МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ

**Национальный исследовательский ядерный университет «МИФИ»**

****

Институт интеллектуальных кибернетических систем

**Кафедра кибернетики (№ 22)**

**Отчёт о работе по курсу**

**«Базы данных (теоретические основы баз данных)»**

Вариант «Нетфликс»

|  |  |
| --- | --- |
| Выполнил | Косенко Н.И. |
| Группа | Б20-524 |
| Вариант | Нетфликс |
| Преподаватель | Петровская А.В. |
| Проверяющий |  |
| Оценка |  |

**Москва 2022**

**Содержание**

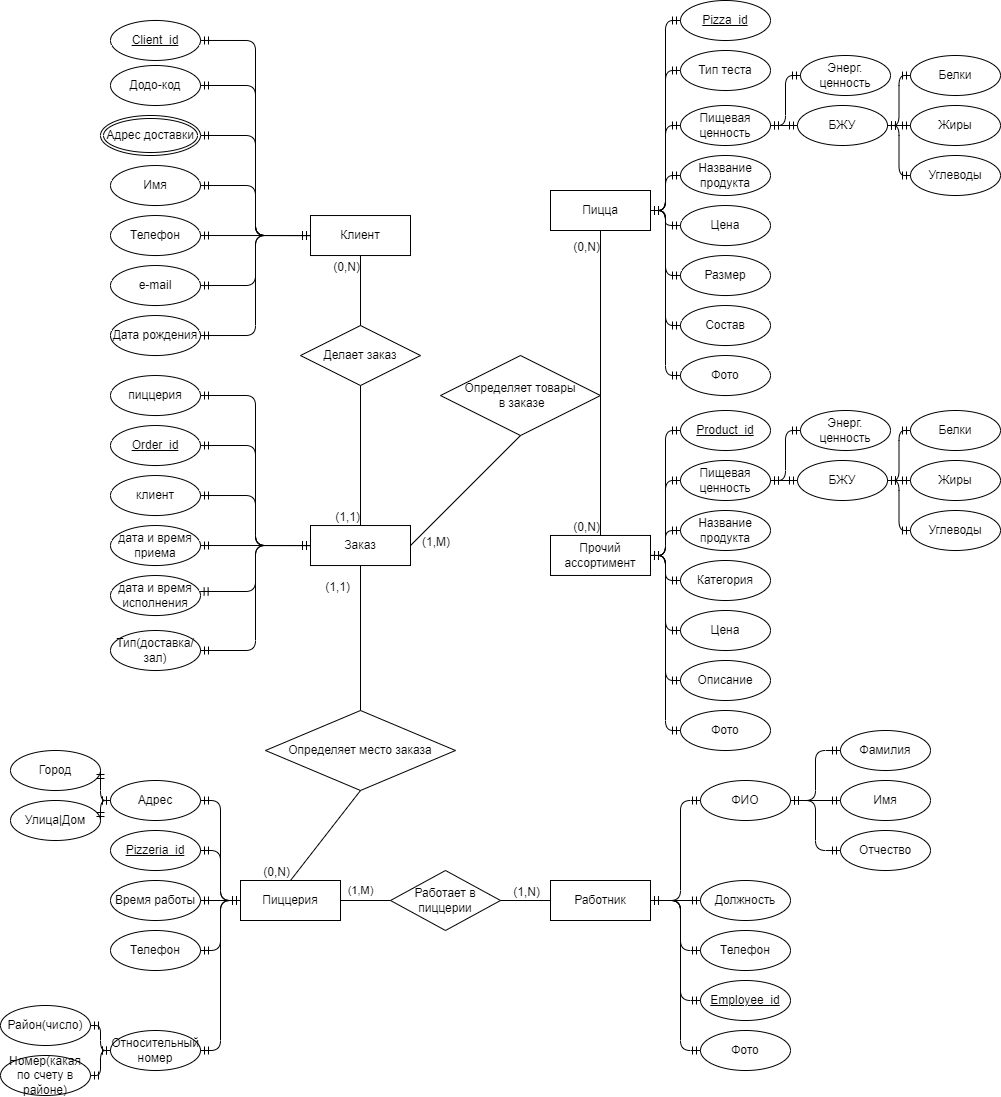
1. [Формулировка задания 3](#_bookmark0)
2. [Концептуальная модель базы данных 3](#_bookmark1)
   1. [Конкретизация предметной области 4](#_bookmark2)
   2. [Описание предметной области и принятых ограничений 4](#_bookmark3)
   3. [Описание атрибутов 5](#_bookmark4)
3. [Логическое проектирование 7](#_bookmark5)
4. [Физическое проектирование 8](#_bookmark6)
   1. [Создание таблиц 8](#_bookmark7)
   2. [Заполнение базы данных 10](#_bookmark8)
      1. [Подготовка данных 11](#_bookmark9)
      2. [Программа заполнения базы данных 13](#_bookmark10)
      3. [Результаты заполнения 15](#_bookmark11)
5. [Выполнение запросов 19](#_bookmark12)

# Формулировка задания

#### Спроектировать базу данных для сети пиццерий «ДОДО пицца», расположенных в разных городах России и осуществляющих розничную продажу еды и напитков. База данных должна содержать информацию о клиентах, отражать текущий штат сотрудников, непосредственно сеть пиццерий и ассортимент товаров, предлагаемых ими.

