Министерство науки и высшего образования Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ ЯДЕРНЫЙ УНИВЕРСИТЕТ «МИФИ»
(НИЯУ МИФИ)

ИНСТИТУТ ИНТЕЛЛЕКТУАЛЬНЫХ КИБЕРНЕТИЧЕСКИХ СИСТЕМ
КАФЕДРА КИБЕРНЕТИКИ

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

Вариант «Grow Food (Доставка здорового питания)»

Выполнил:
Студент группы Б22-534
Баранов А. Т.
Преподаватель:
Петровская А. В.

Москва, осень 2024

Содержание

Формулировка задания

Спроектировать базу данных для сервиса доставки здорового питания «Grow Food». База данных должна содержать информацию о пользователях, заказах, а также курьерах и поставщиках продуктов.

Концептуальная модель базы данных

После проведения анализа предметной области была спроектирована следующая концептуальная модель:

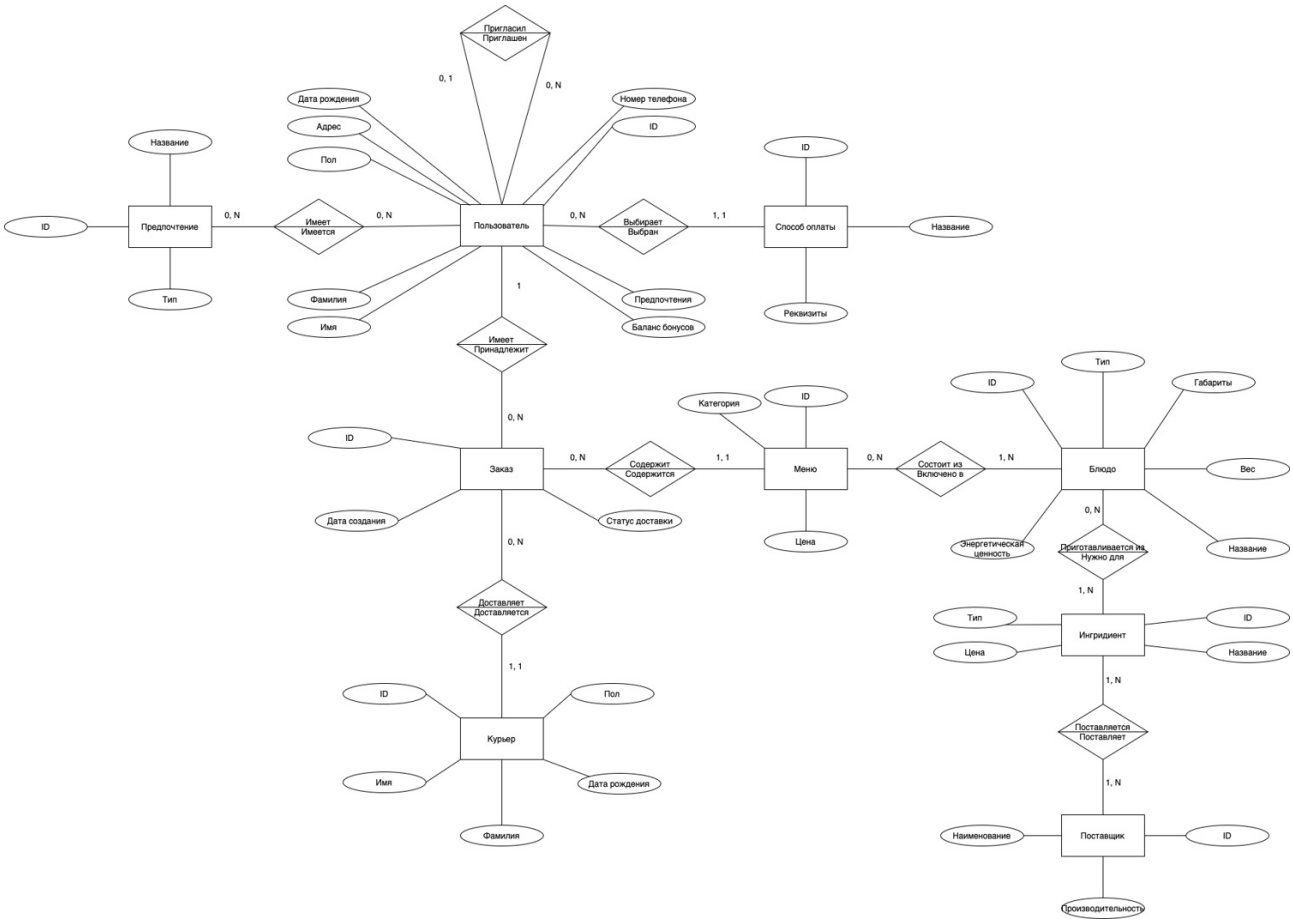


Рис. 1: Концептуальная модель базы данных

Конкретизация предметной области

Необходимо создать систему, отражающую информацию о работе сервиса в сферах: работа с поставщиками, приготовление готовых блюд, составление меню, доставка до

пользователей, оплата. Меню составляется на следующий месяц. База данных должна учитывать предпочтения пользователя по ингредиентам.

Описание предметной области

Система рассчитана на работу с сотрудниками сервиса. Пользователи сервиса доступа к базе данных не имеют.

Каждому пользователю в его личном кабинете доступна информация о выбранных им предпочтениях, расписании меню на следующий месяц, количестве накопленных им баллов, а также предложении оформить заказ.

Курьерам доступна информация о доставленных или доставляемых ими заказах: дата, адрес доставки, способ оплаты и сумма к оплате (при необходимости).

Сотрудники кухни могут воспользоваться информацией о ингредиентах, необходимых для того или иного блюда, а также о самих блюдах: их количестве, размере порции, калорийности.

Описание атрибутов

В процессе анализа были выделены следующие атрибуты, название и описание которых приведены ниже в таблице:

Название атрибута	Описание атрибута
id	Уникальный идентификатор. Есть у каждого объекта
Имя, фамилия, отчество	Имя, фамилия и отчество пользователей сервиса или курьеров, доставляющих заказы
пол	Пол пользователя
Дата рождения	Дата рождения пользователя или курьера
Номер телефона	Российский номер телефона пользователя
id пригласившего пользователя	`id` пользователя, который пригласил данного в сервис

Название атрибута	Описание атрибута
Количество бонусов	Количество бонусов, находящихся на счету у данного пользователя
Адрес	Адрес пользователя для доставки в пределах города Москва
Дата	Дата создания заказа, дата для меню, где указываются доступные наборы блюд на день
Название меню	Название для меню. Например “Похудение”
Цена	Розничная цена меню, закупочная цена ингредиентов
Коллораж	Количество колорий, на которое рассчитано данное меню
Тип	Тип блюда или тип ингридиента
Вес	Вес блюда в граммах
Название поставщика	Название компании или иное имя для организации, поставляющей ингредиенты
Продуктивность	Количество килограмм ингредиента в месяц, которое может быть закуплено у поставщика
Название категории предпочтений	Название для каждой категории предпочтений из некоторого перечня
Название предпочтения	Название для сущности “Предпочтение”, которое выбирает пользователь, чтобы потреблять еду, более ему подходящую
Название способа оплаты	Название для способа оплаты
Реквизиты	Необходимые для данного способа оплаты данные. Например, номер карты, номер счёта или номер чека

