МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ

УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

Национальный исследовательский ядерный университет «МИФИ»



Институт

интеллектуальных кибернетических систем

Кафедра кибернетики (№22)

Направление подготовки 09.03.04 Программная инженерия

Курсовая работа по предмету

«Основы автоматизированных информационных технологий»

Тема: «Grow Food — сервис доставки рационов готовой полезной еды»

Преподаватель: Тихомирова Дарья Валерьевна

Студент: Баранов Александр Тимурович

Группа: Б22-534

**Содержание**

[**1 Описание предметной области**](#_63ixgoa7puc5) **3**

[1.1 Формулировка задания](#_ij1yfy5lfahe) 3

[1.2 Конкретизация предметной области](#_6pgyb2udl98k) 3

[1.3 Пользователи системы](#_9ecadpdhxr7b) 4

[1.4 Сроки хранения информации](#_ybf798x8h1m3) 5

[1.5 События, изменяющие состояние базы данных](#_twss7tb2n468) 6

[1.6 Основные запросы к базе данных (на естественном языке)](#_bdyoy4z7n7ht) 7

[**2 Концептуально-информационная модель предметной области**](#_ce7oqc573w78) **8**

[2.1 ER-диаграмма модели](#_nq7av54y0gqa) 8

[2.2 Оценка мощностных характеристик сущностей и связей](#_9wuto25fgx37) 8

[**3 Концептуальное проектирование**](#_84dzg1moavuf) **10**

[3.1 Принятые проектные соглашения](#_fjzly3xjbk2t) 10

[3.2 Обоснование выбора модели базы данных](#_sfohdro5h7g2) 10

[3.3 Используемые в системе кодификаторы](#_o9xufdry8vok) 10

[3.4 Концептуальная модель базы данных](#_o5z38ndohxus) 11

[**4 Логическое проектирование**](#_iyv0lkony6j3) **12**

[4.1 ER-диаграмма базы данных](#_h8iaqp7un4ys) 12

[4.2 Схемы отношений базы данных](#_hu0fdo17qj8r) 12

[4.3 Схема реляционной базы данных](#_cccr08a9t1xg) 12

[4.4 Схемы основных запросов на реляционной алгебре](#_uqeyte3tdw0i) 12

[**5 Физическое проектирование**](#_6t77l0vvpray) **14**

[5.1 Обоснование выбора конкретной СУБД](#_gp2jra4zqt76) 14

[5.2 Создание базы данных](#_koklzpreblex) 14

[5.3 Создание таблиц](#_2rcoru9aea5x) 15

[5.4 Заполнение таблиц. ETL-процессы загрузки базы данных](#_x3xa7onfv7l9) 15

[5.5 Запросы в терминах SQL](#_p7t5flpaukt5) 16

[5.6 Оценка размеров базы данных и каждого из файлов](#_oyi9da8cpeza) 18

[**6 Приложение. Отчеты**](#_f822v0lxa273) **20**

[6.1 Приложения](#_kzn37ez3v6h2) 20

[6.2 Отчеты](#_z65sh4z8l9mz) 58

[6.2.1 Отчет №1](#_ldew7dffk1gt) 58

[6.2.2 Отчет №2](#_ulmeksfp09bu) 60

[6.2.3 Отчет №3](#_bqw79u5uqii3) 62

# Описание предметной области

## Формулировка задания

Спроектировать базу данных для сервиса доставки здорового питания «Grow Food», поддерживающую полный жизненный цикл заказа: от его создания и обработки до доставки клиенту и взаимодействия с поставщиками продуктов и курьерами.

## Конкретизация предметной области

Сервис доставки здорового питания «Grow Food» – это информационная система, предназначенная для автоматизации процессов работы сервиса. Система рассчитана на взаимодействие с сотрудниками сервиса, при этом пользователи (клиенты) не имеют прямого доступа к

базе данных.

В рамках системы предполагается реализация следующего функционала:

* Работа с поставщиками: Учёт поставок продуктов и сроков доставки ингредиентов.
* Составление меню: Формирование меню на основе предпочтений пользователей и доступности ингредиентов. Меню составляется на следующий месяц и может быть адаптировано под индивидуальные запросы клиентов.
* Приготовление блюд: Учёт рецептов и формирование готовых рационов. Сотрудники кухни имеют доступ к информации об ингредиентах, необходимых для приготовления блюд, а также о самих блюдах: их количестве, размере порции, калорийности и других характе-

ристиках.

* Доставка: Курьеры имеют доступ к информации о доставленных или доставляемых заказах: дата, адрес доставки, способ оплаты и сумма к оплате (при необходимости).
* Оплата: Поддержка различных способов оплаты заказов, включая онлайн-платежи, а также учёт финансовых операций.
* Учёт предпочтений пользователей: Хранение информации о диетических ограничениях, аллергиях и предпочтениях клиентов для персонализации меню. Каждому пользователю в его личном кабинете доступна информация о выбранных предпочтениях, расписании меню на следующий месяц, количестве накопленных баллов, а также предложении оформить заказ.

Система обеспечивает взаимодействие между всеми участниками процесса: сотрудниками кухни, курьерами, поставщиками и администраторами сервиса, при этом пользователи взаимодействуют с системой через личный кабинет без прямого доступа к базе данных.

## Пользователи системы

1. Клиенты:
   * Это физические лица, которые заказывают готовые рационы питания через сервис «Grow Food».
   * Пользователи не имеют прямого доступа к базе данных, но взаимодействуют с системой через личный кабинет.
   * В личном кабинете клиентам доступна следующая информация:
     + Выбранные предпочтения (диетические ограничения, аллергии, предпочтения по

ингредиентам).

* + - Расписание меню на следующий месяц.
    - Количество накопленных баллов (если предусмотрена бонусная система).
    - Возможность оформления новых заказов и просмотра истории заказов.

1. Курьеры:
   * Это сотрудники сервиса, отвечающие за доставку готовых рационов клиентам.
   * Курьеры имеют доступ к информации о доставленных или доставляемых заказах, вклю-

чая:

* + - Дату и время доставки.
    - Адрес доставки.
    - Способ оплаты и сумму к оплате (если требуется).
  + Их задача – обеспечить своевременную и корректную доставку заказов, а также взаимо-

действие с клиентами при необходимости.

1. Поставщики:
   * Это юридические лица или индивидуальные предприниматели, поставляющие ингредиенты для приготовления блюд.
   * Поставщики взаимодействуют с системой через сотрудников сервиса, предоставляя информацию о поставках:
     + Наименование и количество поставляемых продуктов.
     + Сроки и условия доставки.
     + Качество и соответствие стандартам.
     + Их роль – обеспечить своевременные и качественные поставки ингредиентов для

приготовления блюд.

1. Администраторы:
   * Это сотрудники сервиса, отвечающие за управление системой и поддержание её работо-

способности.

* + Администраторы имеют полный доступ к базе данных и выполняют следующие функции:
    - Управление пользователями (клиентами, курьерами, поставщиками).
    - Контроль за составлением меню и процессом приготовления блюд.
    - Мониторинг доставки заказов и взаимодействие с курьерами.
    - Настройка и поддержка системы, включая обработку ошибок и обновление данных.

## Сроки хранения информации

В системе «Grow Food» предусмотрены различные сроки хранения информации в зависимости от её типа и назначения:

1. Информация о заказах:
   * Данные о завершённых заказах (история заказов) хранятся в течение 3 лет для обеспечения возможности анализа и отчётности.
   * По истечении этого срока заказы перемещаются в архив, где хранятся ещё 2 года перед

окончательным удалением.

1. Информация о пользователях:
   * Данные о клиентах (личные данные, предпочтения, история заказов) хранятся до момента удаления аккаунта пользователем.
   * Вслучаенеактивностиаккаунтавтечение2лет,данныеперемещаютсявархивихранятся

там ещё 1 год перед удалением.

1. Информация о поставщиках и ингредиентах:
   * Данные о поставщиках и поставках хранятся в течение 5 лет для обеспечения контроля качества и анализа сотрудничества.
   * Информация о списанных или более неиспользуемых ингредиентах перемещается в ар-

хив через 1 год после их последнего использования.

1. Информация о курьерах и доставках:
   * Данные о курьерах и их доставках хранятся в течение 3 лет для анализа эффективности

работы.

* + По истечении этого срока информация перемещается в архив, где хранится ещё 1 год.

1. Архивные данные:
   * Архивные данные хранятся в отдельной структуре базы данных для снижения нагрузки на основную систему.
   * Доступ к архивным данным ограничен и предоставляется только администраторам для

целей отчётности или анализа.

## События, изменяющие состояние базы данных

Основные события, которые влияют на состояние базы данных системы «Grow Food»,

включают:

* Регистрация новых пользователей или удаление (деактивация) аккаунтов старых пользо-

вателей.

* Оформление нового заказа (изменение его статуса).
* Устройство нового курьера на работу.
* Добавление нового поставщика или изменение договора со старым поставщиком.
* Поступление ингредиентов от поставщиков.
* Изменение меню.
* Начисление баллов пользователям.
* Изменение в статусе оплаты заказа.

Эти события обеспечивают актуальность данных в системе и позволяют отслеживать все ключевые процессы, связанные с работой сервиса.

## Основные запросы к базе данных (на естественном языке)

Основные запросы к БД в разрезе аналитики сервиса:

1. Вычисление ежедневной выручки и её прироста.
2. Вычисление НДС и цен без учёта налога.
3. Анализ возвращённых или отменённых заказов.
4. Анализ распределения пользователей по количеству их заказов, количеству бонусов.
5. Нахождение блюд с популярными (по предпочтению пользователей) ингредиентами.
6. Вычисление необходимого количества ингредиентов и прогнозирование поставок.
7. Определение прибыльности и актуальности меню на основе заказов

# Концептуально-информационная модель предметной области

## ER-диаграмма модели

Для создания диаграммы использовался сервис [drawio](https://www.drawio.com/). Получившийся рисунок приведен в приложении.

## Оценка мощностных характеристик сущностей и связей

| **Сущность/связь** | **Мощность** | | |
| --- | --- | --- | --- |
|  | **минимальная** | **средняя** | **максимальная** |
| Пользователь | 0 | 10000 | 1000000 |
| Предпочтение | 0 | 10 | 20 |
| Способ оплаты | 0 | 5 | 10 |
| Заказ | 0 | 50000 | 5000000 |
| Курьер | 0 | 100 | 1000 |
| Меню | 0 | 50 | 500 |
| Блюдо | 0 | 200 | 2000 |
| Ингредиент | 0 | 500 | 5000 |
| Поставщик | 0 | 20 | 100 |
| Пользователь имеет предпочтения | 0 | 20000 | 2000000 |
| Пользователь пригласил пользователя | 0 | 2000 | 200000 |
| Пользователь выбирает способ оплаты | 0 | 10000 | 1000000 |
| Пользователь имеет заказы | 0 | 50000 | 5000000 |
| Курьер доставляет заказы | 0 | 50000 | 5000000 |
| Заказ содержит меню | 0 | 50000 | 5000000 |

Все представленные мощностные характеристики основываются на следующих сообра-

жениях:

* Минимальное количество соответствует состоянию, когда база данных только заведена и зарегестрированных пользователей еще нет.
* Средние значения указаны из расчёта на «нормальную» работу системы.
* Максимальные значения указаны для случая работы системы в режиме большой нагруз-

ки.

* Сложно оценить мощностные характеристики числа приглашенных пользователей, а также числа выбранных пользователем предпочтений, так как это число напрямую зависит от характера пользователей.

# Концептуальное проектирование

## Принятые проектные соглашения

* Имена таблиц указываются в snake\_case во множественном числе: users, orders.
* Составные атрибуты (например, “Имя, фамилия, отчество”) разделяются на отдельные

поля: first\_name, last\_name.

* Для внешних ключей используется шаблон {table\_name}\_id (например, user\_id для связи с пользователем).
* Первичные ключи (PK) являются автоинкрементными целыми числами (SERIAL в PostgreSQL) для всех сущностей.
* Внешние ключи (FK) реализуются с указанием действий при удалеии (CASCADE, SET NULL).
* Таблицы нормализованы по третьей нормальной форме (атрибуты зависят от PK и нет

транзитивных связей).

## Обоснование выбора модели базы данных

В качестве модели базы данных была выбрана реляционная модель. Это решение обос-

новано несколькими факторами:

1. Структурированность данных. Информация, обрабатываемая системой (пользователи, заказы, курьеры, предпочтения, способы оплаты), имеет чёткую табличную структуру с фиксированными атрибутами. Это естественно укладывается в формат таблиц с чётко заданными связями между ними.
2. С реляционными БД можно работать с помощью такой активно поддерживаемой СУБД, как PostgreSQL. Это позволит эффективно нормализовать данные и поддерживать целостность благодаря механизмам внешних ключей, ограничений и транзакций. Эта СУБД предоставляет мощные инструменты для выполнения аналитических запросов (в том числе оконные функции, CTE, агрегации), что соответствует потребностям проекта в постро-

ении аналитических отчётов и статистики.

## Используемые в системе кодификаторы

В проектируемой информационной системе используются следующие кодификаторы:

1. preference\_categories — категории предпочтений пользователей

Данный кодификатор содержит категории пищевых предпочтений и ограничений, поз-

воляющие учитывать индивидуальные особенности клиентов при формировании рационов:

* + аллергены
  + десерты, выпечка, сахар
  + мясо, рыба
  + овощи, лук, чеснок
  + гарниры, каши

1. payment\_methods — методы оплаты

Кодификатор содержит список возможных способов оплаты, а также признак необходимости реквизитов:

| Метод оплаты | Требует реквизиты |
| --- | --- |
| Наличные курьеру | Нет |
| Карта курьеру | Да |
| Картой онлайн | Да |
| Яндекс.Сплит | Да |

1. order\_status — статусы заказов

Кодификатор содержит возможные состояния заказа в процессе его обработки:

* + new — заказ создан
  + in\_delivery — передан курьеру
  + delivered — доставлен
  + cancelled — отменён
  + returned — возвращён

## Концептуальная модель базы данных

Для создания диаграммы использовался сервис [drawio](https://www.drawio.com/). Получившийся рисунок приведен в приложении.

# Логическое проектирование

## ER-диаграмма базы данных

Для создания диаграммы использовался сервис [drawio](https://www.drawio.com/). Получившийся рисунок приведен в приложении.

## Схемы отношений базы данных

Схема была построена встроеными средствами инструмента с графическим интерфейсом для управления базами данных PostgreSQL [PgAdmin](https://www.pgadmin.org/). Получившийся рисунок приведен в приложении.

## Схема реляционной базы данных

| **Отношение** | **Атрибуты** |
| --- | --- |
| R1 | (#id, first\_name, middle\_name, last\_name, sex, birth\_date, phone\_number, invited\_by, bonuses, address) |
| R2 | (#id, preference, user) |
| R3 | (#id, title, preference\_category) |
| R4 | (#id, preference, preference\_category) |
| R5 | (#id, title) |
| R6 | (#id, title, require\_requisites) |
| R7 | (#id, user, payment\_method, requisites) |
| R8 | (#id, user, menu, created\_at, status, courier, payment\_info) |
| R9 | (#id, first\_name, middle\_name, last\_name, birth\_date) |
| R10 | (#id, title, cost, count\_dishes, colorfulness) |
| R11 | (#id, title, type, weight, colorfulness) |
| R12 | (#id, dish, ingredient) |
| R13 | (#id, title, type, cost) |
| R14 | (#id, ingredient, supplier) |
| R15 | (#id, title, productivity) |
| R16 | (#id, ingredient, preference) |

## Схемы основных запросов на реляционной алгебре

1. Получить всех пользователей, у которых задано хотя бы одно предпочтение

| Users ( i d ) = u s e r . i d ] | ( u s e r [ u s e r . i d | = | u s e r s \_ p r e f e r e n c e s . u s e r ] u s e r s \_ p r e f e r e n c e s ) [ |
| --- | --- | --- | --- |

1. Получить все блюда, в которых используются ингредиенты дороже 100

| Dis he s ( i d ) | = ( ( d i s h [ d i s h . i d = d i s h e s \_ i n g r e d i e n t s . d i s h ] d i s h e s \_ i n g r e d i e n t s |
| --- | --- |
| ) | [ d i s h e s \_ i n g r e d i e n t s . i n g r e d i e n t = i n g r e d i e n t . i d & i n g r e d i e n t .  c o s t > 100] i n g r e d i e n t ) [ d i s h . i d ] |

1. Получить заказы, оплаченные через метод, требующий реквизиты

| Or de rs ( i d ) | = | ( ( o r d e r [ o r d e r . p a y m e n t \_ i n f o = p a y m e n t \_ i n f o . i d ] p a y m e nt \_ i n f o ) [ p a y m e n t \_ i n f o . payment\_method = payment\_method . i d & payment\_metho r e q u i r e \_ r e q u i s i t e s = **true** ] payment\_method ) [ o r d e r . i d ] |
| --- | --- | --- |

1. Получить всех поставщиков, которые поставляют хотя бы один ингредиент типа ‘овощ’

| S u p p l i e r s ( i d ) = ( ( s u p p l i e r [ s u p p l i e r . i d = i n g r e d i e n t s \_ s u p p l i e r s . s u p p l i e r ] i n g r e d i e n t s \_ s u p p l i e r s )  [ i n g r e d i e n t s \_ s u p p l i e r s . i n g r e d i e n t = i n g r e d i e n t . i d & i n g r e d i e n t . t y p e = ' овощ ' ] i n g r e d i e n t )  [ s u p p l i e r . i d ] |
| --- |

1. Найти заказы, содержащие блюда с калорийностью выше 300

| Or de rs ( i d ) = ( o r d e r [ o r d e r . menu = menu . i d & menu . c o l o r f u l n e s s  [ o r d e r . i d ] | > | 300] menu ) |
| --- | --- | --- |

1. Получить все категории предпочтений, в которых есть хотя бы одно предпочтение

| C a t e g o r i e s ( i d ) | = | ( p r e f e r e n c e \_ c a t e g o r y  [ p r e f e r e n c e \_ c a t e g o r y . i d = p r e f e r e n c e s \_ c a t e g o r i e s . p r e f e r e n c e \_ c a t e g o r y ] p r e f e r e n c e s \_ c a t e g o r i e s ) [ p r e f e r e n c e \_ c a t e g o r y . i d ] |
| --- | --- | --- |

# Физическое проектирование

## Обоснование выбора конкретной СУБД

В качестве системы управления базами данных (СУБД) для реализации проекта была

выбрана PostgreSQL. Этот выбор обоснован рядом факторов:

* PostgreSQL — это мощная объектно-реляционная СУБД с открытым исходным кодом, которая поддерживает расширенные возможности SQL и соответствует стандартам.
* Система обладает высокой производительностью и эффективной планировкой запросов, что важно при работе с большим объемом данных и сложными связями между таблицами.
* PostgreSQL широко используется в индустрии и хорошо поддерживается сообществом, что обеспечивает доступность документации, инструментов и решений типовых задач.
* Также немаловажным преимуществом является наличие встроенной поддержки пользовательских типов, таких как перечисления (ENUM), которые активно применяются в данной базе данных.
* Наконец, PostgreSQL хорошо интегрируется с Python через библиотеки psycopg и SQLAlchemy, что важно на этапе генерации и заполнения данных.

