

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Национальный Исследовательский Ядерный Университет «МИФИ»

Институт интеллектуальных кибернетических систем
Кафедра «Кибернетика» (№22)
Направление подготовки 09.03.04. Программная инженерия



**Отчет о работе по курсу
«Базы данных (теоретические основы баз данных)»**

Вариант «*Delivery Club*»

Выполнил	Корнишев Д.О.
Группа	Б22-514
Вариант	« <i>Delivery Club</i> »
Преподаватель	Петровская А.В.
Проверяющий	Панин И.С
Оценка	

Москва, 2024

Оглавление

1. Формулировка задания.....	3
2. Концептуальная модель базы данных.....	3
2.1. Конкретизация предметной области.....	4
2.2 Описание предметной области.....	4
2.3 Описание атрибутов.....	5
3. Логическое проектирование.....	6
4. Физическое проектирование.....	7
4.1 Создание таблиц.....	8
4.2 Заполнение таблиц.....	12
4.2.1 Подготовка данных.....	12
4.2.2 Программа заполнения базы данных.....	12
4.2.3 Результаты заполнения.....	19
5. Выполнение запросов.....	23

1. Формулировка задания

Спроектировать и разработать эффективную базу данных, которая будет поддерживать функциональность приложения Delivery Club, обеспечивая хранение и управление данными о пользователях, ресторанах, магазинах, меню, заказах и доставках.

2. Концептуальная модель базы данных

После проведения анализа предметной области была спроектирована следующая концептуальная модель:

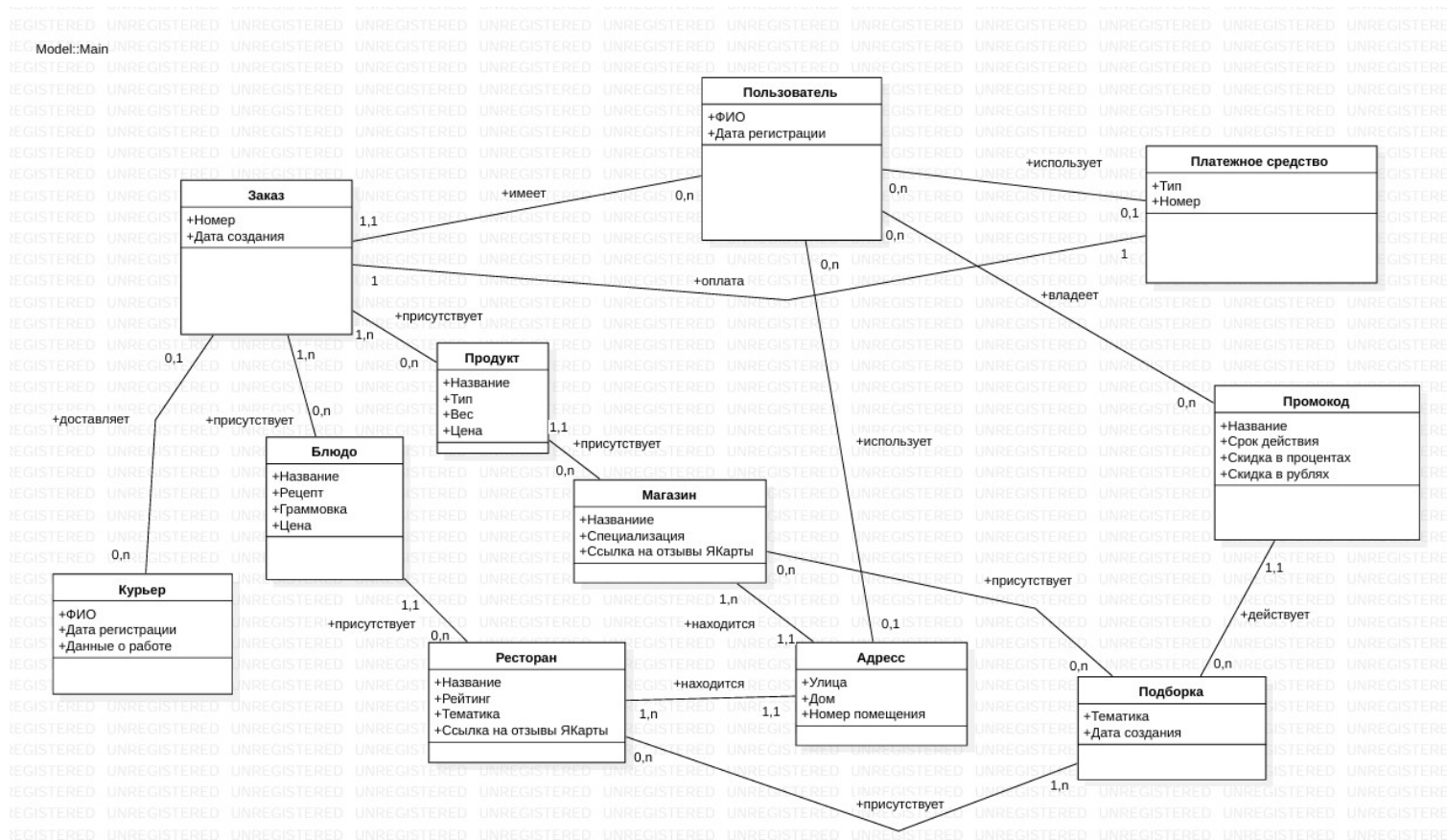


Рисунок 1 – Концептуальная модель базы данных

2.1. Конкретизация предметной области

Необходимо создать систему, отражающую информацию о товарах и блюдах, размещаемых ресторанами и магазинами в системе, с последующей возможностью заказа и курьерской доставки пользователям. Также необходимо хранить информацию о самих заказах, ресторанах и магазинах, курьерах, осуществляющих доставку. Наконец, система должна хранить информацию о платежных средствах и промокодах, которыми пользователь обладает и может воспользоваться для заказа.

2.2 Описание предметной области

Система рассчитана на зарегистрированных пользователей и предоставляет возможности просмотра доступных ресторанов и магазинов, их каталога, а также совершения заказов в них.

Пользователь также может получать и использовать промокоды, действующие на определенные подборки товаров.

Система в полной мере отражает внутренние процессы, такие как заказ и доставка, этапы доставки, её исполнителя, средство оплаты.

Пользователь может получать информацию необходимую для выбора требуемой продукции. Поэтому каждому ресторану и магазину сопоставлена информация о его местонахождении, среднем рейтинге, отзывах, специализации. В свою очередь каждый товар предоставляемый определенным рестораном или магазином обладает описанием.

Исходя из описанных возможностей системы, выделены следующие сущности:

1. Пользователь
2. Курьер
3. Заказ
4. Платежное средство
5. Магазин
6. Ресторан
7. Блюдо
8. Товар

9. Подборка

10. Промокод

11. Адрес

2.3 Описание атрибутов

Имя атрибута	Расшифровка
id	Уникальный идентификатор объекта
ФИО	Фамилия, имя, отчество зарегистрированного пользователя
Номер заказа	Специфический идентификатор заказа в системе
Дата создания, регистрации	Дата внесения объекта в систему
Название	Название блюда, товара, ресторана, магазина или промокода в системе
Тип, специализация, тематика	Характерные свойства объекта среди объектов множества
Граммовка, цена, рецепт, вес	Основные характеристики продукта
Рейтинг	Средний рейтинг заведения по отзывам
Ссылка на отзывы	Ссылка на конкретные отзывы пользователей заведения в стороннем сервисе (ЯКарты)
Улица, дом, номер помещения	Данные об адресе определенного объекта (ресторана, магазина или пользователя)
Срок действия	Интервал действия определенного промокода
Скидка в процентах, скидка в рублях	Объем скидки предоставляемый определенным промокодом
Статус	Статус заказа из множества: доставлен/отменен
Данные о работе	Данные о работе курьера : активен или нет
Сумма	Сумма заказа в рублях
Номер	Номер платежного средства

3. Логическое проектирование

Следующим шагом на основе КМПО была разработана логическая модель базы данных, представленная ниже:

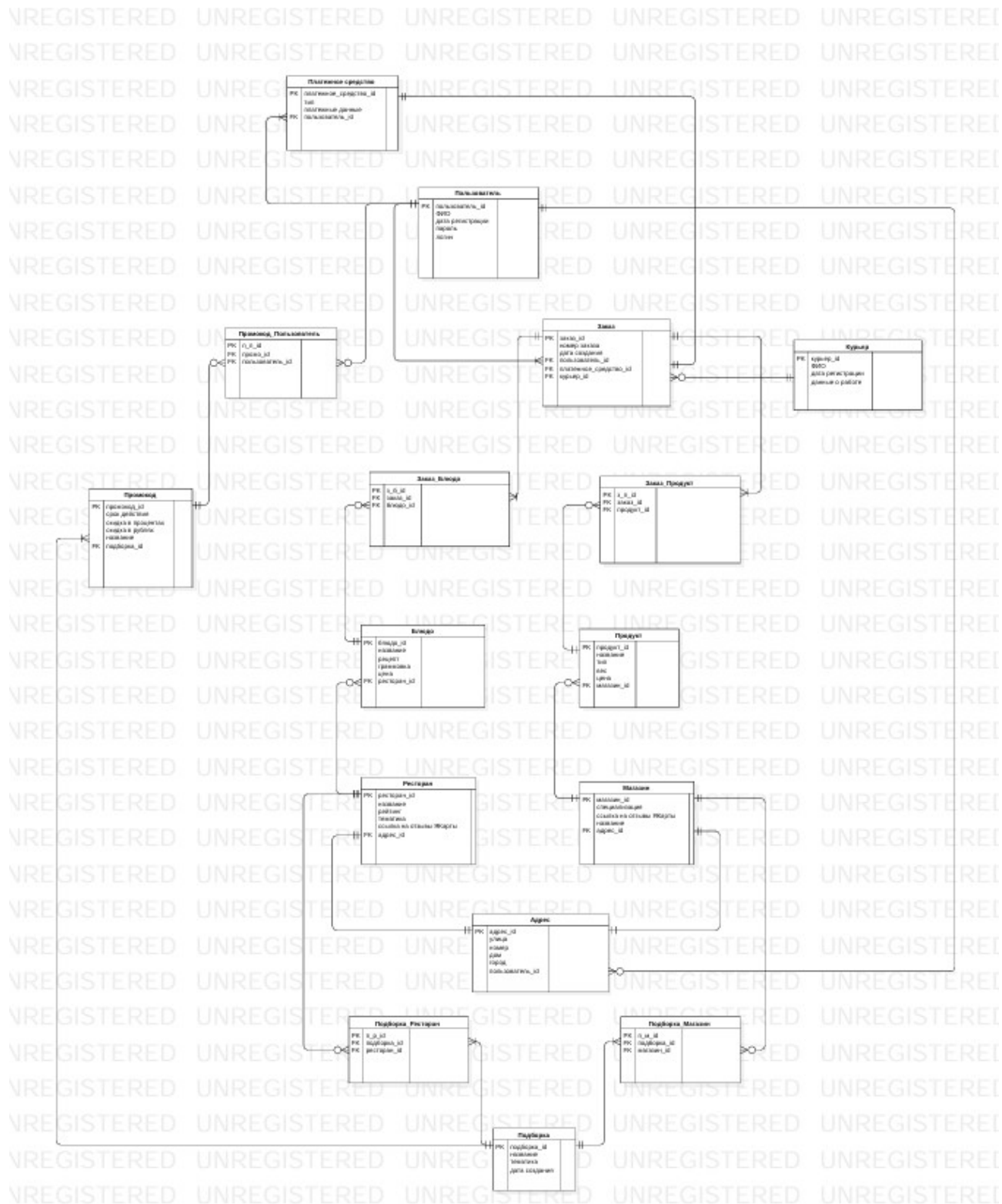


Рисунок 2 – Логическая модель базы данных

Все таблицы находятся в нормальной форме. На схеме приведены все необходимые связи один-многие, многие-многие.

4. Физическое проектирование

В качестве СУБД для реализации разработанной базы данных была выбрана PostgreSQL. В связи с проведённым анализом предметной области была проработана следующая физическая схема БД. Она представлена на следующем рисунке:

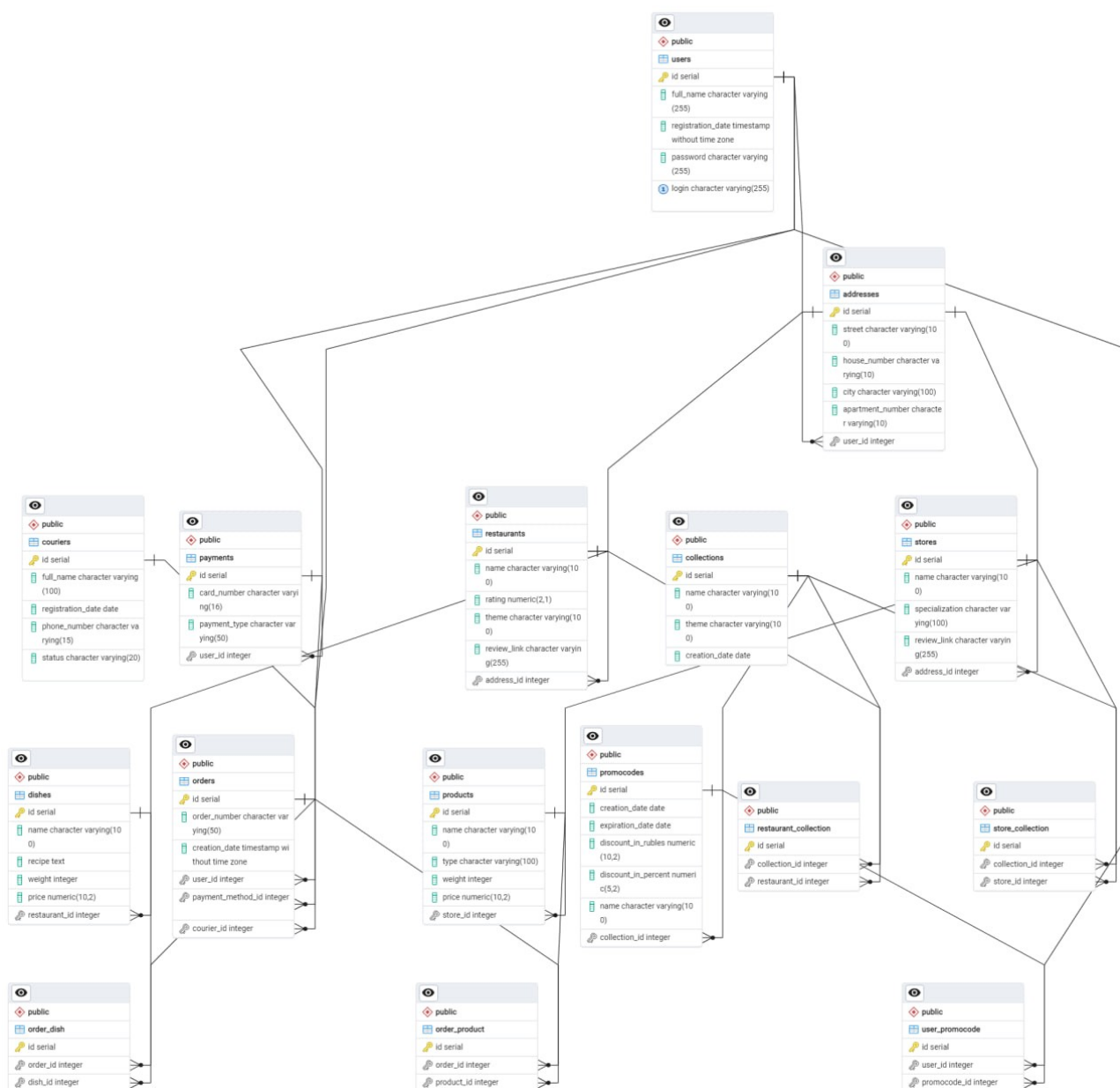


Рисунок 3 – Графическое представление базы данных

4.1 Создание таблиц

Ниже приведены запросы на языке SQL (диалект PostgreSQL) для создания таблиц, описанных выше.