Логическое проектирование

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

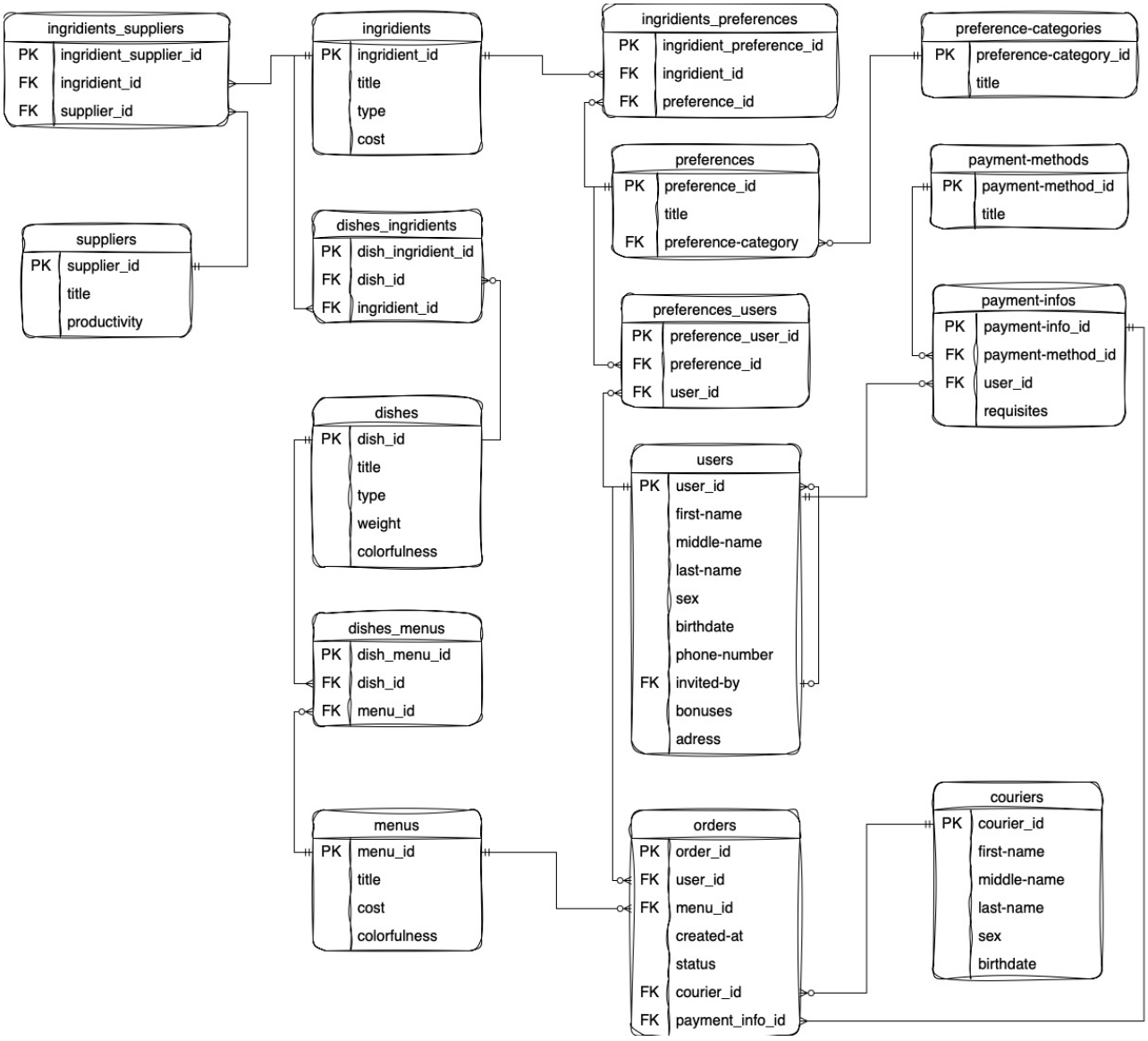


Рис. 2: Логическая модель базы данных

Для соблюдения третьей нормальной формы информация об оплате и категории предпочтений были вынесены в отдельные таблицы. Также отмечены связи “один ко многим”.

Ключевую роль играют таблицы “Пользователь” и “Заказ”, так как в них включены основные сущности сервиса доставки здорового питания: курьеры, предпочтения, оплата и меню.

Таким образом, все таблицы базы данных находятся в третьей нормальной форме.

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:

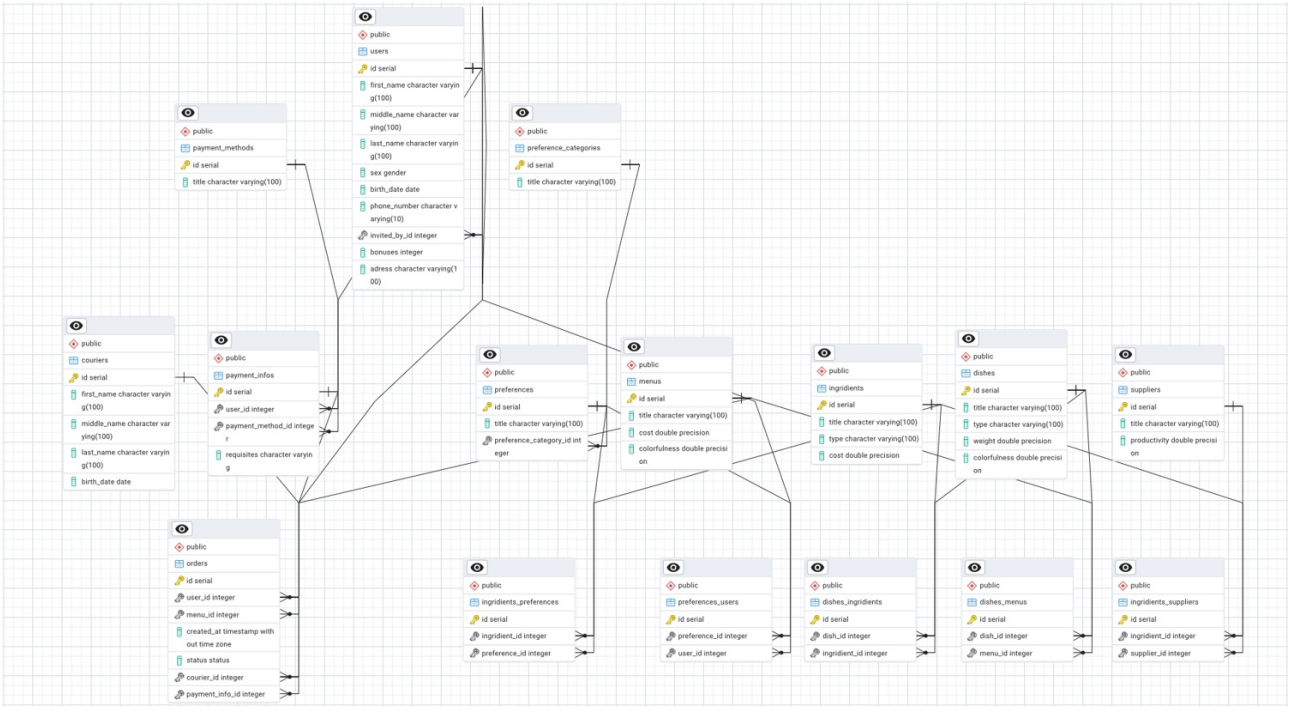


Рис. 3: Графическое представление базы данных

Создание таблиц

Ниже представлен код на языке Python для создания таблиц базы данных. В качестве ORM использована библиотека SQLAlchemy.

- Код для создания таблицы “users”:

```python
class Gender(enum.Enum):
    male = "male"
    female = "female"


class Users(Base):
    __tablename__ = "users"

    id: Mapped[intpk]
    first_name: Mapped[str_100]
    middle_name: Mapped[str_100 | None]
    last_name: Mapped[str_100]
```

```
    sex: Mapped[Gender | None]
    birth_date: Mapped[datetime.date]
    phone_number: Mapped[phone_number]
    invited_by_id: Mapped[int | None] = mapped_column(
        ForeignKey("users.id", ondelete="SET NULL")
    )
    bonuses: Mapped[int] = mapped_column(server_default=text("0"))
    address: Mapped[str_100 | None]
```

- Код для создания таблицы “payment_methods”:

```
class Payment_methods(Base):
    __tablename__ = "payment_methods"
    id: Mapped[intpk]
    title: Mapped[str_100]
    require_requisites: Mapped[bool]
```

- Код для создания таблицы “payment_infos”:

```
class Payment_infos(Base):
    __tablename__ = "payment_infos"
    id: Mapped[intpk]
    user_id: Mapped[int | None] = mapped_column(
        ForeignKey("users.id", ondelete="CASCADE")
    )
    payment_method_id: Mapped[int | None] = mapped_column(
        ForeignKey("payment_methods.id", ondelete="CASCADE",
           server_default="1")
    )
    requisites: Mapped[str] = mapped_column(nullable=True)
```

- Код для создания таблицы “preference_categories”:

```
class Preference_categories(Base):
    __tablename__ = "preference_categories"
    id: Mapped[intpk]
    title: Mapped[str_100]
```

- Код для создания таблицы “preferences”:

```
class Preferences(Base):
    __tablename__ = "preferences"
    id: Mapped[intpk]
    title: Mapped[str_100]
    preference_category_id: Mapped[int] = mapped_column(ForeignKey("
       preference_categories.id", ondelete="CASCADE"))
```