# Концептуальная модель базы данных

#### После проведения анализа предметной области была спроектирована следующая концептуальная модель:



#### Рисунок 1 – Концептуальная модель базы данных

## Конкретизация предметной области

#### Необходимо создать систему, отражающую информацию о заказах зарегистрированных клиентов в пиццериях по всей стране. Ассортимент разбить на две основные группы: пицца и прочие продукты, — т.к. они имеют разные характеристики.

#### База данных должна отражать в какой пиццерии и на какой должности работает сотрудник. Сами пиццерии должны содержать информацию о местоположении и времени работы.

* 1. **Описание предметной области**

#### Система ориентирована на следующих пользователей:

#### Незарегистрированные пользователи;

#### Зарегистрированные клиенты пиццерии, делающие заказы;

#### Сотрудники компании, проводящие отбор и назначение на должности в пиццерии по всей стране.

#### Любой пользователь может узнать информацию об интересующей его пиццерии — адрес, время работы, телефон.

#### Каждый человек может зарегистрироваться в приложении или на сайте компании- заказчика, указав свой номер телефона и имя, став клиентом сети.

#### Клиент может приобретать позиции из ассортимента, информация о котором хранится в базе данных и отражается в приложении и на сайте. Также, клиенту доступна информация о всех заказах, совершённых им с момента регистрации. Даже если товар, содержащийся в заказе, больше не активен, т.е. его невозможно приобрести в сети пиццерий, клиент увидит его. То же касается и пиццерии, к которой привязывается заказ при оформлении. Клиент также может оформлять доставку, указав при этом адрес доставки, которые впоследствии будет сохранён в системе.

#### Сотрудники могут проводить анализ и учёт всех заказов, совершенных с момента начала функционирования системы. Кроме того, В базе данных отражаются текущая должность каждого сотрудника, место его работы, а также личная информация.

#### Определенные лица компании: HR-менеджеры, администраторы сети, директора компании, - также могут вносить изменения в назначениях сотрудников, информации о времени работы пиццерий и вносить данные новых работников и зданий.

#### В итоге, система реализует следующий функционал:

#### учёт представленного ассортимента;

#### регистрация всех заказов в каждой пиццерии;

#### учёт персональной информации клиентов;

#### учёт персональной информации непосредственных сотрудников пиццерий;

#### Предоставление справочной информации о каждой пиццерии. Таким образом, были выделены следующие сущности:

#### Клиент

#### Заказ

#### Пицца

#### Прочий ассортимент

#### Пиццерия

#### Работник

## Описание атрибутов

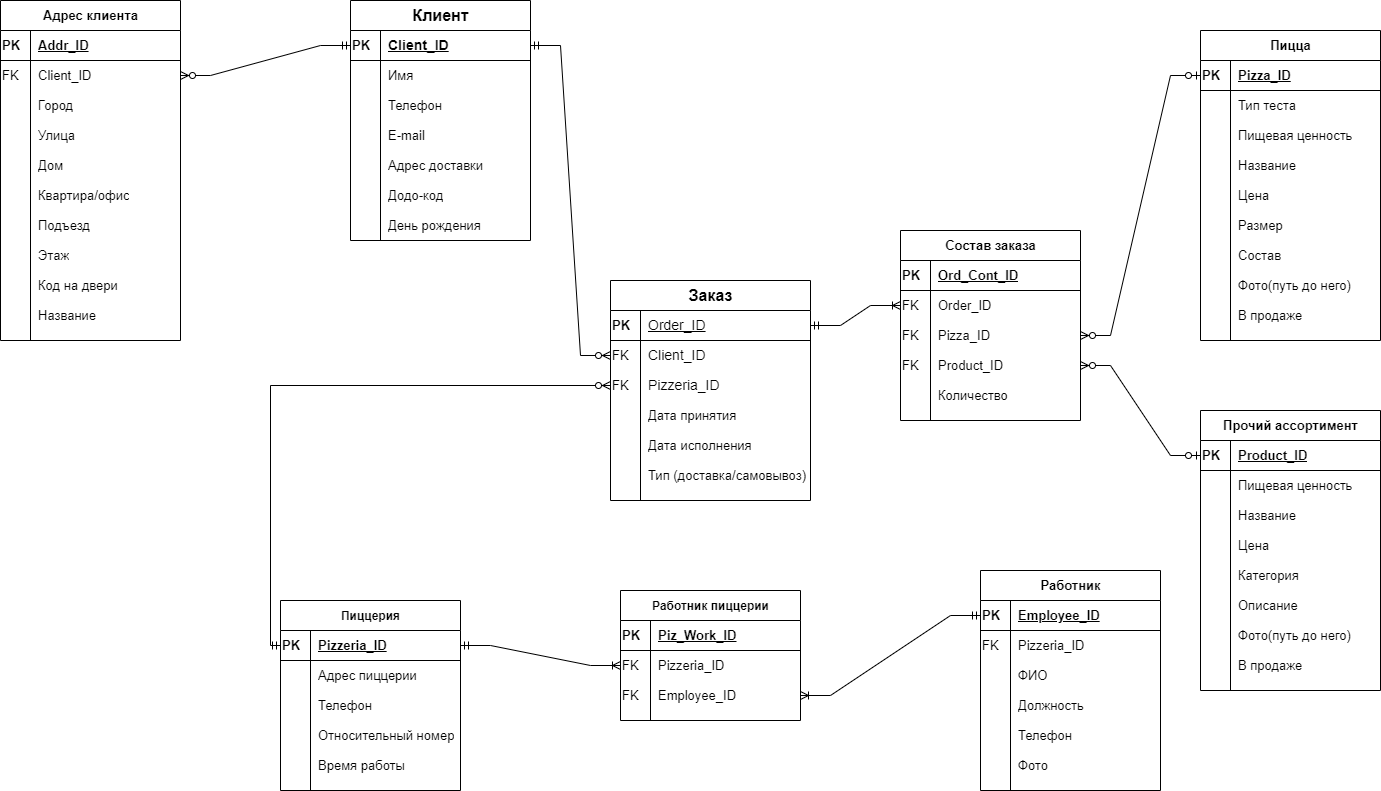
#### В процессе анализа были выделены следующие атрибуты, название и описание которых приведены в таблице ниже.

