернет-магазина.

В работе будут использованы различные источники информации, такие как литература, статьи и онлайн-ресурсы, чтобы обеспечить полноту и достоверность полученных результатов.

Целью данной курсовой работы является разработка качественной базы данных для интернет-магазина, способной эффективно обрабатывать информацию о товарах, заказах и клиентах. Это поможет повысить эффективность работы интернет-магазина, улучшить пользовательский опыт и обеспечить успешное функционирование бизнеса.

Ссылка на репозиторий GitHub c проделанной работой: <https://github.com/main-scientist/course_data_base>

# 

# 2. Анализ требований к базе данных интернет-магазина

Анализ требований к базе данных является важным этапом разработки системы интернет-магазина. Он позволяет определить функциональность и структуру базы данных, а также выявить основные сущности и связи между ними. Ниже приведены ключевые аспекты, подлежащие анализу требований к базе данных интернет-магазина:

Функциональность интернет-магазина:

Определение основных операций, выполняемых в интернет-магазине, таких как просмотр товаров, добавление и удаление товаров из корзины, оформление заказов и оплата.

Идентификация дополнительных функций, таких как фильтрация товаров по категориям, поиск по ключевым словам, отзывы и рейтинги товаров.

Сущности и атрибуты:

Определение основных сущностей, которые должны быть представлены в базе данных, например: товары, категории, клиенты, заказы, платежи и т.д.

Анализ атрибутов каждой сущности, таких как название товара, описание, цена, количество на складе, персональные данные клиентов и другие.

Связи между сущностями:

Идентификация связей между сущностями, например, связь "один-ко-многим" между категорией и товарами (одна категория может содержать множество товаров), связь "многие-ко-многим" между клиентами и заказами (один клиент может иметь несколько заказов, и один заказ может быть связан с несколькими клиентами в случае покупки от имени компании).

Определение типов связей, таких как первичные и внешние ключи, которые обеспечат целостность данных и возможность связи между таблицами.

Обработка транзакций:

Разработка механизмов обработки транзакций для обеспечения целостности данных и защиты от конфликтов при одновременном доступе к базе данных, например, при оформлении заказов или обновлении количества товаров в корзине.

Безопасность данных:

Анализ требований к безопасности данных, таких как защита персональных данных клиентов, безопасность платежных транзакций и предотвращение несанкционированного доступа к базе данных.

Масштабируемость и производительность:

Учет требований к масштабируемости базы данных с ростом числа товаров, клиентов и заказов.

Определение потенциальных узких мест и требований к производительности запросов для обеспечения отзывчивости и эффективности работы интернет-магазина.

Анализ требований к базе данных интернет-магазина позволяет установить основу для дальнейшего проектирования и разработки базы данных, а также обеспечить соответствие функциональности интернет-магазина ожиданиям пользователей и бизнес-процессам.

# 

# 3. Проектирование базы данных

Проектирование базы данных является ключевым этапом разработки системы интернет-магазина. В этом разделе мы определим структуру базы данных, включая таблицы, их поля и связи между ними. Ниже приведены основные аспекты проектирования базы данных:

Выбор модели базы данных:

Рассмотрение различных моделей баз данных, таких как реляционная, иерархическая или объектно-ориентированная, и выбор наиболее подходящей модели для интернет-магазина.

В большинстве случаев реляционная модель базы данных является наиболее распространенной и подходящей для интернет-магазинов.

Определение таблиц и полей:

Идентификация основных сущностей, определенных на этапе анализа требований, и создание соответствующих таблиц в базе данных.

Определение полей для каждой таблицы, которые будут хранить необходимую информацию, например, название товара, цена, описание, дата добавления и т.д.

Разработка правильной структуры таблиц, чтобы обеспечить нормализацию данных и устранить избыточность и несогласованность.

Нормализация базы данных:

Применение нормализации для устранения избыточности данных и обеспечения эффективности хранения и обработки информации.

Проектирование базы данных в соответствии с нормальными формами (например, первая, вторая и третья нормальные формы), чтобы минимизировать дублирование данных и обеспечить целостность информации.

Определение связей между таблицами:

Определение связей между таблицами на основе анализа требований и сущностей.

Использование первичных ключей и внешних ключей для связей между таблицами и обеспечения целостности данных.

Учет типов связей, таких как "один-ко-многим" (например, категория товаров имеет множество товаров) или "многие-ко-многим" (например, клиенты могут иметь несколько заказов, и один заказ может быть связан с несколькими клиентами).

Определение ограничений целостности:

Определение ограничений для обеспечения целостности данных, таких как ограничения уникальности, ограничения ссылочной целостности и ограничения проверки значений полей.

Например, можно установить ограничение на уникальность электронной почты клиента, чтобы каждый клиент имел уникальный адрес электронной почты.

Обеспечение производительности:

Учет требований к производительности базы данных и оптимизация структуры таблиц, индексов и запросов для обеспечения быстрого доступа к данным.

Разработка оптимальных индексов для ускорения выполнения запросов, особенно для часто используемых операций, таких как поиск товаров по категориям или фильтрация по цене.

Разработка хранимых процедур и триггеров:

Реализация хранимых процедур и триггеров для обработки сложных операций и поддержания целостности данных.

Например, можно разработать хранимую процедуру для оформления заказа, которая автоматически обновляет информацию о доступном количестве товаров на складе и генерирует уведомление клиенту о подтверждении заказа.

Проектирование базы данных интернет-магазина требует внимательного анализа требований и грамотного планирования структуры и связей между таблицами. Правильное проектирование базы данных обеспечивает эффективное хранение, управление и доступ к информации, необходимой для работы интернет-магазина.

Таблицы необходимо хранить в схеме, создаём схему shop.

Использование UUID (уникальных идентификаторов) вместо простого целочисленного типа (int) для идентификации записей в базе данных имеет несколько преимуществ:

1. Уникальность: UUID обеспечивает глобальную уникальность идентификаторов. Это означает, что вы можете создавать уникальные идентификаторы, которые будут гарантированно уникальными не только в пределах одной таблицы или базы данных, но и во всей системе или даже в масштабе всего мира. Это полезно, когда вам нужно объединить данные из разных источников или систем.

2. Независимость от последовательности: В отличие от целочисленных идентификаторов, UUID не зависят от последовательности или значения предыдущих идентификаторов. Это особенно важно при создании новых записей в распределенных системах или при использовании параллельных операций, где идентификаторы могут быть созданы одновременно на разных узлах. UUID позволяют избежать конфликтов идентификаторов.

3. Безопасность: UUID могут быть сгенерированы с использованием криптографически стойких алгоритмов, таких как UUID v4. Это обеспечивает высокую степень безопасности и предотвращает предсказуемость идентификаторов.

4. Использование в разных системах: UUID могут быть использованы для интеграции данных между разными системами или базами данных. Поскольку они являются универсальными идентификаторами, они облегчают сопоставление и связывание данных из разных источников.

Однако следует отметить, что использование UUID также имеет некоторые недостатки. Они занимают больше места в базе данных, чем целочисленные идентификаторы, и могут замедлить процессы индексации и поиска в больших таблицах. Кроме того, UUID менее удобны для чтения и запоминания, чем простые числовые идентификаторы.

UUID имеет фиксированный размер в 128 бит, что эквивалентно 16 байтам. Он состоит из двух компонентов: время и уникальный идентификатор узла. Время составляет 60 бит и включает информацию о текущем времени, а уникальный идентификатор узла занимает 48 бит и представляет собой идентификатор сетевого интерфейса.

Table “customers”:

1. uuid - уникальных идентификаторов. Type: string, space: 36 byte.
2. name - имя покупателя. Type: string, space: 16 byte.
3. surname - фамилия покупателя. Type: string, space: 32 byte.
4. password - пароль. Type: string, space: 32 byte.
5. email - электронная почта покупателя. Type: string, space: 64 byte.
6. phone - номер мобильного телефона . Type: string, space: 32 byte.
7. sex - пол покупателя. Type: boolean, space: 1 bit.
8. country - страна покупателя. Type: boolean, space: 32 byte
9. city - город покупателя. Type: string, space: 32 byte.
10. address - адрес покупателя. Type: string, space: 128 byte.
11. postcode - почтовый индекс. Type: string, space: 16 byte.
12. amount of orders - общее количество заказов покупателя. Type: int, space: 4 byte.
13. amount of orders - общая сумма заказов. Type: float, space: 4 byte.
14. registration at - дата регистрации. Type: datetime, space: 8 byte.
15. last order at - последняя дата заказа. Type: datetime, space: 8 byte.

Таблица "customers" используется для хранения информации о покупателях магазина. Она содержит различные атрибуты и данные, связанные с каждым клиентом, позволяя магазину эффективно управлять информацией о своей клиентской базе и предоставлять персонализированный сервис. Информация, такая как уникальный идентификатор (uuid) в качестве первичного ключа, а также обязательные поля - имя (name), фамилия (surname), пароль (password), электронная почта (email) и страна покупателя (country) является критически важной для корректной работы программы и обеспечения удобства пользователей. Уникальный идентификатор гарантирует уникальность каждой записи, а имя, фамилия, пароль и электронная почта служат основными элементами идентификации, обеспечивают безопасность данных, персонализацию опыта пользователя и эффективную коммуникацию с ним. Вместе эти данные обеспечивают правильную работу программы и важны для обеспечения удобства и безопасности пользователей.

Table "sellers":

1. uuid - уникальный идентификатор. Type: string, space: 36 byte.
2. name - имя продавца. Type: string, space: 32 byte.
3. surname - фамилия продавца. Type: string, space: 32 byte.
4. password - пароль. Type: string, space: 32 byte.
5. name\_of\_shop - название магазина. Type: string, space: 128 byte.
6. email - электронная почта продавца. Type: string, space: 64 byte.
7. sex - пол продавца. Type: boolean, space: 1 bit.
8. phone - номер мобильного телефона. Type: string, space: 32 byte.
9. country - страна продавца. Type: string, space: 32 byte.
10. city - город продавца. Type: string, space: 32 byte.
11. address - адрес продавца. Type: string, space: 32 byte.
12. postcode - почтовый индекс. Type: string, space: 32 byte.
13. amount of sales - общее количество продаж. Type: int, space: 4 byte.
14. cost of sales - общая стоимость продаж. Type: float, space: 4 byte.
15. rating - рейтинг продавца. Type: float, space: 4 byte.

Таблица "sellers" используется для хранения информации о продавцах в магазине. Она содержит данные и атрибуты, связанные с каждым продавцом, позволяя магазину управлять информацией о своих продавцах и обеспечивать эффективное функционирование бизнеса. Информация, такая как уникальный идентификатор (uuid) в качестве первичного ключа, а также обязательные поля - имя (name), фамилия (surname), пароль (password) и электронная почта (email).

Table "delivery\_systems":

1. uuid - уникальный идентификатор. Type: int, space: 4 byte.
2. name - название системы доставки. Type: string, space: 16 byte.
3. cost - стоимость доставки. Type: float, space: 4 byte.

Таблица "delivery\_systems" используется для хранения информации о системах доставки, которые могут быть использованы в магазине. Она содержит данные и атрибуты, связанные с каждой системой доставки. Информация, такая как уникальный идентификатор (id) в качестве первичного ключа, а также обязательные поля - название (name) и стоимость (cost) доставки.