- Код для создания таблицы “couriers”:

```
class Couriers(Base):
```

```
    __tablename__ = "couriers"

    id: Mapped[intpk]
    first_name: Mapped[str_100]
    middle_name: Mapped[str_100 | None]
    last_name: Mapped[str_100]
    birth_date: Mapped[datetime.date]
```

- Код для создания таблицы “menus”:

```
class Menus(Base):
    __tablename__ = "menus"

    id: Mapped[intpk]
    title: Mapped[str_100]
    cost: Mapped[float]
    count_dishes: Mapped[int]
    colorfulness: Mapped[float]
```

- Код для создания таблицы “orders”:

```
class Status(enum.Enum):
    new = "new"
    in_delivery = "in_delivery"
    delivered = "delivered"
    cancelled = "cancelled"
    returned = "returned"

class Orders(Base):
    __tablename__ = "orders"

    id: Mapped[intpk]
    user_id: Mapped[int] = mapped_column(ForeignKey("users.id", ondelete
       ="CASCADE"))
    menu_id: Mapped[int | None] = mapped_column(
        ForeignKey("menus.id", ondelete="SET NULL")
    )
    created_at: Mapped[datetime.datetime] = mapped_column(server_default
       =text("now()"))
    status: Mapped[Status] = mapped_column(server_default=text("'new'"))
    courier_id: Mapped[int | None] = mapped_column(
        ForeignKey("couriers.id", ondelete="SET NULL")
    )
    payment_info_id: Mapped[int | None] = mapped_column(
        ForeignKey("payment_infos.id", ondelete="SET NULL")
    )
```

- Код для создания таблицы “dishes”:

```
class Dishes(Base):
    __tablename__ = "dishes"
```

```
    id: Mapped[intpk]
    title: Mapped[str_100]
    type: Mapped[str_100]
    weight: Mapped[float]
    colorfulness: Mapped[float]
```

- Код для создания таблицы “dishes_menus”:

```
class Dishes_Menus(Base):
    __tablename__ = "dishes_menus"

    id: Mapped[intpk]
    dish_id: Mapped[int] = mapped_column(ForeignKey("dishes.id",
       ondelete="CASCADE"))
    menu_id: Mapped[int] = mapped_column(ForeignKey("menus.id", ondelete
       ="CASCADE"))
    date: Mapped[datetime.date] = mapped_column(server_default="now()")
```

- Код для создания таблицы “ingredients”:

```
class Ingredients(Base):
    __tablename__ = "ingredients"
    id: Mapped[intpk]
    title: Mapped[str_100]
    type: Mapped[str_100]
    cost: Mapped[float]
```

- Код для создания таблицы “dishes_ingredients”:

```
class Dishes_Ingredients(Base):
    __tablename__ = "dishes_ingredients"

    id: Mapped[intpk]
    dish_id: Mapped[int] = mapped_column(ForeignKey("dishes.id",
       ondelete="CASCADE"))
    ingredient_id: Mapped[int] = mapped_column(
        ForeignKey("ingredients.id", ondelete="CASCADE")
    )
```

- Код для создания таблицы “suppliers”:

```
class Suppliers(Base):
    __tablename__ = "suppliers"

    id: Mapped[intpk]
    title: Mapped[str_100]
    productivity: Mapped[float]
```

- Код для создания таблицы “ingredients_suppliers”:

```
class Ingredients_Suppliers(Base):
    __tablename__ = "ingredients_suppliers"

    id: Mapped[intpk]
    ingredient_id: Mapped[int] = mapped_column(
        ForeignKey("ingredients.id", ondelete="CASCADE")
    )
    supplier_id: Mapped[int] = mapped_column(
        ForeignKey("suppliers.id", ondelete="CASCADE")
    )
```

- Код для создания таблицы “ingredients_preferences”:

```
class Ingredients_Preferences(Base):
    __tablename__ = "ingredients_preferences"
    id: Mapped[intpk]
    ingredient_id: Mapped[int] = mapped_column(
        ForeignKey("ingredients.id", ondelete="CASCADE")
    )
    preference_id: Mapped[int] = mapped_column(
        ForeignKey("preferences.id", ondelete="CASCADE")
    )
```

Заполнение базы данных

Заполнение базы данных производилось при помощи библиотек SQLAlchemy, Faker и пакета random для языка программирования Python.

Подготовка данных

Были подготовлены данные о блюдах, ингредиентах, а также связях между ними. Подготовлены данные о связи ингредиентов с предпочтениями пользователей. Данные записывались в формате CSV.

Данные о меню, способах оплаты, категориях предпочтений и самих предпочтениях были взяты с оригинального сайта сервиса доставки здорового питания “Grow Food”.

Придуманы функции для корректной генерации имён в зависимости от пола, генерации дат рождения. Добавлены алгоритмы для реалистичного заполнения данных о заказах, выбранных предпочтениях, бонусах, информации о платежах, расписание меню на следующий месяц.

Программа заполнения базы данных

- Код, заполняющий таблицу “users”:

```
fake = Faker("ru_RU")


def generate_phone_number():
    """Форматирует номертелефонаввиде    10 цифрбезкодастраны   ."""
    return fake.numerify("9#########")


def generate_name_by_gender(gender):
    """Генерация имени, отчестваифамилиивзависимостиотпола      ."""
    if gender == models.Gender.male:
        first_name = fake.first_name_male()
        middle_name = fake.middle_name_male()
        last_name = fake.last_name_male()
    else:
        first_name = fake.first_name_female()
        middle_name = fake.middle_name_female()
        last_name = fake.last_name_female()
    return first_name, middle_name, last_name


def generate_birth_date(min_age=18, max_age=80):
    """Генерация датырожденияв диапазонеот     min_age до max_age
       летназад ."""
    today = datetime.date.today()
    start_date = today.replace(year=today.year - max_age)
    end_date = today.replace(year=today.year - min_age)
    return fake.date_between(start_date=start_date, end_date=end_date)


def seed_users(n=50):
    users = []
    for _ in range(n):
        sex = random.choice([models.Gender.male, models.Gender.female])
        first_name, middle_name, last_name = generate_name_by_gender(sex
           )

        user = models.Users(
            first_name=first_name,
            middle_name=middle_name,
            last_name=last_name,
            sex=sex,
            birth_date=generate_birth_date(),
            phone_number=generate_phone_number(),
            address=fake.street_address() if random.random() < 0.3 else
               None,
            bonuses=0,
        )
        users.append(user)
```

```
    session.bulk_save_objects(users)
    session.commit()

    all_users = session.query(models.Users).all()
    for user in all_users:
        if random.random() < 0.3:  # 30%
           пользователейбудутиметьпригласившего   ""
            user.invited_by_id = random.choice(all_users).id
    session.commit()
```

- Код, заполняющий таблицу “payment_methods”:

```
def seed_payment_methods():
    payment_methods = [
        models.Payment_methods(title="Наличные курьеру",
           require_requisites=False),
        models.Payment_methods(title="Карта курьеру", require_requisites
           =True),
        models.Payment_methods(title="Карта", require_requisites=True),
        models.Payment_methods(title="ЯндексСплит.", require_requisites=
           True),
    ]

    session.bulk_save_objects(payment_methods)
    session.commit()
```

- Код, заполняющий таблицу “payment_infos”:

```
fake = Faker("ru_RU")

def seed_payment_infos():
    users = session.query(models.Users).all()
    payment_methods = session.query(models.Payment_methods).all()

    payment_infos = []
    for user in users:
        for payment_method in payment_methods:
            if random.random() < 0.5:
                continue
            payment_info = models.Payment_infos(
                user_id=user.id,
                payment_method_id=payment_method.id,
                requisites=fake.credit_card_number() if payment_method.
                   require_requisites else None
            )
            payment_infos.append(payment_info)

    session.bulk_save_objects(payment_infos)
    session.commit()
```

- Код, заполняющий таблицу “preference_categories”:

```
def seed_preference_categories():
    categories = [
        "аллергены",
        "десерты, выпечка, сахар",
        "мясо, рыба",
        "овощи, лук, чеснок",
        "гарниры, каши"
    ]
    session.bulk_save_objects([Preference_categories(title=title) for
       title in categories])
    session.commit()
```