|  |  |
| --- | --- |
| Имя атрибута | Расшифровка |
| Client\_id, Pizza\_id, Product\_id, Order\_id, Pizzeria\_id, Employee\_id | Уникальный идентификатор соответственно клиента, пиццы, прочего продукта из  ассортимента, заказа, пиццерии и сотрудника |
| Имя | Имя клиента |
| Адрес доставки | Адреса, по которым осуществляется доставка  заказов клиенту |
| Телефон | Номер телефона клиента, пиццерии или  сотрудника |
| e-mail | Адрес электронной почты клиента |
| Дата рождения | Дата рождения пользователя |
| Додо код | Трёхзначный номер, присваиваемый клиенту |
| Пиццерия | Пиццерия, в которой был сделан заказ |
| Клиент | Клиент, оформивший заказ |
| Дата и время приёма/исполнения | Точное время оформления/выполнения заказа |
| Тип | Клиент заказал доставку или придёт за  заказом лично |
| Название продукта | Название позиции из ассортимента |
| Тип теста | Пицца на толстом или тонком тесте |
| Размер | Размер пиццы (Маленькая, средняя, большая) |

|  |  |
| --- | --- |
| Пищевая ценность | Показатели БЖУ и энергетическая ценность  продукта |
| Цена | Стоимость продукта |
| Состав | Подробный состав пиццы |
| Фото | Путь до файла с фотографией продукта |
| Категория | К какому типу относится продукт из прочего  ассортимента (Напитки, снеки и т.д.) |
| Описание | Словесное описания продукта |
| Адрес | Город и точный адрес пиццерии |
| Время работы | Время работы пиццерии |
| Относительный номер | Содержит код пиццерии в зависимости от  района города и номера по счёту в нём |
| ФИО | ФИО сотрудника |
| Должность | Позиция сотрудника в пиццерии |

# Логическое проектирование

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



#### Рисунок 2 – Логическая модель базы данных

#### Многозначный атрибут «Адрес доставки» у сущности «Клиент» был вынесен в отдельную таблицу, чтобы соблюсти нормализацию — по крайней мере 1НФ.