Table "deliveries":

1. uuid - уникальный идентификатор. Type: string, space: 36 byte.
2. delivery system - идентификатор системы доставки. Type: int, space: 4 byte.
3. postcode - почтовый индекс. Type: string, space: 16 byte.
4. city - город. Type: string, space: 32 byte.
5. address - адрес доставки. Type: string, space: 128 byte.

Информация, такая как уникальный идентификатор (uuid) в качестве первичного ключа, а также обязательные поля - идентификатор системы доставки (delivery system), почтовый индекс (postcode), город (city) и адрес доставки (address), является важной для управления процессом доставки в магазине. Уникальный идентификатор гарантирует уникальность каждой записи, а идентификатор системы доставки связывает доставку с определенной системой доставки из таблицы "delivery\_systems". Почтовый индекс, город и адрес доставки определяют место доставки для заказов. Вместе эти данные обеспечивают правильное функционирование процесса доставки и важны для эффективной обработки заказов и доставки товаров покупателям.

Table "payment\_systems":

1. id - уникальный идентификатор. Type: int, space: 4 byte.
2. type - тип платежной системы. Type: string, space: 16 byte.
3. tax - налог платежной системы. Type: float, space: 4 byte.

Информация, такая как уникальный идентификатор (id) в качестве первичного ключа, а также обязательные поля - тип (type) платежной системы и налог (tax) платежной системы.

Table "orders":

1. uuid - уникальный идентификатор. Type: string, space: 36 byte.
2. customer id - идентификатор покупателя. Type: string, space: 36 byte.
3. delivery id - идентификатор доставки. Type: string, space: 36 byte.
4. date - дата заказа. Type: datetime, space: 8 byte.
5. cost - стоимость заказа. Type: int, space: 4 byte.

Таблица "orders" используется для хранения информации о заказах в магазине. Она содержит данные и атрибуты, связанные с каждым заказом, позволяя магазину эффективно управлять процессом заказов, отслеживать статусы и обеспечивать своим клиентам качественное обслуживание. Информация, такая как уникальный идентификатор (uuid) в качестве первичного ключа, а также обязательные поля - идентификатор покупателя (customer\_id), идентификатор доставки (delivery\_id), дата заказа (date) и стоимость заказа (cost), является важной для управления заказами в магазине. Уникальный идентификатор гарантирует уникальность каждой записи, а идентификаторы покупателя и доставки связывают заказ с соответствующими таблицами "customers" и "delivery". Дата заказа и стоимость заказа отражают информацию о времени создания заказа и его стоимости. Вместе эти данные обеспечивают правильное функционирование процесса заказов и важны для отслеживания, обработки и анализа заказов в магазине.

Table "categories":

1. id - уникальный идентификатор. Type: int, space: 4 byte.
2. name - название категории. Type: string, space: 16 byte.

Информация, такая как уникальный идентификатор (id) в качестве первичного ключа, а также поле названия (name) категории, является важной для классификации товаров или услуг в магазине по категориям.

Table "products":

1. uuid - уникальный идентификатор. Type: string, space: 36 byte.
2. name - название продукта. Type: string, space: 16 byte.
3. category id - идентификатор категории. Type: int, space: 4 byte.
4. seller id - идентификатор продавца. Type: string, space: 36 byte.
5. cost - стоимость продукта. Type: float, space: 4 byte.
6. amount of - количество продукта. Type: int, space: 4 byte.
7. code - код продукта. Type: string, space: 16 byte.
8. create at - дата создания продукта. Type: datetime, space: 8 byte.
9. rating - рейтинг продукта. Type: float, space: 4 byte.

Информация, такая как уникальный идентификатор (uuid) в качестве первичного ключа, а также обязательные поля - название (name) продукта, идентификатор категории (category\_id), идентификатор продавца (seller\_id) и стоимость (cost) продукта, является важной для управления информацией о продуктах в магазине. Уникальный идентификатор гарантирует уникальность каждой записи, а название продукта, категория, продавец, стоимость и другие атрибуты определяют информацию о продукте. Категория и продавец связаны с соответствующими таблицами "categories" и "sellers". Количество продукта, код, дата создания и рейтинг отражают дополнительные характеристики и статистику продукта. Вместе эти данные обеспечивают правильное функционирование управления информацией о продуктах и важны для отображения, поиска, сортировки и анализа продуктов в магазине.

Table "orders\_products":

1. uuid - уникальный идентификатор. Type: string, space: 36 byte.
2. orders\_id - идентификатор заказа. Type: string, space: 36 byte.
3. product\_id - идентификатор продукта. Type: string, space: 36 byte.

Информация, такая как уникальный идентификатор (uuid) в качестве первичного ключа, а также обязательные поля - идентификатор заказа (orders\_id) и идентификатор продукта (product\_id), является важной для связи заказов и продуктов в магазине. Идентификаторы заказа и продукта связывают связанные записи в таблицах "orders" и "products". Вместе эти данные обеспечивают правильное функционирование связи между заказами и продуктами, позволяя отслеживать, какие продукты были включены в каждый заказ, и обеспечивают связность данных между соответствующими таблицами.

Table "invoices":

1. uuid - уникальный идентификатор. Type: string, space: 36 byte.
2. customer\_id - идентификатор покупателя. Type: string, space: 36 byte.
3. seller\_id - идентификатор продавца. Type: string, space: 36 byte.
4. order\_id - идентификатор заказа. Type: string, space: 36 byte.
5. create at - дата создания. Type: datetime, space: 8 byte.
6. cost - стоимость. Type: float, space: 4 byte.
7. cost tax - стоимость с налогом. Type: float, space: 4 byte.
8. payment system - идентификатор платежной системы. Type: int, space: 4 byte.

Таблица "invoices" используется для хранения информации о счетах-фактурах в магазине. Информация, такая как уникальный идентификатор (uuid) в качестве первичного ключа, а также обязательные поля - идентификатор покупателя (customer\_id), идентификатор продавца (seller\_id), идентификатор заказа (order\_id) и идентификатор платежной системы (payment\_system), является важной для управления информацией о счетах в магазине. Уникальный идентификатор гарантирует уникальность каждой записи, а идентификаторы покупателя, продавца, заказа и платежной системы связывают счет с соответствующими таблицами "customers", "sellers", "orders" и "payment\_systems". Дата создания, стоимость и стоимость с налогом отражают информацию о счете. Вместе эти данные обеспечивают правильное функционирование управления информацией о счетах и важны для отслеживания платежей, финансового учета и анализа в магазине.

Table "basket":

1. uuid - уникальный идентификатор. Type: string, space: 36 byte.
2. customer\_id - идентификатор покупателя. Type: string, space: 36 byte, обязательное поле.
3. amount of product - количество продуктов. Type: int, space: 4 byte.
4. sum\_of\_product - сумма продуктов. Type: float, space: 4 byte).

Таблица "basket" используется для хранения информации о корзинах покупателей в магазине. Она содержит уникальный идентификатор для каждой корзины, идентификатор покупателя, количество продуктов и сумму продуктов. Внешний ключ связывает поле customer\_id с таблицей customers, позволяя установить связь между корзиной и соответствующим покупателем. Это позволяет магазину отслеживать содержимое корзин, связывать их с конкретными покупателями и рассчитывать сумму и количество продуктов в корзине.

Table "products\_basket":

1. uuid - уникальный идентификатор. Тип: строка (string), размер: 36 байт.
2. basket\_id - идентификатор корзины. Тип: строка (string), размер: 36 байт.
3. product\_id - идентификатор продукта. Тип: строка (string), размер: 36 байт.

Таблица "product\_basket" используется для хранения информации о продуктах, которые находятся в корзинах в магазине. Она содержит уникальный идентификатор для каждой записи, а также связывает продукты с конкретными корзинами, используя внешние ключи. Это позволяет магазину отслеживать содержимое корзин и связывать продукты с соответствующими корзинами. Поле “basket\_id” будет ссылать на поле “uuid” из таблицы “basket”, а поле “product\_id” будет ссылаться на поле “uuid” из таблицы “product”.

Table "product\_reviews":

1. uuid - уникальный идентификатор. Type: string, space: 36 byte.
2. customer\_id - идентификатор покупателя. Type: string, space: 36 byte.
3. product\_id - идентификатор продукта. Type: string, space: 36 byte.
4. review - отзыв о продукте. Type: string, space: 1024 byte.

Таблица "product\_reviews" используется для хранения отзывов покупателей о продуктах в магазине. Она содержит уникальный идентификатор для каждого отзыва, идентификаторы покупателя и продукта, а также текстовое поле для самого отзыва. Внешние ключи связывают поля customer\_id и product\_id с соответствующими таблицами customers и products, что позволяет установить связь между отзывами и соответствующими покупателями и продуктами. Это позволяет магазину хранить и отображать отзывы покупателей, связанные с конкретными продуктами, и предоставлять информацию о качестве и удовлетворенности покупателей с целью помочь другим пользователям принять информированное решение при покупке продукта.

# 4. Выбор программного обеспечения

MySQL (Structured Query Language) - это открытая реляционная система управления базами данных (СУБД), которая использует язык SQL для управления и доступа к данным. Она была разработана компанией MySQL AB и в настоящее время поддерживается и развивается компанией Oracle.

MySQL предоставляет мощные возможности для хранения, управления и обработки структурированных данных. Она широко применяется во многих приложениях и веб-сайтах, включая интернет-магазины, социальные сети, блоги, форумы и другие проекты.

Основные характеристики MySQL:

Реляционная модель: MySQL основана на реляционной модели баз данных, где данные организованы в таблицы с рядами и столбцами. Она поддерживает возможность создания и управления связями между таблицами, что обеспечивает эффективное хранение и связывание данных.

Масштабируемость: MySQL обладает гибкостью и масштабируемостью, позволяя обрабатывать как небольшие базы данных для небольших проектов, так и крупные базы данных с миллионами записей для масштабных приложений. Она также поддерживает репликацию и шардинг, что позволяет распределять нагрузку на несколько серверов для обеспечения высокой доступности и производительности.

Поддержка языка SQL: MySQL полностью совместим с языком SQL, что облегчает создание и выполнение запросов для извлечения, изменения и удаления данных в базе данных. Она также поддерживает различные функции и операторы SQL для манипулирования данными и выполнения сложных операций.

IDE DataGrid - это компонент пользовательского интерфейса (UI), который используется для отображения и редактирования данных в табличной форме. Он предоставляет удобный способ визуализации и взаимодействия с данными, обычно из базы данных или другого источника данных.

DataGrid позволяет отображать данные в виде сетки, где каждая строка представляет отдельную запись данных, а каждый столбец представляет отдельное поле или атрибут данных. Он обычно предоставляет возможности сортировки, фильтрации и группировки данных, а также позволяет редактировать значения в ячейках таблицы.

Основные функции и возможности DataGrid:

Отображение данных: DataGrid отображает данные в удобной и структурированной форме, что позволяет пользователям легко просматривать и анализировать информацию.

Сортировка и фильтрация: DataGrid обычно предоставляет возможности сортировки данных по выбранным столбцам, а также фильтрации данных для отображения только тех записей, которые соответствуют определенным критериям.

Редактирование данных: DataGrid позволяет пользователям редактировать значения в ячейках таблицы. Он может поддерживать различные типы редакторов, такие как текстовые поля, флажки, выпадающие списки и другие, в зависимости от типа данных в каждой ячейке.