- Код, заполняющий таблицу “preferences”:

```
def seed_preferences():
    preferences = [
        {"title": "Без творога", "preference_category_id": 1},
        {"title": "Без орехов", "preference_category_id": 1},
        {"title": "Без меда", "preference_category_id": 1},
        {"title": "Без морепродуктов", "preference_category_id": 1},
        {"title": "Без горчицы", "preference_category_id": 1},
        {"title": "Без шоколада", "preference_category_id": 1},

        {"title": "Без десертов", "preference_category_id": 2},
        {"title": "Без выпечки", "preference_category_id": 2},
        {"title": "Без сэндвичейикруассанов  ", "preference_category_id"
           : 2},
        {"title": "Без белогосахара ", "preference_category_id": 2},

        {"title": "Без свининыиветчины  ", "preference_category_id": 3},
        {"title": "Без красногомяса ", "preference_category_id": 3},
        {"title": "Без рыбы", "preference_category_id": 3},
        {"title": "Без мясаиптицы  ", "preference_category_id": 3},

        {"title": "Без сельдерея", "preference_category_id": 4},
        {"title": "Без грибов", "preference_category_id": 4},
        {"title": "Без стручковойфасоли ", "preference_category_id": 4},
        {"title": "Без брокколи", "preference_category_id": 4},
        {"title": "Без кабачков", "preference_category_id": 4},
        {"title": "Без лука", "preference_category_id": 4},
        {"title": "Без чеснока", "preference_category_id": 4},

        {"title": "Без нута", "preference_category_id": 5},
        {"title": "Без булгура", "preference_category_id": 5},
        {"title": "Без кускуса", "preference_category_id": 5},
        {"title": "Без гречки", "preference_category_id": 5},
        {"title": "Без молочныхкаш ", "preference_category_id": 5}
    ]

    session.bulk_insert_mappings(Preferences, preferences)
    session.commit()
```

- Код, заполняющий таблицу “couriers”:

```
fake = Faker("ru_RU")


def generate_courier_name():
    first_name = fake.first_name_male()
    middle_name = fake.middle_name_male()
    last_name = fake.last_name_male()
    return first_name, middle_name, last_name


def generate_birth_date(min_age=18, max_age=60):
    today = datetime.date.today()
    start_date = today.replace(year=today.year - max_age)
    end_date = today.replace(year=today.year - min_age)
    return fake.date_between(start_date=start_date, end_date=end_date)


def seed_couriers(n=50):
    couriers = []
    for _ in range(n):
        first_name, middle_name, last_name = generate_courier_name()

        courier = models.Couriers(
            first_name=first_name,
            middle_name=middle_name,
            last_name=last_name,
            birth_date=generate_birth_date(),
        )
        couriers.append(courier)

    session.bulk_save_objects(couriers)
    session.commit()
```

- Код, заполняющий таблицу “menus”:

```
def seed_menus():
    menus = [
        models.Menus(title="Похудение", cost=700, count_dishes=3,
           colorfulness=1000),
        models.Menus(title="Баланс", cost=850, count_dishes=4,
           colorfulness=1500),
        models.Menus(title="Набор", cost=1000, count_dishes=5,
           colorfulness=2000),
    ]

    session.bulk_save_objects(menus)
    session.commit()
```

- Код, заполняющий таблицу “orders”:

```
fake = Faker("ru_RU")


def generate_order_created_time(duration):
    now = datetime.datetime.now()
    start_time = now - datetime.timedelta(days=duration)
    return fake.date_time_between(start_date=start_time, end_date=now)


def generate_order_status(created_time):
    now = datetime.datetime.now()
    if now - created_time <= datetime.timedelta(days=1):
        status_weights = {
            models.Status.new: 0.5,
            models.Status.in_delivery: 0.5,
        }
        status = random.choices(
            list(status_weights.keys()), list(status_weights.values()),
               k=1
        )[0]
    elif datetime.timedelta(days=1) < now - created_time <= datetime.
       timedelta(days=3):
        status_weights = {models.Status.in_delivery: 0.1, models.Status.
           delivered: 0.9}
        status = random.choices(
            list(status_weights.keys()), list(status_weights.values()),
               k=1
        )[0]
    else:
        status = models.Status.delivered

    if status == models.Status.delivered and random.random() < 0.05:
        status = models.Status.returned

    if random.random() < 0.05:
        status = models.Status.cancelled

    return status


def seed_orders(n=100, duration=30):
    users = session.query(models.Users).all()
    menus = session.query(models.Menus).all()
    couriers = session.query(models.Couriers).all()
    payment_infos = (
        session.query(models.Payment_infos)
        .filter(models.Payment_infos.user_id == models.Users.id)
        .all()
    )

    orders = []
    for _ in range(n):
        user = random.choice(users)
```

```
        menu = random.choice(menus)
        courier = random.choice(couriers)
        payment_info = random.choice(payment_infos)
        created_at = generate_order_created_time(duration)

        order = models.Orders(
            user_id=user.id,
            menu_id=menu.id,
            created_at=created_at,
            status=generate_order_status(created_at),
            courier_id=courier.id,
            payment_info_id=payment_info.id,
        )

        user.bonuses += menu.cost * 0.05
        user.address = fake.street_address()

        orders.append(order)

    session.bulk_save_objects(orders)
    session.commit()
    session.flush()

    session.commit()
```

- Код, заполняющий таблицу “dishes”:

```
def seed_dishes():
    file_path = os.path.join(os.path.dirname(__file__), "csv", "dishes.
       csv")
    with open(file_path, "r") as file:
        reader = csv.DictReader(file)
        dishes = []

        for row in reader:
            dish = Dishes(
                title=row["title"], type=row["type"], weight=float(row["
                   weight"]), colorfulness=float(row["colorfulness"]),
            )
            dishes.append(dish)

    session.bulk_save_objects(dishes)
    session.commit()
```

- Код, заполняющий таблицу “dishes_menus”:

```
def seed_dishes_menus():
    # Задаемдиапазондатнамесяцвперед
    today = datetime.date.today()
    date_range = [today + datetime.timedelta(days=i) for i in range(30)]

    # Получаемвсеменю
```

```
    menus = session.query(Menus).all()
    values = []
    for date in date_range:
        for menu in menus:
            selected_dishes = []
            colorfulness_accumulated = 0
            # Поканевыбранонужноеколичествоблюддляменю
            while len(selected_dishes) < menu.count_dishes:
                # Выбираемслучайноеблюдо
                dishes = (
                    session.query(Dishes)
                    .where(
                        Dishes.colorfulness
                        <= menu.colorfulness - colorfulness_accumulated
                    )
                    .all()
                )
                dish = random.choice(dishes)

                # Проверяем, укладываетсяблюдовпараметрыменю
                if colorfulness_accumulated + dish.colorfulness <= menu.
                   colorfulness:
                    selected_dishes.append(dish)
                    colorfulness_accumulated += dish.colorfulness

            # Добавляемвыбранныеблюдав    dishes_menus
            for dish in selected_dishes:
                value = Dishes_Menus(dish_id=dish.id, menu_id=menu.id,
                   date=date)
                values.append(value)
    session.bulk_save_objects(values)
    session.commit()
```

- Код, заполняющий таблицу “ingredients”:

```
def seed_ingredients():
    file_path = os.path.join(os.path.dirname(__file__), "csv", "
       ingredients.csv")
    with open(file_path, "r") as file:
        reader = csv.DictReader(file)
        ingredients = []

        for row in reader:
            ingredient = Ingredients(
                title=row["title"], type=row["type"], cost=float(row["
                   cost"])
            )
            ingredients.append(ingredient)

    session.bulk_save_objects(ingredients)
    session.commit()
```

- Код, заполняющий таблицу “dishes_ingredients”:

```
def seed_dishes_ingredients():
    file_path = os.path.join(os.path.dirname(__file__), "csv", "
       dishes_ingredients.csv")
    with open(file_path, "r") as file:
        reader = csv.DictReader(file)
        values = []

        for row in reader:
            value = Dishes_Ingredients(
                dish_id=row["dish_id"],
                ingredient_id=row["ingredient_id"],
            )
            values.append(value)

    session.bulk_save_objects(values)
    session.commit()
```

- Код, заполняющий таблицу “suppliers”:

```
fake = Faker("ru_RU")


def seed_suppliers(n=10):
    suppliers = [
        Suppliers(title=fake.company(), productivity=random.randint(1,
           200))
        for _ in range(n)
    ]

    session.bulk_save_objects(suppliers)
    session.commit()
```