Код создания таблицы «users»

```
CREATE TABLE IF NOT EXISTS users ( -- 1
    id SERIAL PRIMARY KEY,
    full_name VARCHAR(255) NOT NULL,
    registration_date TIMESTAMP NOT NULL DEFAULT NOW(),
    password VARCHAR(255) NOT NULL,
    login VARCHAR(255) NOT NULL UNIQUE
);
```

Код создания таблицы «payments»

```
CREATE TABLE IF NOT EXISTS payments ( -- 2
    id SERIAL PRIMARY KEY,
    card_number VARCHAR(16) NOT NULL,
    payment_type VARCHAR(50) NOT NULL,
    user_id INT NOT NULL,
    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
);
```

Код создания таблицы «couriers»

```
CREATE TABLE IF NOT EXISTS couriers ( -- 3
    id SERIAL PRIMARY KEY,
    full_name VARCHAR(100) NOT NULL,
    registration_date DATE NOT NULL,
    phone_number VARCHAR(15),
    status VARCHAR(20) DEFAULT 'active'
);
```

Код создания таблицы «orders»

```
CREATE TABLE IF NOT EXISTS orders ( -- 4
    id SERIAL PRIMARY KEY,
    order_number VARCHAR(50) NOT NULL,
    creation_date TIMESTAMP NOT NULL DEFAULT NOW(),
    user_id INT NOT NULL,
    payment_method_id INT NOT NULL,
    courier_id INT,
    order_status varchar(15) not null,
    close_date timestamp,

    FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE,
    FOREIGN KEY (payment_method_id) REFERENCES payments(id) ON DELETE CASCADE,
    FOREIGN KEY (courier_id) REFERENCES couriers(id) ON DELETE CASCADE
);
```


Код создания таблицы «collections»

```
CREATE TABLE IF NOT EXISTS collections ( -- 5
  id SERIAL PRIMARY KEY,
  name VARCHAR(100) NOT NULL,
  theme VARCHAR(100),
  creation_date DATE NOT NULL
);
```

Код создания таблицы «promocodes»

```
CREATE TABLE IF NOT EXISTS promocodes ( -- 6
  id SERIAL PRIMARY KEY,
  creation_date DATE NOT NULL,
  expiration_date DATE NOT NULL,
  discount_in_rubles NUMERIC(10, 2),
  discount_in_percent NUMERIC(5, 2),
  name VARCHAR(100) NOT NULL,
  collection_id INT,
  FOREIGN KEY (collection_id) REFERENCES collections(id)
);
```

Код создания таблицы «user_promocode»

```
CREATE TABLE IF NOT EXISTS user_promocode ( -- 7
  id SERIAL PRIMARY KEY,
  user_id INT,
  promocode_id INT,
  FOREIGN KEY (user_id) REFERENCES users(id),
  FOREIGN KEY (promocode_id) REFERENCES promocodes(id)
);
```

Код создания таблицы «addresses»

```
CREATE TABLE IF NOT EXISTS addresses ( -- 8
  id SERIAL PRIMARY KEY,
  street VARCHAR(100) NOT NULL,
  house_number VARCHAR(10) NOT NULL,
  city VARCHAR(100) NOT NULL,
  apartment_number VARCHAR(10),
  user_id INT,
  FOREIGN KEY (user_id) REFERENCES users(id)
);
```

Код создания таблицы «restaurants»

```
CREATE TABLE IF NOT EXISTS restaurants ( -- 9
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    rating DECIMAL(2, 1) CHECK (rating >= 0 AND rating <= 5),
    theme VARCHAR(100),
    review_link VARCHAR(255),
    address_id INT,
    FOREIGN KEY (address_id) REFERENCES addresses(id)
);
```

Код создания таблицы «stores»

```
CREATE TABLE IF NOT EXISTS stores ( -- 10
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    specialization VARCHAR(100),
    review_link VARCHAR(255),
    address_id INT,
    FOREIGN KEY (address_id) REFERENCES addresses(id)
);
```

Код создания таблицы «dishes»

```
CREATE TABLE IF NOT EXISTS dishes ( -- 11
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    recipe TEXT,
    weight INT,
    price DECIMAL(10, 2) NOT NULL,
    restaurant_id INT,
    FOREIGN KEY (restaurant_id) REFERENCES restaurants(id)
);
```

Код создания таблицы «products»

```
CREATE TABLE IF NOT EXISTS products ( -- 12
    id SERIAL PRIMARY KEY,
    name VARCHAR(100) NOT NULL,
    type VARCHAR(100),
    weight INT,
    price DECIMAL(10, 2) NOT NULL,
    store_id INT,
    FOREIGN KEY (store_id) REFERENCES stores(id)
);
```

Код создания таблицы «order_dish»

```
CREATE TABLE IF NOT EXISTS order_dish ( -- 13
    id SERIAL PRIMARY KEY,
    order_id INT,
    dish_id INT,
    FOREIGN KEY (order_id) REFERENCES orders(id),
    FOREIGN KEY (dish_id) REFERENCES dishes(id)
);
```

Код создания таблицы «order_product»

```
CREATE TABLE IF NOT EXISTS order_product ( -- 14
    id SERIAL PRIMARY KEY,
    order_id INT,
    product_id INT,
    FOREIGN KEY (order_id) REFERENCES orders(id),
    FOREIGN KEY (product_id) REFERENCES products(id)
);
```

Код создания таблицы «restaurant_collection»

```
CREATE TABLE IF NOT EXISTS restaurant_collection ( -- 15
    id SERIAL PRIMARY KEY,
    collection_id INT,
    restaurant_id INT,
    FOREIGN KEY (collection_id) REFERENCES collections(id),
    FOREIGN KEY (restaurant_id) REFERENCES restaurants(id)
);
```

Код создания таблицы «store_collection»

```
CREATE TABLE IF NOT EXISTS store_collection ( -- 16
    id SERIAL PRIMARY KEY,
    collection_id INT,
    store_id INT,
    FOREIGN KEY (collection_id) REFERENCES collections(id),
    FOREIGN KEY (store_id) REFERENCES stores(id)
);
```

4.2 Заполнение таблиц

Заполнение базы данных проводилось при помощи библиотеки для работы с PostgreSQL `psycopg2` языка программирования Python (библиотека `faker`).

4.2.1 Подготовка данных

Были сгенерированы и подготовлены массивы данных для блюд, товаров, ресторанов, подборок. По возможности для генерации тематических данных использованы готовые функции библиотеки `faker`, например:

`fake.date_between(start_date='-1y',end_date='today')` для дат,

`fake.credit_card_number()[:16]` для номера карты,

`random.randint(7, 30)` для чисел и тд.

Функции заполнения данных разработаны с учетом возможных коллизий.

4.2.2 Программа заполнения базы данных

Код для заполнения всех таблиц имеет похожую структуру и представляет собой `INSERT`-запрос с подготовленными данными. При необходимости до основного запроса проводится несколько `SELECT`-запросов для получения корректных данных для выбора и вставки. Данные генерируются случайно либо с помощью `random` и массивов данных, либо с помощью `faker`. В цикле производится последовательное заполнение таблицы. С помощью `psycopg2` производится подключение и закрытие подключения. Далее можно видеть примеры кода:

- Код заполнения таблицы products

```
import random
from faker import Faker
import psycopg2

# Создаем экземпляр Faker
fake = Faker()

# Массив названий продуктов
product_names = [
    "Яблоко", "Банан", "Апельсин", "Груша", "Киви",
    "Морковь", "Картофель", "Помидор", "Огурец", "Брокколи",
    "Курица", "Говядина", "Свинина", "Индейка", "Рыба",
    "Молоко", "Сыр", "Йогурт", "Сметана", "Творог",
    "Рис", "Гречка", "Овсянка", "Пшено", "Киноа",
    "Фасоль", "Чечевица", "Горошек", "Консервированные перцы", "Консервированные помидоры",
    "Шоколад", "Печенье", "Конфеты", "Торт", "Мороженое",
    "Чай", "Кофе", "Сок", "Вода", "Лимонад",
    "Хлеб", "Булочка", "Круассан", "Пирог", "Пирожное",
    "Спагетти", "Рагу", "Пицца", "Бургер", "Салат",
    "Каша", "Суп", "Блины", "Оладьи", "Запеканка",
    "Паста", "Лазанья", "Котлета", "Шашлык", "Фажитас",
    "Суши", "Роллы", "Тартар", "Крем-суп", "Фондю",
    "Салат Цезарь", "Теплый салат", "Салат с тунцом", "Салат с курицей", "Салат с авокадо",
    "Салат из свеклы", "Салат с морепродуктами", "Салат с картошкой", "Салат с фасолью", "Салат с грибами",
    "Салат с горошком", "Салат с курицей и ананасами", "Салат с курицей и помидорами", "Салат с рыбой", "Салат с копченым лососем",
    "Салат с овощами", "Салат с картошкой и горошком", "Салат с тунцом и фасолью", "Салат с курицей и грибами", "Салат с курицей и картошкой",
    "Суп с фрикадельками", "Суп с чечевицей", "Суп с овощами", "Суп с грибами", "Суп с курицей",
    "Суп с лапшой", "Суп с брокколи", "Суп с картошкой", "Суп из морепродуктов", "Суп-пюре",
    "Каша овсяная", "Каша гречневая", "Каша рисовая", "Каша манная", "Каша с фруктами",
    "Каша с медом", "Каша с курагой", "Каша с ягодами", "Каша с тыквой", "Каша с орехами",
    "Каша с яблоками", "Каша с бананами", "Каша с изюмом", "Каша с корицей", "Каша с молоком"
]

product_types = [
    "Фрукты", "Овощи", "Мясо", "Молочные продукты",
    "Крупы", "Консервы", "Сладости", "Напитки",
    "Хлебобулочные изделия", "Специи", "Замороженные продукты",
    "Косметика", "Бытовая химия", "Здоровое питание", "Готовая еда"
]

conn = psycopg2.connect(
    dbname="postgres",
    user="postgres",
    password="2022",
    host="localhost",
    port="5432"
)

cur = conn.cursor()

for _ in range(1100):
    name = random.choice(product_names)
    product_type = random.choice(product_types)
    weight = random.randint(100, 5000)
    price = round(random.uniform(1.0, 100.0), 2)
    store_id = random.randint(1, 10)
    insert_query = """
        INSERT INTO products (name, type, weight, price, store_id)
        VALUES (%s, %s, %s, %s, %s);
    """
    cur.execute(insert_query, (name, product_type, weight, price, store_id))

conn.commit()
cur.close()
conn.close()
```

- Код заполнения таблицы dishes

```
import random
from faker import Faker
import psycopg2
fake = Faker()
dishes_names = [
    "Паста карбонара", "Суши", "Пицца Маргарита", "Бургер с говядиной",
    "Том Ям", "Цезарь с курицей", "Ризотто с грибами", "Курица терияки",
    "Лазанья", "Греческий салат", "Фахитас", "Крем-брюле", "Тарт Татен",
    "Шашлык", "Салат Оливье", "Фондю", "Куриный суп", "Блинчики с мясом",
    "Котлеты по-киевски", "Кабачковые оладьи", "Пирог с картошкой",
    "Салат из тунца", "Мусака", "Пельмени", "Суп-пюре из тыквы",
    "Стейк из говядины", "Запеченная рыба", "Борщ", "Суп с фрикадельками",
    "Курица по-французски", "Пирожки с вишней", "Роллы с лососем",
    "Куриные крылышки", "Капрезе", "Теплый салат с курицей",
    "Баклажаны по-баклажански", "Овощное рагу", "Каша овсяная",
    "Салат с авокадо", "Куриный шашлык", "Крем-суп из шпината",
    "Сырники", "Омлет с помидорами", "Кабачки, запеченные с сыром",
    "Торт Наполеон", "Пирог с яблоками", "Кексы", "Торт медовик",
    "Теплый салат с говядиной", "Паста с соусом песто", "Салат с креветками",
    "Тортилья с курицей", "Пирожное эклер", "Печеный картофель",
    "Суп с грибами", "Рагу из говядины", "Каша гречневая",
    "Паста с морепродуктами", "Курица с лимоном", "Блины с творогом",
    "Тосты с авокадо", "Суп с лапшой", "Рыбные котлеты",
    "Салат с курицей и ананасами", "Творожная запеканка",
    "Пирог с капустой", "Салат из свеклы", "Овощи на гриле",
    "Паста с томатным соусом", "Пирожные с заварным кремом",
    "Курица с грибами", "Печеные овощи", "Салат с тунцом и фасолью",
    "Фриттата", "Торт с малиной", "Паста с креветками",
    "Суп из чечевицы", "Курица с медом", "Салат с гречкой",
    "Буррито", "Салат с кукурузой", "Котлеты из индейки",
    "Каша рисовая", "Пирожки с картошкой", "Суп с брокколи",
    "Запеканка из картофеля", "Салат с курицей и грибами",
    "Творожные оладьи", "Суп из морепродуктов", "Блины с икрой",
    "Торт с орехами", "Салат с фасолью и овощами", "Суп с овощами",
    "Курица с апельсинами", "Пирог с мясом", "Паста с шпинатом",
    "Каша манная", "Салат с редисом", "Котлеты по-домашнему",
    "Суп с картошкой", "Запеченные яблоки", "Блинчики с творогом",
    "Салат из морепродуктов", "Ризотто с морепродуктами",
    "Пирожное картошка", "Куриные наггетсы", "Запеканка из макарон",

    "Салат с морепродуктами и авокадо", "Котлеты из индейки с овощами",
    "Блины с творожным кремом", "Суп с картошкой и грибами",
    "Запеканка из картофеля с мясом", "Пирог с ягодами",
    "Салат с курицей и авокадо", "Каша с яблоками",
    "Суп с овощами и мясом", "Котлеты из рыбы с картошкой",
    "Бургер с овощами", "Салат с помидорами и огурцами",
    "Паста с томатами и базиликом", "Торт с малиной",
    "Рагу из говядины с картошкой", "Каша с ягодами и медом",
    "Суп с грибами и зеленью", "Пирожное с кремом и ягодами",
    "Салат с курицей и грибами", "Котлеты из картофеля с сыром",
    "Блины с вареньем", "Суп с чечевицей и овощами",
    "Запеканка из кабачков с мясом", "Пирог с мясом и овощами",
    "Салат с рыбой и картошкой", "Каша с медом и орехами",
    "Суп с картошкой и зеленью"
]
conn = psycopg2.connect(
    dbname="postgres",
    user="postgres",
    password="2022",
    host="localhost",
    port="5432"
)
cur = conn.cursor()
for _ in range(1100):
    name = random.choice(dishes_names)
    recipe = fake.text(max_nb_chars=200)
    weight = random.randint(100, 2000)
    price = round(random.uniform(5.0, 50.0), 2)
    restaurant_id = random.randint(1, 50)

    # SQL-запрос для вставки данных
    insert_query = """
        INSERT INTO dishes (name, recipe, weight, price, restaurant_id)
        VALUES (%s, %s, %s, %s, %s);
    """
    cur.execute(insert_query, (name, recipe, weight, price, restaurant_id))
conn.commit()
cur.close()
conn.close()
```

- Код заполнения таблицы user_promocode

```
import psycopg2
import random

conn = psycopg2.connect(
    dbname="postgres",
    user="postgres",
    password="2022",
    host="localhost",
    port="5432"
)

cur = conn.cursor()

cur.execute("SELECT id FROM users;")
user_ids = [row[0] for row in cur.fetchall()]

cur.execute("SELECT id FROM promocodes;")
promocode_ids = [row[0] for row in cur.fetchall()]

for _ in range(15000):
    user_id = random.choice(user_ids)
    promocode_id = random.choice(promocode_ids)

    insert_query = """
    INSERT INTO user_promocode (user_id, promocode_id)
    VALUES (%s, %s);
    """

    cur.execute(insert_query, (user_id, promocode_id))
conn.commit()
cur.close()
conn.close()
```

- Код заполнения таблицы collections

```
import psycopg2
from faker import Faker
import random
conn = psycopg2.connect(
    dbname="postgres",
    user="postgres",
    password="2022",
    host="localhost",
    port="5432"
)
cur = conn.cursor()
fake = Faker()
food_themes = [
    'Итальянская кухня',
    'Мексиканская кухня',
    'Японская кухня',
    'Французская кухня',
    'Индийская кухня',
    'Китайская кухня',
    'Греческая кухня',
    'Американская кухня',
    'Испанская кухня',
    'Бразильская кухня'
]
for _ in range(50):
    theme = random.choice(food_themes)
    creation_date = fake.date_this_decade()

    insert_query = """
    INSERT INTO collections (name, theme, creation_date)
    VALUES (%s, %s, %s);
    """
    cur.execute(insert_query, (name, theme, creation_date))
conn.commit()
cur.close()
conn.close()
```