Выделение и выбор данных: DataGrid позволяет выделять и выбирать одну или несколько строк данных для выполнения определенных операций, таких как удаление, обновление или другие манипуляции с данными.

Поддержка событий: DataGrid генерирует события при взаимодействии с данными, такие как изменение значения в ячейке или выбор строки. Это позволяет разработчикам реагировать на эти события и выполнять соответствующие действия.

DataGrid является популярным компонентом во многих средах разработки программного обеспечения и часто используется в приложениях, где требуется отображение и редактирование табличных данных, включая системы управления базами данных, CRM-системы, административные панели и другие приложения, где работа с данными играет важную роль.

Выбор MySQL сервера и IDE DataGrid обоснован их преимуществами, такими как надежность, производительность, широкая поддержка, гибкость, масштабируемость, безопасность, удобство, визуальные возможности, функции редактирования и поддержка SQL. Эти компоненты помогут эффективно разрабатывать и управлять базой данных интернет-магазина.

# 5. Разработка схемы базы данных на SQL

<https://github.com/main-scientist/course_data_base/blob/main/data_base/create_tables/init_tables.sql>



Рисунок 1. Создание схемы shop

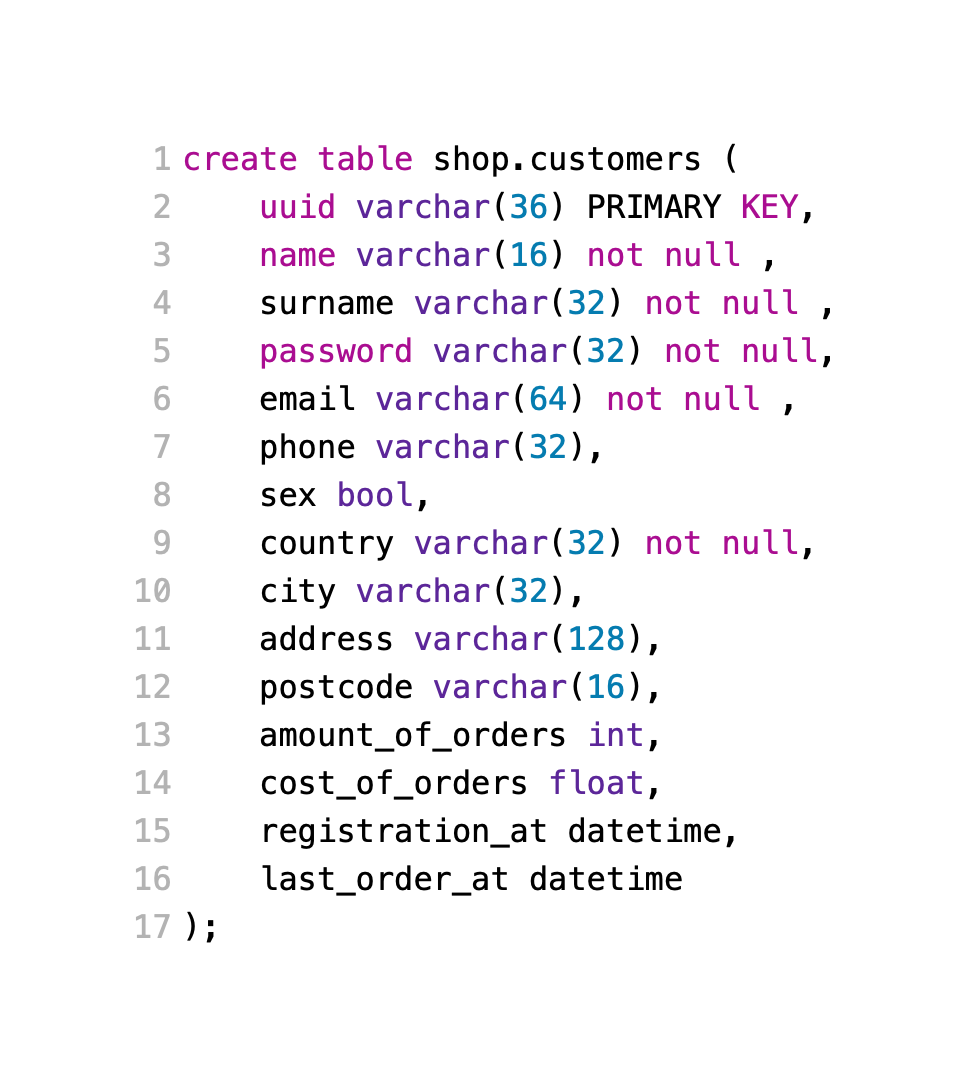


Рисунок 2. Создание таблицы “customers”

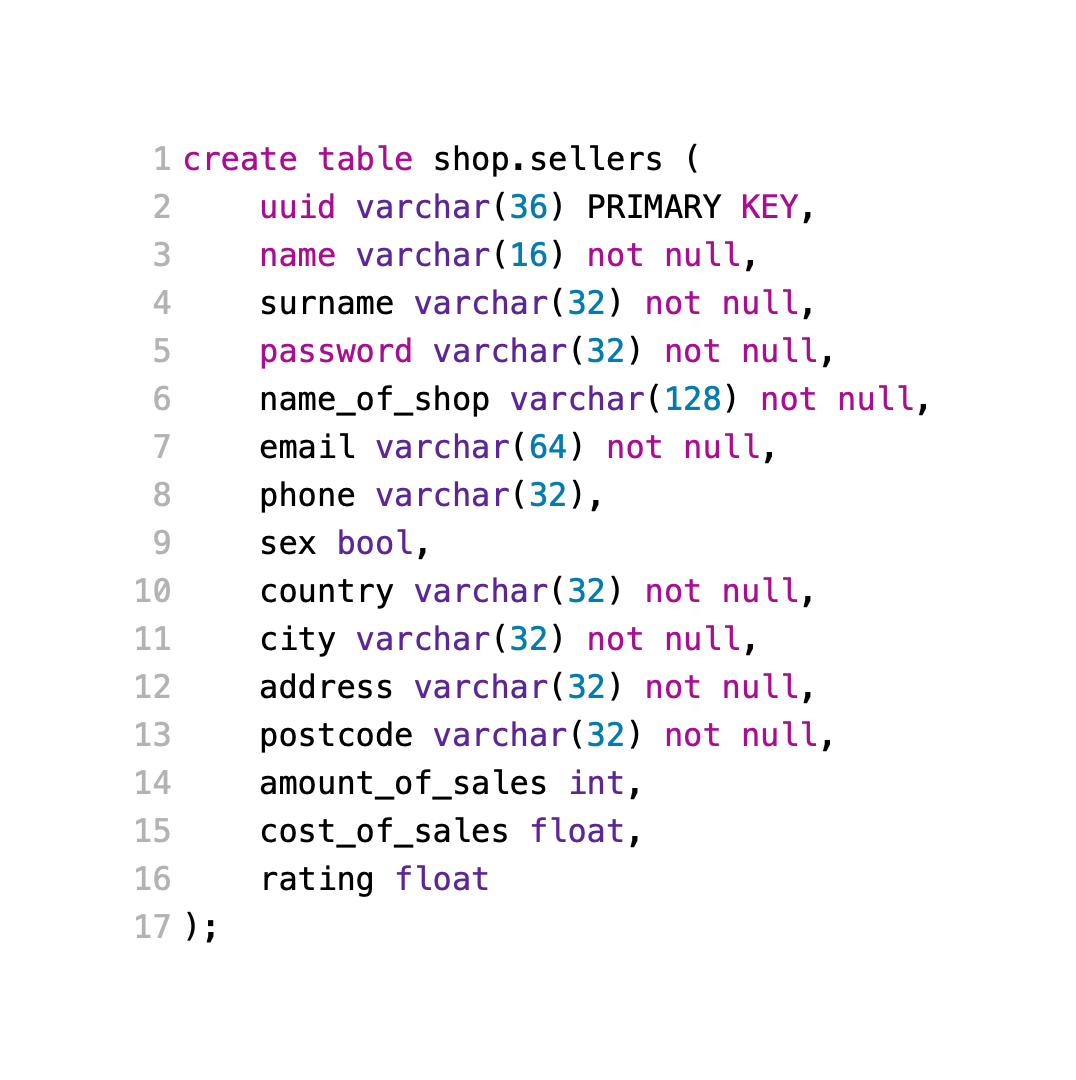


Рисунок 3. Создание таблицы “sellers”

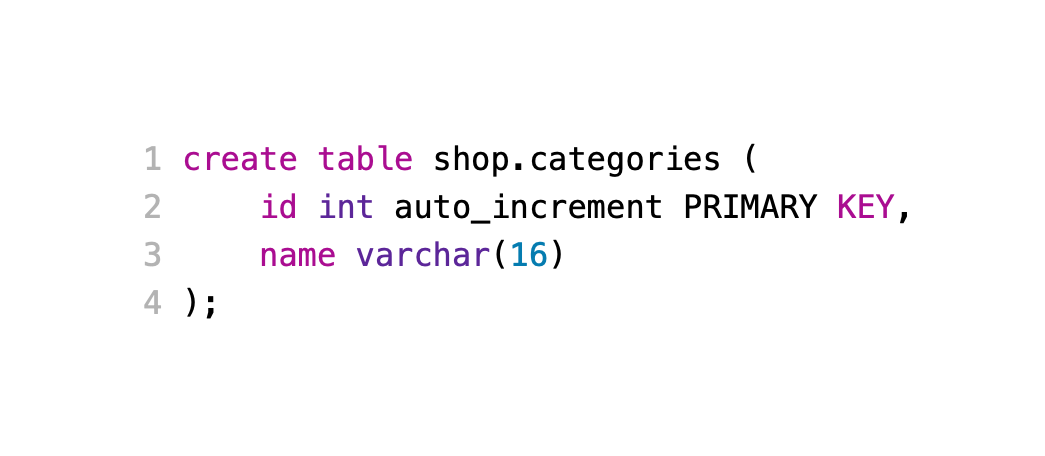


Рисунок 4. Создание таблицы “categories”

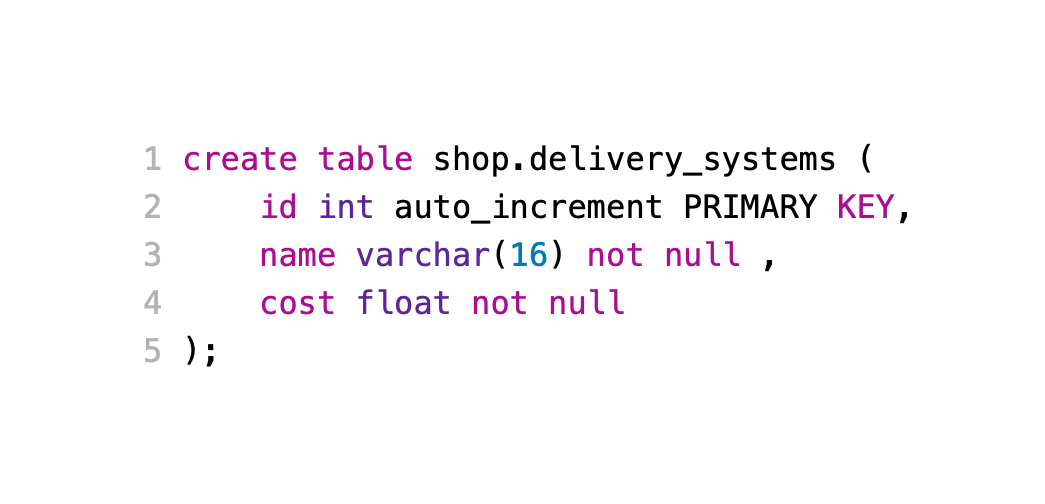


Рисунок 5. Создание таблицы “delivery\_systems”