- Код, заполняющий таблицу “ingredients_suppliers”:

```
def seed_ingredients_suppliers():
    ingredients = session.query(Ingredients).all()
    suppliers = session.query(Suppliers).all()

    ingredients_suppliers = []
    for ingredient in ingredients:
        random_suppliers = random.sample(suppliers, k=random.randint(1,
           5))

        for supplier in random_suppliers:
            ingredients_suppliers.append(
                Ingredients_Suppliers(ingredient_id=ingredient.id,
                   supplier_id=supplier.id)
            )

    session.bulk_save_objects(ingredients_suppliers)
    session.commit()
```

- Код, заполняющий таблицу “ingredients_preferences”:

```
def seed_ingredients_preferences():
    file_path = os.path.join(os.path.dirname(__file__), "csv", "
       ingredients_preferences.csv")
    with open(file_path, "r") as file:
        reader = csv.DictReader(file)
        values = []

        for row in reader:
            value = Ingredients_Preferences(
                ingredient_id=row["ingredient_id"],
                preference_id=row["preference_id"]
            )
            values.append(value)

    session.bulk_save_objects(values)
    session.commit()
```

Результаты заполнения базы данных

После выполнения всех функций заполнения базы данных были получены следующие результаты:

	id	first_name	middle_name	last_name	sex	birth_date	phone_number	invited_by_id	bonuses	address
0	14	Демьян	Ильич	Романов	male	1991-05-11	9419405564	NaN	0	None
1	47	Панфил	Валерьянович	Назаров	male	1993-02-12	9381812017	NaN	0	None
2	1	Фёкла	Владиславовна	Мартынова	female	1982-02-01	9896072920	NaN	128	пр. Леонова, д. 537
3	2	Раиса	Кузьминична	Игнатова	female	1955-05-26	9190322073	NaN	78	пр. Щербакова, д. 114
4	4	Евпраксия	Мироновна	Гуляева	female	2003-05-26	9336371496	NaN	220	алл. Мусы Джалиля, д. 9/9
5	5	Пелагея	Оскаровна	Комарова	female	1960-12-18	9844123848	NaN	232	наб. Докучаева, д. 36 к. 424
6	6	Оксана	Кирилловна	Беспалова	female	1974-06-18	9524441984	NaN	212	ш. Чайковского, д. 8
7	9	Нонна	Олеговна	Лобанова	female	1986-10-17	9268817100	NaN	212	пр. Октября, д. 72 к. 696
8	10	Ираклий	Бенедиктович	Максимов	male	1998-05-26	9920674128	NaN	298	бул. Красина, д. 2 к. 38
9	11	Мариан	Александрович	Стрелков	male	1952-10-06	9966287019	NaN	128	ш. Культуры, д. 3/5 стр. 345
10	15	Тамара	Аркадьевна	Жукова	female	2006-03-19	9333688096	NaN	155	бул. Бригадный, д. 4/6 стр. 1
11	16	Виктория	Валентиновна	Калашникова	female	1963-03-16	9905105073	NaN	270	ул. Громова, д. 6
12	17	Герман	Бенедиктович	Ермаков	male	1952-03-09	9511625218	NaN	185	ул. Юбилейная, д. 1 стр. 34
13	20	Фотий	Валерьевич	Стрелков	male	1947-12-19	9491186390	NaN	178	наб. Аэродромная, д. 65 стр. 36
14	21	Майя	Львовна	Бурова	female	1997-02-02	9504390341	NaN	178	бул. Монтажников, д. 9 к. 856

Рис. 4: Таблица “users”

	id	title	require_requisites
0	1	Наличные курьеру	False
1	2	Карта курьеру	True
2	3	Карта	True
3	4	Яндекс.Сплит	True

Рис. 5: Таблица “payment_methods”

	id	user_id	payment_method_id	requisites
0	1	1	2	2201907139552375
1	2	1	3	349717068130383
2	3	1	4	8118732266029974
3	4	2	3	4643746123709936
4	5	2	4	5148949858501083
5	6	4	3	377704027535519
6	7	5	1	None
7	8	5	2	8197220808101745
8	9	5	4	2704328270012826
9	10	6	1	None
10	11	6	2	370127313355573
11	12	6	3	2201107745986451
12	13	9	1	None
13	14	9	3	6309636871527178
14	15	10	2	2683778571749002

Рис. 6: Таблица “payment_infos”

	id	title
0	1	аллергены
1	2	десерты, выпечка, сахар
2	3	мясо, рыба
3	4	овощи, лук, чеснок
4	5	гарниры, каши

Рис. 7: Таблица “preference_categories”

	id	title	preference_category_id
0	1	Без творога	1
1	2	Без орехов	1
2	3	Без меда	1
3	4	Без морепродуктов	1
4	5	Без горчицы	1
5	6	Без шоколада	1
6	7	Без десертов	2
7	8	Без выпечки	2
8	9	Без сэндвичей и круассанов	2
9	10	Без белого сахара	2
10	11	Без свинины и ветчины	3
11	12	Без красного мяса	3
12	13	Без рыбы	3
13	14	Без мяса и птицы	3
14	15	Без сельдерея	4

Рис. 8: Таблица “preferences”

	id	first_name	middle_name	last_name	birth_date
0	1	Демьян	Викторович	Рожков	1991-04-03
1	2	Измаил	Ефимьевич	Константинов	1988-05-06
2	3	Всеволод	Гертрудович	Бирюков	1965-11-29
3	4	Мокей	Александрович	Маслов	1973-06-16
4	5	Ефим	Иларионович	Селиверстов	1973-09-06
5	6	Станислав	Якубович	Самойлов	1979-04-21
6	7	Лаврентий	Августович	Виноградов	1990-11-01
7	8	Архип	Гурьевич	Егоров	2003-12-15
8	9	Феофан	Феофанович	Нестеров	1993-06-02
9	10	Лукьян	Валентинович	Хохлов	1972-08-20
10	11	Эраст	Аверьянович	Данилов	1979-12-08
11	12	Василий	Ефимьевич	Калашников	1991-11-25
12	13	Вацлав	Ермилович	Гришин	1990-07-20
13	14	Пимен	Эдгардович	Абрамов	1992-07-18
14	15	Никодим	Эдуардович	Макаров	1967-03-18

Рис. 9: Таблица “couriers”

	id	title	cost	count_dishes	colorfulness
0	1	Похудение	700.0	3	1000.0
1	2	Баланс	850.0	4	1500.0
2	3	Набор	1000.0	5	2000.0

Рис. 10: Таблица “menus”

	id	user_id	menu_id	created_at	status	courier_id	payment_info_id
0	1	1563	1	2024-08-23 02:38:24.791040	delivered	914	4770
1	2	3793	3	2024-10-24 13:24:12.374363	delivered	128	20906
2	3	6084	3	2024-10-29 18:55:40.730061	delivered	262	13826
3	4	14926	1	2024-10-18 06:45:32.744788	delivered	765	23951
4	5	1731	3	2024-11-03 02:24:43.124879	delivered	92	8290
5	6	1499	3	2024-11-03 09:22:10.711858	delivered	594	7856
6	7	10796	2	2024-08-25 11:24:10.927954	delivered	28	27027
7	8	6288	1	2024-09-06 02:24:32.935803	delivered	987	13893
8	9	12789	2	2024-10-14 04:27:13.514588	delivered	200	538
9	10	8749	2	2024-11-14 13:53:48.831527	delivered	938	11768
10	11	7636	2	2024-11-26 22:15:27.299888	new	503	11547
11	12	5434	1	2024-09-24 10:39:06.321188	returned	624	271
12	13	12614	3	2024-11-14 07:35:27.733649	delivered	570	16965
13	14	12391	2	2024-10-14 04:28:40.513829	delivered	658	24199
14	15	12069	2	2024-08-30 11:19:44.221203	delivered	52	17097

Рис. 11: Таблица “orders”

	id	title	type	weight	colorfulness
0	1	Окрошка	Суп	250.0	150.0
1	2	Борщ	Суп	300.0	250.0
2	3	Щи	Суп	300.0	230.0
3	4	Солянка	Суп	350.0	350.0
4	5	Рассольник	Суп	250.0	220.0
5	6	Пельмени	Второе	350.0	450.0
6	7	Вареники с картошкой	Второе	400.0	400.0
7	8	Жаркое	Второе	450.0	600.0
8	9	Котлеты по-киевски	Второе	300.0	550.0
9	10	Сельдь под шубой	Салат	250.0	300.0
10	11	Оливье	Салат	200.0	320.0
11	12	Винегрет	Салат	180.0	200.0
12	13	Мимоза	Салат	250.0	350.0
13	14	Шуба с рыбой	Салат	220.0	330.0
14	15	Курник	Выпечка	350.0	600.0