- Заполнение таблицы restaurants

```
import psycopg2
from faker import Faker
import random
DB_NAME = 'postgres'
DB_USER = 'postgres'
DB_PASSWORD = '2022'
DB_HOST = 'localhost'
DB_PORT = '5432'
fake = Faker()

# Список фейковых названий ресторанов
restaurant_names = [
    "The Gourmet Kitchen", "Savory Bites", "Culinary Delights", "Taste of Italy",
    "Spice Route", "Ocean's Catch", "Farm to Table", "Urban Bistro",
    "The Green Plate", "Flavors of Asia", "Rustic Table", "The Hungry Chef",
    "Sweet Tooth Café", "Herb & Spice", "The Cozy Corner", "Dine Divine",
    "Sizzle & Grill", "The Social Fork", "Epicurean Escape", "Charming Eats",
    "Café Aroma", "Bistro Bliss", "The Rustic Spoon", "Gastronomy Haven",
    "Taste Buds", "Fusion Flavors", "The Food Lab", "Savory Seasons",
    "The Local Diner", "Elegant Eats", "Culinary Canvas", "The Flavor Factory",
    "Artisan Eatery", "The Dining Room", "Taste & Toast", "The Savory Chef",
    "Bite Me Bistro", "The Foodie Spot", "Harvest Table", "The Spice Rack",
    "The Culinary Studio", "Whisk & Ladle", "The Flavor Vault", "The Urban Plate",
    "Global Grub", "The Tasty Table", "Bite & Brew", "The Flavor Kitchen",
    "Gourmet Garden", "The Chef's Table", "Café Delights", "Gastronomic Galore",
    "The Flavor House", "Taste of Home", "The Dining Experience", "Savory Sensations",
    "The Artisan Kitchen", "The Flavor Journey", "The Culinary Corner", "Taste Explorers",
    "The Flavor Bar", "The Gourmet Spot", "The Taste Lab", "The Flavor Society",
    "The Dining Hub", "Café Culture", "The Flavor Workshop", "The Epicurean Table",
    "The Savory Experience", "The Flavor District", "The Culinary Voyage", "The Taste Collective",
    "The Flavor Haven", "The Gourmet Plaza", "The Flavor Oasis", "The Culinary Escape",
    "The Taste House", "The Flavor Lounge", "The Savory Kitchen", "The Culinary Retreat",
    "The Flavor Bistro", "The Taste Haven", "The Culinary Oasis", "The Flavor Nook",
    "The Savory Spot", "The Culinary Hideaway", "The Flavor Retreat", "The Taste Studio",
    "The Culinary Nook", "The Flavor Cove", "The Savory Retreat", "The Taste Haven",
    "The Culinary Oasis", "The Flavor Den", "The Savory Lounge", "The Taste Retreat",
    "The Culinary Sanctuary", "The Flavor Sanctuary", "The Savory Sanctuary", "The Taste Sanctuary",
]
```

```
def insert_fake_restaurants():
    try:
        connection = psycopg2.connect(
            dbname=DB_NAME,
            user=DB_USER,
            password=DB_PASSWORD,
            host=DB_HOST,
            port=DB_PORT
        )
        cursor = connection.cursor()

        for _ in range(100):
            name = random.choice(restaurant_names)
            rating = round(random.uniform(0, 5), 1)
            theme = fake.word().capitalize()
            review_link = fake.url()
            address_id = random.randint(1, 100)
            insert_query = """
            INSERT INTO restaurants (name, rating, theme, review_link, address_id)
            VALUES (%s, %s, %s, %s, %s);
            """
            cursor.execute(insert_query, (name, rating, theme, review_link, address_id))
        connection.commit()
    except Exception as e:
        print(f"Произошла ошибка: {e}")
    finally:
        if connection:
            cursor.close()
            connection.close()
insert_fake_restaurants()
```

Во всех запросах :

```
conn = psycopg2.connect(  
    dbname="postgres",  
    user="postgres",  
    password="2022",  
    host="localhost",  
    port="5432"  
)
```

- подключение

```
for _ in range(100):  
    full_name = fake.name()  
    registration_date = fake.date_between(start_date='-1y', end_date='today')  
    login = fake.user_name()  
    password = fake.password()  
  
    cur.execute("""  
    INSERT INTO users (full_name,registration_date,password,login)  
    VALUES (%s,%s,%s,%s);  
    """, (full_name,registration_date,password,login))
```

- заполнение в цикле

```
# Получение всех существующих user_id и promocode_id  
cur.execute("SELECT id FROM users;")  
user_ids = [row[0] for row in cur.fetchall()]  
  
cur.execute("SELECT id FROM promocodes;")  
promocode_ids = [row[0] for row in cur.fetchall()]
```

- отбор данных

```
streets = ["Main Street", "Maple Avenue", "Oak Lane", "Pine Road", "Cedar Drive", "Elm Street", "Birch Way", "Sunset Boulevard", "River Street",  
    "Hilltop Avenue", "Lakeview Drive", "Cherry Blossom Lane", "Willow Way", "Broadway", "Park Avenue", "Spring Street", "Country Road",  
    "West End Drive", "South Hill Road", "Northgate Street", "Greenwood Avenue", "Mountain View Road", "Seaside Boulevard", "Forest Lane",  
    "Ocean Drive", "Crescent Road", "Lakeshore Drive", "Valley View Street", "Sunrise Avenue", "Meadow Lane"]
```

- массивы сгенерированных данных

4.2.3 Результаты заполнения

Далее представлены результаты работы программы на примере таблиц:

Таблица users

	id [PK] integer	full_name character varying (255)	registration_date timestamp without time zone	password character varying (255)	login character varying (255)
34	50	Jonathan Johnson	2024-04-11 00:00:00	G7u1cKhqu#	martha35
35	51	Kimberly Cameron	2024-06-08 00:00:00	J7Zyi@8N*M	zacharyfarmer
36	52	Jeremy King	2024-01-10 00:00:00	YO+M1Xoetb	jacksonapril
37	53	Mr. Erik Sherman	2024-01-20 00:00:00	(7vV\$Nigw5	michellerodriguez
38	54	Amy Hill	2024-02-09 00:00:00	(CXrubQQ07	kristinburns
39	55	Kimberly Thompson	2024-01-09 00:00:00	%uq3Ejzh*a	roberthenderson
40	56	Richard Patel	2024-10-16 00:00:00	e@lL8hqT!7	hfrank
41	57	Timothy James	2024-05-12 00:00:00	(#*0M0mx6c	melissajimenez
42	58	Christine Kane	2024-05-20 00:00:00	d07n7TSb+c	ckemp
43	59	Larry Burgess	2024-07-02 00:00:00	7GN2*eTp\$^	fergusonshannon
44	60	Mary Keller	2024-05-24 00:00:00	r\$t8lFBvCY	romeroearl
45	61	Timothy Graham	2023-11-21 00:00:00	%20SnDr1bV	jerryjohnson

Таблица products

	id [PK] integer	name character varying (100)	type character varying (100)	weight integer	price numeric (10,2)	store_id integer
4409	4409	Коричневый рис	Зерновые	1000	1.79	7
4410	4410	Оливковое масло	Приправы	500	5.99	9
4411	4411	Яйца от кур свободног...	Молочные продукты	600	3.29	11
4412	4412	Киноа	Зерновые	500	4.99	12
4413	4413	Свежий лосось	Рыба	300	14.99	14
4414	4414	Нут	Бобовые	400	1.29	15
4415	4415	Сладкий картофель	Овощи	800	0.79	13
4416	4416	Мед	Приправы	250	6.99	18
4417	4417	Греческий сыр фета	Молочные продукты	200	3.49	19
4418	4418	Сауэрдоу хлеб	Выпечка	400	4.49	20
4419	4419	Смешанные орехи	Сладости	200	8.99	17
4420	4420	Травяной чай	Напитки	100	2.99	16