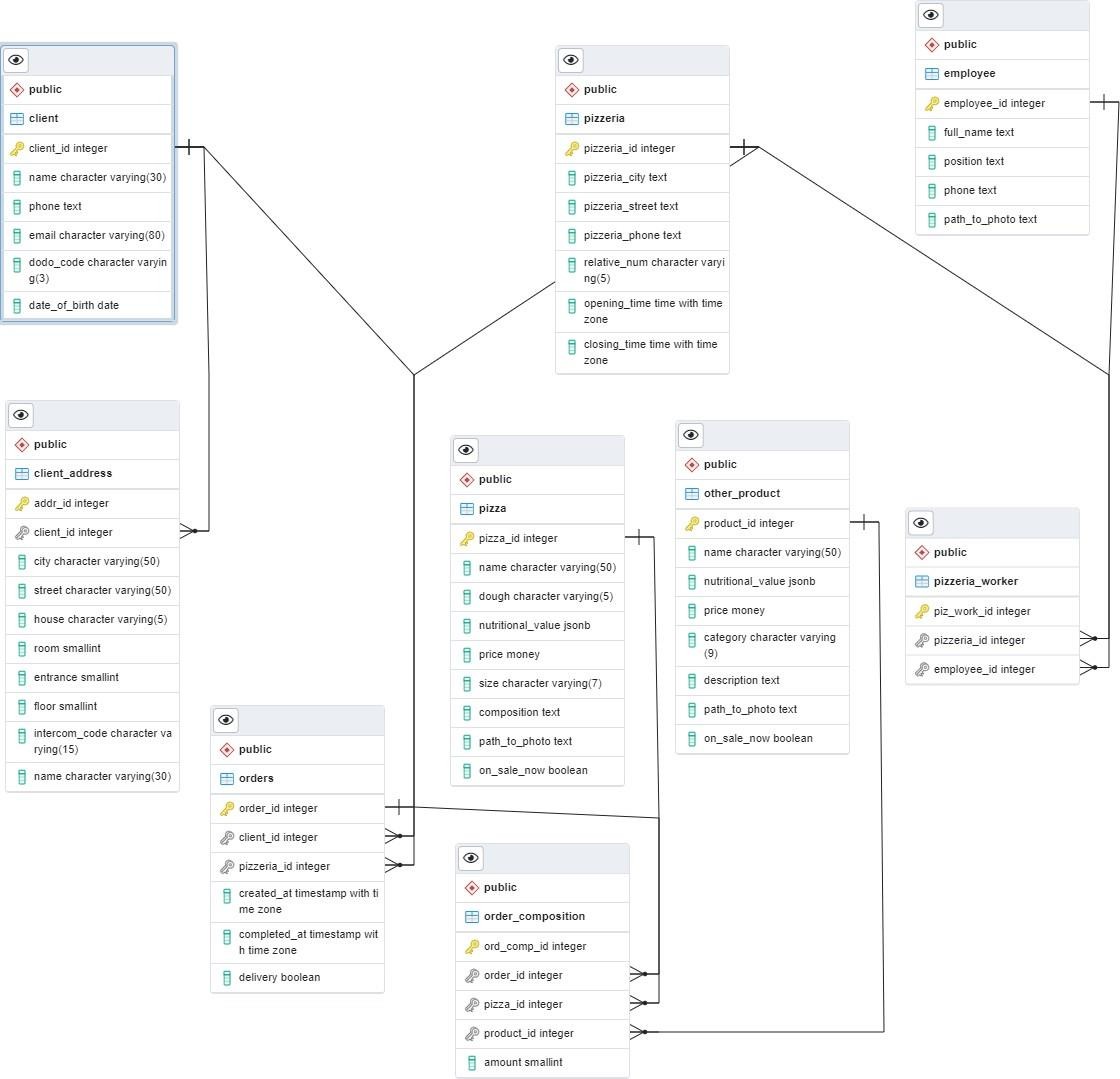
#### Все связи типа «Многие ко многим» были реализованы через отдельные таблицы.

#### Заказ и его позиции связаны через таблицу «Состав заказа», которая содержит идентификатор заказа, одни из идентификаторов ассортимента, т.е. в одной записи или идентификатор пиццы не пустой или наоборот, а также количество выбранной позиции ассортимента.

#### Таким образом, все таблицы базы данных находятся в 3НФ форме, т.к. все атрибуты являются простыми, отсутствуют функциональные зависимости между неключевыми атрибутами и первичными ключами, а также нет транзитивных зависимостей.

# Физическое проектирование

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



#### Рисунок 3 – Графическое представление базы данных

## Создание таблиц

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

#### Код для создания таблицы client

#### Код для создания таблицы client\_address

#### Код для создания таблицы pizzeria

#### Код для создания таблицы orders

#### Код для создания таблицы pizza

#### Код для создания таблицы other\_product

#### Код для создания таблицы order\_composition

#### Код для создания таблицы employee

#### Код для создания таблицы pizzeria\_worker

## Заполнение базы данных

#### Заполнение базы данных производилось при помощи ЯП kotlin и ORM фреймворка Exposed.

### Подготовка данных

#### В сети Интернет были найдены списки имён, фамилий, названий городов и улиц, а также часовые пояса городов, различные домены электронных адресов, и были обёрнуты в объект Data. Краткое описание объекта представлено ниже:

*object* Data {

*val* pizzeriasAmount = HashMap<String, Int>()

*val* clientsAddresses = HashMap<Int, Int>()

*const val* pathToEmployee = "/home/employees/photos/"

*const val* pathToPizza = "/home/pizza/photos/"

*val* domains = *arrayOf*( "gmail.com", "hotmail.com", "mail.ru",

… "ymail.com",

"libero.it", "outlook.com"

)

*val* names = *arrayOf*( "Анна",

"София",

… "Давид", "Андрей"

)

*val* surnames = *arrayOf*( "Смирнов", "Иванов",

… "Блинов",

"Колесников"

)

*val* phoneCodes = *arrayOf*( "900",

"901",

.. "997",

"999"

)

*val* positions = *arrayOf*("Повар", "Кассир", "Пиццемейкер", "Тестомес", "Пекарь", "Руководитель", "Менеджер")

*val* cities = *hashMapOf*( Pair("Москва", 0),

Pair("Санкт-Петербург", 1),

Pair("Астрахань", 2),

…

Pair("Югорск", 33),

Pair("Якутск", 34)

)

*val* citiesTimezones = *hashMapOf*( Pair("Москва", "+03"),

Pair("Санкт-Петербург", "+03"),

…

Pair("Югорск", "+05"),

Pair("Якутск", "+09")

)

*val* streets = *arrayOf*( "Абрамцевская просека", "Абрамцевская улица",

#### Также, с сайта ДОДО пиццы был взят текущий ассортимент и записан в файлы в специальном виде с разделителем, например:

#### Для пиццы: Название.Состав

#### Энерг\_ценность.Белки.Жиры.Углеводы Цена маленькой пиццы

#### Цена средней Цена большой

#### Для прочего ассортимента Название|Описание Энерг\_ценность.Белки.Жиры.Углеводы Цена

#### Также, т.к. ещё имеются категории, то ассортимент был разбит по файлам, соответствующим категориям продуктов.

### Программа заполнения базы данных

#### Выполнение запросов к БД осуществляется посредством транзакций:

#### Здесь «filling fun» одна из следующих функций:

#### Функция заполнения таблицы client

#### Функция заполнения таблицы orders

0,

24

)

}".*padStart*(2, '0')

}:${"${Random.nextInt(0, 60)}".*padStart*(2, '0')}:${ "${

Random.nextInt( 0,

60

)

}".*padStart*(2, '0')

}"

times[createdAt] = LocalDateTime.parse(createdAt.*substringBefore*('+'),

DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"))

.plusMinutes(Random.nextLong(15, 240)).toString().*replace*('T', ' ')

}

*val* timesList = times.*toList*()

*for* (i *in* 1..quantity) {

*val* client = Random.nextInt(1, 201)

*val* clientCitiesArr = ArrayList<String>() ClientCities.slice(ClientCities.city)

.*select* **{** *// SELECT public.city FROM public.client\_address WHERE client\_id = 12*

ClientCities.clientId *eq* client

**}**.*toList*().*forEach* **{**

clientCitiesArr.add(**it**.toString().*substringAfter*('='))

##### }

*val* ids = ArrayList<Int>()

*val* city: String =

*if* (clientCitiesArr.*isNotEmpty*()) { *//SELECT pizzeria\_id FROM public.pizzeria WHERE pizzeria\_city = 'Оренбург' OR pizzeria\_city = 'Коломна'. Use random instead of OR*

clientCitiesArr.*random*()

} *else* {

Data.cities.keys.*toTypedArray*().*random*()

}

PizzeriasIds.slice(PizzeriasIds.pizzeriaId).*select* **{**

PizzeriasIds.pizzeriaCity *eq* city

**}**.*toList*().*forEach* **{**

ids.add(**it**.toString().*substringAfter*('=').*toInt*()) **}**

*val* createdAt = timesList[i - 1].first + Data.citiesTimezones[city]

*val* completedAt = timesList[i - 1].second + Data.citiesTimezones[city]

TransactionManager.current().exec(

"INSERT INTO public.orders( *order\_id*, *client\_id*, *pizzeria\_id*, *created\_at*, *completed\_at*, *delivery*) VALUES ( NEXTVAL('orders\_order\_id\_seq'), '$client', '${ids.*random*()}', '$createdAt', '$completedAt', '${Random.nextBoolean()}');"

)

}

}

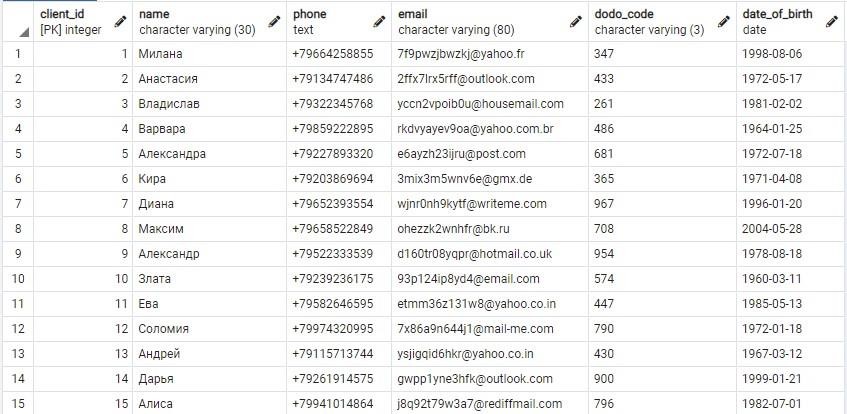
#### Как видно, происходит обыкновенный INSERT запрос, который содержит данные, выбранные случайным образом. Во втором примере также осуществляется SELECT запрос к БД, чтобы на выходе получить относительно достоверные данные. В данном случае, заказы клиента осуществляются в городах, который он указывал в своих адресах.