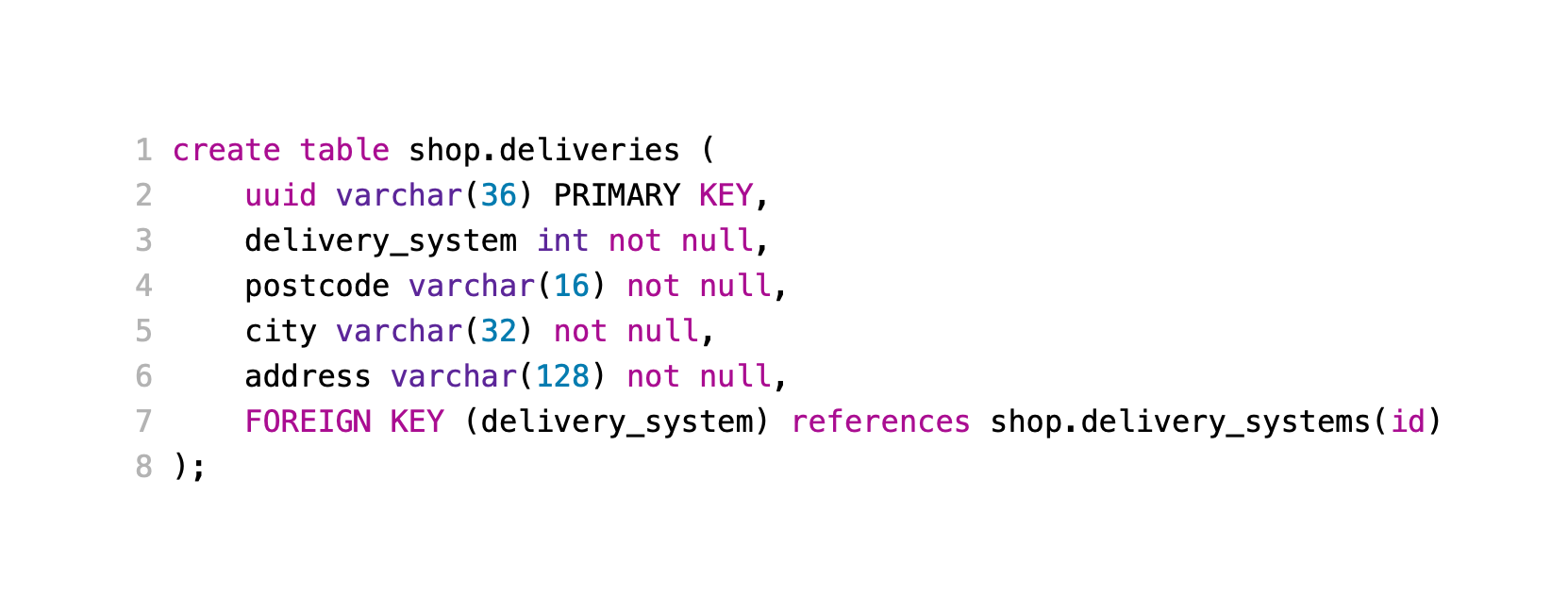


Рисунок 6. Создание таблицы “deliveries”

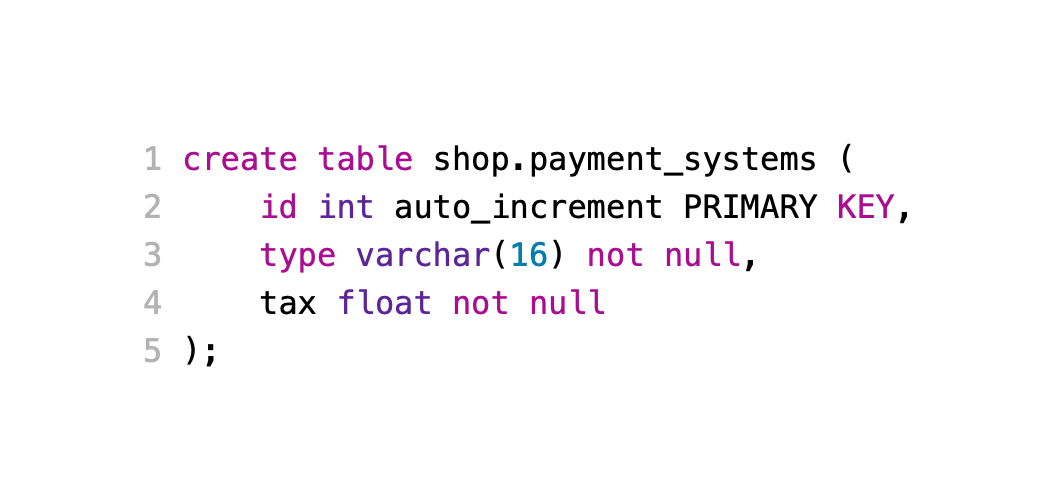
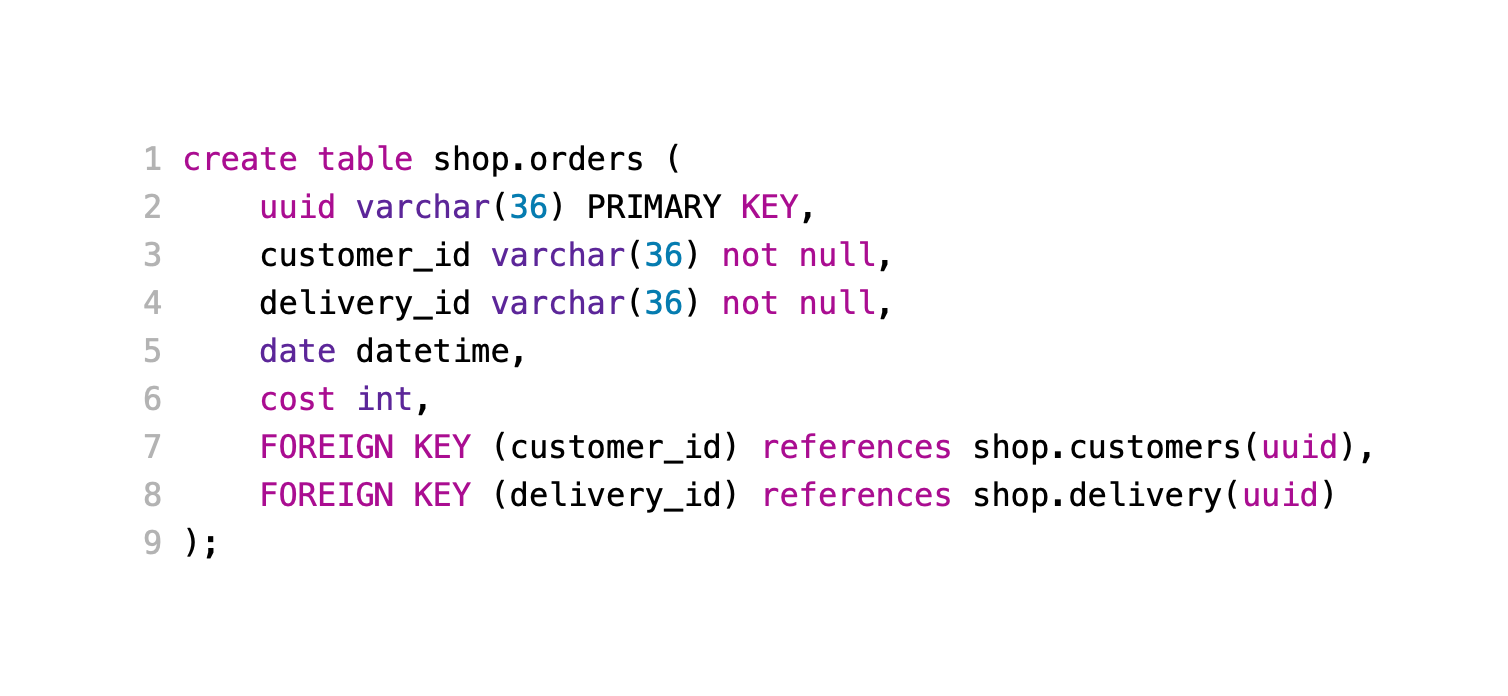


Рисунок 7. Создание таблицы “payment\_systems”

Рисунок 8. Создание таблицы “orders”

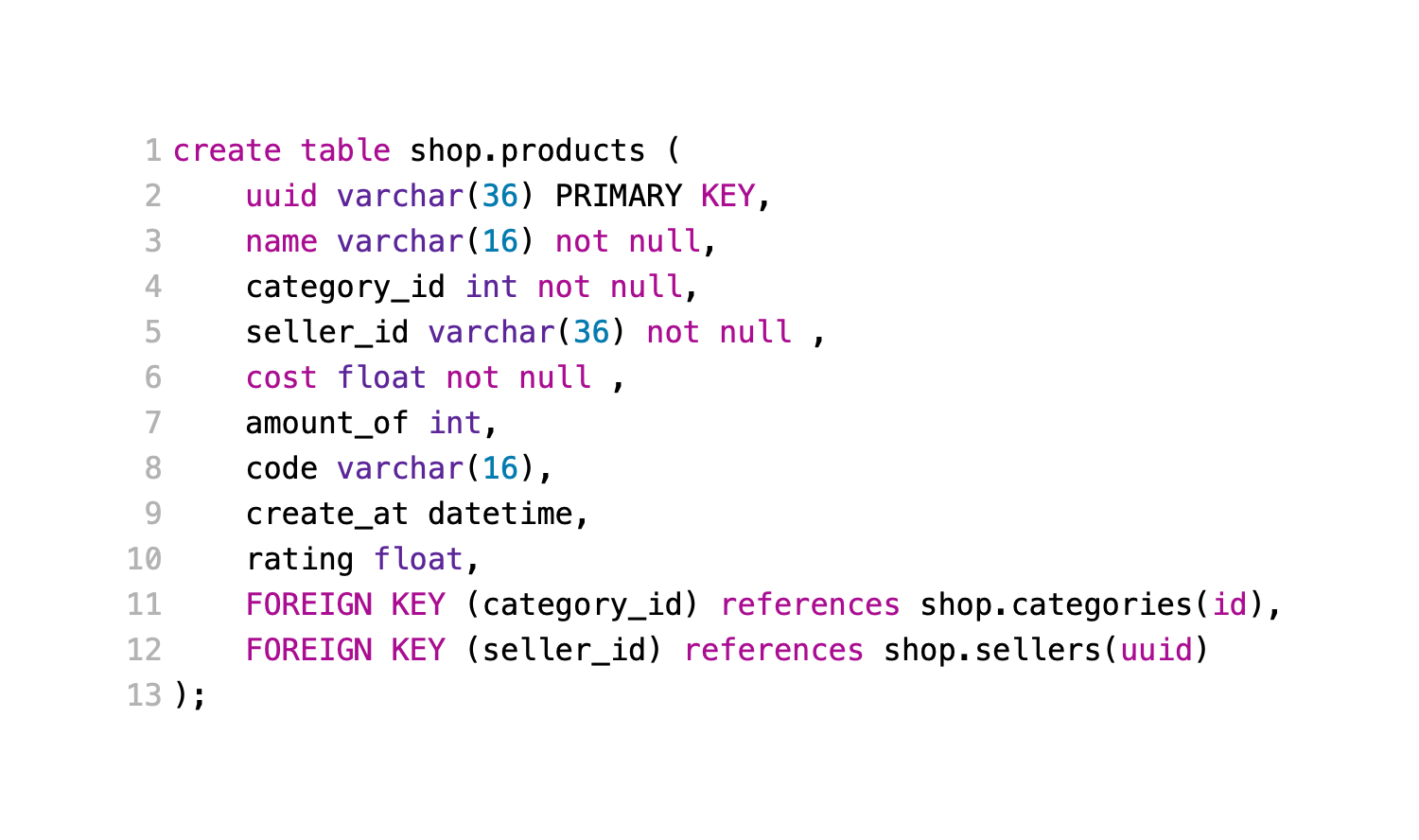


Рисунок 9. Создание таблицы “products”