Таким образом, PostgreSQL является оптимальным выбором как с технической, так и с практической точек зрения.

## Создание базы данных

Создание и запуск базы данных производится с использованием среды контейнеризации Docker. Это позволяет упростить настройку окружения, добиться воспроизводимости конфигу-

рации и обеспечить удобство развертывания.

Для запуска PostgreSQL используется официальный образ postgres, настраиваемый через docker-compose. Все параметры подключения (имя пользователя, пароль, имя базы данных

и порт) передаются через файл .env, что обеспечивает гибкость и безопасность конфигурации.

Для работы с базой и отслеживания её состояния применяется PGAdmin, также запускаемый как отдельный контейнер. Он позволяет удобно просматривать содержимое таблиц, выполнять SQL-запросы и отслеживать структуру базы данных через графический интерфейс.

Создание таблиц производится автоматически при старте контейнера с помощью специального SQL-файла schema.sql, который монтируется в контейнер базы данных и исполняется через docker-entrypoint-initdb.d. Эта схема позволяет без участия пользователя разворачивать структуру БД при первом запуске.

Таким образом, вся инфраструктура для базы данных развернута в виде контейнеров, а структура таблиц автоматически формируется при старте, что делает процесс развертывания гибким и надежным.

## Создание таблиц

Создание таблиц производилось с помощью инициирующего SQL-запроса schema.sql, исполняющегося через docker-entrypoint-initdb.d. Код для создания всех таблиц вынесен в приложение.

## Заполнение таблиц. ETL-процессы загрузки базы данных

Заполнение базы данных проводилось в несколько этапов с использованием различных источников и инструментов, в зависимости от характера данных.

1. Источники данных

Для так называемых статических сущностей — то есть данных, которые либо редко изменяются, либо представляют собой заранее известные справочники, были использованы как

реальные источники, так и логически обоснованные предположения:

* + Данные о предпочтениях пользователей, категориях предпочтений и стоимости меню были заимствованы из существующего сервиса по доставке здорового питания [GrowFood](http://growfood.pro/).
  + Информация о типах ингредиентов, блюдах и их составе формировалась на основе анализа ассортимента типовых рационов здорового питания, а также общих гастрономических принципов. Эти данные предварительно подготавливались в виде CSV-файлов и импор-

тировались в базу.

1. Генерация и вставка данных

Остальные сущности, в том числе:

* + Пользователи,
  + Курьеры,
  + Заказы,
  + Платёжные методы,
  + Поставщики и поставки,

генерировались программно с использованием библиотеки [Faker (ru\_RU)](https://fakerjs.dev/), обеспечивающей ре-

алистичные русскоязычные данные (имена, адреса, даты рождения и т.д.).

Для этого были реализованы Python-скрипты, которые подключаются к базе данных с помощью библиотеки [psycopg2](https://www.psycopg.org/docs/). Скрипты построены по модульному принципу: каждая таблица имеетсобственныйскрипт-загрузчик(seeder),аглавныймодульзапускаетзаполнениевнужной последовательности, учитывая зависимости между таблицами (например, сначала пользовате-

ли — затем заказы).

Код для Python-скриптов вынесен в приложение

После выполнения всех функций заполнения базы данных были получены заполненные таблицы, проиллюстрированные в приложении

## Запросы в терминах SQL

1. Найти выручку сервиса за последний месяц. Код запроса, результат выполнения.
2. Какая доля (в %) общей выручки приходится на меню с названием «Похудение»? Про-

центы округлить до двух знаков после запятой. Код запроса, результат выполнения.

1. Вычислить НДС каждого меню и рассчитать цену каждого меню не включая НДС. Вывести название меню, его текущую цену, НДС и цену без НДС. Значения округлить до двух

знаков после запятой. Код запроса, результат выполнения.

1. Найти заказы, которые оказались возвращены и были оплачены с помощью «Ян-

декс.Сплит». Вывести id заказа, дату заказа и реквизиты платежа. Код запроса, результат

выполнения.

1. Найти топ-5 пользователей по количеству бонусов, которые большинство своих заказов оплатили наличными курьеру. Вывести id пользователя, его имя и количество бонусов. Код запроса, результат выполнения.
2. Найти блюда, содержащие хотя бы один ингредиент, который относится к самой популярной категории предпочтений. Вывести только названия блюд, отсортированные в ал-

фавитном порядке. Код запроса, результат выполнения.

1. Вывести 2 строки с названием меню, массивом входящих в него блюд и даты, которые принесут больше всего прибыли (численно и в % относительно себестоимости). Код запроса, результат выполнения.

Пусть блюдо состоит из ингридиентов 𝑥1,𝑥2,…,𝑥𝑛. Закупочная стоимость каждого ингридиента равна 𝑐1,𝑐2,…,𝑐𝑛 соответственно. Тогда я предполагаю, что себестоимость блюда

равна 𝑝 = ∑𝑛𝑖=1 𝑐𝑖 (я не учитываю, что блюдо содержит 𝑎 грамм данного ингридиента).

Пусть меню на конкретный день состоит из блюд 𝑦1,𝑦2,…,𝑦𝑚. Тогда себестоимость

меню будет равна 𝑃 = ∑𝑚𝑗=1 𝑝𝑗.

Розничная цена на данное меню равна 𝑆. Тогда прибыль от данного меню равна 𝑆 − 𝑃. Будем находить 2 меню: с наибольшим значением 𝑆 − 𝑃 и с наибольшим значением 𝑆−𝑃𝑃 .

Заметим, что цена ингредиента указана в долларах, а цена меню в рублях. Поэтому для вычисления прибыли нужно умножить цену ингредиента на курс доллара. Курс доллара принять равным 1 доллар = 7.55 рублей (поправка на нереалистичную заполненность базы данных

и предположение о составе блюда).

1. Какиепары блюднаходятсявместевменючащевсего?Вывестиихназванияиколичество

нахождений вместе. Код запроса, результат выполнения.

1. Найти медианное количество заказов у пользователей. Код запроса, результат выполне-

ния.

Если количество строк чётное, то медиана - это среднее двух средних значений. Если

количество строк нечётное, то медиана - это значение в середине.

Если всего строк чётное количество, то total\_rows / 2 и total\_rows / 2 + 1 - целые чис-

ла, на выходе будет две строки и среднее арифметическое возьмётся от двух значений.

Если всего строк нечётное количество, то total\_rows / 2 и total\_rows / 2 + 1 - нецелые числа, на выходе будет одна строка и это и будет медиана.

1. Рассчитать ежедневную выручку сервиса, рассчитать ежедневный прирост выручки (численно и в %) относительно предыдущего дня. Код запроса, результат выполнения.
2. Построить иерархию приглашенных и пригласивших пользователей. Код запроса, результат выполнения.
3. Пусть сегодня 1 число какого-то месяца. У нас есть список заказов за предыдущий месяц. В предположении, что количество заказов на каждую позицию в меню останется таким же, вывести список ингредиентов, для которых следует нанять ещё поставщиков (которых не хватит для изготовления нужных блюд на ближайший месяц). Рассчитывать со следующим допущением: если блюдо весит 𝑎 грамм и для его изготовления нужно 𝑛 ингредиентов, то каждого ингредиента нужно в количестве 𝑛𝑎 грамм. Код запроса, результат выполнения.

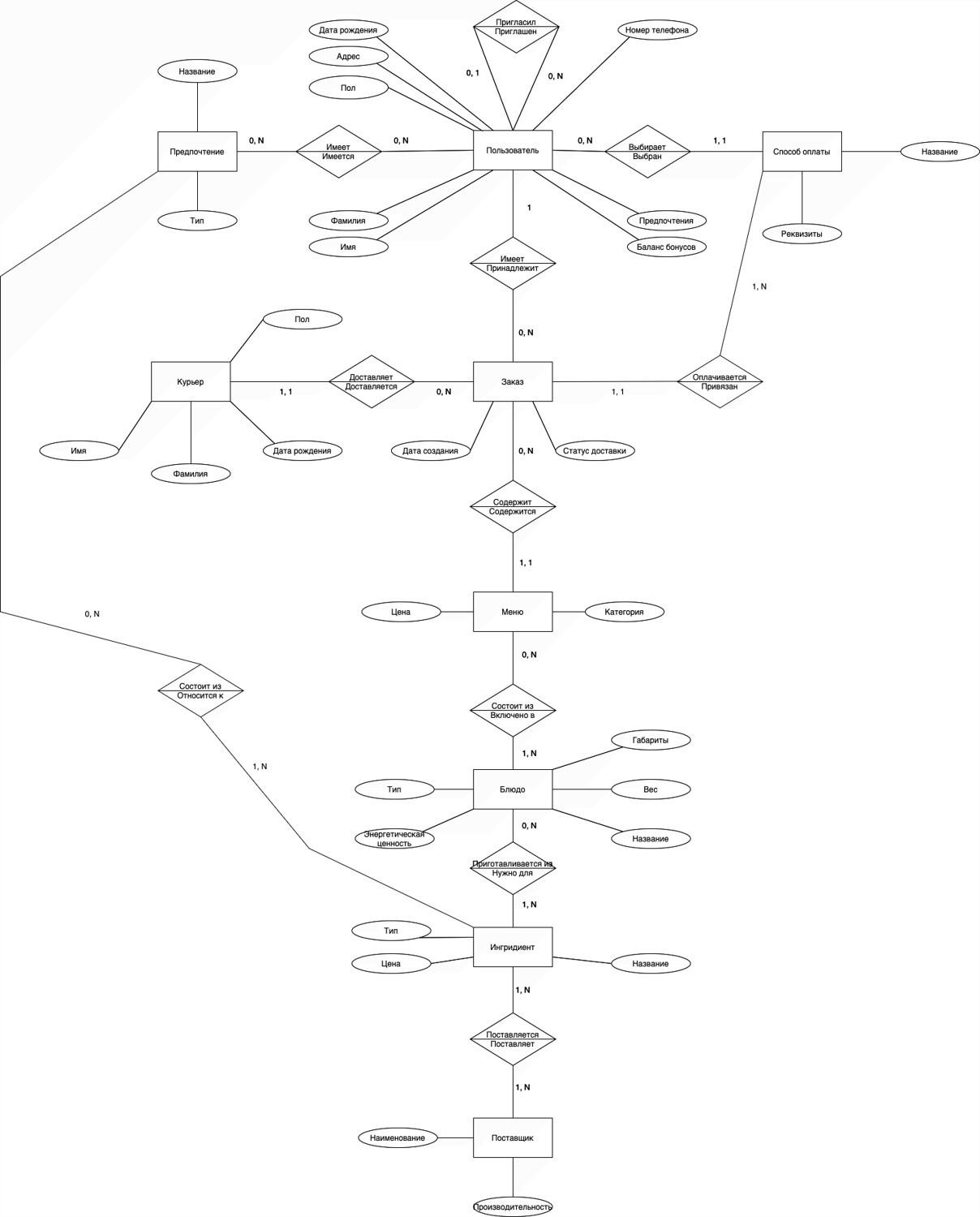
## Оценка размеров базы данных и каждого из файлов

| **Отношение** | **Атрибут** | **Тип данных** | **Размер, байт** | **Среднее количество** | **Объем, байт** | **Объём, МБ** |
| --- | --- | --- | --- | --- | --- | --- |
| users | id  sex birth\_date invited\_by\_id bonuses | integer  gender date integer integer | 4  4  4  4  4 | 10000 | 1687552 | 1.61 |
|  | address | varchar(200) | 400 |  |  |  |
|  | first\_name | varchar(50) | 100 |  |  |  |
|  | middle\_name | varchar(50) | 100 |  |  |  |
|  | last\_name | varchar(50) | 100 |  |  |  |
|  | phone\_number | varchar(10) | 20 |  |  |  |
| dishes | id  weight colorfulness | integer  double double | 4  8  8 | 106 | 57344 | 0.05 |
|  | title | varchar(100) | 200 |  |  |  |
|  | type | varchar(100) | 200 |  |  |  |
| dishes\_ingredients | id  dish\_id | integer  integer | 4  4 | 534 | 81920 | 0.08 |
|  | ingredient\_id | integer | 4 |  |  |  |
| ingredients | id  title | integer  varchar(100) | 4  200 | 149 | 57344 | 0.05 |
|  | type | varchar(100) | 200 |  |  |  |
|  | cost | double | 8 |  |  |  |
| dishes\_menus | id  dish\_id | integer  integer | 4  4 | 360 | 57344 | 0.05 |
|  | menu\_id | integer | 4 |  |  |  |
|  | date | date | 4 |  |  |  |
| menus | id  title  cost | integer varchar(100) double | 4  200  8 | 3 | 24576 | 0.02 |
|  | count\_dishes | integer | 4 |  |  |  |
|  | colorfulness | double | 8 |  |  |  |

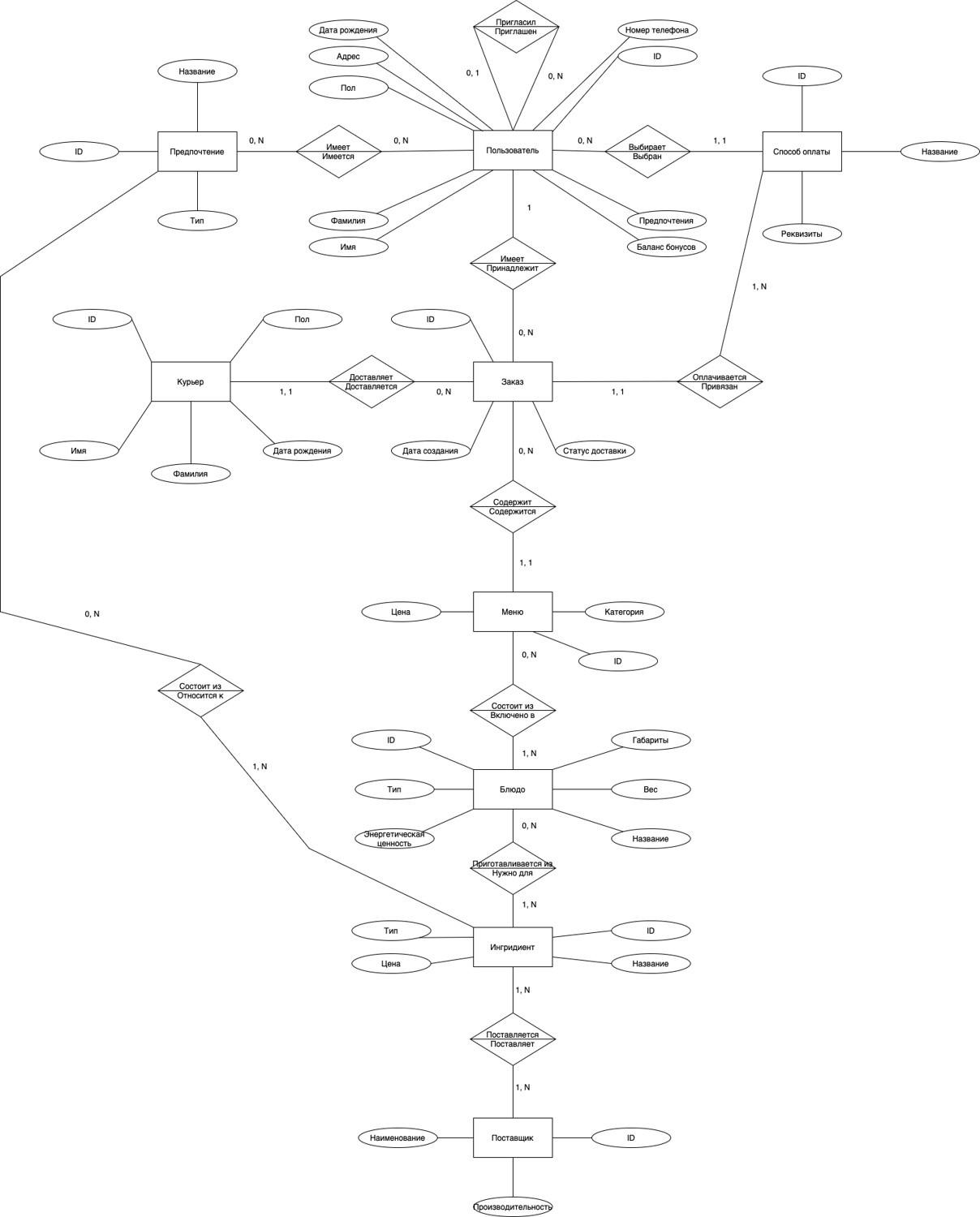
| **Отношение** | **Атрибут** | **Тип данных** | **Размер, байт** | **Среднее количество** | **Объем, байт** | **Объём, МБ** |
| --- | --- | --- | --- | --- | --- | --- |
| ingredients\_suppliers | id  ingredient\_id | integer  integer | 4  4 | 408 | 81920 | 0.08 |
|  | supplier\_id | integer | 4 |  |  |  |
| suppliers | id  title | integer  varchar(100) | 4  200 | 20 | 24576 | 0.02 |
|  | productivity | double | 8 |  |  |  |
| payment\_infos | id  user\_id | integer  integer | 4  4 | 19936 | 1638400 | 1.56 |
|  | payment\_method\_id | integer | 4 |  |  |  |
|  | requisites | varchar(100) | 200 |  |  |  |
| payment\_methods | id  title | integer  varchar(100) | 4  200 | 4 | 24576 | 0.02 |
|  | require\_requisites | boolean | 1 |  |  |  |
| preferences\_categories | id | integer | 4 | 5 | 24576 | 0.02 |
|  | title | varchar(100) | 200 |  |  |  |
| couriers | id first\_name  middle\_name | integer varchar(50) varchar(50) | 4  100  100 | 100 | 57344 | 0.05 |
|  | last\_name | varchar(50) | 100 |  |  |  |
|  | birth\_date | date | 4 |  |  |  |
| preferences | id  title | integer  varchar(100) | 4  200 | 26 | 24576 | 0.02 |
|  | preference\_category\_id | integer | 4 |  |  |  |
| ingredients\_preferences | id  ingredient\_id | integer  integer | 4  4 | 86 | 24576 | 0.02 |
|  | preference\_id | integer | 4 |  |  |  |
| orders | id user\_id courier\_id payment\_info\_id | integer integer integer integer | 4  4  4  4 | 50000 | 4587520 | 4.38 |
|  | menu\_id | integer | 4 |  |  |  |
|  | created\_at | timestamp | 8 |  |  |  |
|  | status | order\_status | 4 |  |  |  |
| preferences\_users | id  user\_id | integer  integer | 4  4 | 9051 | 655360 | 0.62 |
|  | preference\_id | integer | 4 |  |  |  |