#### Полный текст программы для заполнения таблиц приложены к отчёту — Main.kt

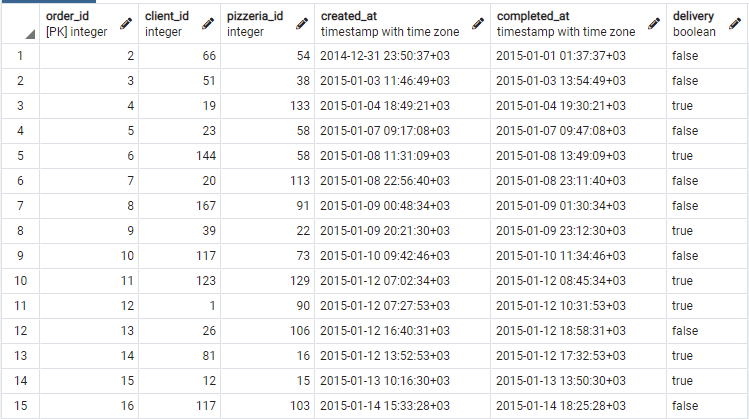
### Результаты заполнения

#### Далее представлены результаты работы программы на примере таблиц, соответствующих функциям, приведенным выше.

#### Таблица client

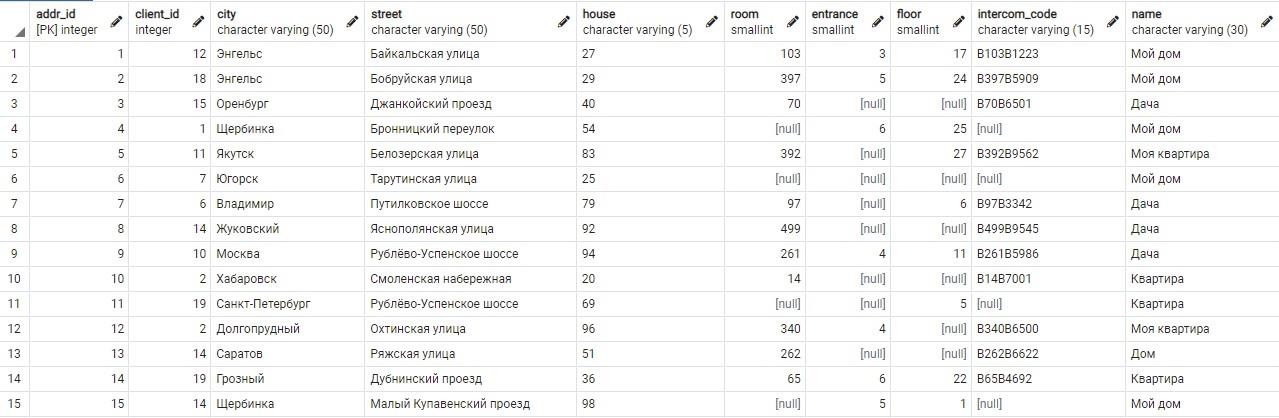


#### Таблица orders

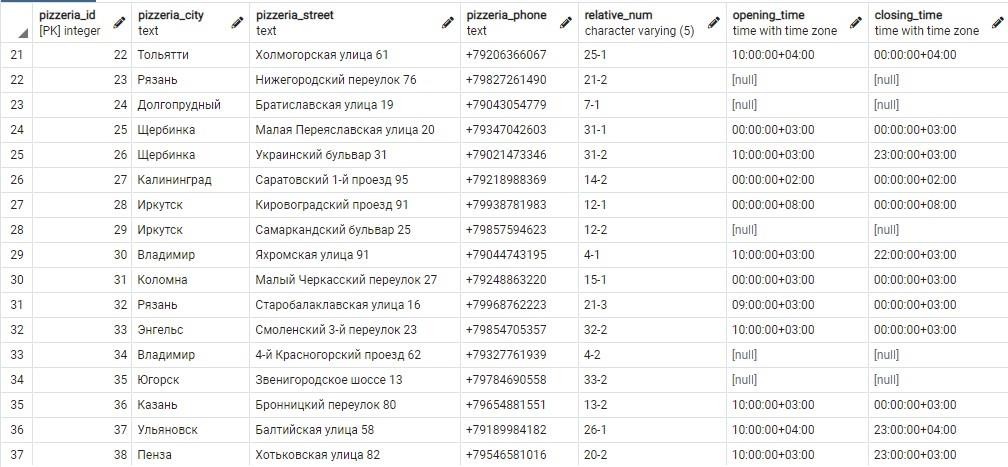


#### Остальные таблицы:

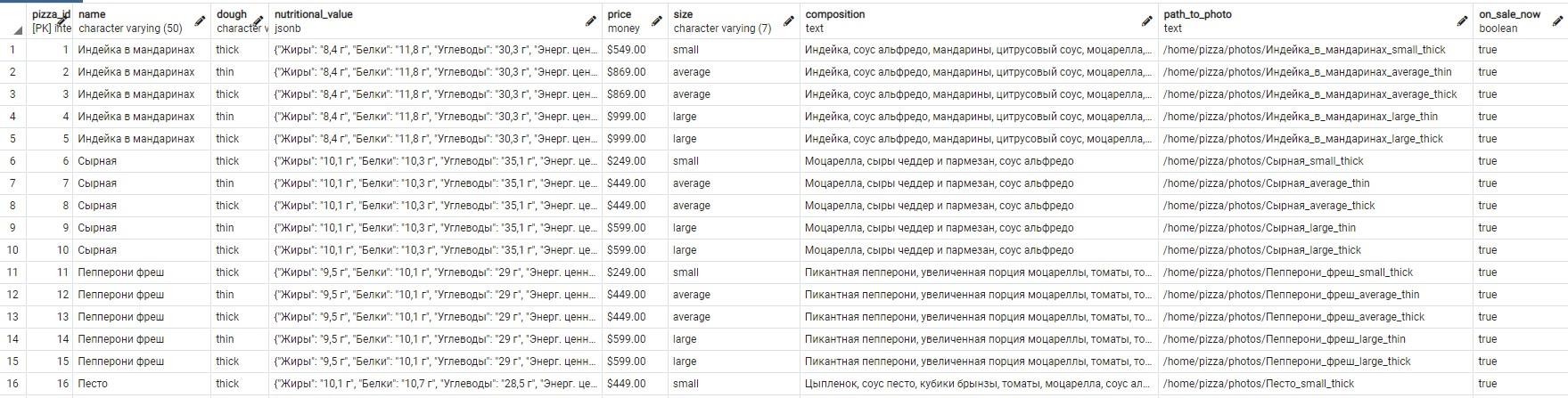
#### Таблица client\_addresses



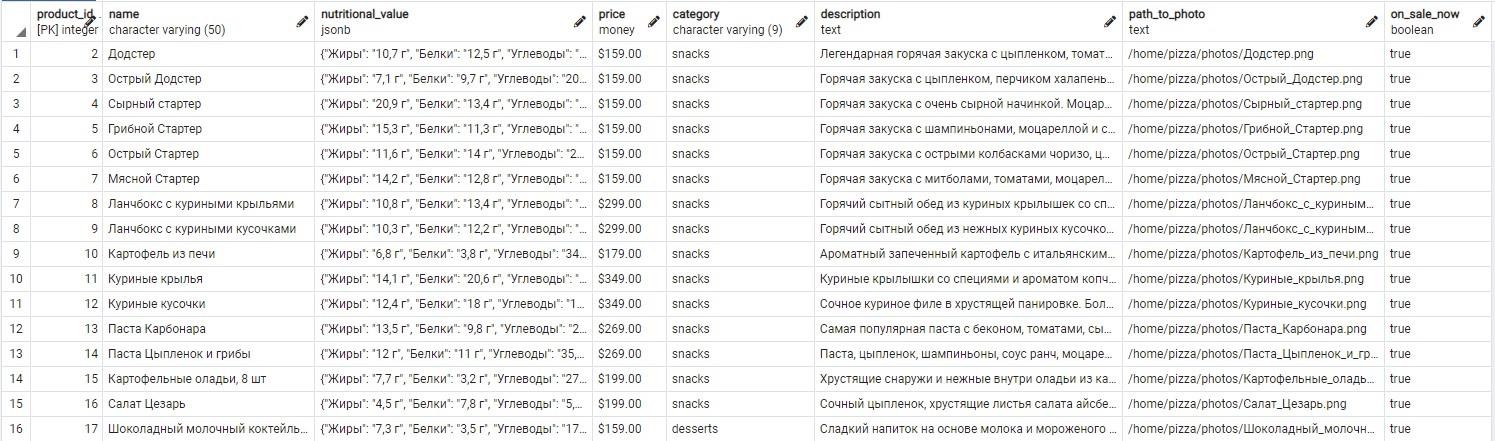
#### Таблица pizzeria



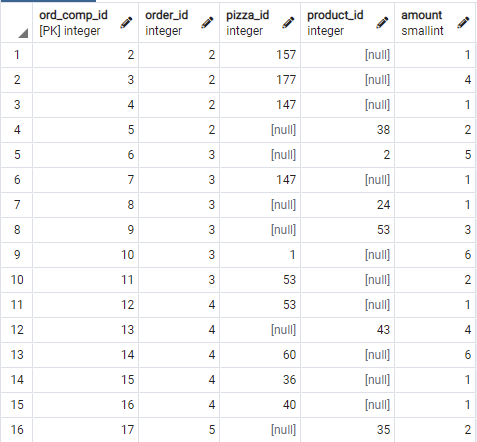
#### Таблица pizza



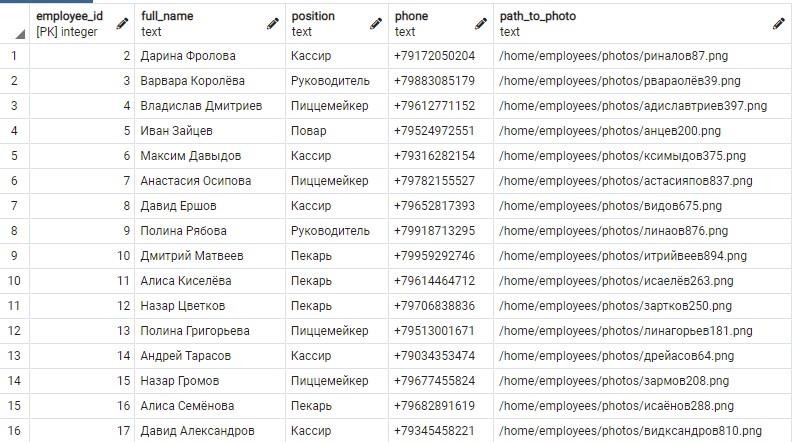
#### Таблица other\_products



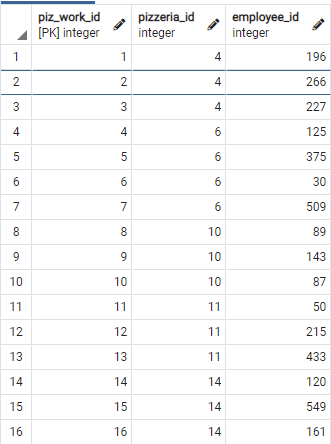
#### Таблица order\_composition



#### Таблица employee



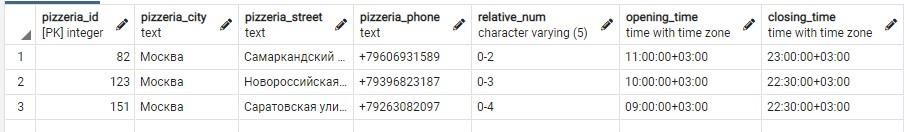
#### Таблица pizzeria\_worker



# Выполнение запросов

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

#### Вывести все работающие пиццерии в городе.



#### Обновить телефон пользователя

#### Было:

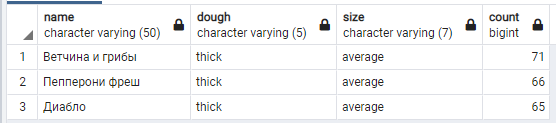