Таблица restaurants

	id [PK] integer	name character varying (100)	rating numeric (2,1)	theme character varying (100)	review_link character varying (255)	address_id integer
66	66	White-Jones	1.4	Mexican	http://nguyen-hopkins.net/	20
67	67	Kaiser-Douglas	4.6	Chinese	https://sharp.com/	76
68	68	Wilson-Underwood	2.5	American	http://burgess-lin.com/	10
69	69	Romero-Ramirez	2.0	Chinese	http://www.allison.info/	34
70	70	Rodriguez Inc	1.5	Italian	https://ramos.net/	103
71	71	Fowler LLC	3.7	Indian	http://www.roberts-bennett.com/	78
72	72	Horton and Sons	3.2	Italian	http://www.garcia.info/	121
73	73	Weeks, Snyder and Henderson	1.3	Chinese	http://www.alvarez-allen.net/	40
74	74	Stevens-Day	0.5	Mexican	http://thompson.com/	136
75	75	King and Sons	3.6	Indian	https://www.young-perez.org/	69
76	76	Lane, Soto and Hart	0.5	American	https://nolan.org/	20
77	77	Cunningham-Church	4.0	Chinese	https://www.mitchell.com/	126

Таблица orders

	id	order_number character varying (50)	creation_date timestamp without time zone	user_id integer	payment_method_id integer	courier_id integer	total_amount numeric (10,2)	order_status character varying (20)	close_date timestamp without time zone
33	55	76c48e54-2dd4-4b62-ab7f-d5789ab5fdca	2024-01-27 06:13:40	110	12	40	4682.18	canceled	2024-01-27 07:08:15.280138
34	56	3576a403-7f94-48d8-b278-b903451a27a2	2024-08-21 02:42:32	130	22	54	3658.57	pending	2024-08-21 04:14:21.88649
35	57	99519b7b-8791-49fd-b103-4a3e919920c8	2024-05-29 22:03:02	125	29	60	3102.53	completed	2024-05-29 22:07:54.459475
36	58	60541e16-49fc-42e5-a5f4-69917f1e3d51	2024-06-02 09:41:51	49	19	22	4596.30	canceled	2024-06-02 10:19:47.443995
37	59	9d4b1137-b171-4710-85dc-5213abb87a...	2024-05-13 18:42:33	75	14	18	4866.17	canceled	2024-05-13 19:37:48.23464
38	60	05420693-aa56-472b-ad61-d24e1e3364...	2024-10-20 22:43:28	118	15	56	3582.54	pending	2024-10-21 00:33:44.506825
39	61	7dc28155-4249-4ec0-bdd9-79678a3537...	2024-06-20 13:46:49	46	12	15	2509.07	pending	2024-06-20 15:18:14.354342
40	487	fc9ef22a-2710-4fce-ad35-cab5f2b4149c	2024-06-21 05:49:55	102	27	5	4191.50	canceled	2024-06-21 06:23:59.323551
41	488	a001ad24-c043-4ea9-91cb-ae833ab837a4	2024-04-28 02:17:08	49	25	35	2591.14	canceled	2024-04-28 03:51:44.251025
42	489	d47d9837-5c40-4dc8-96e4-6f113622fa48	2024-05-02 14:16:18	100	20	35	1209.41	canceled	2024-05-02 14:58:41.705855
43	490	97f50ae4-36c5-42a6-82c4-da78a6750f68	2024-07-23 08:25:30	25	14	33	153.58	pending	2024-07-23 09:32:17.791524
44	491	32b50613ff44-125f-8011-13b104...	2024-06-02 02:04:03	105	26	20	4570.74	pending	2024-06-02 02:06:00.001705




	id [PK] integer 	order_id integer 	dish_id integer 
19	19	1089	360
20	20	595	401
21	21	613	491
22	22	850	983
23	23	1158	929
24	24	1333	36
25	25	14	878
26	26	1412	977
27	27	868	104
28	28	691	571
29	29	719	643
30	30	1460	473

Таблица order_dish

Таблица payments





	id [PK] integer 	card_number character varying (16) 	payment_type character varying (50) 	user_id integer 
13	18	3565971455805021	Банковский перевод	49
14	19	38249953660494	Кредитная карта	91
15	20	30327071992698	Кредитная карта	29
16	21	30394911924562	Банковский перевод	37
17	22	4637171285089412	Кредитная карта	50
18	23	4913091326993125	Пейпал	98
19	24	630488760368	Дебетовая карта	21
20	25	3550368590443387	Пейпал	128
21	26	4768633210131	Кредитная карта	90
22	27	502018012315	Банковский перевод	53
23	28	4982403662794312	Пейпал	105
24	29	2616821201278562	Дебетовая карта	56

Таблица promocodes

	id [PK] integer	creation_date date	expiration_date date	discount_in_rubles numeric (10,2)	discount_in_percent numeric (5,2)	name character varying (100)	collection_id integer
1	1	2024-10-26	2024-11-02	631.70	18.53	Magazine	4
2	2	2024-04-25	2024-05-22	270.94	56.61	One	44
3	3	2024-07-06	2024-07-12	733.79	49.40	Today	25
4	4	2024-06-09	2024-07-03	871.74	79.74	Strategy	18
5	5	2024-07-06	2024-08-03	708.27	28.73	Matter	18
6	6	2024-06-24	2024-07-01	328.13	8.59	Expect	34
7	7	2024-04-03	2024-04-23	743.23	71.67	Reach	38
8	8	2024-07-01	2024-07-23	41.38	61.16	Million	46
9	9	2024-01-08	2024-01-27	545.79	68.82	Man	23
10	10	2024-07-14	2024-07-21	541.43	93.65	Hundred	4
11	11	2023-11-24	2023-12-16	542.70	80.78	Training	10
12	12	2024-04-27	2024-04-29	311.92	7.97	Southern	17

Таблица user_promocode

	id [PK] integer	user_id integer	promocode_id integer
36	36	87	37
37	37	62	76
38	38	119	22
39	39	84	89
40	40	39	51
41	41	28	57
42	42	104	37
43	43	28	73
44	44	26	95
45	45	95	79
46	46	125	97
47	47	95	89
48	48	95	89

5. Выполнение запросов

В этом разделе приведены различные запросы к реализованной базе данных — их краткие описания, непосредственно запрос на языке SQL и результат выполнения.

1. Топ-100 пользователей с наибольшим количеством заказов

```
1 SELECT u.id, u.full_name, COUNT(o.id) AS order_count
2 FROM users u
3 JOIN orders o ON u.id = o.user_id
4 GROUP BY u.id, u.full_name
5 ORDER BY order_count DESC
6 LIMIT 100;
```

	id [PK] integer	full_name character varying (255)	order_count bigint
1	57	Timothy James	180
2	130	Kristen Parsons	174
3	23	Екатерина Васильева	174
4	105	Andrew Alvarado	168
5	85	Melinda Coleman	167
6	41	Ashley Riley	167
7	35	Mark Riddle	166
8	106	Erika Ford	166
9	43	Pamela Johnson	163

2. Рестораны с их средним рейтингом и количеством блюд, отсортированные по среднему рейтингу

```
8 SELECT r.id, r.name, AVG(r.rating) AS average_rating, COUNT(d.id) AS dish_count
9 FROM restaurants r
10 LEFT JOIN dishes d ON r.id = d.restaurant_id
11 GROUP BY r.id, r.name
12 ORDER BY average_rating DESC;
```

	id [PK] integer	name character varying (100)	average_rating numeric	dish_count bigint
1	87	Black-Welch	5.0000000000000000	8
2	43	Holland, Zhang and Miller	4.9000000000000000	56
3	90	Trevino-Rhodes	4.8000000000000000	10
4	81	Chang, Jacobs and Johnson	4.8000000000000000	12
5	62	Harris, Haynes and Phelps	4.8000000000000000	9
6	52	Conrad-Taylor	4.7000000000000000	7
7	57	Banks Ltd	4.7000000000000000	10
8	7	Garcia Ltd	4.7000000000000000	53
9	67	Kaiser-Douglas	4.6000000000000000	11

3. Курьеры и количество заказов, которые они выполнили, отсортированные по количеству заказов

```
14 SELECT c.id, c.full_name, COUNT(o.id) AS order_count
15 FROM couriers c
16 LEFT JOIN orders o ON c.id = o.courier_id
17 GROUP BY c.id, c.full_name
18 ORDER BY order_count DESC;
```

	id [PK] integer	full_name character varying (100)	order_count bigint
1	66	Jenny Gray	250
2	16	Молодой Траппа	246
3	13	Елена Григорьева	241
4	8	Иван Ковалев	241
5	21	Brenda White	241
6	25	Charles Clark	240
7	49	Ralph Pitts	237
8	44	Renee Harris	237
9	36	Thomas Rogers	236