# Приложение. Отчеты

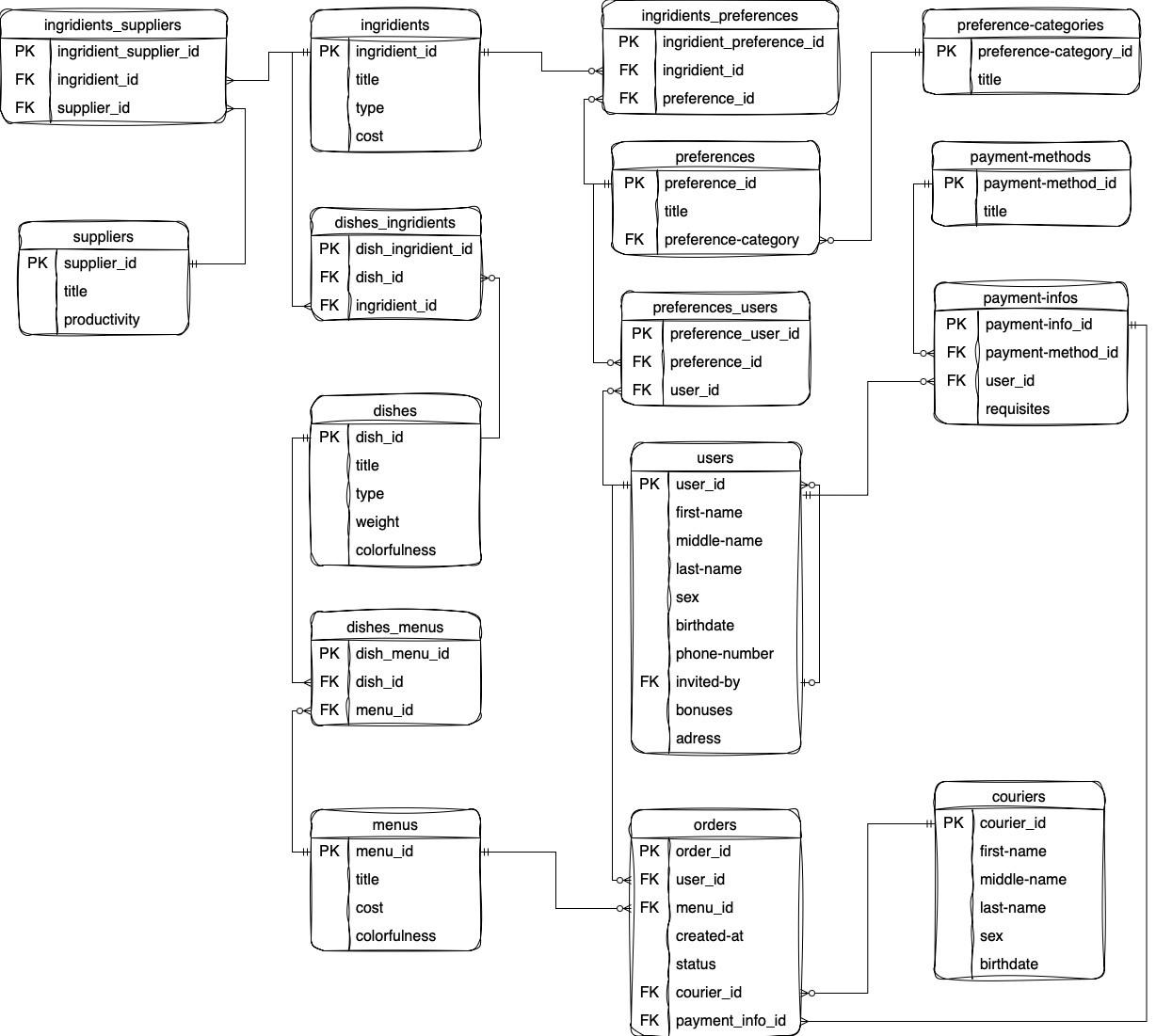
## Приложения



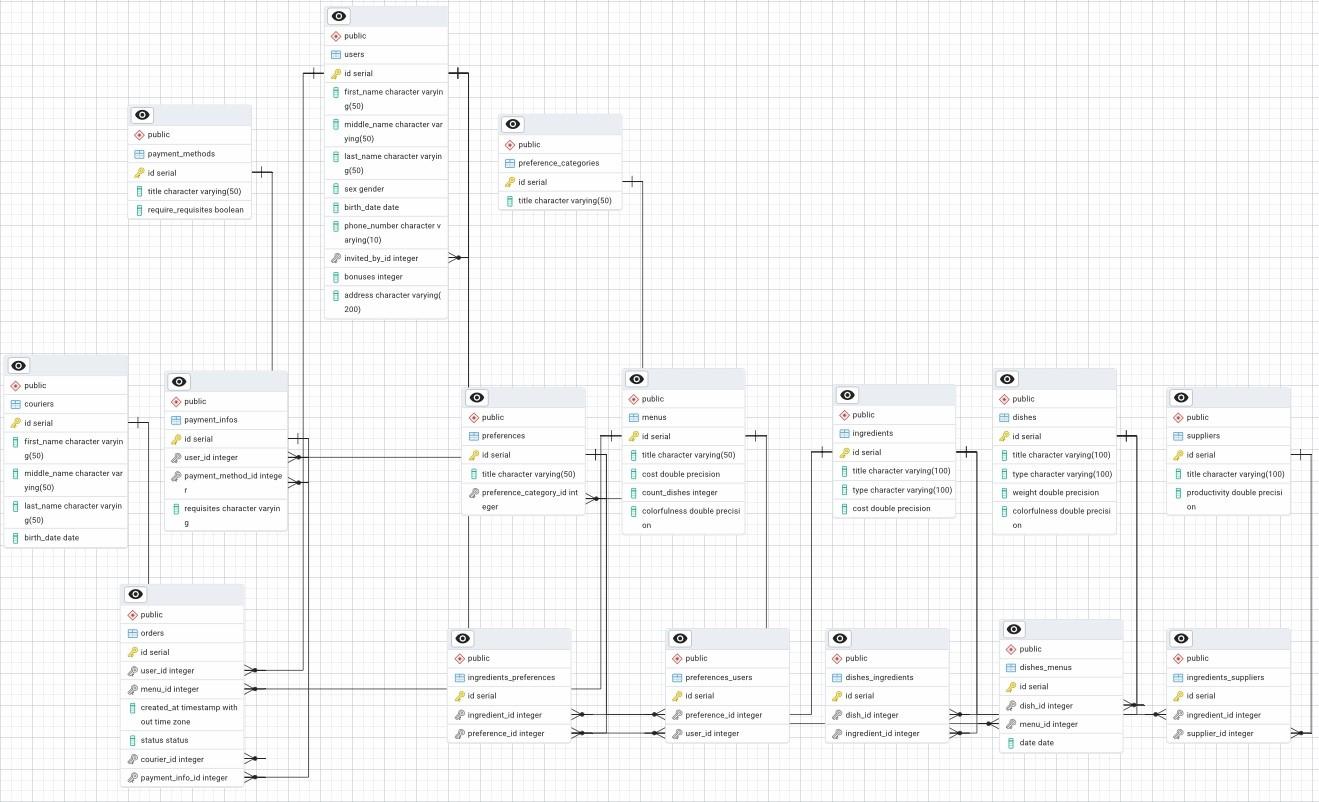
**Рис. 1:** ER-диаграмма концептуальной модели предметной области



**Рис. 2:** ER-диаграмма концептуальной модели базы данных



**Рис. 3:** ER-диаграмма логической модели базы данных



**Рис. 4:** Схемы отношений базы данных

**Листинг 1:** Создание ENUM-типов gender и status

| **CREATE** TYPE ge nd e r **AS** ENUM ( ' male ' , | ' fe m al e ' ) ; |  |  |
| --- | --- | --- | --- |
| **CREATE** TYPE s t a t u s **AS** ENUM ( ' new ' ,  ' , ' r e t u r n e d ' ) ; | ' i n \_ d e l i v e r y ' , | ' d e l i v e r e d ' , | ' c a n c e l e d |

**Листинг 2:** Создание таблицы couriers

| **CREATE TABLE** c o u r i e r s ( i d SERIAL **NOT NULL**, f i r s t \_ n a m e **VARCHAR**( 5 0 ) **NOT NULL**,  middle\_name **VARCHAR**( 5 0 ) , l a s t \_ n a m e **VARCHAR**( 5 0 ) **NOT NULL**,  b i r t h \_ d a t e **DATE NOT NULL**,  **PRIMARY KEY** ( i d )  ) ; |
| --- |

**Листинг 3:** Создание таблицы dishes

| **CREATE TABLE** d i s h e s ( i d SERIAL **NOT NULL**,  t i t l e **VARCHAR**( 1 0 0 ) **NOT NULL**,  t y p e **VARCHAR**( 1 0 0 ) **NOT NULL**, wei gh t **FLOAT NOT NULL**,  c o l o r f u l n e s s **FLOAT NOT NULL**, |
| --- |
| **PRIMARY KEY** ( i d )  ) ; |

**Листинг 4:** Создание таблицы ingredients

| **CREATE TABLE** i n g r e d i e n t s ( i d SERIAL **NOT NULL**,  t i t l e **VARCHAR**( 1 0 0 ) **NOT NULL**,  t y p e **VARCHAR**( 1 0 0 ) **NOT NULL**, c o s t **FLOAT NOT NULL**, **PRIMARY KEY** ( i d )  ) ; |
| --- |

**Листинг 5:** Создание таблицы menus

| **CREATE TABLE** menus ( i d SERIAL **NOT NULL**, t i t l e **VARCHAR**( 5 0 ) **NOT NULL**,  c o s t **FLOAT NOT NULL**, c o u n t \_ d i s h e s **INTEGER NOT NULL**, c o l o r f u l n e s s **FLOAT NOT NULL**,  **PRIMARY KEY** ( i d )  ) ; |
| --- |

**Листинг 6:** Создание таблицы payment\_methods

| **CREATE TABLE** payment\_methods ( i d SERIAL **NOT NULL**,  t i t l e **VARCHAR**( 5 0 ) **NOT NULL**,  r e q u i r e \_ r e q u i s i t e s BOOLEAN **NOT NULL**,  **PRIMARY KEY** ( i d )  ) ; |
| --- |

**Листинг 7:** Создание таблицы preference\_categories

| **CREATE TABLE** p r e f e r e n c e \_ c a t e g o r i e s i d SERIAL **NOT NULL**,  t i t l e **VARCHAR**( 5 0 ) **NOT NULL**,  **PRIMARY KEY** ( i d )  ) ; | ( |
| --- | --- |

**Листинг 8:** Создание таблицы suppliers

| **CREATE TABLE** s u p p l i e r s ( i d SERIAL **NOT NULL**,  t i t l e **VARCHAR**( 1 0 0 ) **NOT NULL**, p r o d u c t i v i t y **FLOAT NOT NULL**,  **PRIMARY KEY** ( i d )  ) ; |
| --- |

24

**Листинг 9:** Создание таблицы users

| **CREATE TABLE** u s e r s ( i d SERIAL **NOT NULL**, f i r s t \_ n a m e **VARCHAR**( 5 0 ) **NOT NULL**,  middle\_name **VARCHAR**( 5 0 ) ,  l a s t \_ n a m e **VARCHAR**( 5 0 ) **NOT NULL**, sex gender ,  b i r t h \_ d a t e **DATE NOT NULL**,  phone\_number **VARCHAR**( 1 0 ) **NOT NULL**, i n v i t e d \_ b y \_ i d **INTEGER**,  bonuses **INTEGER DEFAULT** 0 **NOT NULL**, a d d r e s s **VARCHAR**( 2 0 0 ) ,  **PRIMARY KEY** ( i d ) ,  **FOREIGN KEY**( i n v i t e d \_ b y \_ i d ) REFERENCES u s e r s  ) ; | ( i d ) **ON DELETE SET NULL** |
| --- | --- |

**Листинг 10:** Создание таблицы dishes\_ingredients

| **CREATE TABLE** d i s h e s \_ i n g r e d i e n t s ( i d SERIAL **NOT NULL**, d i s h \_ i d **INTEGER NOT NULL**, i n g r e d i e n t \_ i d **INTEGER NOT NULL**,  **PRIMARY KEY** ( i d ) ,  **FOREIGN KEY**( d i s h \_ i d ) REFERENCES d i s h e s ( i d ) **ON DELETE CASCADE**,  **FOREIGN KEY**( i n g r e d i e n t \_ i d ) REFERENCES i n g r e d i e n t s ( i d ) **ON DELETE CASCADE**  ) ; |
| --- |

**Листинг 11:** Создание таблицы dishes\_menus

| **CREATE TABLE** dishes\_menus ( i d SERIAL **NOT NULL**, d i s h \_ i d **INTEGER NOT NULL**, menu\_id **INTEGER NOT NULL**,  **date DATE DEFAULT** ' now ( ) ' **NOT NULL**,  **PRIMARY KEY** ( i d ) ,  **FOREIGN KEY**( d i s h \_ i d ) REFERENCES d i s h e s ( i d ) **ON DELETE CASCADE**,  **FOREIGN KEY**( menu\_id ) REFERENCES menus ( i d ) **ON DELETE CASCADE**  ) ; |
| --- |

**Листинг 12:** Создание таблицы ingredients\_suppliers

| **CREATE TABLE** i n g r e d i e n t s \_ s u p p l i e r s ( i d SERIAL **NOT NULL**,  i n g r e d i e n t \_ i d **INTEGER NOT NULL**, s u p p l i e r \_ i d **INTEGER NOT NULL**,  **PRIMARY KEY** ( i d ) ,  **FOREIGN KEY**( i n g r e d i e n t \_ i d ) REFERENCES i n g r e d i e n t s ( i d ) **ON DELETE CASCADE**,  **FOREIGN KEY**( s u p p l i e r \_ i d ) REFERENCES s u p p l i e r s ( i d ) **ON DELETE CASCADE** |
| --- |

) ;

**Листинг 13:** Создание таблицы payment\_infos

| **CREATE TABLE** p a y m e n t \_ i n f o s ( i d SERIAL **NOT NULL**, u s e r \_ i d **INTEGER**,  payment\_method\_id **INTEGER**,  r e q u i s i t e s **VARCHAR**,  **PRIMARY KEY** ( i d ) ,  **FOREIGN KEY**( u s e r \_ i d ) REFERENCES u s e r s ( i d ) **ON DELETE CASCADE**,  **FOREIGN KEY**( payment\_method\_id ) REFERENCES payment\_methods ( i d ) **ON**  **DELETE CASCADE**  ) ; |
| --- |

**Листинг 14:** Создание таблицы preferences

| **CREATE TABLE** p r e f e r e n c e s ( i d SERIAL **NOT NULL**,  t i t l e **VARCHAR**( 5 0 ) **NOT NULL**,  p r e f e r e n c e \_ c a t e g o r y \_ i d **INTEGER NOT NULL**,  **PRIMARY KEY** ( i d ) ,  **FOREIGN KEY**( p r e f e r e n c e \_ c a t e g o r y \_ i d ) REFERENCES p r e f e r e n c e \_ c a t e g o r i e s ( i d ) **ON DELETE CASCADE**  ) ; |
| --- |

**Листинг 15:** Создание таблицы ingredients\_preferences

| **CREATE TABLE** i n g r e d i e n t s \_ p r e f e r e n c e s ( i d SERIAL **NOT NULL**,  i n g r e d i e n t \_ i d **INTEGER NOT NULL**, p r e f e r e n c e \_ i d **INTEGER NOT NULL**,  **PRIMARY KEY** ( i d ) ,  **FOREIGN KEY**( i n g r e d i e n t \_ i d ) REFERENCES i n g r e d i e n t s | ( i d ) **ON DELETE** |
| --- | --- |
| **CASCADE**,  **FOREIGN KEY**( p r e f e r e n c e \_ i d ) REFERENCES p r e f e r e n c e s  **CASCADE**  ) ; | ( i d ) **ON DELETE** |

**Листинг 16:** Создание таблицы orders

| **CREATE TABLE** o r d e r s ( i d SERIAL **NOT NULL**, u s e r \_ i d **INTEGER NOT NULL**,  menu\_id **INTEGER**,  c r e a t e d \_ a t **TIMESTAMP** WITHOUT **TIME ZONE DEFAULT** now ( ) **NOT NULL**, s t a t u s s t a t u s **DEFAULT** ' new ' **NOT NULL**, c o u r i e r \_ i d **INTEGER**,  p a y m e n t \_ i n f o \_ i d **INTEGER**,  **PRIMARY KEY** ( i d ) ,  **FOREIGN KEY**( u s e r \_ i d ) REFERENCES u s e r s ( i d ) **ON DELETE CASCADE**, |
| --- |
| **FOREIGN KEY**( menu\_id ) REFERENCES menus ( i d ) **ON DELETE SET NULL**,  **FOREIGN KEY**( c o u r i e r \_ i d ) REFERENCES c o u r i e r s ( i d ) **ON DELETE SET NULL**,  **FOREIGN KEY**( p a y m e n t \_ i n f o \_ i d ) REFERENCES p a y m e n t \_ i n f o s ( i d ) **ON DELETE SET NULL**  ) ; |

**Листинг 17:** Создание таблицы preferences\_users

| **CREATE TABLE** p r e f e r e n c e s \_ u s e r s ( i d SERIAL **NOT NULL**,  p r e f e r e n c e \_ i d **INTEGER NOT NULL**, u s e r \_ i d **INTEGER NOT NULL**,  **PRIMARY KEY** ( i d ) ,  **FOREIGN KEY**( p r e f e r e n c e \_ i d ) REFERENCES p r e f e r e n c e s ( i d ) **ON DELETE CASCADE**,  **FOREIGN KEY**( u s e r \_ i d ) REFERENCES u s e r s ( i d ) **ON DELETE CASCADE**  ) ; |
| --- |

%python

**Листинг 18:** Подключение к базе данных с помощью psycopg2

| **from** f a k e r **import** Faker  **import** psycopg **import** os f a k e = Faker ( " ru\_RU " )  **def** g e t \_ c o n n ( ) : dsn = os . e n v i r o n . g e t ( "DSN" )  **i f not** dsn :  **raise** R u n t i m e E r r o r ( "DSN n o t **return** psycopg . c o n n e c t ( dsn , | s e t autocommit | i n  = | e n v i r o n m e n t  True ) | v a r i a b l e s " ) |
| --- | --- | --- | --- | --- |

**Листинг 19:** Вызов функций заполнения всех таблиц

| **from** s e e d e r s **import** \* **from** common **import** g e t \_ c o n n  **def** s e e d \_ a l l ( ) :  **with** g e t \_ c o n n ( ) **as** conn : seed\_payment\_methods ( conn )  s e e d \_ u s e r s ( conn , 10000) s e e d \_ p a y m e n t \_ i n f o s ( conn )  s e e d \_ c o u r i e r s ( conn , 100)  seed\_menus ( conn )  s e e d \_ o r d e r s ( conn , 50000 , 100) |
| --- |
| s e e d \_ p r e f e r e n c e \_ c a t e g o r i e s ( conn ) s e e d \_ p r e f e r e n c e s ( conn ) s e e d \_ p r e f e r e n c e s \_ u s e r s ( conn )  s e e d \_ s u p p l i e r s ( conn , 20) s e e d \_ i n g r e d i e n t s ( conn ) s e e d \_ i n g r e d i e n t s \_ s u p p l i e r s ( conn ) s e e d \_ d i s h e s ( conn ) s e e d \_ d i s h e s \_ i n g r e d i e n t s ( conn ) s e e d \_ i n g r e d i e n t s \_ p r e f e r e n c e s ( conn ) s e e d \_ d i s h e s \_ m e n u s ( conn ) |