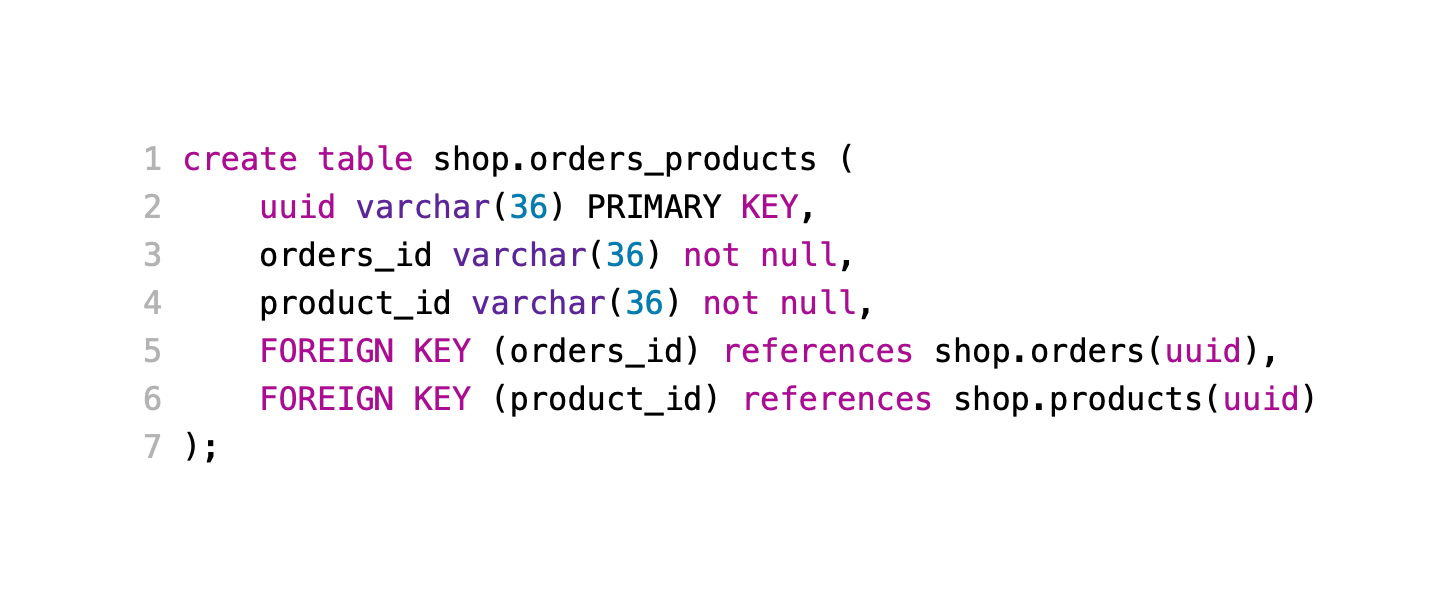


Рисунок 10. Создание таблицы “orders\_products”

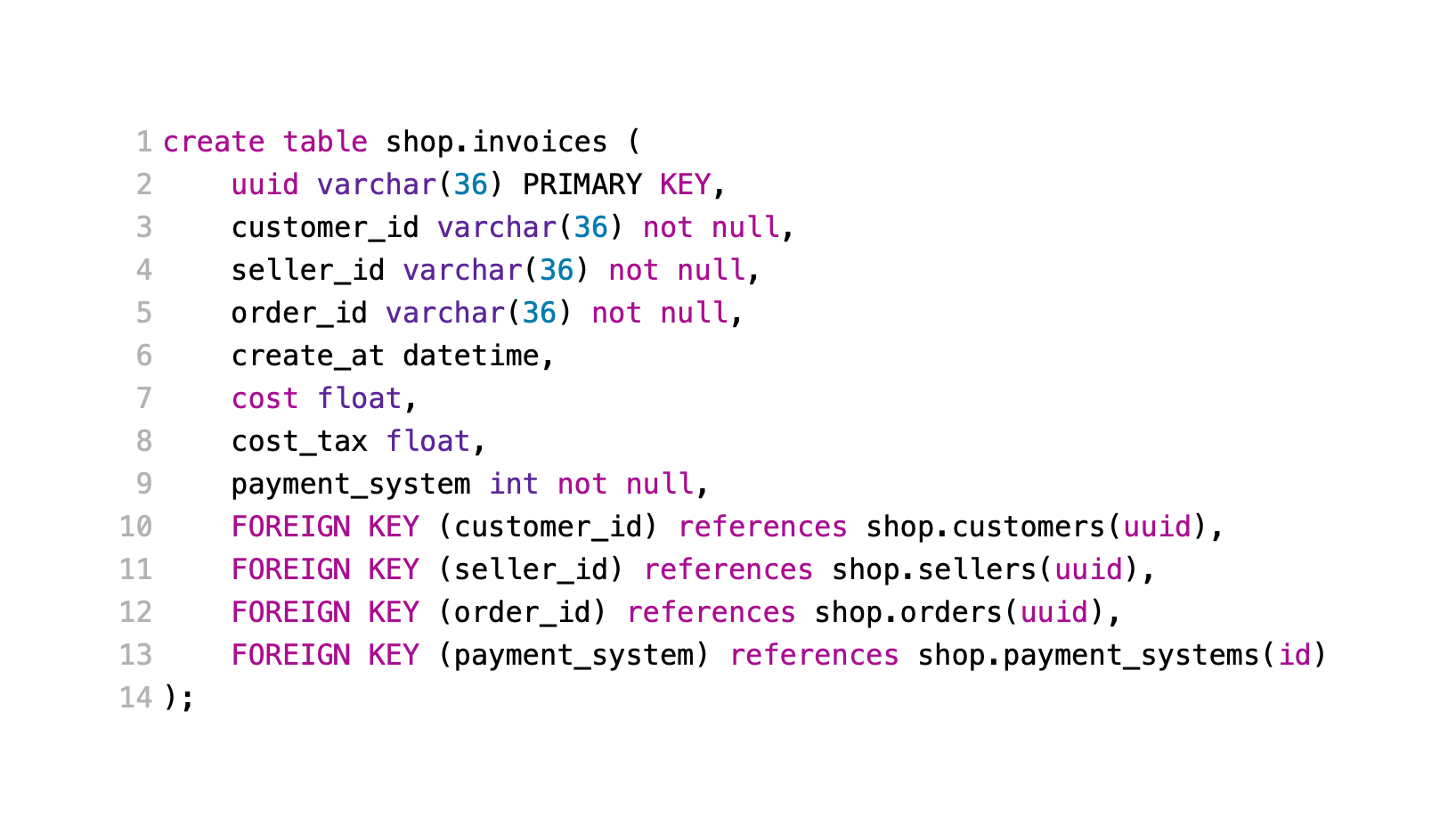
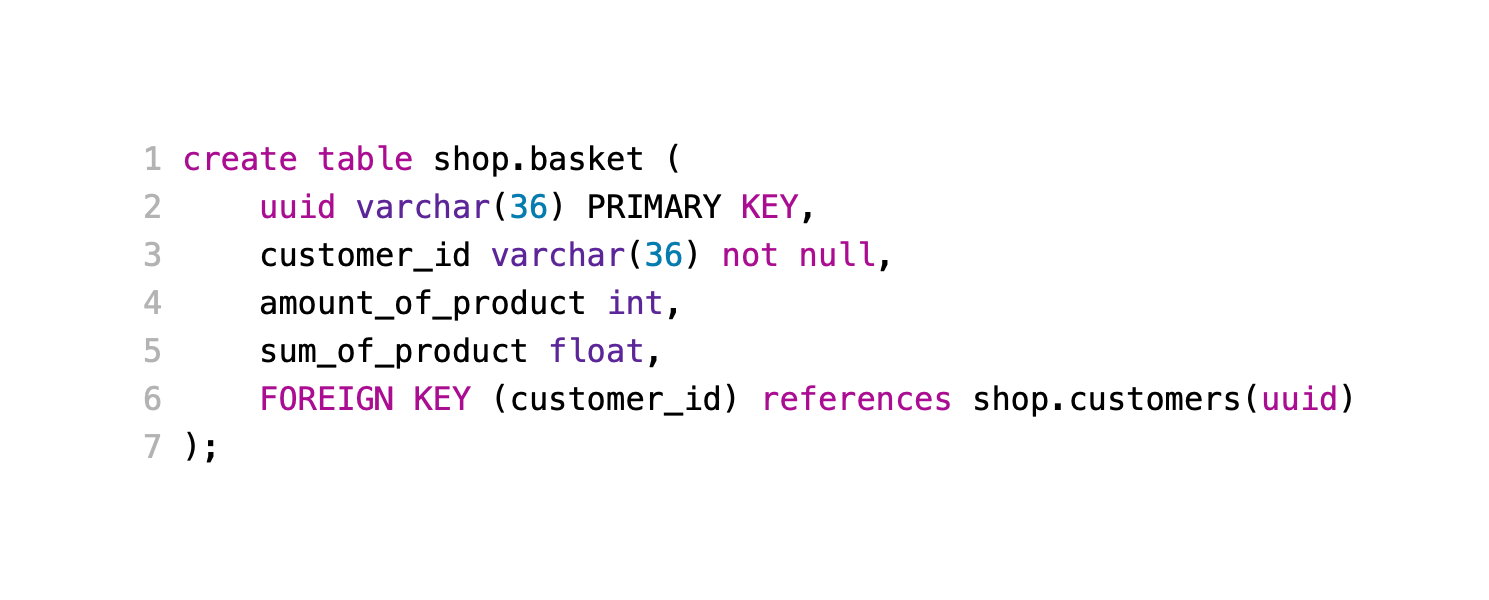


Рисунок 11. Создание таблицы “invoices”

Рисунок 12. Создание таблицы “basket”