#### Стало:

#### Сравнить средние стоимости на пиццы и на прочий ассортимент и вывести наибольшее из них



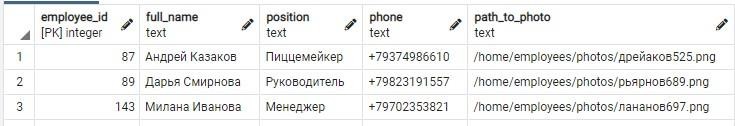
#### Найти топ 3 самых заказываемых пицц



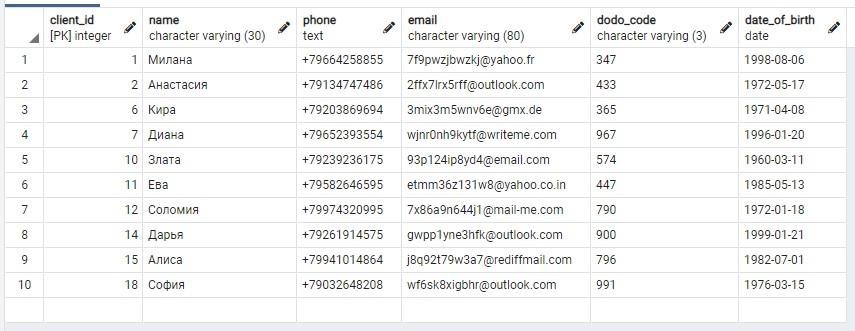
#### Вывести действующие пиццерии в городе клиента.



#### Получить работников определенной пиццерии

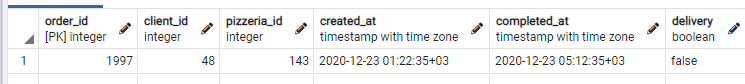


#### Найти клиентов с >3 указанными адресами доставки



#### Вывести последний заказ содержащий определенный продукт с количеством

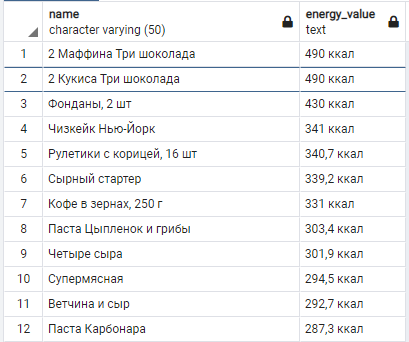
#### >3



#### Вывести список заказов, на которые определённый пользователь потратил более 1500 рублей



#### Найти позиции ассортимента с энергетической ценностью >250 Ккал. Фильтрация ассортимента в приложении по энерг. ценности.



#### Вывести количество незанятых сотрудников каждой должности



#### Найти повара(-ов) и сколько разных пицц он приготовил для определённого клиента во время выполнения определённого заказа

ON client\_order.order\_id = order\_composition.order\_id WHERE order\_composition.pizza\_id IS NOT NULL



GROUP BY client\_order.order\_id