**Листинг 20:** Заполнение таблицы payment\_methods

| **def** | seed\_payment\_methods ( conn ) :  payment\_methods = [  {  " t i t l e " : "Наличные курьеру" ,  " r e q u i r e \_ r e q u i s i t e s " : F a l s e ,  } ,  {  " t i t l e " : "Карта курьеру" ,  " r e q u i r e \_ r e q u i s i t e s " : True ,  } ,  {  " t i t l e " : "Карта" ,  " r e q u i r e \_ r e q u i s i t e s " : True ,  } ,  {  " t i t l e " : "ЯндексСплит . " ,  " r e q u i r e \_ r e q u i s i t e s " : True ,  } ,  ]  **with** conn . c u r s o r ( ) **as** c u r :  **for** method **in** payment\_methods :  c u r . e x e c u t e (  " " "  INSERT INTO payment\_methods | ( t i t l e , | r e q u i r e \_ r e q u i s i t e s ) |
| --- | --- | --- | --- |
|  | VALUES (%s , %s )  " " " ,  ( method [ " t i t l e " ] , method [ " r e q u i r e \_ r e q u i s i t e s " ] ) ,  )  conn . commit ( )  **print** ( " I n s e r t e d payment methods i n t o t h e d a t a b a s e . " ) | | |

**Листинг 21:** Заполнение таблицы users

| **from** common **import**  **import** d a t e t i m e **import** random | f a k e |
| --- | --- |

| **def** | generate\_phone\_number ( ) :  " " "Форматирует номертелефонаввиде 10 цифрбезкодастраны . " " " **return** f a k e . n u m e r i f y ( " 9######### " ) | | |  |
| --- | --- | --- | --- | --- |
| **def** | g e n e r a t e \_ n a m e \_ b y \_ g e n d e r ( ge nd e r ) :  " " "Генерация имени , отчестваифамилиивзависимостиотпола **i f** ge nd e r == " male " :  f i r s t \_ n a m e = f a k e . f i r s t \_ n a m e \_ m a l e ( )  middle\_name = f a k e . middle\_name\_male ( ) l a s t \_ n a m e = f a k e . l a s t \_ n a m e \_ m a le ( )  **else** :  f i r s t \_ n a m e = f a k e . f i r s t \_ n a m e \_ f e m a l e ( )  middle\_name = f a k e . middle\_name\_female ( )  l a s t \_ n a m e = f a k e . l a s t \_ n a m e \_ f e m a l e ( )  **return** f i r s t \_ n a m e , middle\_name , l a s t \_ n a m e | | . " " " |  |
| **def** | g e n e r a t e \_ b i r t h \_ d a t e ( min\_age =18 , max\_age =80) :  " " "Генерация датырождениявдиапазонеот min\_age до | | max\_age летназад | . " " " |
|  | t o d a y = d a t e t i m e . d a t e . t o d a y ( )  s t a r t \_ d a t e = t o d a y . r e p l a c e ( y e a r = t o d a y . y e a r - max\_age )  e n d \_ d a t e = t o d a y . r e p l a c e ( y e a r = t o d a y . y e a r - min\_age ) **return** f a k e . d a t e \_ b e t w e e n ( s t a r t \_ d a t e = s t a r t \_ d a t e , e n d \_ d a t e = e n d \_ d a t e | | | ) |
| **def** | s e e d \_ u s e r s  **with** conn  **for** | ( conn , n =50) :  . c u r s o r ( ) **as** c u r :  \_ **in range** ( n ) : sex = random . c h o i c e ( [ " male " , " fe m al e " ] ) f i r s t \_ n a m e , middle\_name , l a s t \_ n a m e = g e n e r a t e \_ n a m e \_ by\_ g e n de r ( sex )  c u r . e x e c u t e (  " " "  INSERT INTO u s e r s ( f i r s t \_ n a m e , middle\_name , last\_name , sex , b i r t h \_ d a t e , phone\_number , a d d r e s s , bonuses )  VALUES (%s , %s , %s , %s , %s , %s , %s , %s )  " " " ,  (   1. i r s t \_ n a m e ,   middle\_name ,  last\_name , sex ,   1. e n e r a t e \_ b i r t h \_ d a t e ( ) ,   generate\_phone\_number ( ) , f a k e . s t r e e t \_ a d d r e s s ( ) **i f** random . random ( ) < 0 . 3 **else** | | |
|  | None ,  0 ,  ) ,  )  conn . commit ( )  **print** ( f " I n s e r t e d {n} u s e r s i n t o t h e d a t a b a s e . " ) c u r . e x e c u t e ( "SELECT i d FROM u s e r s " ) | | | |
| a l l \_ u s e r s = c u r . f e t c h a l l ( )  **for** u s e r \_ i d **in** a l l \_ u s e r s :  **i f** random . random ( ) < 0 . 3 :  c u r . e x e c u t e (  " " "  UPDATE u s e r s  SET i n v i t e d \_ b y \_ i d = %s  WHERE i d = %s  " " " ,  ( random . c h o i c e ( a l l \_ u s e r s ) [ 0 ] , u s e r \_ i d [ 0 ] ) ,  )  conn . commit ( )  **print** ( " Updated i n v i t e d \_ b y \_ i d f o r some u s e r s . " ) | | | | |

**Листинг 22:** Заполнение таблицы payment\_infos

| **from** common **import** f a k e **import** random  **def** s e e d \_ p a y m e n t \_ i n f o s ( conn ) :  **with** conn . c u r s o r ( ) **as** c u r : c u r . e x e c u t e ( "SELECT i d FROM u s e r s " ) u s e r s = c u r . f e t c h a l l ( )  c u r . e x e c u t e ( "SELECT id , r e q u i r e \_ r e q u i s i t e s FROM payment\_methods "  )  payment\_methods = c u r . f e t c h a l l ( )  p a y m e n t \_ i n f o s = [ ] **for** u s e r **in** u s e r s :  **for** payment\_method **in** payment\_methods :  **i f** random . random ( ) < 0 . 5 :  **continue**  r e q u i s i t e s = f a k e . c r e d i t \_ c a r d \_ n u m b e r ( ) **i f** payment\_method  [ 1 ] **else** None p a y m e n t \_ i n f o s . append (  ( u s e r [ 0 ] , payment\_method [ 0 ] , r e q u i s i t e s )  )  c u r . executemany (  " " "  INSERT INTO p a y m e n t \_ i n f o s ( u s e r \_ i d , payment\_method\_id , r e q u i s i t e s )  VALUES (%s , %s , %s )  " " " , p a y m e n t \_ i n f o s  )  conn . commit ( ) **print** ( f " I n s e r t e d { l e n ( p a y m e n t \_ i n f o s ) } payment i n f o s i n t o t h e d a t a b a s e . " ) |
| --- |

**Листинг 23:** Заполнение таблицы couriers

| **from** common **import** f a k e **import** d a t e t i m e  **def** g e n e r a t e \_ c o u r i e r \_ n a m e ( ) :  f i r s t \_ n a m e = f a k e . f i r s t \_ n a m e \_ m a l e ( )  middle\_name = f a k e . middle\_name\_male ( )  l a s t \_ n a m e = f a k e . l a s t \_ n a m e \_ m a l e ( ) **return** f i r s t \_ n a m e , middle\_name , l a s t \_ n a m e  **def** g e n e r a t e \_ b i r t h \_ d a t e ( min\_age =18 , max\_age =60) :  t o d a y = d a t e t i m e . d a t e . t o d a y ( )  s t a r t \_ d a t e = t o d a y . r e p l a c e ( y e a r = t o d a y . y e a r - max\_age )   1. n d \_ d a t e = t o d a y . r e p l a c e ( y e a r = t o d a y . y e a r - min\_age ) **return** f a k e . d a t e \_ b e t w e e n ( s t a r t \_ d a t e = s t a r t \_ d a t e , e n d \_ d a t e = e n d \_ d a t e )   **def** s e e d \_ c o u r i e r s ( conn , n =50) :  **with** conn . c u r s o r ( ) **as** c u r :  **for** \_ **in range** ( n ) :   1. i r s t \_ n a m e , middle\_name , l a s t \_ n a m e = g e n e r a t e \_ c o u r i e r \_ n a m e ( ) b i r t h \_ d a t e = g e n e r a t e \_ b i r t h \_ d a t e ( )   c u r . e x e c u t e (  " " "  INSERT INTO c o u r i e r s ( f i r s t \_ n a m e , middle\_name , last\_name , b i r t h \_ d a t e )  VALUES (%s , %s , %s , %s )  " " " ,  ( f i r s t \_ n a m e , middle\_name , last\_name , b i r t h \_ d a t e ) ,  )  conn . commit ( )  **print** ( f " I n s e r t e d {n} c o u r i e r s i n t o t h e d a t a b a s e . " ) |
| --- |

**Листинг 24:** Заполнение таблицы menus

| **def** | seed\_menus ( conn ) : menus = [ | | | | |
| --- | --- | --- | --- | --- | --- |
|  | { | " t i t l e " : "Похудение" ,  " c o s t " : 700 ,  " c o u n t \_ d i s h e s " : 3 ,  " c o l o r f u l n e s s " : 1000 , | | | |
|  | } ,  { | " t i t l e " : "Баланс" ,  " c o s t " : 850 ,  " c o u n t \_ d i s h e s " : 4 ,  " c o l o r f u l n e s s " : 1500 , | | | |
|  | } ,  { | " t i t l e " : "Набор" ,  " c o s t " : 1000 , | | | |
| " c o u n t \_ d i s h e s " : 5 ,  " c o l o r f u l n e s s " : 2000 ,  } ,  ]  **with** conn . c u r s o r ( ) **as** c u r :  **for** menu **in** menus :  c u r . e x e c u t e (  " " "  INSERT INTO menus ( t i t l e , | | | c o s t , | c o u n t \_ d i s h e s , |  | |
| c o l o r f u l n e s s )  VALUES (%s , %s , %s , %s )  " " " ,  ( menu [ " t i t l e " ] , menu [ " c o s t " ] , menu [ " c o u n t \_ d i s h e s " ] , [ " c o l o r f u l n e s s " ] ) ,  )  conn . commit ( )  **print** ( f " I n s e r t e d { l e n ( menus ) } menus i n t o t h e d a t a b a s e . " ) | | | | | menu | |

**Листинг 25:** Заполнение таблицы orders

| **from** common **import** f a k e **import** random **import** d a t e t i m e  **def** g e n e r a t e \_ o r d e r \_ c r e a t e d \_ t i m e ( d u r a t i o n ) :  now = d a t e t i m e . d a t e t i m e . now ( )  s t a r t \_ t i m e = now - d a t e t i m e . t i m e d e l t a ( days = d u r a t i o n )  **return** f a k e . d a t e \_ t i m e \_ b e t w e e n ( s t a r t \_ d a t e = s t a r t \_ t i m e , e n d \_ d a t e =now )  **def** g e n e r a t e \_ o r d e r \_ s t a t u s ( c r e a t e d \_ t i m e ) :  now = d a t e t i m e . d a t e t i m e . now ( )  **i f** now - c r e a t e d \_ t i m e <= d a t e t i m e . t i m e d e l t a ( days =1) :  s t a t u s \_ w e i g h t s = {  ' new ' : 0 . 5 ,  ' i n \_ d e l i v e r y ' : 0 . 5 ,  }  s t a t u s = random . c h o i c e s ( **l i s t** ( s t a t u s \_ w e i g h t s . keys ( ) ) , **l i s t** ( s t a t u s \_ w e i g h t s . v a l u e s ( ) ) , k=1  ) [ 0 ]  **e l i f** d a t e t i m e . t i m e d e l t a ( days =1) < now - c r e a t e d \_ t i m e <= d a t e t i m e .  t i m e d e l t a ( days =3) :  s t a t u s \_ w e i g h t s = { ' i n \_ d e l i v e r y ' : 0 . 1 , ' d e l i v e r e d ' : 0 . 9 } s t a t u s = random . c h o i c e s ( **l i s t** ( s t a t u s \_ w e i g h t s . keys ( ) ) , **l i s t** ( s t a t u s \_ w e i g h t s . v a l u e s ( ) ) , k=1  ) [ 0 ] **else** :  s t a t u s = ' d e l i v e r e d ' **i f** s t a t u s == ' d e l i v e r e d ' **and** random . random ( ) < 0 . 0 5 : |
| --- |

|  | s t a t u s = ' r e t u r n e d '  **i f** random . random ( ) < 0 . 0 5 :  s t a t u s = ' c a n c e l e d ' **return** s t a t u s | | | |  | |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **def** | s e e d \_ o r d e r s ( conn , n =100 , d  **with** conn . c u r s o r ( ) **as** c u r : c u r . e x e c u t e ( "SELECT i d | u r a t i o  FROM | n =30) : u s e r s " ) | |  | |  |  |
|  | u s e r s = c u r . f e t c h a l l ( ) c u r . e x e c u t e ( "SELECT i d | FROM | menus " ) | |  | |  |  |
|  | menus = c u r . f e t c h a l l ( ) c u r . e x e c u t e ( "SELECT i d | FROM | c o u r i e r s " ) | |  | |  |  |
|  | c o u r i e r s = c u r . f e t c h a l l ( ) c u r . e x e c u t e ( "SELECT i d FROM | | p a y m e n t \_ i n f o s WHERE | | u s e r \_ i d | | IN | ( |
|  | SELECT i d FROM u s e r s ) " ) p a y m e n t \_ i n f o s = c u r . f e t c h a l l ( )   * r d e r s = [ ] **for** \_ **in range** ( n ) :   u s e r = random . c h o i c e ( u s e r s ) menu = random . c h o i c e ( menus )  c o u r i e r = random . c h o i c e ( c o u r i e r s )  p a y m e n t \_ i n f o = random . c h o i c e ( p a y m e n t \_ i n f o s )  c r e a t e d \_ a t = g e n e r a t e \_ o r d e r \_ c r e a t e d \_ t i m e ( d u r a t i o n )   * r d e r = (   u s e r [ 0 ] , menu [ 0 ] ,  c r e a t e d \_ a t ,  g e n e r a t e \_ o r d e r \_ s t a t u s ( c r e a t e d \_ a t ) , c o u r i e r [ 0 ] ,  p a y m e n t \_ i n f o [ 0 ] ,  )   * r d e r s . append ( o r d e r )   c u r . executemany (  " " "  INSERT INTO o r d e r s ( u s e r \_ i d , menu\_id , c r e a t e d \_ a t , | | | | | | s t a t u s | , |
|  | c o u r i e r \_ i d , p a y m e n t \_ i n f o \_ i d ) VALUES (%s , %s , %s , %s , %s , %s )  " " " , o r d e r s ,  )  conn . commit ( )  **print** ( f " I n s e r t e d { l e n ( o r d e r s ) } o r d e r s i n t o  # Update u s e r bonuses **for** o r d e r **in** o r d e r s : | | | t h e d a t a b a s e | | . " ) | | |
| u s e r \_ i d = o r d e r [ 0 ] menu\_id = o r d e r [ 1 ]  c u r . e x e c u t e (  " " "  UPDATE u s e r s  SET bonuses = bonuses + (SELECT c o s t FROM menus WHERE i d = %s ) \* 0 . 0 5  WHERE i d = %s  " " " ,  ( menu\_id , u s e r \_ i d ) ,  )  conn . commit ( )  **print** ( " Updated u s e r bonuses based on o r d e r s . " ) | | | | | | | | | |

**Листинг 26:** Заполнение таблицы preference\_categories

| **def** | s e e d \_ p r e f e r e n c e \_ c a t e g o r i e s ( conn ) :  c a t e g o r i e s = [  "аллергены" ,  "десерты , выпечка , сахар" ,  "мясо , рыба" ,  "овощи , лук , чеснок" ,  "гарниры , каши"  ]  **with** conn . c u r s o r ( ) **as** c u r :  **for** c a t e g o r y **in** c a t e g o r i e s :  c u r . e x e c u t e (  " " "  INSERT INTO p r e f e r e n c e \_ c a t e g o r i e s | ( t i t l e ) |  |
| --- | --- | --- | --- |
|  | VALUES (%s )  " " " ,  ( c a t e g o r y , ) ,  )  conn . commit ( )  **print** ( f " I n s e r t e d { l e n ( c a t e g o r i e s ) } p r e f e r e n c e t h e d a t a b a s e . " ) | c a t e g o r i e s | i n t o |

**Листинг 27:** Заполнение таблицы preferences

| **def** | s e e d \_ p r e f e r e n c e s ( c p r e f e r e n c e s = [ | | onn ) : |  |  |
| --- | --- | --- | --- | --- | --- |
|  | { " t i t l e " : | "Без | творога" , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 1} , |  |  |
|  | { " t i t l e " : | "Без | орехов" , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 1} , |  |  |
|  | { " t i t l e " : | "Без | меда" , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 1} , |  |  |
|  | { " t i t l e " : | "Без | морепродуктов" , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : | 1} , |  |
|  | { " t i t l e " : | "Без | горчицы" , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 1} , |  |  |
|  | { " t i t l e " : | "Без | шоколада" , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 1} , |  |  |
|  | { " t i t l e " : | "Без | десертов" , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 2} , |  |  |
|  | { " t i t l e " : | "Без | выпечки" , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 2} , |  |  |
|  | { " t i t l e " : | "Без | сэндвичейикруассанов " , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : | | 2} , |
|  | { " t i t l e " : | "Без | белогосахара " , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 2} , | |  |
| { " t i t l e " : "Без свининыиветчины " , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 3} ,  { " t i t l e " : "Без красногомяса " , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 3} ,  { " t i t l e " : "Без рыбы" , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 3} ,  { " t i t l e " : "Без мясаиптицы " , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 3} ,  { " t i t l e " : "Без сельдерея" , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 4} ,  { " t i t l e " : "Без грибов" , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 4} ,  { " t i t l e " : "Без стручковойфасоли " , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 4} ,  { " t i t l e " : "Без брокколи" , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 4} ,  { " t i t l e " : "Без кабачков" , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 4} ,  { " t i t l e " : "Без лука" , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 4} ,  { " t i t l e " : "Без чеснока" , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 4} ,  { " t i t l e " : "Без нута" , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 5} ,  { " t i t l e " : "Без булгура" , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 5} , { " t i t l e " : "Без кускуса" , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 5} ,  { " t i t l e " : "Без гречки" , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 5} ,  { " t i t l e " : "Без молочныхкаш " , " p r e f e r e n c e \_ c a t e g o r y \_ i d " : 5}  ]  **with** conn . c u r s o r ( ) **as** c u r :  **for** p r e f e r e n c e **in** p r e f e r e n c e s :  c u r . e x e c u t e (  " " "  INSERT INTO p r e f e r e n c e s ( t i t l e , p r e f e r e n c e \_ c a t e g o r y \_ i d )  VALUES (%s , %s )  " " " ,  ( p r e f e r e n c e [ " t i t l e " ] , p r e f e r e n c e [ " p r e f e r e n c e \_ c a t e g o r y \_ i d " ] ) ,  )  conn . commit ( ) **print** ( f " I n s e r t e d { l e n ( p r e f e r e n c e s ) } p r e f e r e n c e s i n t o t h e d a t a b a s e . " ) | | | | | |