Рис. 12: Таблица “dishes”

	id	dish_id	menu_id	date
0	1	26	1	2024-11-27
1	2	83	1	2024-11-27
2	3	26	1	2024-11-27
3	4	11	2	2024-11-27
4	5	44	2	2024-11-27
5	6	96	2	2024-11-27
6	7	89	2	2024-11-27
7	8	13	3	2024-11-27
8	9	55	3	2024-11-27
9	10	49	3	2024-11-27
10	11	102	3	2024-11-27
11	12	85	3	2024-11-27
12	13	40	1	2024-11-28
13	14	40	1	2024-11-28
14	15	59	1	2024-11-28

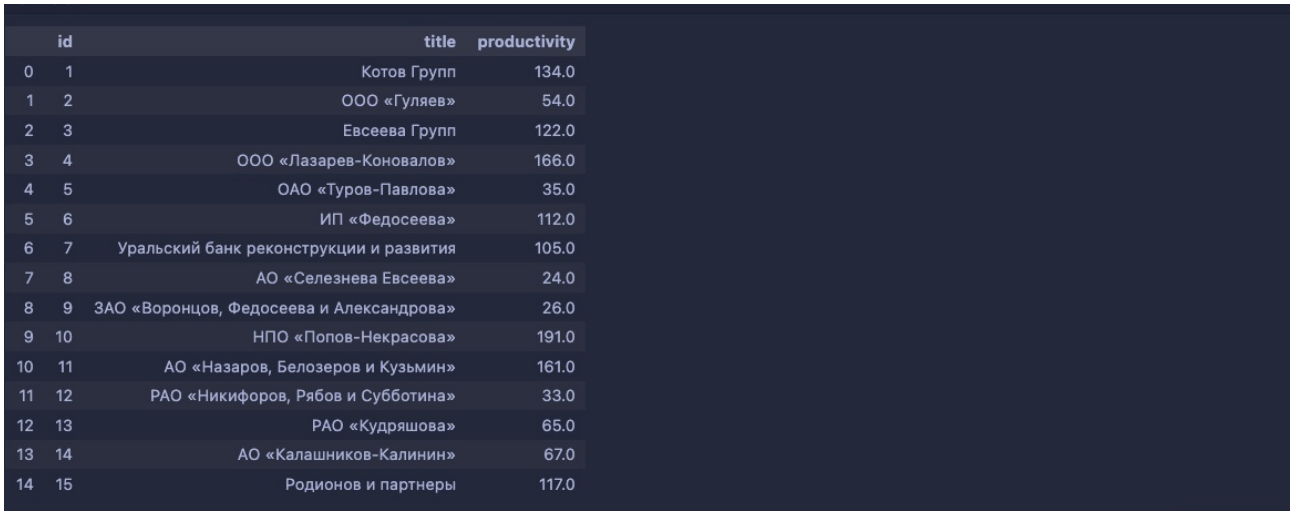
Рис. 13: Таблица “dishes_menus”

	id	title	type	cost
0	1	Куриное филе	Мясо	4.99
1	2	Говядина	Мясо	6.49
2	3	Свинина	Мясо	5.99
3	4	Кролик	Мясо	7.99
4	5	Баранина	Мясо	8.99
5	6	Индейка	Мясо	5.49
6	7	Фарш мясной	Мясо	4.99
7	8	Лосось	Рыба	10.99
8	9	Треска	Рыба	7.49
9	10	Сельдь	Рыба	4.99
10	11	Скумбрия	Рыба	6.49
11	12	Камбала	Рыба	8.99
12	13	Морепродукты	Морепродукты	12.99
13	14	Окунь	Рыба	9.49
14	15	Крабовое мясо	Морепродукты	14.99

Рис. 14: Таблица “ingredients”

	id	dish_id	ingredient_id
0	1	1	70
1	2	1	92
2	3	1	130
3	4	1	42
4	5	1	44
5	6	1	100
6	7	1	14
7	8	2	48
8	9	2	134
9	10	2	22
10	11	2	107
11	12	2	130
12	13	2	101
13	14	3	69
14	15	3	12

Рис. 15: Таблица “dishes_ingredients”



	id	title	productivity
0	1	Котов Групп	134.0
1	2	ООО «Гуляев»	54.0
2	3	Евсеева Групп	122.0
3	4	ООО «Лазарев-Коновалов»	166.0
4	5	ОАО «Туров-Павлова»	35.0
5	6	ИП «Федосеева»	112.0
6	7	Уральский банк реконструкции и развития	105.0
7	8	АО «Селезнева Евсеева»	24.0
8	9	ЗАО «Воронцов, Федосеева и Александрова»	26.0
9	10	НПО «Попов-Некрасова»	191.0
10	11	АО «Назаров, Белозеров и Кузьмин»	161.0
11	12	РАО «Никифоров, Рябов и Субботина»	33.0
12	13	РАО «Кудряшова»	65.0
13	14	АО «Калашников-Калинин»	67.0
14	15	Родионов и партнеры	117.0

Рис. 16: Таблица “suppliers”

	id	ingredient_id	supplier_id
0	1	1	366
1	2	1	88
2	3	1	357
3	4	1	450
4	5	2	63
5	6	2	470
6	7	2	89
7	8	3	393
8	9	3	128
9	10	4	62
10	11	4	244
11	12	4	472
12	13	4	162
13	14	5	79
14	15	5	467

Рис. 17: Таблица “ingredients_suppliers”

	id	ingredient_id	preference_id
0	1	26	1
1	2	67	2
2	3	68	2
3	4	69	2
4	5	70	2
5	6	71	2
6	7	72	2
7	8	73	2
8	9	74	2
9	10	75	2
10	11	76	2
11	12	77	2
12	13	78	2
13	14	143	3
14	15	12	4

Рис. 18: Таблица “ingredients_preferences”

Выполнение запросов

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

1. Найти выручку сервиса за последний месяц.

```sql
WITH
    orders_last_month AS (
        SELECT
            *
        FROM
            orders
        WHERE
            created_at >= CURRENT_DATE - INTERVAL '1 month'
    )

SELECT
    SUM(m.cost) AS Выручка
FROM
    orders_last_month AS o
    INNER JOIN menus AS m on o.menu_id = m.id
```

	Выручка
0	13422150.0

Рис. 19: Результат выполнения запроса 1

2. Какая доля (в %) общей выручки приходится на меню с названием «Похудение»?

Проценты округлить до двух знаков после запятой.

```
SELECT
    ROUND(
        SUM(m.cost) FILTER (WHERE m.title = 'Похудение')::DECIMAL /
        SUM(m.cost)::DECIMAL * 100, 2) AS losing_weight_revenue_ratio
FROM
    orders AS o
    INNER JOIN menus AS m on o.menu_id = m.id
```

	losing_weight_revenue_ratio
0	27.56

Рис. 20: Результат выполнения запроса 2

3. 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.

```
SELECT
    title,
    cost,
    ROUND(cost::DECIMAL / 1.2 * 0.2, 2) AS tax,
    ROUND(cost::DECIMAL / 1.2, 2) AS cost_before_tax
FROM
    menus
```

	title	cost	tax	cost_before_tax
0	Похудение	700.0	116.67	583.33
1	Баланс	850.0	141.67	708.33
2	Набор	1000.0	166.67	833.33

Рис. 21: Результат выполнения запроса 3

4. Найти заказы, которые оказались возвращены и были оплачены с помощью «Яндекс.Сплит». Вывести `id` заказа, дату заказа и реквизиты платежа.

```
WITH
    yandex_split_payments AS (
        SELECT
            i.*
        FROM
            payment_infos i
            JOIN payment_methods m ON i.payment_method_id = m.id
        WHERE
            m.title = 'ЯндексСплит.'
    )
```

```sql
SELECT
    o.id AS order_id,
    DATE(o.created_at) AS order_created_at,
    p.requisites AS requisites
FROM
    orders o
    JOIN yandex_split_payments p ON o.payment_info_id = p.id
WHERE
    o.status = 'returned'
```

	order_id	order_created_at	requisites
0	10	2024-09-17	348003776203841
1	86	2024-11-02	2257648768555601
2	110	2024-09-21	8100939730499929
3	136	2024-11-16	6635525845179221
4	432	2024-11-09	4037182764191594
...	...	...	...
591	49466	2024-11-14	4214473832989134
592	49543	2024-10-04	6908438154829347
593	49789	2024-08-24	6546165912392635
594	49880	2024-10-09	6831288755820316
595	49953	2024-11-20	2712010099594008