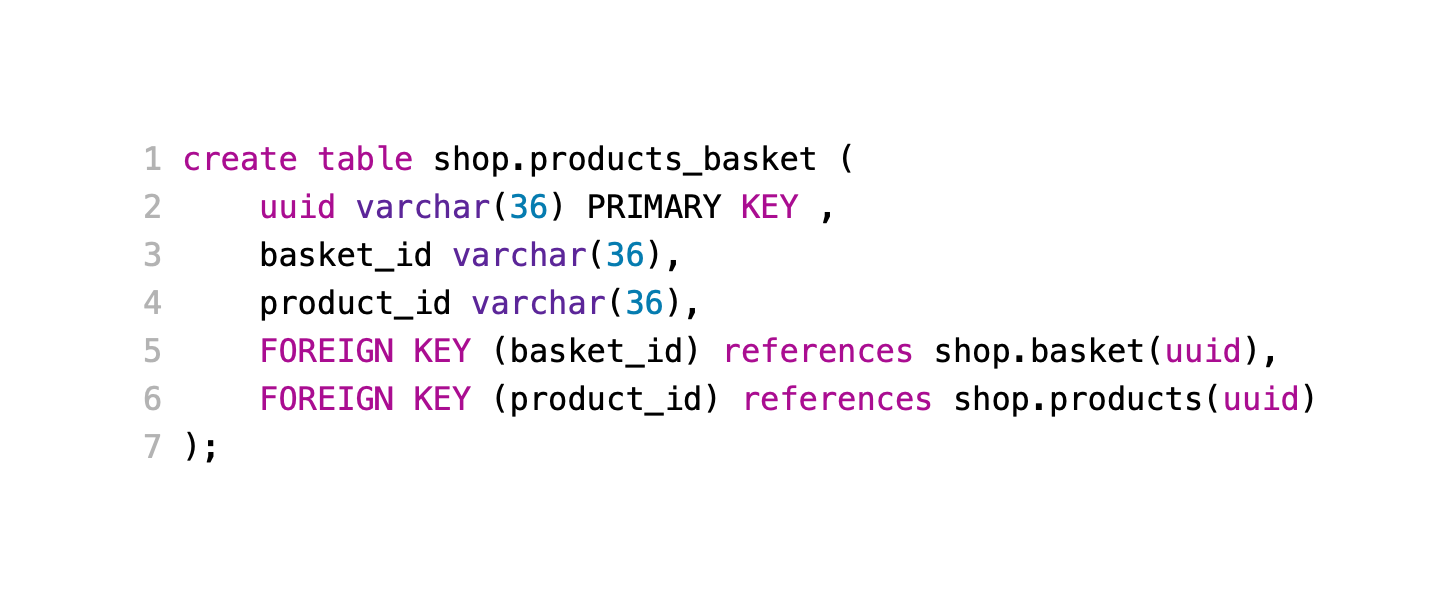


Рисунок 13. Создание таблицы “products\_basket”

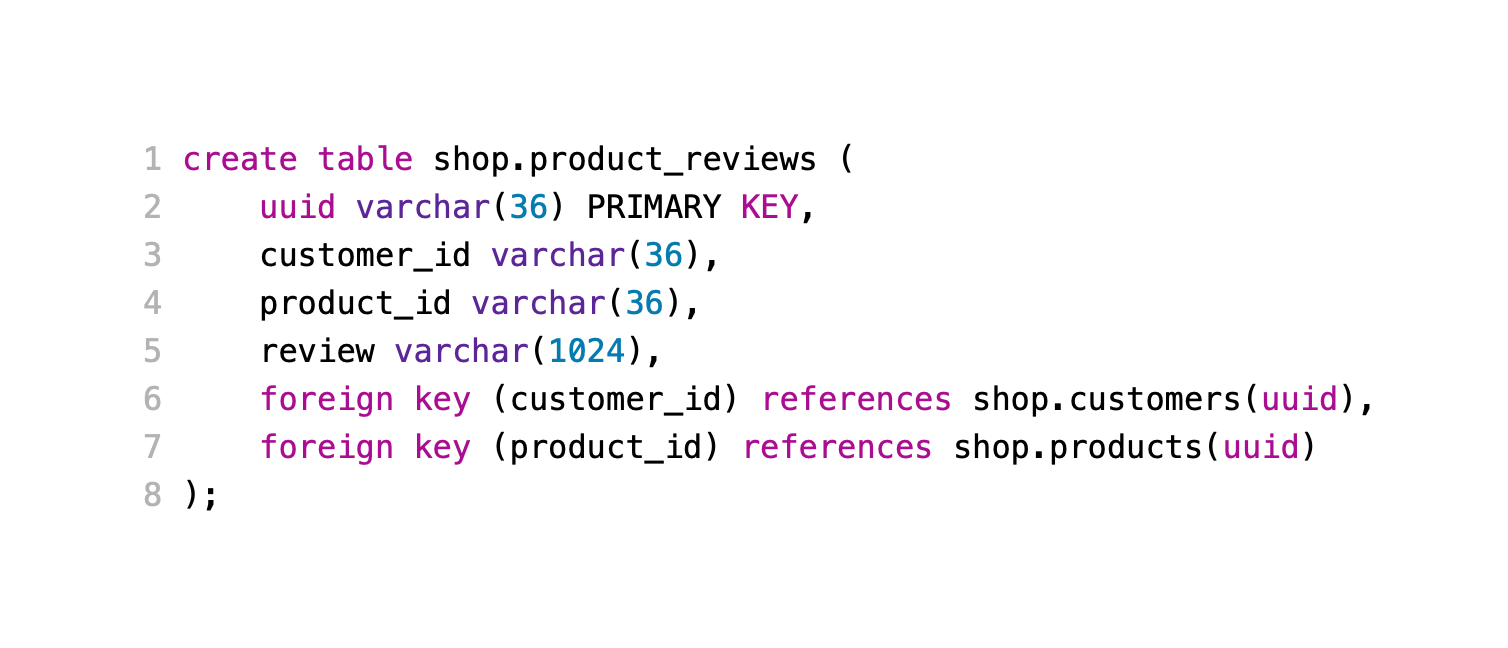


Рисунок 14. Создание таблицы “product\_rewiews”

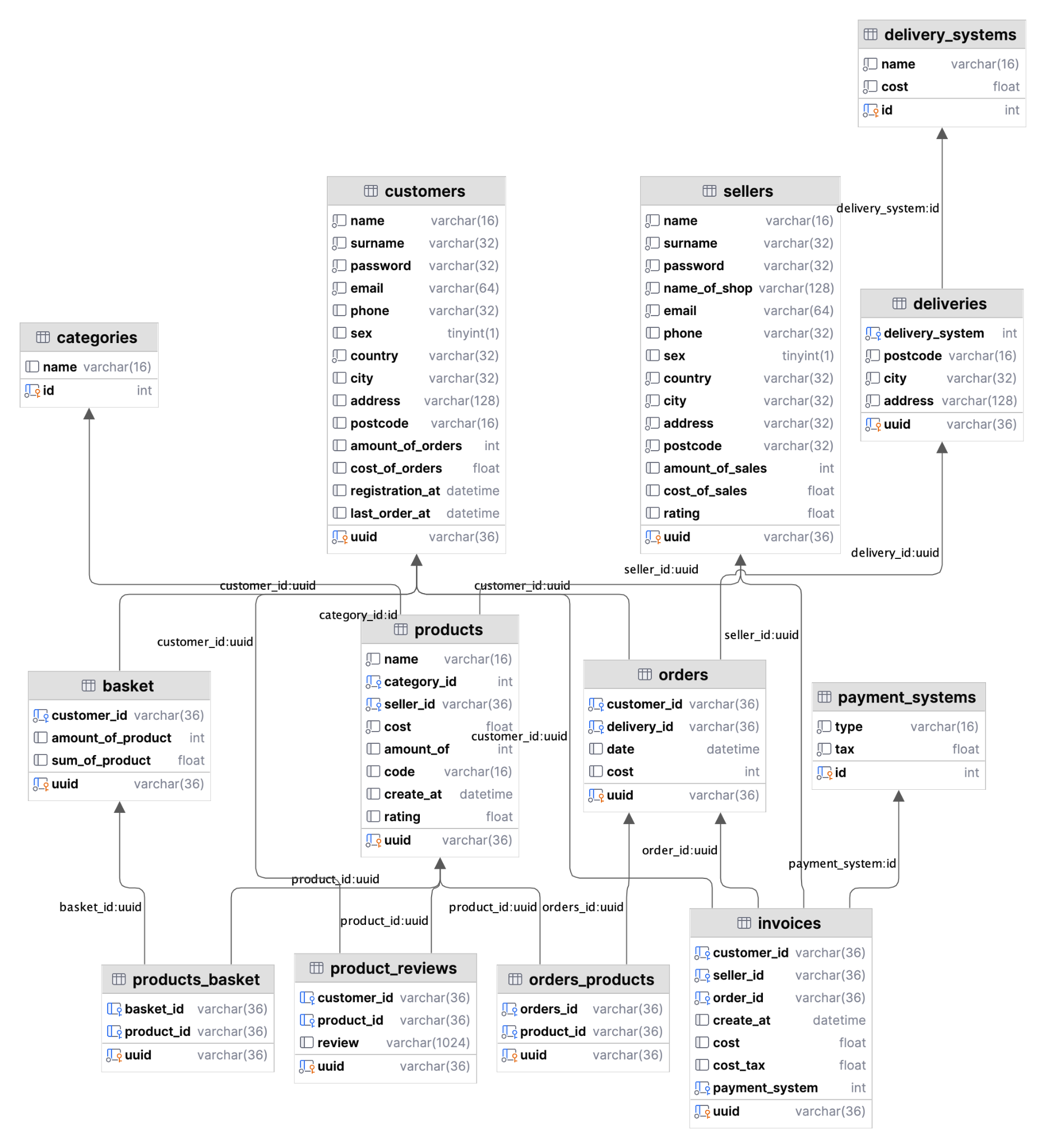


Рисунок 14. Схема “shop”