**Листинг 28:** Заполнение таблицы preferences\_users

| **import def** | random  s e e d \_ p r e f e r e n c e s \_ u s e r s ( conn ) :  **with** conn . c u r s o r ( ) **as** c u r : c u r . e x e c u t e ( "SELECT i d FROM u s e r s " ) u s e r s = c u r . f e t c h a l l ( )  c u r . e x e c u t e ( "SELECT i d FROM p r e f e r e n c e s " ) p r e f e r e n c e s = c u r . f e t c h a l l ( )  u s e r s = [ u s e r [ 0 ] **for** u s e r **in** u s e r s ] p r e f e r e n c e s = [ p r e f [ 0 ] **for** p r e f **in** p r e f e r e n c e s ] **for** u s e r **in** random . sample ( u s e r s , **int** ( **len** ( u s e r s ) | | | \* 0 . 3 ) ) : |
| --- | --- | --- | --- | --- |
|  | s e l e c t e d \_ p r e f s = random . sample ( p r e f e r e n c e s ,  ( 1 , 5) ) | | | random . r a n d i n t |
| **for** | | p r e f \_ i d **in** s e l e c t e d \_ p r e f s :  c u r . e x e c u t e (  " " "  INSERT INTO p r e f e r e n c e s \_ u s e r s p r e f e r e n c e \_ i d )  VALUES (%s , %s )  " " " ,  ( user , p r e f \_ i d ) ,  ) | ( u s e r \_ i d , | | |
| conn . commit ( )  **print** ( " I n s e r t e d p r e f e r e n c e s \_ u s e r s i n t o t h e | | | d a t a b a s e . " ) | | |

**Листинг 29:** Заполнение таблицы suppliers

| **from** common **import** f a k e **import** random  **def** s e e d \_ s u p p l i e r s ( conn , n =10) :  **with** conn . c u r s o r ( ) **as** c u r :  **for** \_ **in range** ( n ) :  t i t l e = f a k e . company ( )  p r o d u c t i v i t y = random . r a n d i n t ( 1 , 200)  c u r . e x e c u t e (  " " "  INSERT INTO s u p p l i e r s ( t i t l e , p r o d u c t i v i t y )  VALUES (%s , %s )  " " " ,  ( t i t l e , p r o d u c t i v i t y ) ,  )  conn . commit ( )  **print** ( f " I n s e r t e d {n} s u p p l i e r s i n t o t h e d a t a b a s e . " ) |
| --- |

**Листинг 30:** Заполнение таблицы ingredients

| **import import def** | csv os  s e e d \_ i n g r e d i e n t s ( conn ) :  f i l e \_ p a t h = os . p a t h . j o i n ( os . p a t h . dirname ( \_ \_ f i l e \_ \_ ) , i n g r e d i e n t s . csv " )  **with open** ( f i l e \_ p a t h , " r " ) **as f i l e** :  r e a d e r = csv . D i c t R e a d e r ( **f i l e** ) i n g r e d i e n t s = [ ]  **for** row **in** r e a d e r :  i n g r e d i e n t = {  " t i t l e " : row [ " t i t l e " ] ,  " t y p e " : row [ " t y p e " ] ,  " c o s t " : **float** ( row [ " c o s t " ] ) ,  } i n g r e d i e n t s . append ( i n g r e d i e n t ) | " csv " , | " |
| --- | --- | --- | --- |
| **with** conn . c u r s o r ( ) **as** c u r :  **for** i n g r e d i e n t **in** i n g r e d i e n t s :  c u r . e x e c u t e (  " " "  INSERT INTO i n g r e d i e n t s ( t i t l e , type , c o s t )  VALUES (%s , %s , %s )  " " " ,  ( i n g r e d i e n t [ " t i t l e " ] , i n g r e d i e n t [ " t y p e " ] , i n g r e d i e n t [ " c o s t " ] ) ,  )  conn . commit ( )  **print** ( f " I n s e r t e d { l e n ( i n g r e d i e n t s ) } i n g r e d i e n t s i n t o t h e d a t a b a s e . " ) | | | | |

**Листинг 31:** Заполнение таблицы ingredients\_suppliers

| **import** random  **def** s e e d \_ i n g r e d i e n t s \_ s u p p l i e r s ( conn ) :  **with** conn . c u r s o r ( ) **as** c u r :  c u r . e x e c u t e ( "SELECT i d FROM i n g r e d i e n t s " )  i n g r e d i e n t s = c u r . f e t c h a l l ( )  c u r . e x e c u t e ( "SELECT i d FROM s u p p l i e r s " )  s u p p l i e r s = c u r . f e t c h a l l ( )  i n g r e d i e n t s \_ s u p p l i e r s = [ ]  **for** i n g r e d i e n t **in** i n g r e d i e n t s :  r a n d o m \_ s u p p l i e r s = random . sample ( s u p p l i e r s , k=random . r a n d i n t ( 1 , 5) )  **for** s u p p l i e r **in** r a n d o m \_ s u p p l i e r s :  i n g r e d i e n t s \_ s u p p l i e r s . append (  ( i n g r e d i e n t [ 0 ] , s u p p l i e r [ 0 ] )  )  c u r . executemany (  " " "  INSERT INTO i n g r e d i e n t s \_ s u p p l i e r s ( i n g r e d i e n t \_ i d , s u p p l i e r \_ i d ) VALUES (%s , %s )  " " " , i n g r e d i e n t s \_ s u p p l i e r s  )  conn . commit ( )  **print** ( f " I n s e r t e d { l e n ( i n g r e d i e n t s \_ s u p p l i e r s ) } i n g r e d i e n t s s u p p l i e r s i n t o t h e d a t a b a s e . " ) |
| --- |

**Листинг 32:** Заполнение таблицы dishes

| **import** | | csv | |
| --- | --- | --- | --- |
| **import** | | os | |
| **def** | s e e d \_ d i s h e s ( conn ) :  f i l e \_ p a t h = os . p a t h . j o i n ( os . p a t h . dirname ( \_ \_ f i l e \_ \_ ) , " csv " , | | " d i s h e s . |
|  | csv " )  **with open** ( f i l e \_ p a t h , " r " ) **as f i l e** :  r e a d e r = csv . D i c t R e a d e r ( **f i l e** ) d i s h e s = [ ]  **for** row **in** r e a d e r :  d i s h = {  " t i t l e " : row [ " t i t l e " ] ,  " t y p e " : row [ " t y p e " ] ,  " wei gh t " : **float** ( row [ " wei gh t " ] ) ,  " c o l o r f u l n e s s " : **float** ( row [ " c o l o r f u l n e s s " ] ) ,  } d i s h e s . append ( d i s h ) **with** conn . c u r s o r ( ) **as** c u r :  **for** d i s h **in** d i s h e s :  c u r . e x e c u t e (  " " "  INSERT INTO d i s h e s ( t i t l e , type , weight , c o l o r f u l n e s s  VALUES (%s , %s , %s , %s )  " " " ,  ( d i s h [ " t i t l e " ] , d i s h [ " t y p e " ] , d i s h [ " wei gh t " ] , c o l o r f u l n e s s " ] ) ,  )  conn . commit ( )  **print** ( f " I n s e r t e d { l e n ( d i s h e s ) } d i s h e s i n t o t h e d a t a b a s e . " ) | | )  d i s h [ " |

**Листинг 33:** Заполнение таблицы dishes\_ingredients

| **import** os **import** csv  **def** s e e d \_ d i s h e s \_ i n g r e d i e n t s ( conn ) :  f i l e \_ p a t h = os . p a t h . j o i n ( os . p a t h . dirname ( \_ \_ f i l e \_ \_ ) , " csv " , " d i s h e s \_ i n g r e d i e n t s . csv " )  **with open** ( f i l e \_ p a t h , " r " ) **as f i l e** :  r e a d e r = csv . D i c t R e a d e r ( **f i l e** ) v a l u e s = [ ]  **for** row **in** r e a d e r :  v a l u e = {  " d i s h \_ i d " : row [ " d i s h \_ i d " ] ,  " i n g r e d i e n t \_ i d " : row [ " i n g r e d i e n t \_ i d " ] ,  } v a l u e s . append ( v a l u e )  **with** conn . c u r s o r ( ) **as** c u r :  c u r . executemany (  " " "  INSERT INTO d i s h e s \_ i n g r e d i e n t s ( d i s h \_ i d , i n g r e d i e n t \_ i d )  VALUES (%s , %s ) | |
| --- | --- |
| " " " ,  [ ( v a l u e [ " d i s h \_ i d " ] , v a l u e [ " i n g r e d i e n t \_ i d " ] ) **for** | v a l u e **in** |
| v a l u e s ]  )  conn . commit ( )  **print** ( f " I n s e r t e d { l e n ( v a l u e s ) } d i s h e s \_ i n g r e d i e n t s i n t o  " ) | t h e d a t a b a s e . |

**Листинг 34:** Заполнение таблицы ingredients\_preferences

| **import** csv **import** os  **def** s e e d \_ i n g r e d i e n t s \_ p r e f e r e n c e s ( conn ) :  f i l e \_ p a t h = os . p a t h . j o i n ( os . p a t h . dirname ( \_ \_ f i l e \_ \_ ) , " csv " , " i n g r e d i e n t s \_ p r e f e r e n c e s . csv " )  **with open** ( f i l e \_ p a t h , " r " ) **as f i l e** :  r e a d e r = csv . D i c t R e a d e r ( **f i l e** ) v a l u e s = [ ]  **for** row **in** r e a d e r :  v a l u e = {  " i n g r e d i e n t \_ i d " : row [ " i n g r e d i e n t \_ i d " ] ,  " p r e f e r e n c e \_ i d " : row [ " p r e f e r e n c e \_ i d " ]  } v a l u e s . append ( v a l u e )  **with** conn . c u r s o r ( ) **as** c u r :  **for** v a l u e **in** v a l u e s :  c u r . e x e c u t e (  " " "  INSERT INTO i n g r e d i e n t s \_ p r e f e r e n c e s ( i n g r e d i e n t \_ i d , p r e f e r e n c e \_ i d )  VALUES (%s , %s )  " " " ,  ( v a l u e [ " i n g r e d i e n t \_ i d " ] , v a l u e [ " p r e f e r e n c e \_ i d " ] ) ,  )  conn . commit ( ) **print** ( f " I n s e r t e d { l e n ( v a l u e s ) } i n g r e d i e n t s p r e f e r e n c e s i n t o t h e d a t a b a s e . " ) |
| --- |

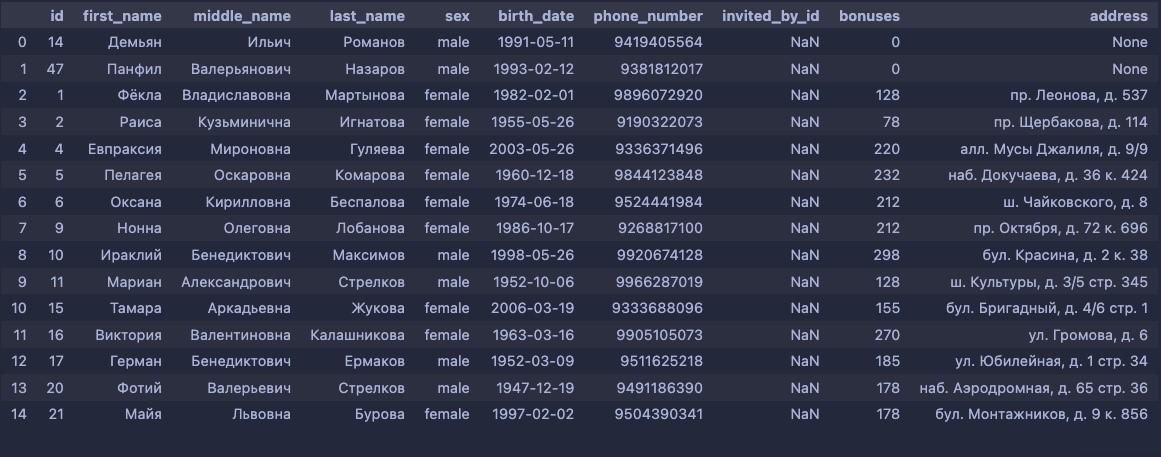
**Листинг 35:** Заполнение таблицы dishes\_menus

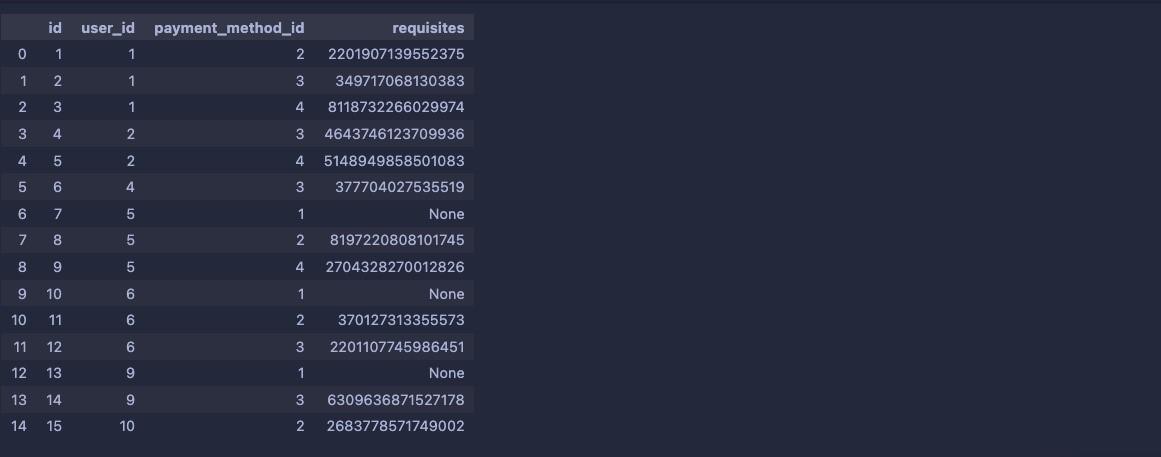
| **import** d a t e t i m e **import** random  **def** s e e d \_ d i s h e s \_ m e n u s ( conn ) :  t o d a y = d a t e t i m e . d a t e . t o d a y ( ) d a t e \_ r a n g e = [ t o d a y + d a t e t i m e . t i m e d e l t a ( days = i )  **with** conn . c u r s o r ( ) **as** c u r : # Получаемвсеменю | **for** | i | **in** | **range** ( 3 0 ) ] |
| --- | --- | --- | --- | --- |

| 1. u r . e x e c u t e ( "SELECT id , c o u n t \_ d i s h e s , c o l o r f u l n e s s FROM menus " )   menus = c u r . f e t c h a l l ( ) # списоккортежей : ( id , c o u n t \_ d i s h e s , c o l o r f u l n e s s )  **for** d a t e **in** d a t e \_ r a n g e :  **for** menu\_id , c o u n t \_ d i s h e s , max\_color **in** menus :  s e l e c t e d \_ d i s h e s = [ ] c o l o r f u l n e s s \_ a c c u m u l a t e d = 0  **while len** ( s e l e c t e d \_ d i s h e s ) < c o u n t \_ d i s h e s : # Выбираемблюда , которыевписываютсявограничения c u r . e x e c u t e (  " " "  SELECT id , c o l o r f u l n e s s FROM d i s h e s  WHERE c o l o r f u l n e s s <= %s  " " " ,  ( max\_color - c o l o r f u l n e s s \_ a c c u m u l a t e d , )  )   1. i s h e s = c u r . f e t c h a l l ( ) # [ ( id , c o l o r f u l n e s s ) , . . . ]   **i f not** d i s h e s :  **break** # нетподходящихблюд—выходим d i s h = random . c h o i c e ( d i s h e s )  **i f** c o l o r f u l n e s s \_ a c c u m u l a t e d + d i s h [ 1 ] <= max\_color :  s e l e c t e d \_ d i s h e s . append ( d i s h ) c o l o r f u l n e s s \_ a c c u m u l a t e d += d i s h [ 1 ]  # Добавляемвыбранныеблюдавтаблицу **for** d i s h \_ i d , \_ **in** s e l e c t e d \_ d i s h e s :  c u r . e x e c u t e (  " " "  INSERT INTO dishes\_menus ( d i s h \_ i d , menu\_id , d a t e  )  VALUES (%s , %s , %s )  " " " ,  ( d i s h \_ i d , menu\_id , d a t e )  )  conn . commit ( ) |
| --- |

**Рис. 5:** Заполненная таблица “users”

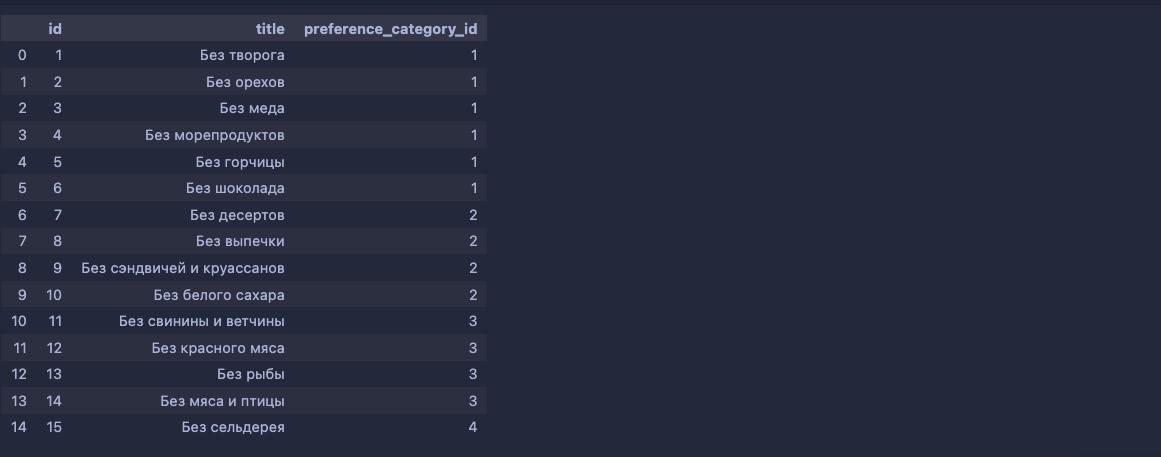


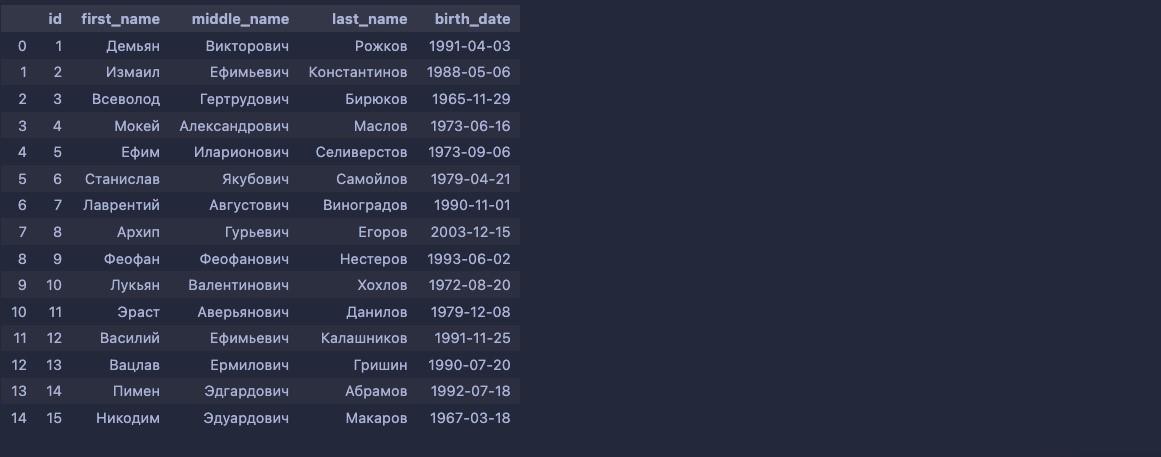
**Рис. 6:** Заполненная таблица “payment\_methods”