),client\_pizza AS (

SELECT client\_id, name, phone, piz.\* FROM client\_order

INNER JOIN piz

ON piz.order\_id = client\_order.order\_id

), workers\_ids AS (

SELECT pizzeria\_worker.employee\_id, client\_order.order\_id FROM pizzeria\_worker

INNER JOIN client\_order

ON pizzeria\_worker.pizzeria\_id = client\_order.pizzeria\_id

), workers AS (

SELECT employee.employee\_id, full\_name, position, phone, path\_to\_photo, order\_id

FROM employee

INNER JOIN workers\_ids

ON employee.employee\_id = workers\_ids.employee\_id

WHERE position = 'Повар' OR position = 'Пиццемейкер' OR position = 'Тестомес' OR position = 'Пекарь'

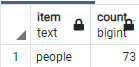
)

SELECT client\_pizza.\*, full\_name, position, workers.phone, path\_to\_photo FROM client\_pizza

INNER JOIN workers

ON client\_pizza.order\_id = workers.order\_id;

#### Определить, сколько клиентов старше 30 лет указали ровно 2 адреса доставки



#### Вычислить суммарную выручку с продаж за 2020 год. Ведение бух. учёта и аудита.



#### Для конкретного города вывести рейтинг поваров по количеству приготовленных пицц за какой-то год