# 

# 6. Реализация функциональности базы данных

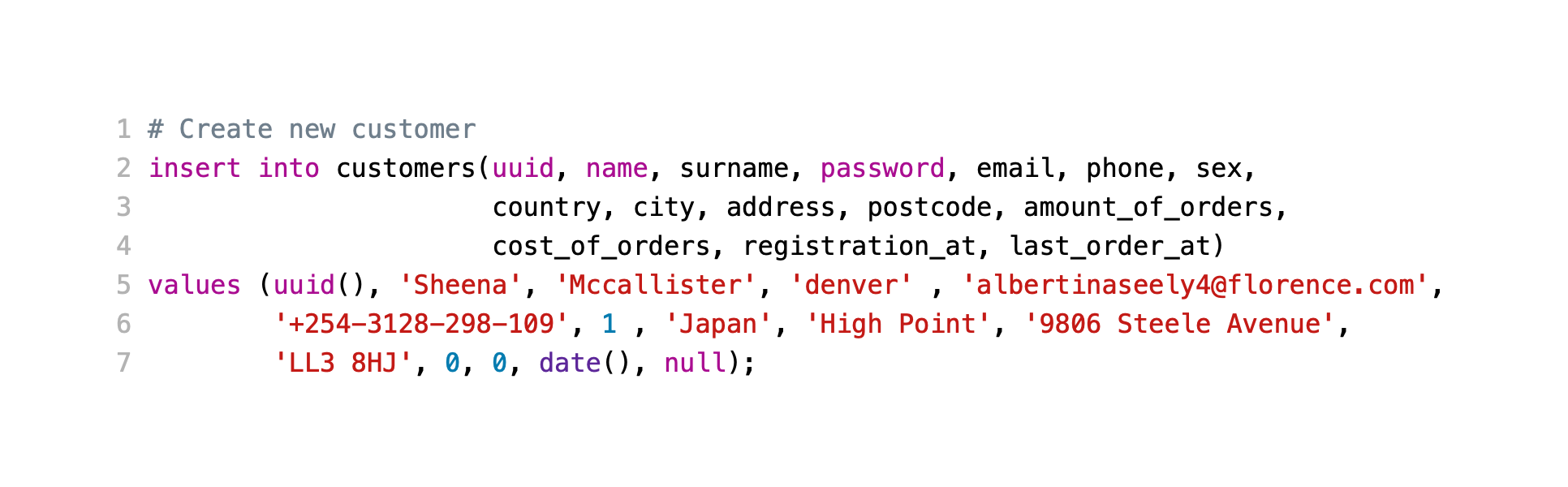


Рисунок 15. Создание нового покупателя

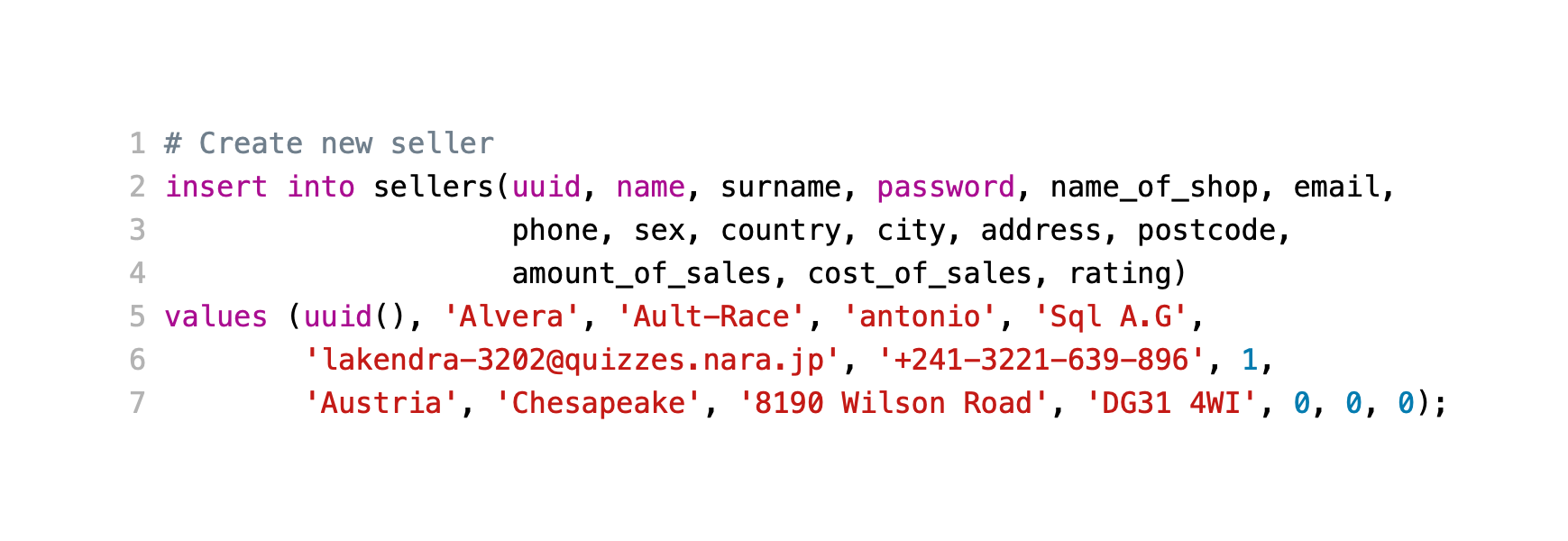


Рисунок 16. Создание нового продавца

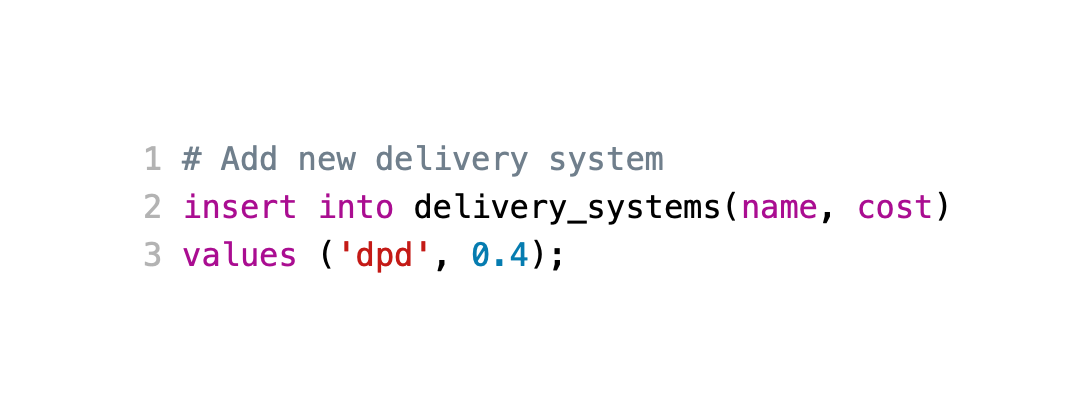


Рисунок 17. Создание новой системы доставки



Рисунок 18. Создание новой системы оплаты

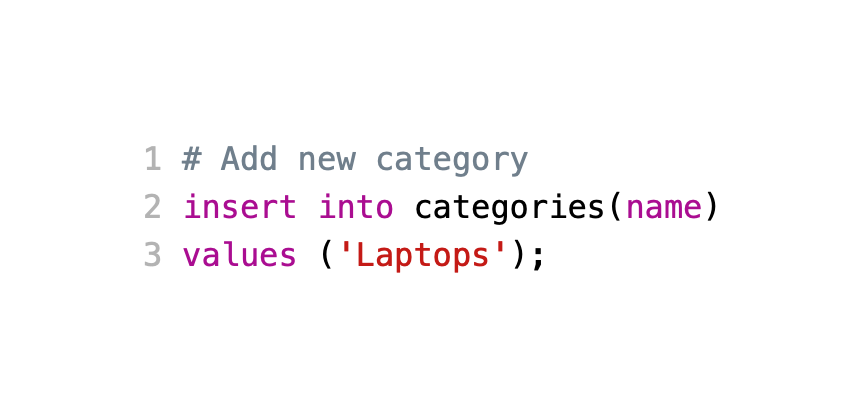


Рисунок 19. Создание новой категории

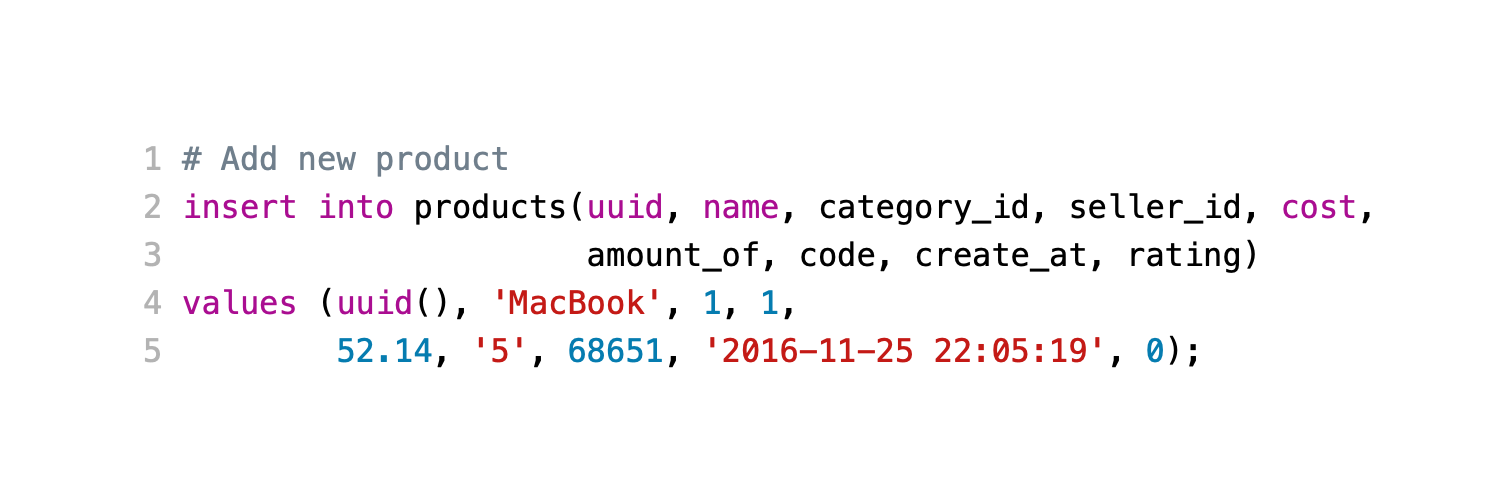


Рисунок 20. Создание нового продукта

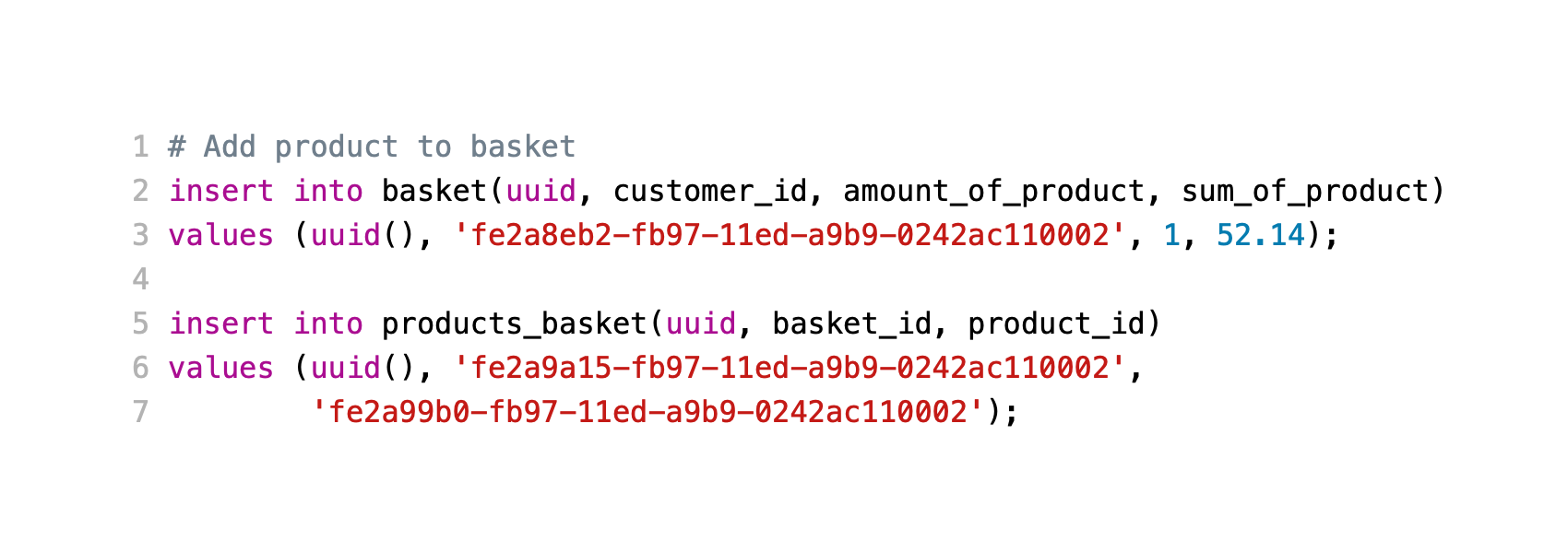


Рисунок 21. Создание новой корзины.