**Рис. 7:** Заполненная таблица “payment\_infos”



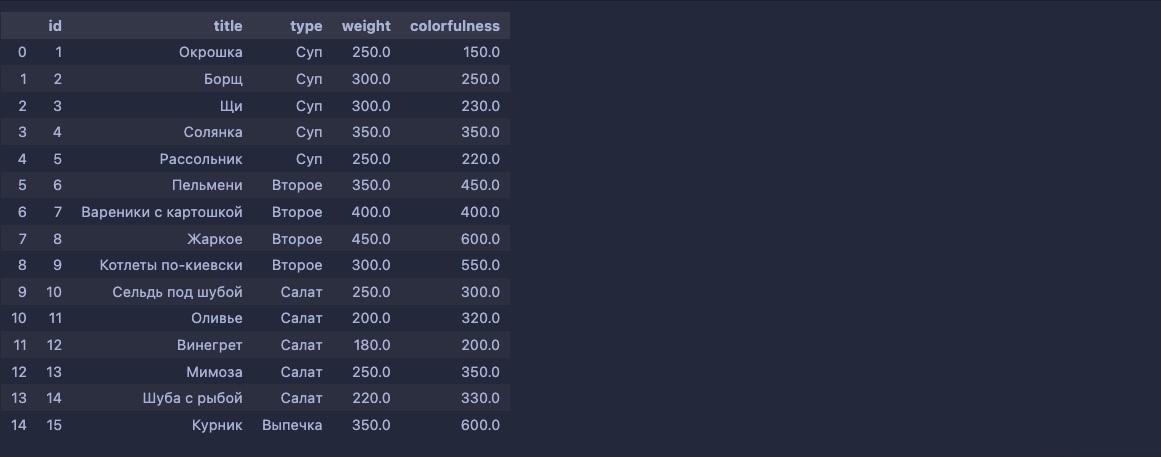
**Рис. 8:** Заполненная таблица “preference\_categories” **Рис. 9:** Заполненная таблица “preferences”

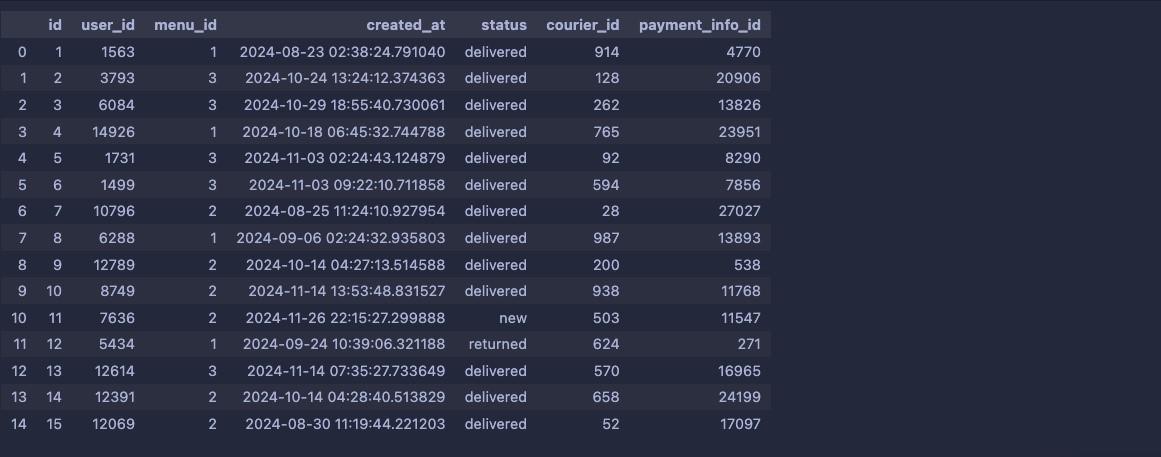


**Рис. 10:** Заполненная таблица “couriers”



**Рис. 11:** Заполненная таблица “menus” **Рис. 12:** Заполненная таблица “orders”

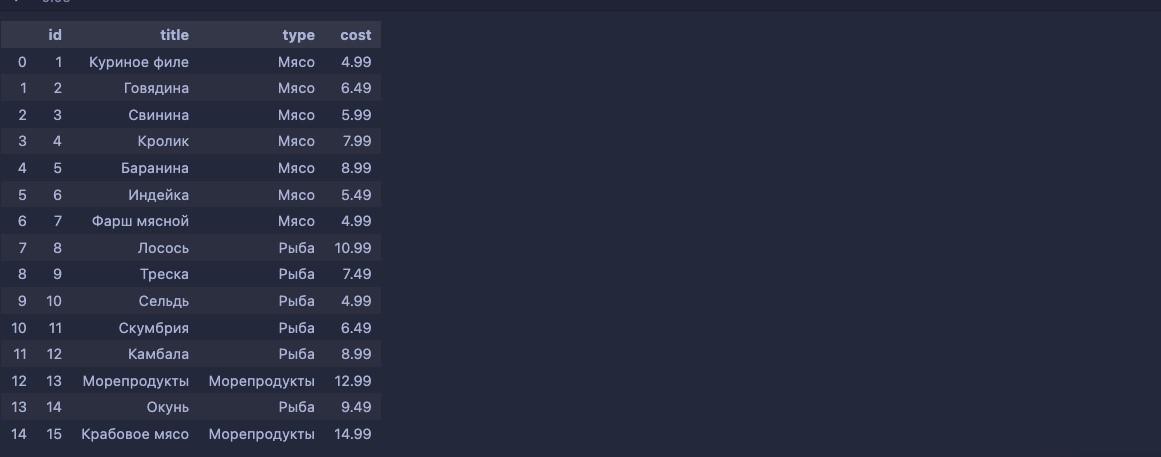


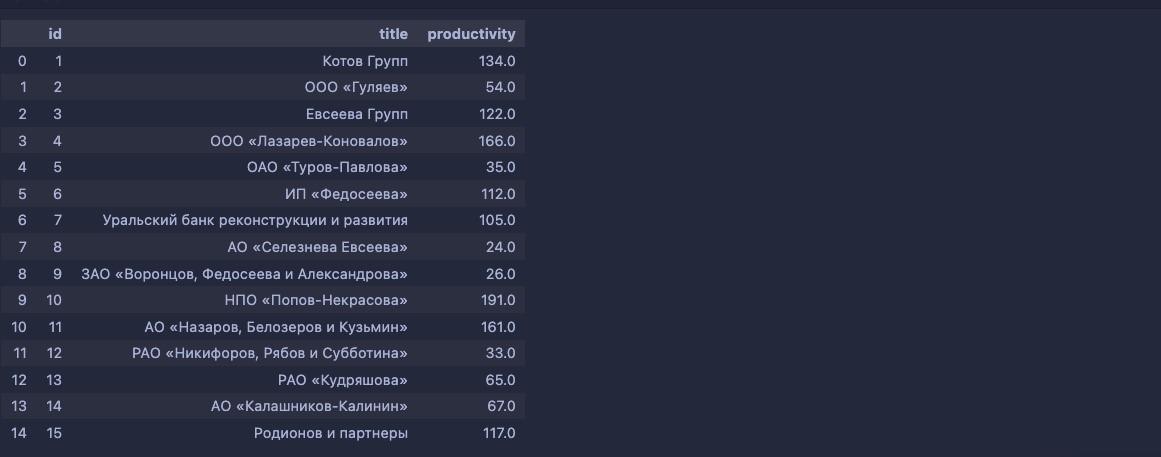
**Рис. 13:** Заполненная таблица “dishes”



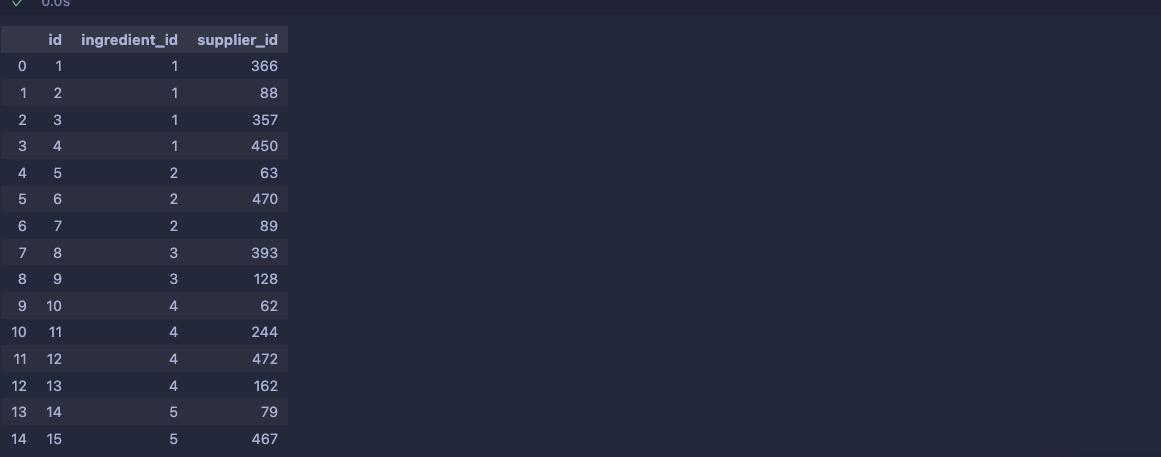
**Рис. 14:** Заполненная таблица “dishes\_menus” **Рис. 15:** Заполненная таблица “ingredients”



**Рис. 16:** Заполненная таблица “dishes\_ingredients”



**Рис. 17:** Заполненная таблица “suppliers”

**Рис. 18:** Заполненная таблица “ingredients\_suppliers”



**Рис. 19:** Заполненная таблица “ingredients\_preferences”

**Листинг 36:** Запрос 1 на SQL

| WITH o r d e r s \_ l a s t \_ m o n t h **AS** (  **SELECT**  \*  **FROM** o r d e r s  **WHERE** c r e a t e d \_ a t >= **CURRENT\_DATE** - **INTERVAL**  )  **SELECT**  **SUM**(m. c o s t ) **AS** Выручка  **FROM** o r d e r s \_ l a s t \_ m o n t h **AS** o  **INNER JOIN** menus **AS** m **on** o . menu\_id = m. i d | ' 1 month ' |
| --- | --- |



**Рис. 20:** Результат выполнения запроса 1

**Листинг 37:** Запрос 2 на SQL

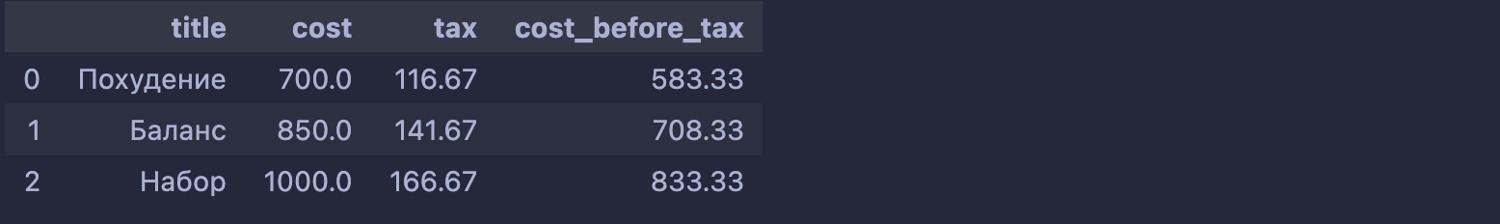
| **SELECT**  ROUND(  **SUM**(m. c o s t ) FILTER (**WHERE** m. t i t l e = ' Похудение ' ) : : **DECIMAL** /  **SUM**(m. c o s t ) : : **DECIMAL** \* 100 , 2) **AS** l o s i n g \_ w e i g h t \_ r e v e n u e \_ r a t i o  **FROM** o r d e r s **AS** o  **INNER JOIN** menus **AS** m **on** o . menu\_id = m. i d |
| --- |



**Рис. 21:** Результат выполнения запроса 2

**Листинг 38:** Запрос 3 на SQL

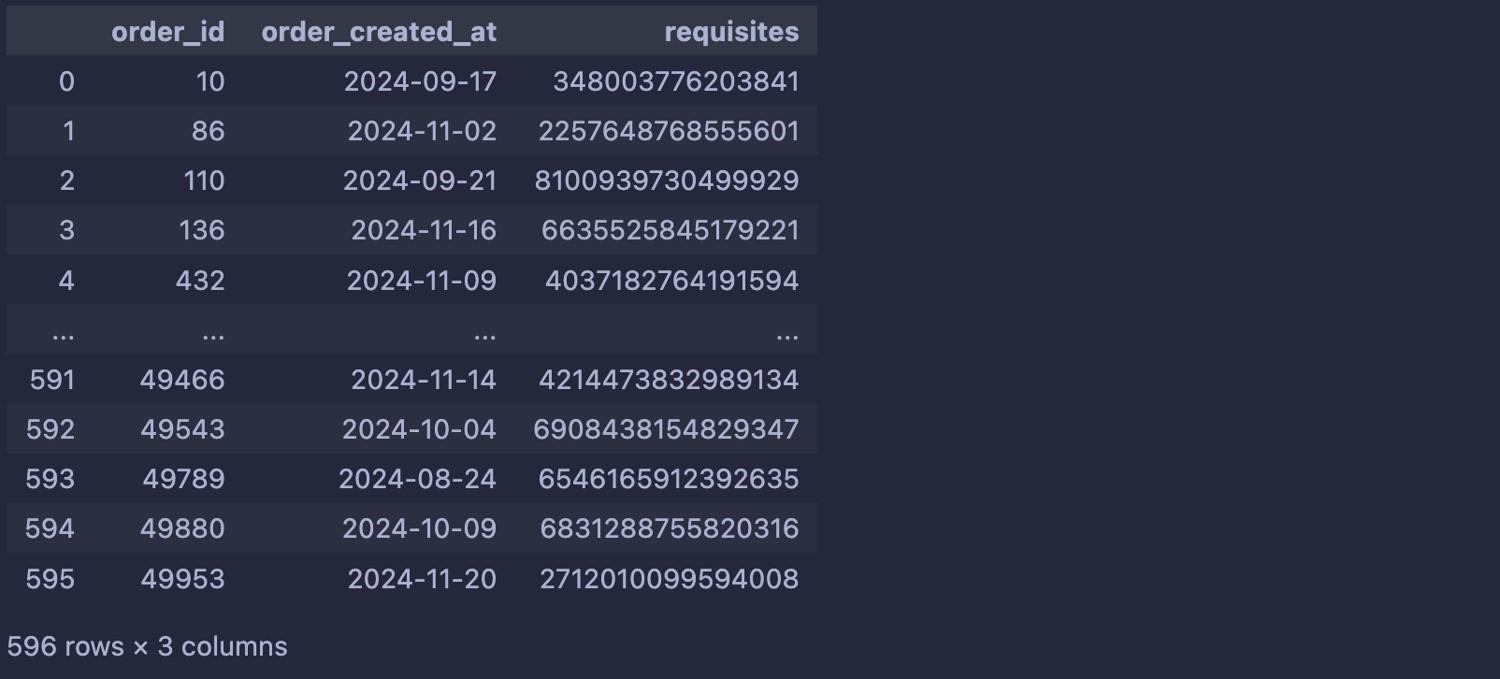
| **SELECT** t i t l e ,  c o s t ,  ROUND( c o s t : : **DECIMAL** / | 1 . 2 | \* 0 . 2 , 2) **AS** tax , |
| --- | --- | --- |
| ROUND( c o s t : : **DECIMAL** /  **FROM** menus | 1 . 2 , | 2) **AS** c o s t \_ b e f o r e \_ t a x |



**Рис. 22:** Результат выполнения запроса 3

**Листинг 39:** Запрос 4 на SQL

| WITH y a n d e x \_ s p l i t \_ p a y m e n t s **AS** (  **SELECT**  i . \*  **FROM** |
| --- |
| p a y m e n t \_ i n f o s i  **JOIN** payment\_methods m **ON** i . payment\_method\_id = m. i d  **WHERE**  m. t i t l e = ' ЯндексСплит . '  )  **SELECT**   * . i d **AS** o r d e r \_ i d ,   **DATE**( o . c r e a t e d \_ a t ) **AS** o r d e r \_ c r e a t e d \_ a t , p . r e q u i s i t e s **AS** r e q u i s i t e s  **FROM** o r d e r s o  **JOIN** y a n d e x \_ s p l i t \_ p a y m e n t s p **ON** o . p a y m e n t \_ i n f o \_ i d = p . i d  **WHERE**   * . s t a t u s = ' r e t u r n e d ' |