596 rows × 3 columns

Рис. 22: Результат выполнения запроса 4

5. Найти топ-5 пользователей по количеству бонусов, которые большинство своих заказов оплатили наличными курьеру. Вывести `id` пользователя, его имя и количество бонусов.

```sql
WITH
    payment_infos_and_methods AS (
        SELECT
            i.id AS payment_info_id,
            m.title AS payment_method
        FROM
            payment_infos i
            JOIN payment_methods m ON i.payment_method_id = m.id
    ),
    user_method_counts AS (
        SELECT
            o.user_id,
            p.payment_method,
            COUNT(o.id) AS orders_count
        FROM
            orders o
            JOIN payment_infos_and_methods p USING (payment_info_id)
```

```
        GROUP BY
            o.user_id,
            p.payment_method
    ),
    user_method_counts_with_total AS (
        SELECT
            *,
            SUM(orders_count) OVER (
                PARTITION BY
                    user_id
            ) AS total_orders_count
        FROM
            user_method_counts
    ),
    users_with_almost_cash_orders AS (
        SELECT
            user_id AS id
        FROM
            user_method_counts_with_total
        WHERE
            payment_method = 'Наличные курьеру'
            AND orders_count > total_orders_count / 2
    )

SELECT
    u.id,
    CONCAT(u.first_name, ' ', u.middle_name, ' ', u.last_name) AS name,
    u.bonuses
FROM
    users u
    JOIN users_with_almost_cash_orders u_cash USING (id)
ORDER BY
    u.bonuses DESC
LIMIT
    5;
```

	id	name	bonuses
0	1227	Ольга Кузьминична Соколова	410
1	5673	Назар Анатольевич Орехов	398
2	3924	Ирина Геннадиевна Петухова	398
3	2461	Александра Романовна Кондратьева	398
4	7821	Ульяна Робертовна Юдина	390

Рис. 23: Результат выполнения запроса 5

6. Найти блюда, содержащие хотя бы один ингредиент, который относится к самой популярной категории предпочтений. Вывести только названия блюд, отсортированные в алфавитном порядке.

```
WITH
    most_popular_category AS (
        SELECT
            preference_category_id,
            COUNT(pu.id) AS count
        FROM
            preferences_users pu
            JOIN preferences p ON pu.preference_id = p.id
        GROUP BY
            preference_category_id
        ORDER BY
            count DESC
        LIMIT
            1
    ),
    preferences_from_most_popular_category AS (
        SELECT
            id,
            title
        FROM
            preferences
        WHERE
            preference_category_id = (
                SELECT
                    preference_category_id
                FROM
                    most_popular_category
            )
    ),
    ingredients_from_most_popular_category AS (
        SELECT
            ip.ingredient_id
        FROM
            ingredients_preferences ip
        JOIN preferences_from_most_popular_category p ON ip.
           preference_id = p.id
    )

SELECT
    DISTINCT d.title
FROM
    dishes_ingredients
    JOIN ingredients_from_most_popular_category i ON dishes_ingredients.
       ingredient_id = i.ingredient_id
    JOIN dishes d ON dishes_ingredients.dish_id = d.id
ORDER BY
    d.title;
```

	title
0	Борщ
1	Борщ с капустой
2	Вареники с картошкой
3	Винегрет
4	Гречка с грибами
5	Гречка с овощами
6	Гречневая каша
7	Жаркое
8	Запеканка с творогом
9	Каша овсяная
10	Каша пшенная
11	Каша рисовая с молоком
12	Котлеты по-киевски
13	Медовик
14	Морс
15	Пельмени с говядиной
16	Печенье "Маковое"
17	Печенье с орехами
18	Пирог с вишней
19	Пирог с картошкой и грибами
20	Пирог с мясом
21	Пирог с мясом и картошкой
22	Пирог с рыбой
23	Ризотто с грибами
24	Рыбный суп
25	Сельдь под шубой
26	Солянка с мясом
27	Суп из чечевицы
28	Суп с фрикадельками
29	Творожная запеканка
30	Творожный десерт
31	Торт "Птичье молоко"
32	Шарлотка
33	Шуба с рыбой
34	Щи

Рис. 24: Результат выполнения запроса 6

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

Пусть блюдо состоит из ингридиентов $x_1, x_2, \dots, x_n$. Закупочная стоимость каждого ингридиента равна $c_1, c_2, \dots, c_n$ соответственно. Тогда я предполагаю, что себестоимость блюда равна $p = \sum_{i=1}^{n} c_i$ (я не учитываю, что блюдо содержит a грамм данного ингридиента).

Пусть меню на конкретный день состоит из блюд $y_1, y_2, \dots, y_m$. Тогда себестоимость меню будет равна $P = \sum_{j=1}^{m} p_j$.

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 $\frac{S-P}{P}$.

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).

```sql
WITH
    ingredients_costs_in_rubles AS (
        SELECT
            id,
            cost * 7.55 AS cost_rubles
        FROM
            ingredients
    ),
    dishes_costs AS (
        SELECT
            d.dish_id AS id,
            SUM(i.cost_rubles) AS cost
        FROM
            dishes_ingredients d
            JOIN ingredients_costs_in_rubles i ON d.ingredient_id = i.id
        GROUP BY
            d.dish_id
    ),
    menus_costs_and_dishes AS (
        SELECT
            menu_id,
            m.date,
            SUM(cost) AS our_cost,
            ARRAY_AGG(dish_id) AS dishes
        FROM
            dishes_menus m
            JOIN dishes_costs c ON m.dish_id = c.id
        GROUP BY
            menu_id,
            m.date
        ORDER BY
            m.date,
            menu_id
```

```
    ),
    menus_profits AS (
        SELECT
            mc.*,
            m.cost,
            ROUND((m.cost - mc.our_cost)::DECIMAL, 2) AS profit,
            ROUND((m.cost - mc.our_cost)::DECIMAL / mc.our_cost::DECIMAL
                * 100, 2) AS profit_percent
        FROM
            menus_costs_and_dishes mc
            JOIN menus m ON mc.menu_id = m.id
    ),
    max_profit AS (
        SELECT
            menu_id,
            date,
            dishes,
            profit,
            profit_percent
        FROM
            menus_profits
        ORDER BY
            profit_percent DESC
        LIMIT 1
    ),
    max_profit_percent AS (
        SELECT
            menu_id,
            date,
            dishes,
            profit,
            profit_percent
        FROM
            menus_profits
        ORDER BY
            profit DESC
        LIMIT 1
    )

SELECT * FROM max_profit UNION ALL SELECT * FROM max_profit_percent;
```

	menu_id	date	dishes	profit	profit_percent
0	1	2024-12-06	[12, 56, 103]	531.48	315.39
1	2	2024-12-08	[81, 56, 56, 65]	536.83	171.41

Рис. 25: Результат выполнения запроса 7

8. Какие пары блюд находятся вместе в меню чаще всего? Вывести их названия и количество нахождений вместе.

```
WITH
    max_count_pair AS (
        SELECT DISTINCT
            ARRAY[dm1.dish_id, dm2.dish_id] AS pair,
            COUNT(*) AS count
        FROM
            dishes_menus dm1 JOIN dishes_menus dm2 USING(menu_id, date)
        WHERE
            dm1.dish_id < dm2.dish_id
        GROUP BY
            pair
        ORDER BY
            count DESC
        LIMIT 1
    )

SELECT
    d1.title AS dish1,
    d2.title AS dish2,
    count
FROM
    max_count_pair
    JOIN dishes d1 ON pair[1] = d1.id
    JOIN dishes d2 ON pair[2] = d2.id
```

	dish1	dish2	count
0	Сельдь под шубой	Кекс	3

Рис. 26: Результат выполнения запроса 8

9. Найти медианное количество заказов у пользователей.