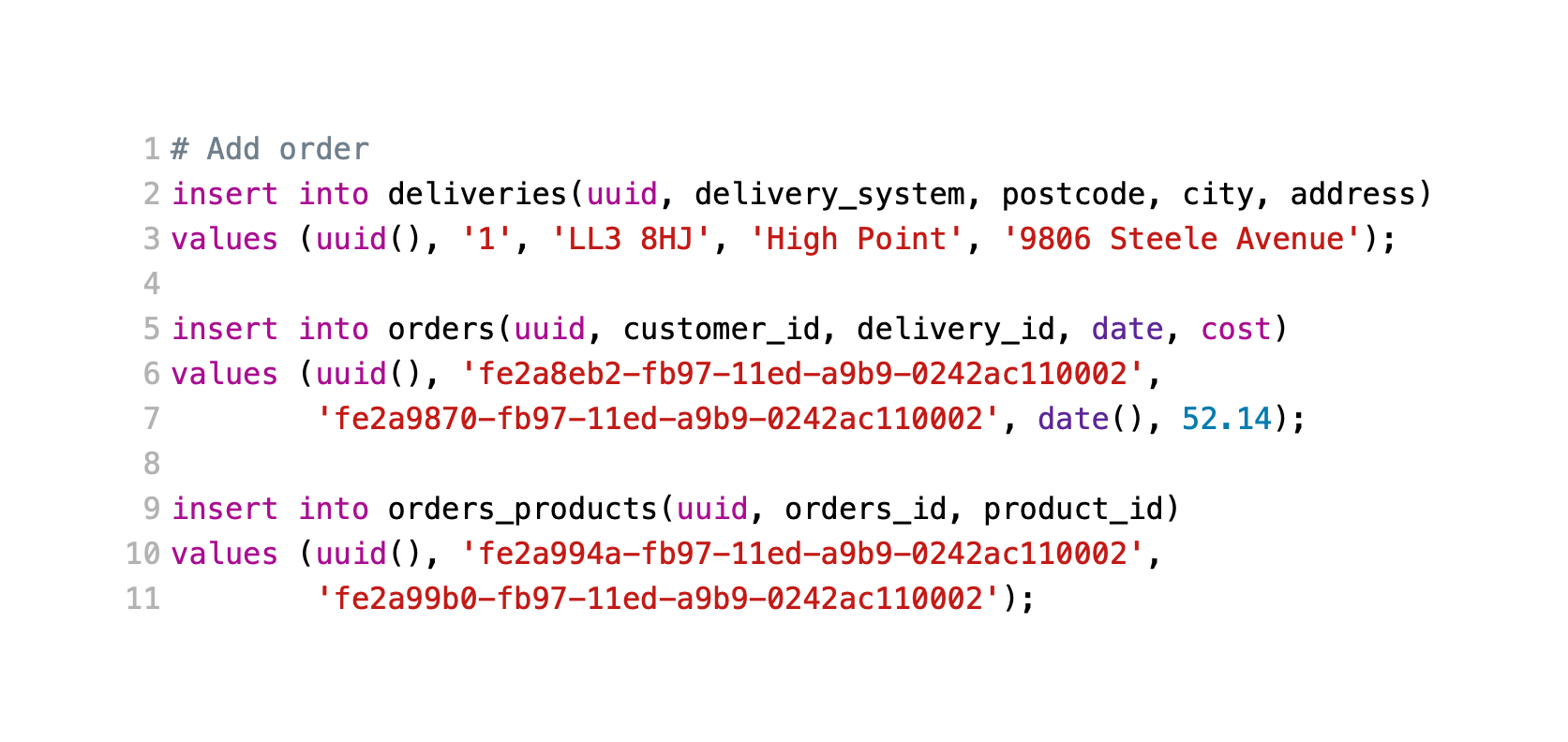


Рисунок 22. Создание нового заказа.

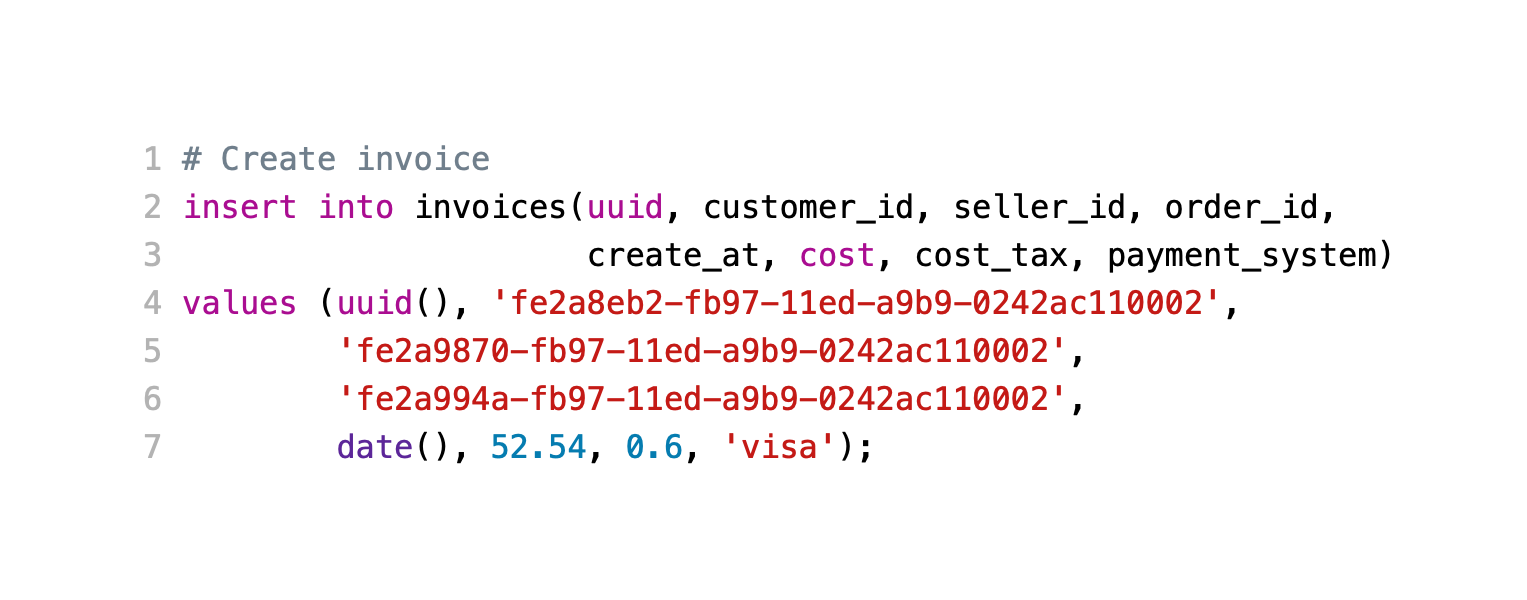


Рисунок 23. Создание нового счёта на оплату



Рисунок 24. Создание нового отзыва на продукт

# 7. Заключение

В ходе данной курсовой работы была проведена разработка базы данных для интернет-магазина. Целью данного проекта было создание эффективной и надежной системы хранения и управления данными, специально адаптированной для требований интернет-магазинов.

Для достижения этой цели были выполнены следующие задачи:

1. Анализ требований: были исследованы и проанализированы основные требования к базе данных интернет-магазина, такие как хранение информации о товарах, клиентах, заказах и т.д.

2. Проектирование схемы базы данных: была разработана логическая и физическая схемы базы данных, определены таблицы, их поля, связи и индексы. При проектировании были учтены принципы нормализации и обеспечение эффективного доступа к данным.

3. Реализация базы данных: была создана база данных с использованием выбранной системы управления базами данных (СУБД) и соответствующего языка запросов. Были созданы таблицы, индексы, представления и триггеры в соответствии с проектированием.

В результате выполнения данной курсовой работы была успешно разработана база данных для интернет-магазина, удовлетворяющая основным требованиям и обеспечивающая эффективное хранение и управление данными. Разработанная база данных предоставляет возможность удобного добавления, обновления и удаления информации о товарах, клиентах, заказах и других сущностях, что является важным компонентом функциональности интернет-магазина.

Однако, необходимо отметить, что разработка базы данных - это лишь одна из составляющих успешного интернет-магазина. Для полноценного функционирования также требуется разработка соответствующего программного обеспечения для обработки заказов, управления клиентскими данными и взаимодействия с базой данных. Кроме того, регулярное обслуживание базы данных и резервное копирование данных являются неотъемлемыми частями поддержки и безопасности системы.

В целом, разработка базы данных для интернет-магазина - это сложный и многогранный процесс, требующий внимательного анализа требований, грамотного проектирования и аккуратной реализации. Однако, правильно спроектированная и реализованная база данных играет важную роль в успешной работе интернет-магазина, обеспечивая эффективное управление данными и удовлетворение потребностей клиентов.

# 8. Список использованных источников

1. Кармайкл, Т. "Базы данных: проектирование, реализация и сопровождение" (Кармайкл, Т., 2018) - книга, которая охватывает основные аспекты разработки баз данных.

2. Сайт Oracle Documentation (https://docs.oracle.com/en/database/) - официальная документация Oracle Database, одной из наиболее популярных СУБД, содержит информацию о разработке баз данных и SQL.

3. Сайт Microsoft SQL Server Documentation (https://docs.microsoft.com/en-us/sql/) - официальная документация Microsoft SQL Server, другой широко используемой СУБД, с подробной информацией о разработке баз данных и SQL.

4. Date, C. J. "An Introduction to Database Systems" (Date, C. J., 2003) - классическая книга, в которой автор рассматривает основы баз данных и SQL.

5. Сайт W3Schools (https://www.w3schools.com/sql/) - онлайн-ресурс, предлагающий обширный набор учебных материалов и примеров по SQL.

6. Плодзе, Д., Шустан, М. "SQL: язык структурированных запросов" (Плодзе, Д., Шустан, М., 2013) - книга, охватывающая основы языка SQL и его применение при разработке баз данных.

7. Сайт MySQL Documentation (https://dev.mysql.com/doc/) - официальная документация MySQL, популярной открытой СУБД, содержит информацию о разработке баз данных и SQL.

8. Garcia-Molina, H., Ullman, J. D., Widom, J. "Database Systems: The Complete Book" (Garcia-Molina, H., Ullman, J. D., Widom, J., 2008) - учебник, охватывающий все аспекты баз данных и SQL.

9. Сайт PostgreSQL Documentation (https://www.postgresql.org/docs/) - официальная документация PostgreSQL, мощной открытой СУБД, с информацией о разработке баз данных и SQL.

10. Рамакришнан, Р., Герхке, Й. "Database Management Systems" (Ramakrishnan, R., Gehrke, J., 2002) - книга, которая представляет современный подход к разработке баз данных и SQL.