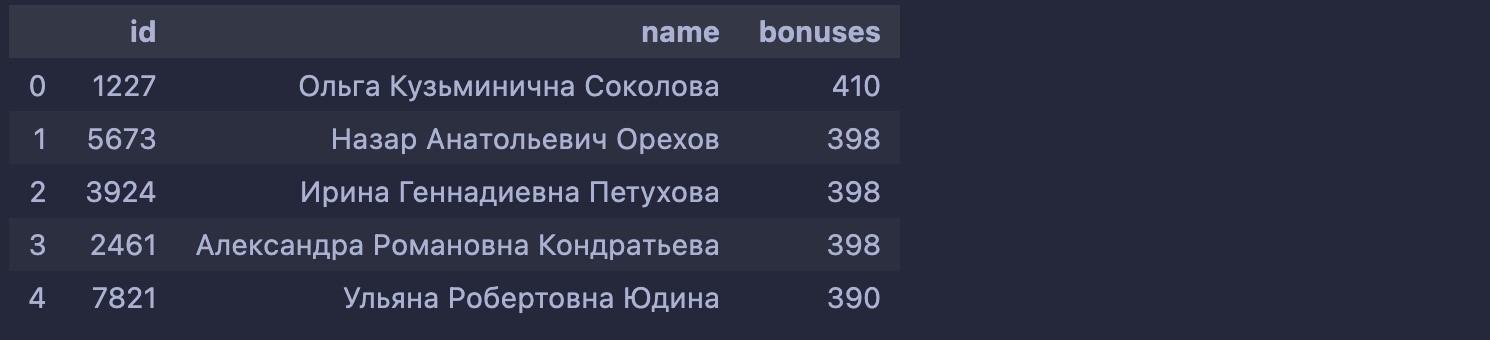
**Рис. 23:** Результат выполнения запроса 4

**Листинг 40:** Запрос 5 на SQL

| WITH payment\_infos\_and\_methods **AS** (  **SELECT**  i . i d **AS** p a y m e n t \_ i n f o \_ i d ,  m. t i t l e **AS** payment\_method  **FROM** p a y m e n t \_ i n f o s i  **JOIN** payment\_methods m **ON** i . payment\_method\_id = m. i d  ) ,  u s e r \_ m e t h o d \_ c o u n t s **AS** (  **SELECT**  o . u s e r \_ i d ,  p . payment\_method ,  **COUNT**( o . i d ) **AS** o r d e r s \_ c o u n t  **FROM** |
| --- |

47

| * r d e r s o   **JOIN** payment\_infos\_and\_methods p **USING** ( p a y m e n t \_ i n f o \_ i d )  **GROUP BY**   * . u s e r \_ i d ,   p . payment\_method  ) ,  u s e r \_ m e t h o d \_ c o u n t s \_ w i t h \_ t o t a l **AS** (  **SELECT**  \* ,  **SUM**( o r d e r s \_ c o u n t ) OVER (  PARTITION **BY** u s e r \_ i d  ) **AS** t o t a l \_ o r d e r s \_ c o u n t  **FROM** u s e r \_ m e t h o d \_ c o u n t s  ) ,  u s e r s \_ w i t h \_ a l m o s t \_ c a s h \_ o r d e r s **AS** (  **SELECT** u s e r \_ i d **AS** i d  **FROM** u s e r \_ m e t h o d \_ c o u n t s \_ w i t h \_ t o t a l  **WHERE** payment\_method = ' Наличные курьеру '  **AND** o r d e r s \_ c o u n t > t o t a l \_ o r d e r s \_ c o u n t / 2  )  **SELECT**  u . id ,  CONCAT( u . f i r s t \_ n a m e , ' ' , u . middle\_name , ' ' , u . l a s t \_ n a m e ) **AS** name , u . bonuses  **FROM** u s e r s u  **JOIN** u s e r s \_ w i t h \_ a l m o s t \_ c a s h \_ o r d e r s u\_cash **USING** ( i d )  **ORDER BY**  u . bonuses **DESC**  **LIMIT**  5 ; |
| --- |

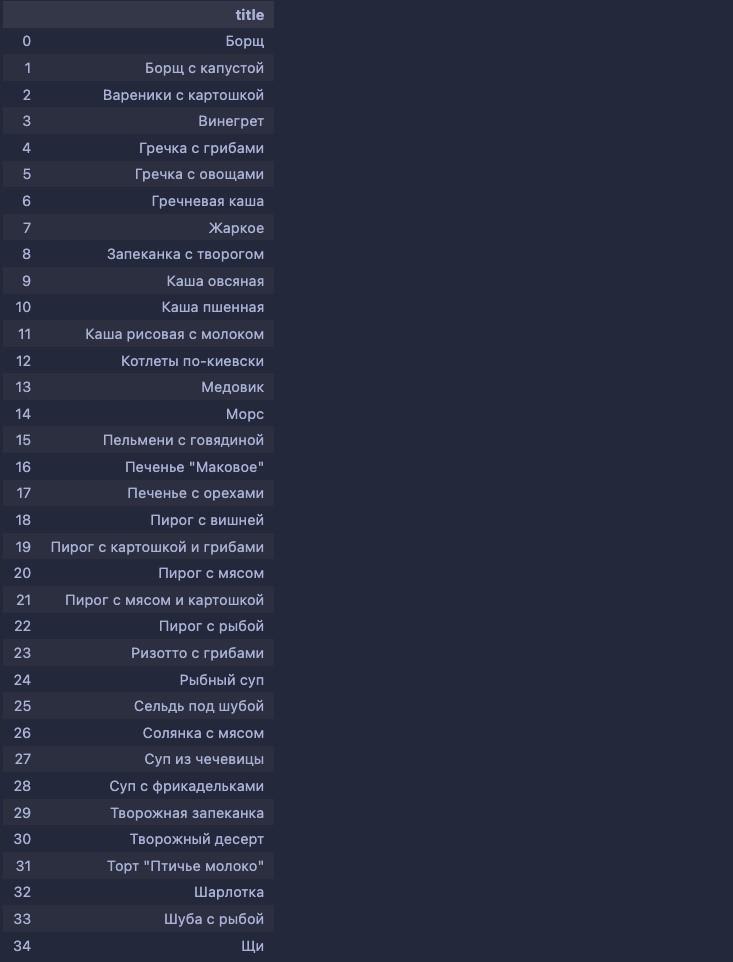


**Рис. 24:** Результат выполнения запроса 5

**Листинг 41:** Запрос 6 на SQL

| WITH m o s t \_ p o p u l a r \_ c a t e g o r y **AS** (  **SELECT** p r e f e r e n c e \_ c a t e g o r y \_ i d ,  **COUNT**( pu . i d ) **AS count**  **FROM** p r e f e r e n c e s \_ u s e r s pu  **JOIN** p r e f e r e n c e s p **ON** pu . p r e f e r e n c e \_ i d = p . i d  **GROUP BY** p r e f e r e n c e \_ c a t e g o r y \_ i d  **ORDER BY count DESC**  **LIMIT**  1  ) ,  p r e f e r e n c e s \_ f r o m \_ m o s t \_ p o p u l a r \_ c a t e g o r y **AS** (  **SELECT**  id ,  t i t l e  **FROM** p r e f e r e n c e s  **WHERE** p r e f e r e n c e \_ c a t e g o r y \_ i d = (  **SELECT** p r e f e r e n c e \_ c a t e g o r y \_ i d  **FROM** m o s t \_ p o p u l a r \_ c a t e g o r y  )  ) ,  i n g r e d i e n t s \_ f r o m \_ m o s t \_ p o p u l a r \_ c a t e g o r y **AS** (  **SELECT** i p . i n g r e d i e n t \_ i d  **FROM** i n g r e d i e n t s \_ p r e f e r e n c e s i p  **JOIN** p r e f e r e n c e s \_ f r o m \_ m o s t \_ p o p u l a r \_ c a t e g o r y p **ON** i p .  p r e f e r e n c e \_ i d = p . i d  )  **SELECT**  **DISTINCT** d . t i t l e  **FROM** d i s h e s \_ i n g r e d i e n t s  **JOIN** i n g r e d i e n t s \_ f r o m \_ m o s t \_ p o p u l a r \_ c a t e g o r y i **ON** d i s h e s \_ i n g r e d i e n t s .  i n g r e d i e n t \_ i d = i . i n g r e d i e n t \_ i d  **JOIN** d i s h e s d **ON** d i s h e s \_ i n g r e d i e n t s . d i s h \_ i d = d . i d  **ORDER BY**  d . t i t l e ; |
| --- |

49



**Рис. 25:** Результат выполнения запроса 6

**Листинг 42:** Запрос 7 на SQL

| WITH i n g r e d i e n t s \_ c o s t s \_ i n \_ r u b l e s **AS** (  **SELECT**  id ,   1. o s t \* 7 . 5 5 **AS** c o s t \_ r u b l e s   **FROM** i n g r e d i e n t s  ) ,   1. i s h e s \_ c o s t s **AS** (   **SELECT**  d . d i s h \_ i d **AS** id ,  **SUM**( i . c o s t \_ r u b l e s ) **AS** c o s t  **FROM** d i s h e s \_ i n g r e d i e n t s d  **JOIN** i n g r e d i e n t s \_ c o s t s \_ i n \_ r u b l e s i **ON** d . i n g r e d i e n t \_ i d = i . i d  **GROUP BY**  d . d i s h \_ i d ) ,  m e n u s \_ c o s t s \_ a n d \_ d i s h e s **AS** (  **SELECT** menu\_id ,  m. **date** ,  **SUM**( c o s t ) **AS** o u r \_ c o s t ,  ARRAY\_AGG( d i s h \_ i d ) **AS** d i s h e s  **FROM** dishes\_menus m  **JOIN** d i s h e s \_ c o s t s c **ON** m. d i s h \_ i d = c . i d  **GROUP BY**  menu\_id ,  m. **date**  **ORDER BY**  m. **date** ,  menu\_id  ) ,  m e n u s \_ p r o f i t s **AS** (  **SELECT** mc . \* ,  m. c o s t ,  ROUND( (m. c o s t - mc . o u r \_ c o s t ) : : **DECIMAL**, 2) **AS** p r o f i t ,  ROUND( (m. c o s t - mc . o u r \_ c o s t ) : : **DECIMAL** / mc . o u r \_ c o s t : : **DECIMAL**  \* 100 , 2) **AS** p r o f i t \_ p e r c e n t  **FROM** m e n u s \_ c o s t s \_ a n d \_ d i s h e s mc  **JOIN** menus m **ON** mc . menu\_id = m. i d  ) ,  m a x \_ p r o f i t **AS** (  **SELECT** menu\_id , **date** ,  d i s h e s , p r o f i t , p r o f i t \_ p e r c e n t  **FROM** |
| --- |

51

| m e n u s \_ p r o f i t s  **ORDER BY** p r o f i t \_ p e r c e n t **DESC**  **LIMIT** 1  ) ,  m a x \_ p r o f i t \_ p e r c e n t **AS** (  **SELECT** menu\_id , **date** ,  d i s h e s , p r o f i t , p r o f i t \_ p e r c e n t  **FROM** m e n u s \_ p r o f i t s  **ORDER BY** p r o f i t **DESC**  **LIMIT** 1  )  **SELECT** \* **FROM** m a x \_ p r o f i t **UNION ALL SELECT** \* **FROM** m a x \_ p r o f i t \_ p e r c e n t ; |
| --- |



**Рис. 26:** Результат выполнения запроса 7

**Листинг 43:** Запрос 8 на SQL

| WITH m a x \_ c o u n t \_ p a i r **AS** (  **SELECT DISTINCT** ARRAY[ dm1 .  **COUNT**( \* ) **AS**  **FROM** dishes\_menus  **WHERE** dm1 . d i s h \_ i d  **GROUP BY**  p a i r  **ORDER BY count DESC**  **LIMIT** 1  )  **SELECT** d1 . t i t l e **AS** dish1 , d2 . t i t l e **AS** dish2 ,  **count**  **FROM** | d i s h \_ i d , dm2 . d i s h \_ i d ] **AS** p a i r , **count** dm1 **JOIN** dishes\_menus dm2 **USING**( menu\_id ,  < dm2 . d i s h \_ i d | | **date** ) |
| --- | --- | --- | --- |
| m a x \_ c o u n t \_ p a i r  **JOIN** d i s h e s d1 **ON** p a i r [ 1 ] | | = d1 . i d | |
| **JOIN** d i s h e s d2 **ON** p a i r [ 2 ] | | = d2 . i d | |



**Рис. 27:** Результат выполнения запроса 8

**Листинг 44:** Запрос 9 на SQL

| WITH m a i n \_ t a b l e **AS** (  **SELECT**  u . id ,  **COUNT**( o . i d ) **AS** o r d e r s \_ c o u n t  **FROM** u s e r s u  **JOIN** o r d e r s o **ON** u . i d = o . u s e r \_ i d  **GROUP BY**  u . i d  **ORDER BY** o r d e r s \_ c o u n t **ASC**  ) ,  s e r i e s **AS** (  **SELECT** o r d e r s \_ c o u n t ,  ROW\_NUMBER( ) OVER (  **ORDER BY**  o r d e r s \_ c o u n t  ) **AS** row\_number ,  **COUNT**( \* ) OVER ( ) **AS** t o t a l \_ r o w s  **FROM** m a i n \_ t a b l e  )  **SELECT**  **AVG**( o r d e r s \_ c o u n t ) **AS** median\_count  **FROM**  s e r i e s  **WHERE** row\_number **BETWEEN** t o t a l \_ r o w s / 2 . 0 **AND** t o t a l \_ r o w s | / | 2 . 0 + 1 |
| --- | --- | --- |

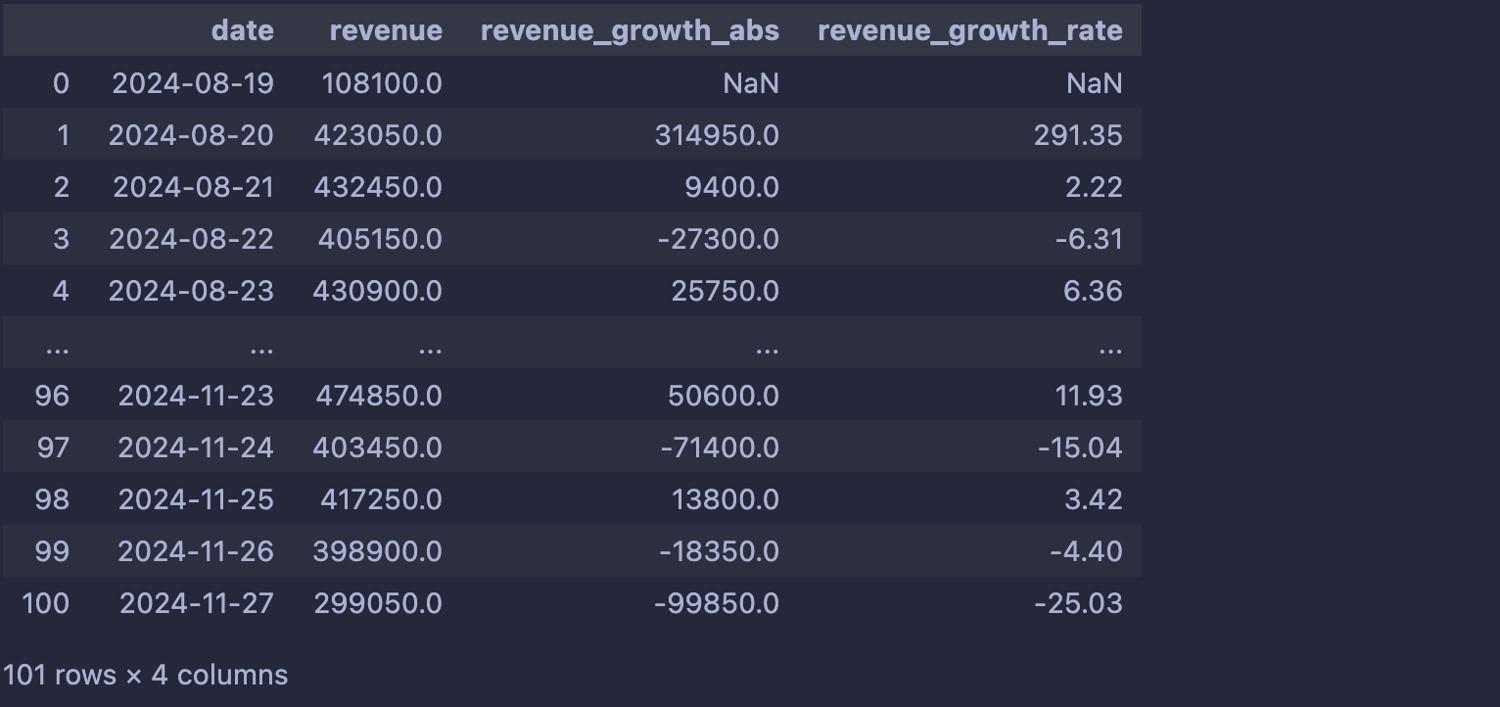


**Рис. 28:** Результат выполнения запроса 9

53

**Листинг 45:** Запрос 10 на SQL

| WITH m a i n \_ t a b l e **AS** (  **SELECT**  **DATE** ( o . c r e a t e d \_ a t ) **AS date** ,  **SUM**(m. c o s t ) **AS** r e v e n u e  **FROM** o r d e r s o  **JOIN** menus m **ON** o . menu\_id = m. i d  **GROUP BY**  **DATE** ( o . c r e a t e d \_ a t )  **ORDER BY**  **date**  )  **SELECT date** , revenue ,  ( r e v e n u e - LAG ( revenue , 1) OVER ( ) ) **AS** revenue\_growth\_abs ,  ROUND(100 \* ( r e v e n u e - LAG ( revenue , 1) OVER ( ) ) : : **DECIMAL** / LAG ( revenue , 1) OVER ( ) : : **DECIMAL**, 2) **AS** r e v e n u e \_ g r o w t h \_ r a t e  **FROM** m a i n \_ t a b l e |
| --- |