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 числа, 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.

```
WITH
    main_table AS (
        SELECT
            u.id,
            COUNT(o.id) AS orders_count
        FROM
            users u
```

```
            JOIN orders o ON u.id = o.user_id
        GROUP BY
            u.id
        ORDER BY
            orders_count ASC
    ),
    series AS (
        SELECT
            orders_count,
            ROW_NUMBER() OVER (
                ORDER BY
                    orders_count
            ) AS row_number,
            COUNT(*) OVER () AS total_rows
        FROM
            main_table
    )

SELECT
    AVG(orders_count) AS median_count
FROM
    series
WHERE
    row_number BETWEEN total_rows / 2.0 AND total_rows  / 2.0 + 1
```

	median_count
0	3.0

Рис. 27: Результат выполнения запроса 9

10. Рассчитать ежедневную выручку сервиса, рассчитать ежедневный прирост выручки (численно и в %) относительно предыдущего дня.

```
WITH
    main_table AS (
        SELECT
            DATE (o.created_at) AS date,
            SUM(m.cost) AS revenue
        FROM
            orders o
            JOIN menus m ON o.menu_id = m.id
        GROUP BY
            DATE (o.created_at)
        ORDER BY
            date
    )

SELECT
    date,
    revenue,
    (revenue - LAG (revenue, 1) OVER ()) AS revenue_growth_abs,
```

```
    ROUND(100 * (revenue - LAG (revenue, 1) OVER ())::DECIMAL / LAG (
       revenue, 1) OVER ()::DECIMAL, 2) AS revenue_growth_rate
FROM
    main_table
```

	date	revenue	revenue_growth_abs	revenue_growth_rate
0	2024-08-19	108100.0	NaN	NaN
1	2024-08-20	423050.0	314950.0	291.35
2	2024-08-21	432450.0	9400.0	2.22
3	2024-08-22	405150.0	-27300.0	-6.31
4	2024-08-23	430900.0	25750.0	6.36
...	...	...	...	...
96	2024-11-23	474850.0	50600.0	11.93
97	2024-11-24	403450.0	-71400.0	-15.04
98	2024-11-25	417250.0	13800.0	3.42
99	2024-11-26	398900.0	-18350.0	-4.40
100	2024-11-27	299050.0	-99850.0	-25.03

101 rows × 4 columns

Рис. 28: Результат выполнения запроса 10

11. Построить иерархию приглашенных и пригласивших пользователей.

```
WITH RECURSIVE
    users_tree AS (
        SELECT
            id,
            CONCAT (first_name, ' ', middle_name, ' ', last_name) AS
               full_name,
            invited_by_id AS parent_id,
            1 AS level
        FROM
            users
        WHERE
            invited_by_id IS NULL
        UNION ALL
        SELECT
            u.id,
            CONCAT (first_name, ' ', middle_name, ' ', last_name) AS
               full_name,
            invited_by_id AS parent_id,
            ut.level + 1
        FROM
            users u
            JOIN users_tree ut ON u.invited_by_id = ut.id
    )

SELECT
    *
```

```
FROM
    users_tree
```

	id	full_name	parent_id	level
0	14	Демьян Ильич Романов	NaN	1
1	47	Панфил Валерьянович Назаров	NaN	1
2	1	Фёкла Владиславовна Мартынова	NaN	1
3	2	Раиса Кузьминична Игнатова	NaN	1
4	4	Евпраксия Мироновна Гуляева	NaN	1
...	...	...	...	...
14995	228	Таисия Тарасовна Новикова	13314.0	6
14996	6681	Лариса Робертовна Афанасьева	6977.0	7
14997	2039	Мариан Дмитриевич Анисимов	10803.0	7
14998	5621	Леонтий Ярославович Никонов	524.0	7
14999	13393	Клавдия Геннадиевна Уварова	10741.0	7

15000 rows × 4 columns

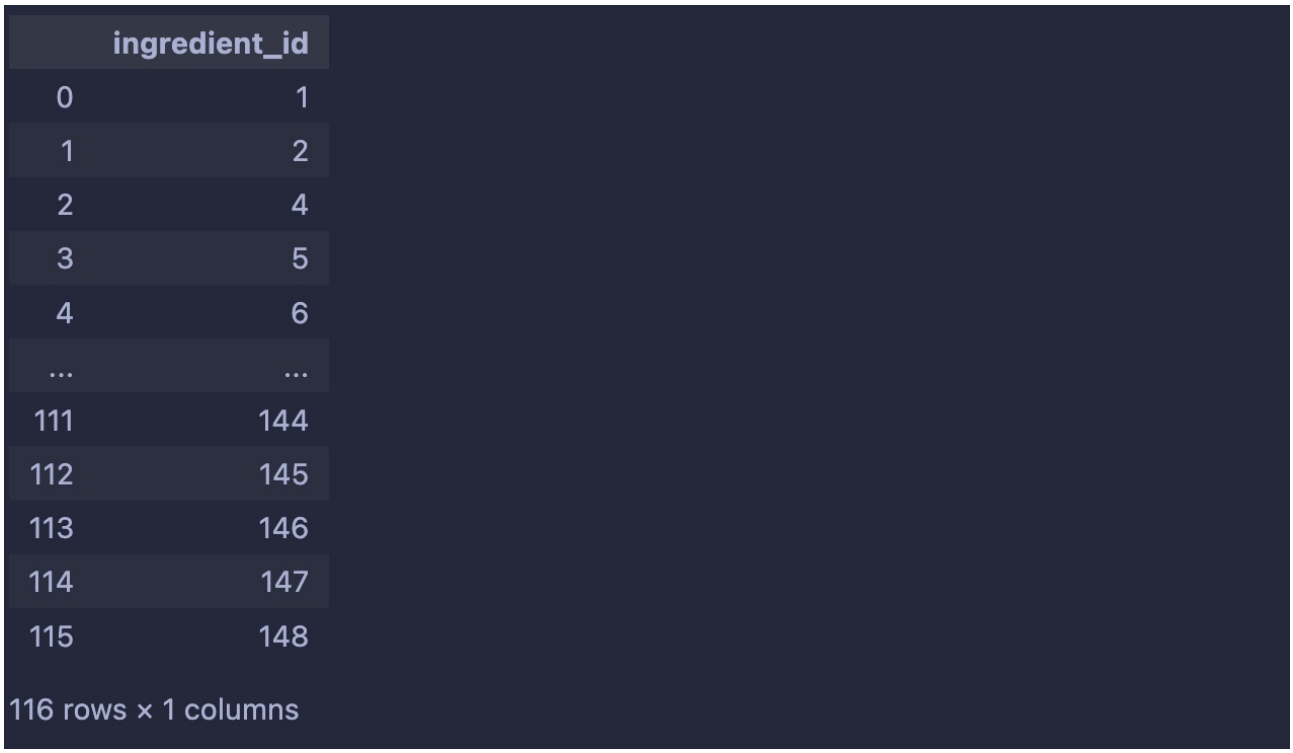
Рис. 29: Результат выполнения запроса 11

12. 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 $\frac{a}{n}$ грамм.

```
WITH
    orders_last_month AS (
        SELECT
            menu_id,
            date (created_at) AS date,
            COUNT(*) AS orders_count
        FROM
            orders
        WHERE
            date (created_at) + INTERVAL '1 day' BETWEEN CURRENT_DATE -
               INTERVAL '1 month' AND CURRENT_DATE
        GROUP BY
            date (created_at),
            menu_id
    ),
    dishes_count AS (
        SELECT
```

```
            dish_id,
            SUM(orders_count) AS count
        FROM
            orders_last_month o
            JOIN dishes_menus dm ON o.menu_id = dm.menu_id
            AND o.date + INTERVAL '1 month 1 day' = dm.date
        GROUP BY
            dish_id
    ),
    dishes_ingredients_with_weights AS (
        SELECT
            dish_id,
            ingredient_id,
            weight::DECIMAL / COUNT(*) OVER (PARTITION BY dish_id)::
               DECIMAL AS ingredients_weight
        FROM
            dishes_ingredients AS di
            JOIN dishes AS d ON di.dish_id = d.id
    ),
    ingredients_total_weight AS (
        SELECT
            ingredient_id,
            SUM(ingredients_weight * count) AS total_weight
        FROM
            dishes_ingredients_with_weights AS weights
            JOIN dishes_count AS counts USING (dish_id)
        GROUP BY
            ingredient_id
    ),
    ingredients_productivity AS (
        SELECT
            ingredient_id,
            SUM(productivity) AS productivity
        FROM
            ingredients_suppliers AS ins
            JOIN suppliers s ON ins.supplier_id = s.id
        GROUP BY
            ingredient_id
    )

SELECT
    ingredient_id
FROM
    ingredients_total_weight AS itw
    JOIN ingredients_productivity AS ip USING (ingredient_id)
WHERE
    productivity * 1000 >= total_weight
ORDER BY
    ingredient_id
```



	ingredient_id
0	1
1	2
2	4
3	5
4	6
...	...
111	144
112	145
113	146
114	147
115	148

116 rows × 1 columns

Рис. 30: Результат выполнения запроса 12