4. Получить список промокодов с количеством использований и их скидками, отсортированный по количеству использований

```
20 SELECT p.id, p.name, COUNT(up.id) AS usage_count,
21 COALESCE(p.discount_in_rubles, 0) AS discount_in_rubles,
22 COALESCE(p.discount_in_percent, 0) AS discount_in_percent
23 FROM promocodes p
24 LEFT JOIN user_promocode up ON p.id = up.promocode_id
25 GROUP BY p.id, p.name, p.discount_in_rubles, p.discount_in_percent
26 ORDER BY usage_count DESC;
```

	id [PK] integer	name character varying (100)	usage_count bigint	discount_in_rubles numeric	discount_in_percent numeric
1	71	Cause	191	599.21	71.93
2	57	Recent	177	513.71	68.07
3	55	Agent	177	117.09	45.27
4	70	Head	175	687.19	42.61
5	30	Every	174	799.41	22.65
6	97	Want	174	317.43	23.85
7	60	Court	172	874.99	22.45
8	5	Matter	172	708.27	28.73
9	43	Enough	171	528.74	95.35

5. Получить пользователей, которые использовали промокод хотя бы 1 раз вместе со средней суммой их заказов

```
31 SELECT u.id, u.full_name, AVG(o.total_amount) AS average_order_amount
32 FROM users u
33 JOIN user_promocode up ON u.id = up.user_id
34 JOIN orders o ON u.id = o.user_id
35 GROUP BY u.id, u.full_name;
```

	id [PK] integer	full_name character varying (255)	average_order_amount numeric
1	87	Jessica Kent	2504.5308280254777070
2	51	Kimberly Cameron	2470.7050322580645161
3	70	Tommy Ray	2768.9083333333333333
4	22	Дмитрий Смирнов	2575.8486986301369863
5	42	Sara Wright	2621.5500000000000000
6	117	Christopher Petty	2457.3160869565217391
7	113	John Guerrero	2373.7035036496350365
8	40	Mark Wolfe	2495.6737500000000000
9	125	Michael Brown	2764.2548076923076923

6. Получить курьеров, которые выполнили заказы на сумму более 10000 рублей, с количеством выполненных заказов

```
37 SELECT c.id, c.full_name, COUNT(o.id) AS order_count
38 FROM couriers c
39 JOIN orders o ON c.id = o.courier_id
40 GROUP BY c.id, c.full_name
41 HAVING SUM(o.total_amount) > 10000;
```

	id [PK] integer	full_name character varying (100)	order_count bigint
1	51	Heather Brennan	198
2	70	Christopher Ross	184
3	22	Chad Hernandez	236
4	42	Michael Brooks	228
5	40	Jeffrey Cooper	235
6	43	Damon Clark	219
7	57	Leslie Robertson	211
8	19	Lauren Bell	218
9	54	James Sweeney	229

7. Получить рестораны с наибольшим количеством уникальных блюд и их средним рейтингом, отсортированные по количеству блюд

```
43 SELECT r.id, r.name, COUNT(DISTINCT d.id) AS unique_dish_count, AVG(r.rating) AS average_rating
44 FROM restaurants r
45 LEFT JOIN dishes d ON r.id = d.restaurant_id
46 GROUP BY r.id, r.name
47 ORDER BY unique_dish_count DESC;
```

	id [PK] integer	name character varying (100)	unique_dish_count bigint	average_rating numeric
1	50	Vaughn, Robinson and Chavez	70	0.500000000000000000
2	19	Banks, Peck and Parks	68	0.400000000000000000
3	38	Strickland Inc	67	1.200000000000000000
4	3	Foster, Green and Giles	66	1.600000000000000000
5	13	Brown LLC	65	2.800000000000000000
6	46	Torres-Jones	64	3.100000000000000000
7	40	Frazier Inc	64	0.500000000000000000
8	41	Madden, Sawyer and Stewart	64	3.100000000000000000
9	18	Munoz-Barnes	64	0.900000000000000000

8. Получить промокоды с их скидками и количеством использований, которые были созданы в последние 30 дней

```
50 SELECT p.id, p.name, COUNT(up.id) AS usage_count,
51 COALESCE(p.discount_in_rubles, 0) AS discount_in_rubles,
52 COALESCE(p.discount_in_percent, 0) AS discount_in_percent
53 FROM promocodes p
54 LEFT JOIN user_promocode up ON p.id = up.promocode_id
55 WHERE p.creation_date >= NOW() - INTERVAL '30 days'
56 GROUP BY p.id, p.name, p.discount_in_rubles, p.discount_in_percent;
```

	id [PK] integer	name character varying (100)	usage_count bigint	discount_in_rubles numeric	discount_in_percent numeric
1	42	Themselves	140	220.67	89.01
2	23	Young	150	396.81	56.72
3	31	Piece	149	675.58	36.29
4	83	Adult	148	525.08	73.78

9. Анализ аномального поведения пользователя и нахождение пользователей для потенциальной блокировки по формуле определения вероятности того, что пользователь бот (резко отличающиеся места заказов, интервалы, наличие большого числа платежных средств, количество отмененных заказов и тд)

```

60 WITH user_order_stats AS (
61     SELECT
62         u.id AS user_id,
63         COUNT(o.id) AS total_orders,
64         COUNT(DISTINCT o.courier_id) AS distinct_couriers,
65         COUNT(DISTINCT o.payment_method_id) AS distinct_payment_methods,
66         COUNT(o.id) FILTER (WHERE o.order_status = 'canceled') AS canceled_orders,
67         COUNT(DISTINCT a.city) AS distinct_cities,
68         MAX(o.creation_date) - MIN(o.creation_date) AS order_time_span
69     FROM
70         users u
71     JOIN
72         orders o ON u.id = o.user_id
73     LEFT JOIN
74         addresses a ON u.id = a.user_id
75     GROUP BY
76         u.id
77 )
78 SELECT
79     user_id,
80     total_orders,
81     distinct_couriers,
82     distinct_payment_methods,
83     canceled_orders,
84     distinct_cities,
85     order_time_span,
86     CASE
87         WHEN total_orders > 20 AND distinct_cities > 5 AND canceled_orders > 5 THEN 'Potential Bot'
88         ELSE 'Normal User'
89     END AS user_type
90 FROM
91     user_order_stats
92 WHERE
93     CASE
94         WHEN total_orders > 2000 AND distinct_cities > 5 AND canceled_orders > 100 THEN 'Potential Bot'
95         ELSE 'Normal User'
96     END = 'Potential Bot';

```

	user_id integer	total_orders bigint	distinct_couriers bigint	distinct_payment_methods bigint	canceled_orders bigint	distinct_cities bigint	order_time_span interval	user_type text
1	24	2224	63	24	1072	16	313 days 23:40:53	Potential Bot
2	29	2312	65	24	1003	17	316 days 04:28:30	Potential Bot
3	31	2352	66	24	992	16	317 days 01:38:59	Potential Bot
4	35	3154	69	24	1368	19	319 days 03:45:11	Potential Bot
5	48	2220	63	24	855	15	316 days 12:46:49	Potential Bot
6	72	2340	66	24	1035	15	319 days 09:22:33	Potential Bot
7	77	2058	64	24	896	14	314 days 23:48:18	Potential Bot
8	84	2106	69	24	1066	13	315 days 12:25:05	Potential Bot
9	89	2054	69	24	923	13	311 days 14:29:12	Potential Bot
10	92	2898	66	24	1332	18	317 days 00:09:16	Potential Bot

10. Анализ активности пользователей (рекурсивно находит цепочки заказов пользователей , находит максимальное число последовательных заказов, общее число заказов , находит средний интервал между заказами)

```

110 WITH RECURSIVE user_orders AS (
111     SELECT
112         user_id,
113         creation_date,
114         ROW_NUMBER() OVER (PARTITION BY user_id ORDER BY creation_date) AS order_seq
115     FROM
116         orders
117 ),
118 order_intervals AS (
119     SELECT
120         u1.user_id,
121         u1.creation_date AS order_date,
122         u2.creation_date AS next_order_date,
123         EXTRACT(EPOCH FROM (u2.creation_date - u1.creation_date)) / 60 AS interval_minutes
124     FROM
125         user_orders u1
126     JOIN
127         user_orders u2 ON u1.user_id = u2.user_id AND u1.order_seq + 1 = u2.order_seq
128 )
129 SELECT
130     user_id,
131     COUNT(order_date) AS total_orders,
132     COUNT(next_order_date) AS consecutive_orders,
133     AVG(interval_minutes) AS average_interval_minutes
134 FROM
135     order_intervals
136 GROUP BY
137     user_id
138 ORDER BY
139     total_orders DESC;