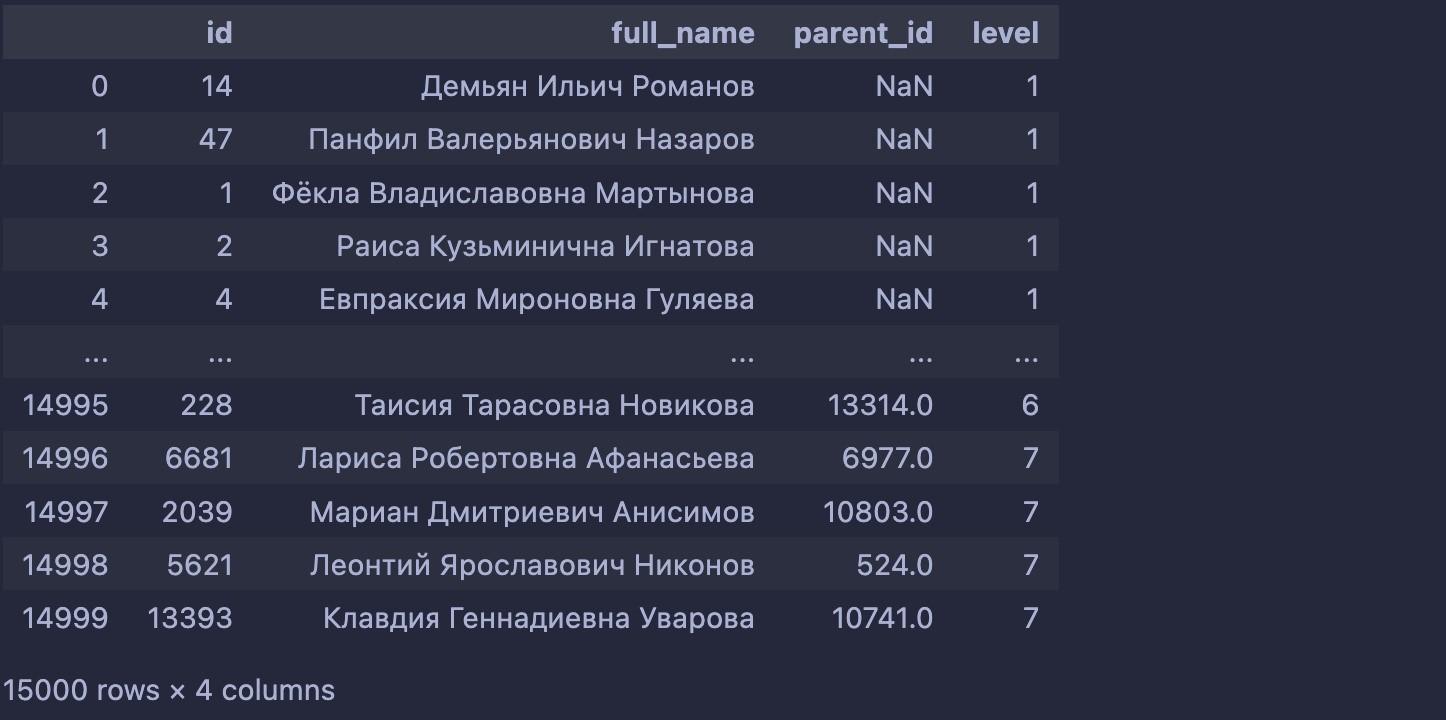


**Рис. 29:** Результат выполнения запроса 10

**Листинг 46:** Запрос 11 на SQL

| WITH **RECURSIVE** u s e r s \_ t r e e **AS** (  **SELECT**  id ,  CONCAT ( f i r s t \_ n a m e , | ' | ' , | middle\_name , | ' | ' , | l a s t \_ n a m e ) **AS** |
| --- | --- | --- | --- | --- | --- | --- |

| full\_name ,  i n v i t e d \_ b y \_ i d **AS** p a r e n t \_ i d ,  1 **AS level**  **FROM**  u s e r s  **WHERE** i n v i t e d \_ b y \_ i d IS **NULL**  **UNION ALL**  **SELECT**  u . id ,  CONCAT ( f i r s t \_ n a m e , ' ' , middle\_name , | ' | ' , l a s t \_ n a m e ) **AS** |
| --- | --- | --- |
| full\_name ,  i n v i t e d \_ b y \_ i d **AS** p a r e n t \_ i d , u t . **level** + 1  **FROM** u s e r s u  **JOIN** u s e r s \_ t r e e u t **ON** u . i n v i t e d \_ b y \_ i d  )  **SELECT**  \*  **FROM** u s e r s \_ t r e e | = | u t . i d |

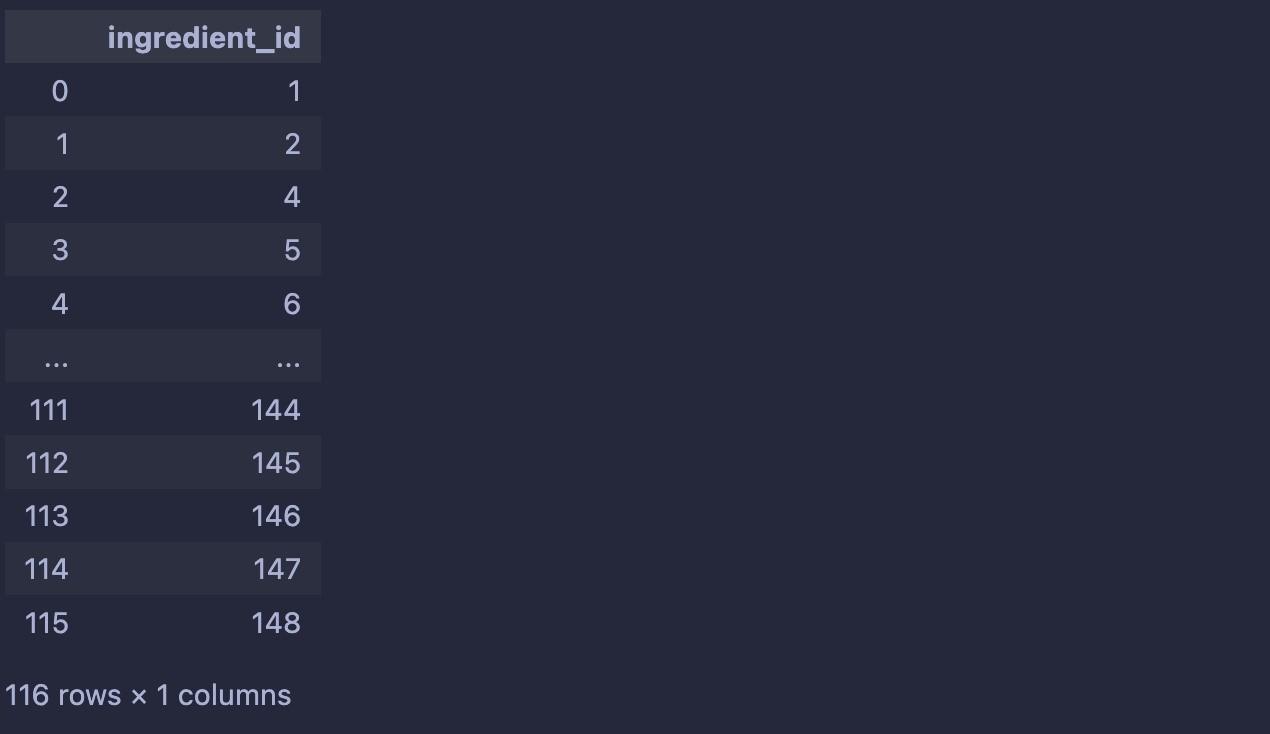


**Рис. 30:** Результат выполнения запроса 11

**Листинг 47:** Запрос 12 на SQL

| WITH o r d e r s \_ l a s t \_ m o n t h **AS** (  **SELECT** menu\_id ,  **date** ( c r e a t e d \_ a t ) **AS date** , |
| --- |

| **COUNT**( \* ) **AS** o r d e r s \_ c o u n t  **FROM** o r d e r s  **WHERE date** ( c r e a t e d \_ a t ) + **INTERVAL** ' 1 day ' **BETWEEN CURRENT\_DATE INTERVAL** ' 1 month ' **AND CURRENT\_DATE**  **GROUP BY date** ( c r e a t e d \_ a t ) ,  menu\_id  ) ,  d i s h e s \_ c o u n t **AS** (  **SELECT** d i s h \_ i d ,  **SUM**( o r d e r s \_ c o u n t ) **AS count**  **FROM** o r d e r s \_ l a s t \_ m o n t h o  **JOIN** dishes\_menus dm **ON** o . menu\_id = dm . menu\_id  **AND** o . **date** + **INTERVAL** ' 1 month 1 day ' = dm . **date**  **GROUP BY**  d i s h \_ i d  ) ,  d i s h e s \_ i n g r e d i e n t s \_ w i t h \_ w e i g h t s **AS** (  **SELECT** d i s h \_ i d , i n g r e d i e n t \_ i d ,  wei gh t : : **DECIMAL** / **COUNT**( \* ) OVER ( PARTITION **BY** d i s h \_ i d ) : : **DECIMAL AS** i n g r e d i e n t s \_ w e i g h t  **FROM** d i s h e s \_ i n g r e d i e n t s **AS** d i  **JOIN** d i s h e s **AS** d **ON** d i . d i s h \_ i d = d . i d  ) ,  i n g r e d i e n t s \_ t o t a l \_ w e i g h t **AS** (  **SELECT** i n g r e d i e n t \_ i d ,  **SUM**( i n g r e d i e n t s \_ w e i g h t \* **count** ) **AS** t o t a l \_ w e i g h t  **FROM** d i s h e s \_ i n g r e d i e n t s \_ w i t h \_ w e i g h t s **AS** w e i g h t s  **JOIN** d i s h e s \_ c o u n t **AS** c o u n t s **USING** ( d i s h \_ i d )  **GROUP BY**  i n g r e d i e n t \_ i d  ) ,  i n g r e d i e n t s \_ p r o d u c t i v i t y **AS** (  **SELECT** i n g r e d i e n t \_ i d ,  **SUM**( p r o d u c t i v i t y ) **AS** p r o d u c t i v i t y  **FROM** i n g r e d i e n t s \_ s u p p l i e r s **AS** i n s  **JOIN** s u p p l i e r s s **ON** i n s . s u p p l i e r \_ i d = s . i d  **GROUP BY**  i n g r e d i e n t \_ i d  )  **SELECT** | | | |
| --- | --- | --- | --- |
| i n g r e d i e n t \_ i d  **FROM** i n g r e d i e n t s \_ t o t a l \_ w e i g h t **AS** i t w  **JOIN** i n g r e d i e n t s \_ p r o d u c t i v i t y **AS** i p  **WHERE** p r o d u c t i v i t y \* 1000 >= t o t a l \_ w e i g h t  **ORDER BY**  i n g r e d i e n t \_ i d | **USING** | ( i n g r e d i e n t \_ i d ) |



**Рис. 31:** Результат выполнения запроса 12

## Отчеты

### Отчет №1

**Название:** Анализ выручки сервиса за предыдущие 30 дней

**Дата создания отчета:** 25.05.2025

**Инструментальные средства:** *Python, Matplotlib, Seaborn* **Диаграмма:**



**Рис. 32:** Диаграмма к отчету №1

**Код запроса:**

58

**Листинг 48:** Код запроса к отчету №1

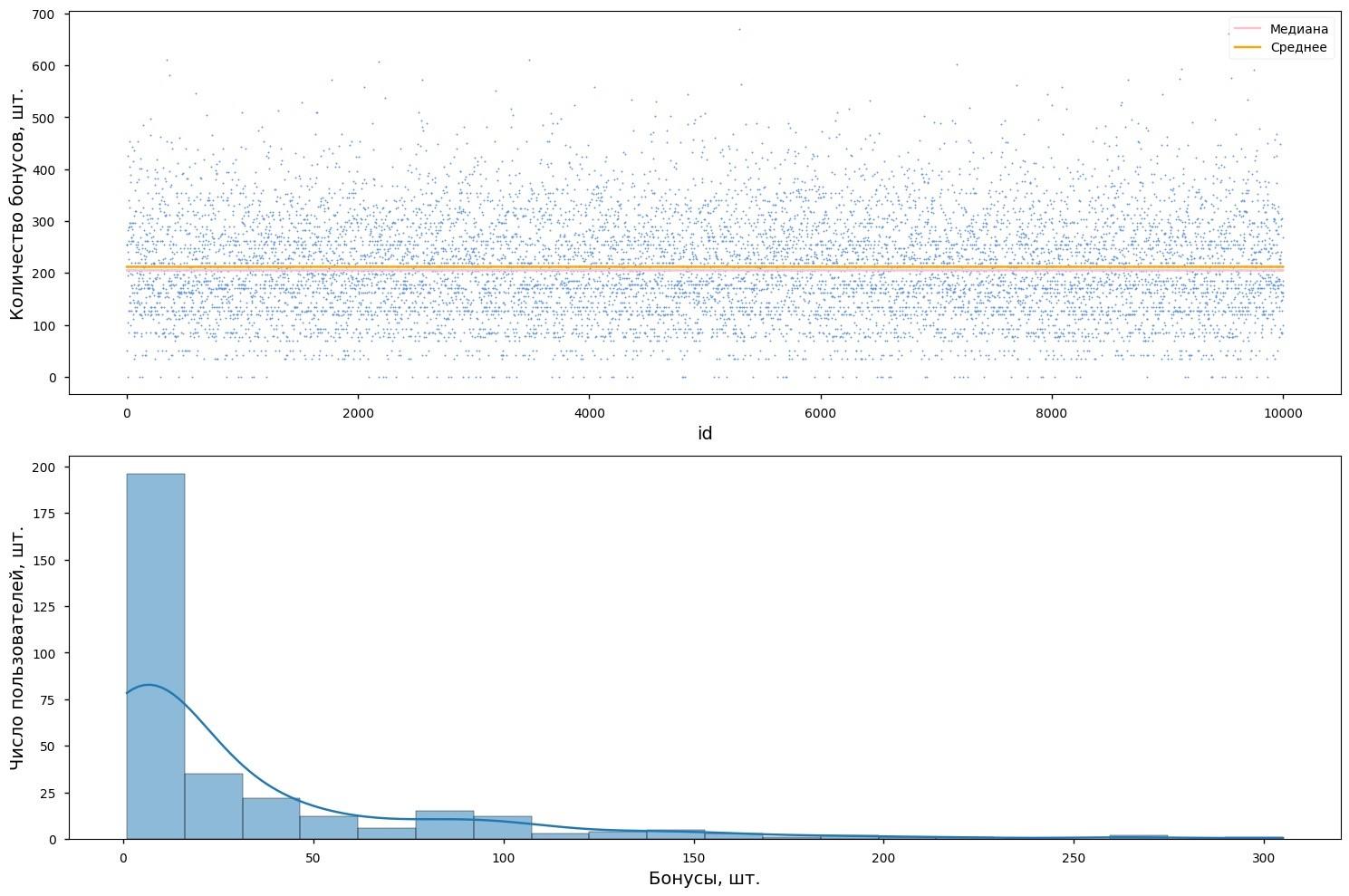
| WITH m a i n \_ t a b l e **AS** (  **SELECT**  **DATE** ( o . c r e a t e d \_ a t ) **AS date** ,  **SUM**(m. c o s t ) **AS** r e v e n u e  **FROM** o r d e r s o  **JOIN** menus m **ON** o . menu\_id = m. i d  **GROUP BY**  **DATE** ( o . c r e a t e d \_ a t )  **ORDER BY**  **date**  )  **SELECT date** , revenue ,  ( r e v e n u e - LAG ( revenue , 1) OVER ( ) ) **AS** revenue\_growth\_abs ,  ROUND(100 \* ( r e v e n u e - LAG ( revenue , 1) OVER ( ) ) : : **DECIMAL** / LAG ( revenue , 1) OVER ( ) : : **DECIMAL**, 2) **AS** r e v e n u e \_ g r o w t h \_ r a t e  **FROM** m a i n \_ t a b l e |
| --- |

### Отчет №2

**Название:** Анализ распределения количества бонусов среди пользователей

**Дата создания отчета:** 25.05.2025

**Инструментальные средства:** *Python, Pandas, Matplotlib, Seaborn* **Диаграмма:**



**Рис. 33:** Диаграмма к отчету №2

**Коды запроса:**

**Листинг 49:** Код запроса к отчету №2

| **SELECT**  id , bonuses  **FROM**  u s e r s  **ORDER BY** i d **ASC** |
| --- |

**Листинг 50:** Код запроса к отчету №2

60

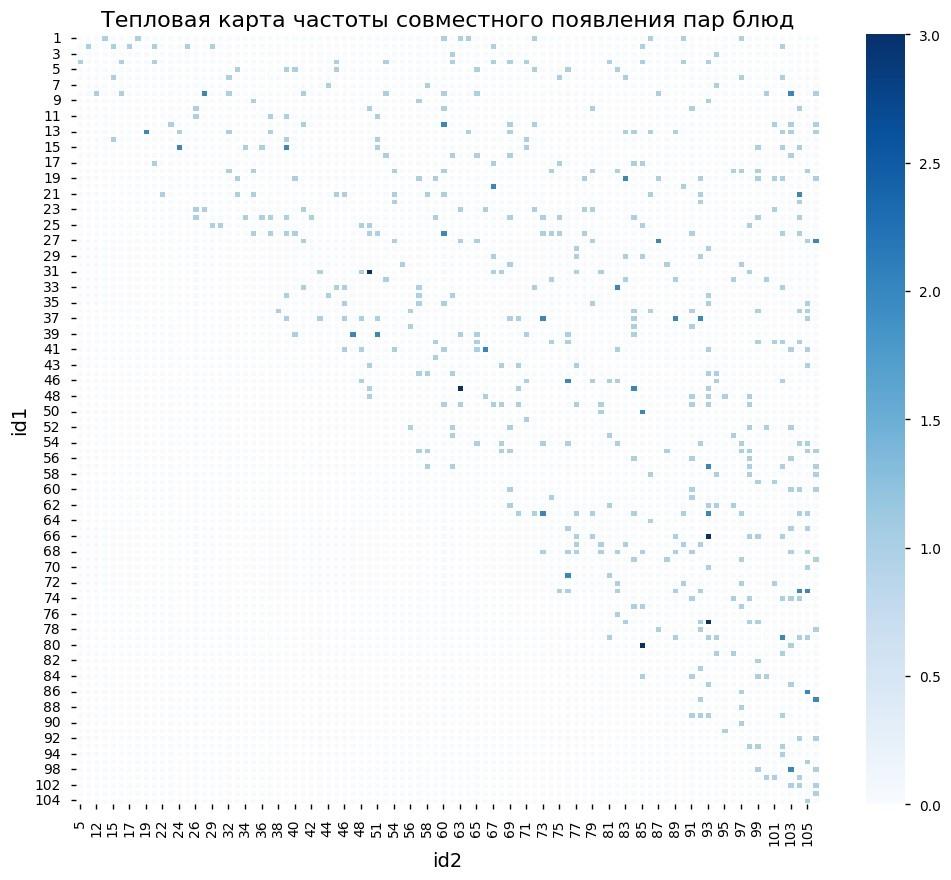
| **SELECT** bonuses ,  **COUNT**( \* ) **AS count**  **FROM**  u s e r s  **GROUP BY**  bonuses  **ORDER BY** bonuses **ASC**; |
| --- |

### Отчет №3

**Название:** Анализ частоты встречаемости различных пар блюд в меню

**Дата создания отчета:** 25.05.2025

**Инструментальные средства:** *Python, Pandas, Matplotlib, Seaborn* **Диаграмма:**



**Рис. 34:** Диаграмма к отчету №2

**Код запроса:**

**Листинг 51:** Код запроса к отчету №3

62

| WITH m a x \_ c o u n t \_ p a i r **AS** (  **SELECT DISTINCT**  ARRAY[ dm1 . d i s h \_ i d , dm2 . d i s h \_ i d ] **AS** p a i r ,  **COUNT**( \* ) **AS count**  **FROM** dishes\_menus dm1 **JOIN** dishes\_menus dm2 **USING**( menu\_id ,  **WHERE** dm1 . d i s h \_ i d < dm2 . d i s h \_ i d  **GROUP BY**  p a i r | | | **date** ) |
| --- | --- | --- | --- |
| )  **SELECT** d1 . i d | **AS** id1 , |  |  |
| d2 . i d | **AS** id2 , |  |  |
| **count**  **FROM** m a x \_ c  **JOIN** | o u n t \_ p a i r d i s h e s d1 | **ON** p a i r [ 1 ] = d1 . i d |  |
| **JOIN** d i s h e s d2 | | **ON** p a i r [ 2 ] = d2 . i d |  |