```

	user_id integer 	total_orders bigint 	consecutive_orders bigint 	average_interval_minutes numeric 
1	57	179	179	2542.4353817504655456
2	130	173	173	2612.6975915221580000
3	23	173	173	2620.1342003853564528
4	105	167	167	2721.7955089820359242
5	85	166	166	2748.5651606425702811
6	41	166	166	2751.3829317269076365
7	35	165	165	2785.3647474747474687
8	106	165	165	2742.9044444444444485
9	43	162	162	2827.5000000000000000

11. Статистика по курьерам с учетом производительности и времени выполнения заказов, ранжирование курьеров (находит число выполненных заказов у курьеров , среднее время доставки заказа , ранжирует по этим двум параметрам)

```

142 SELECT
143     c.id AS courier_id,
144     c.full_name,
145     COUNT(o.id) AS total_deliveries,
146     AVG(EXTRACT(EPOCH FROM (o.close_date - o.creation_date))) / 60 AS average_delivery_time
147 FROM
148     couriers c
149 LEFT JOIN
150     orders o ON c.id = o.courier_id
151 GROUP BY
152     c.id
153 HAVING
154     AVG(EXTRACT(EPOCH FROM (o.close_date - o.creation_date))) / 60 >= 0
155 ORDER BY
156     total_deliveries DESC, average_delivery_time ASC;

```



	courier_id integer	full_name character varying (100)	total_deliveries bigint	average_delivery_time numeric
1	66	Jenny Gray	250	62.21373300339999999760
2	16	Молодой Траппа	246	59.16950742046070460434
3	8	Иван Ковалев	241	56.1853825113416321
4	13	Елена Григорьева	241	59.74497634190871369087
5	21	Brenda White	241	59.86434041251728907261
6	25	Charles Clark	240	58.2600877742361111
7	44	Renee Harris	237	58.90150342883263009930
8	49	Ralph Pitts	237	59.46793532510548523249
9	22	Chad Hernandez	236	57.11350981186440677881
10	36	Thomas Rogers	236	60.01186151659604519689
11	68	Lisa Jackson	236	60.51399955903954802133
12	65	Keith Wise	235	58.70482577645390070738

12. Вычисление среднего времени доставки по региону и ранжирование их

```

204 WITH average_times AS (
205     SELECT
206         a.city,
207         AVG(EXTRACT(EPOCH FROM (o.close_date - o.creation_date))) / 60 AS average_delivery_time
208     FROM
209         orders o
210     JOIN
211         couriers c ON o.courier_id = c.id
212     JOIN
213         addresses a ON o.user_id = a.user_id
214     GROUP BY
215         a.city
216     HAVING
217         AVG(EXTRACT(EPOCH FROM (o.close_date - o.creation_date))) / 60 >= 0
218 ),
219 ranked_times AS (
220     SELECT
221         city,
222         average_delivery_time,
223         RANK() OVER (ORDER BY average_delivery_time) AS rn
224     FROM
225         average_times
226 )
227 SELECT
228     DISTINCT ON (average_delivery_time)
229     city,
230     average_delivery_time
231 FROM
232     ranked_times
233 WHERE
234     rn >= 1
235
236     rn >= 1
237     AND EXISTS (
238         SELECT
239             1
240         FROM
241             ranked_times rti
242         WHERE
243             rti.average_delivery_time = ranked_times.average_delivery_time
244             AND rti.rn != ranked_times.rn
245     )
246 ORDER BY
247     average_delivery_time ASC, city ASC;

```

	city character varying (100) 	average_delivery_time numeric 
6	Annetteland	34.2787055900621118
7	Anthonyton	47.2477873417721519
8	Bellside	56.0684920863309353
9	Ashleyfurt	65.5525383561643836
10	North Christina	84.6852681818181818
11	East Michael	86.5236662068965517
12	Hooperborough	93.7695440993788820
13	Johnsonport	99.40342262996941896024
14	Adamside	108.3579677966101695
15	Carterview	110.5241563380281690
16	Dawnview	111.6248197183098592

13. Вычисление любимого ресторана , магазина , блюда , товара.

```
WITH user_orders AS (  
    SELECT  
        user_id,  
        COUNT(*) AS total_orders,  
        COUNT(DISTINCT order_number) AS unique_orders  
    FROM  
        orders  
    GROUP BY
```

```
        o.user_id  
336 ),  
337 favorite_restaurant AS (  
338     SELECT  
339         o.user_id,  
340         d.restaurant_id,  
341         COUNT(d.id) AS dish_count  
342     FROM  
343         orders o  
344     JOIN  
345         order_dish od ON o.id = od.order_id  
346     JOIN  
347         dishes d ON od.dish_id = d.id  
348     GROUP BY  
349         o.user_id, d.restaurant_id  
350 ),  
351 favorite_store AS (  
352     SELECT  
353         o.user_id,  
354         p.store_id,  
355         COUNT(p.id) AS product_count  
356     FROM  
357         orders o  
358     JOIN  
359         order_product op ON o.id = op.order_id  
360     JOIN  
361         products p ON op.product_id = p.id  
362     GROUP BY  
363         o.user_id, p.store_id  
364 ),  
365
```

```
366 favorite_dish AS (  
367     SELECT  
368         o.user_id,  
369         d.id AS dish_id,  
370         COUNT(d.id) AS count  
371     FROM  
372         orders o  
373     JOIN  
374         order_dish od ON o.id = od.order_id  
375     JOIN  
376         dishes d ON od.dish_id = d.id  
377     GROUP BY  
378         o.user_id, d.id  
379 ),  
380 favorite_product AS (  
381     SELECT  
382         o.user_id,  
383         p.id AS product_id,  
384         COUNT(p.id) AS count  
385     FROM  
386         orders o  
387     JOIN  
388         order_product op ON o.id = op.order_id  
389     JOIN  
390         products p ON op.product_id = p.id  
391     GROUP BY  
392         o.user_id, p.id  
393 )  
394
```

```

394
395 SELECT
396     u.id AS user_id,
397     u.full_name,
398     /*COALESCE(TO_CHAR((CAST(r.repeated_order_count AS FLOAT) / NULLIF(uo.total_orders, 0)) * 100, 'FM99999999.00'), '0.00') AS rep
399     (SELECT restaurant_id FROM favorite_restaurant WHERE user_id = u.id ORDER BY dish_count DESC LIMIT 1) AS favorite_restaurant_id,
400     (SELECT store_id FROM favorite_store WHERE user_id = u.id ORDER BY product_count DESC LIMIT 1) AS favorite_store_id,
401     (SELECT dish_id FROM favorite_dish WHERE user_id = u.id ORDER BY count DESC LIMIT 1) AS favorite_dish_id,
402     (SELECT product_id FROM favorite_product WHERE user_id = u.id ORDER BY count DESC LIMIT 1) AS favorite_product_id
403 FROM
404     users u
405 JOIN
406     user_orders uo ON u.id = uo.user_id
407 LEFT JOIN
408     repeated_orders r ON u.id = r.user_id
409 ORDER BY
410     u.id;

```

	user_id integer	full_name character varying (255)	favorite_restaurant_id integer	favorite_store_id integer	favorite_dish_id integer	favorite_product_id integer
23	42	Sara Wright	39	32	74	52
24	43	Pamela Johnson	94	7	77	17
25	44	Dominique Mcdonald	3	54	54	290
26	45	Brent Miller	36	13	5	386
27	46	Matthew Reyes	53	2	4	69
28	47	Francisco Brown	24	10	60	82
29	48	Jamie Smith	38	33	5	56
30	49	Kimberly Wood	72	3	116	25
31	50	Jonathan Johnson	39	43	38	15
32	51	Kimberly Cameron	91	81	53	2
33	52	Jeremy King	53	5	219	92
34	53	Mr. Erik Sherman	63	27	10	829
35	54	Amy Hill	17	3	8	7
36	55	Kimberly Thompson	1	26	1	747