# **ГЛАВА 1. АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ**

## **1.1 Общие сведения о предметной области**

В рамках проекта рассматривается создание базы данных для ресторана. Ресторан — это предприятие общественного питания, которое предоставляет услуги по приему и обслуживанию клиентов, предлагая им различные блюда и напитки. Основная деятельность ресторана - приготовление и продажа пищи, но также он может предоставлять другие услуги, такие как организация мероприятий и банкетов, доставка еды на дом и т.д.

## **1.2 Цель создания базы данных**

Цель создания базы данных для ресторана заключается в обеспечении эффективного управления всеми аспектами ресторанного бизнеса. База данных позволяет хранить информацию о блюдах, ингредиентах, клиентах, сотрудниках, заказах, поставках и партнерах, а также связи между ними. Благодаря этому ресторан может получить целостное представление о своей деятельности и принимать управленческие решения на основе данных, что повышает эффективность и прибыльность бизнеса.

* Хранение данных о блюдах и ингредиентах. База данных может использоваться для хранения информации о блюдах, которые предлагает ресторан, а также о всех необходимых ингредиентах, которые используются для приготовления этих блюд. Это позволяет ресторану быстро и легко находить информацию о своих продуктах, а также контролировать запасы ингредиентов.
* Хранение данных о клиентах. База данных может использоваться для хранения информации о клиентах ресторана, включая их имена, контактные данные, предпочтения и историю заказов. Это позволяет ресторану предоставлять персонализированный сервис и улучшать опыт клиентов.
* Хранение данных о заказах. База данных может использоваться для хранения информации о заказах, которые делают клиенты, включая дату, время, состав заказа, стоимость, комментарии и т.д. Это позволяет ресторану контролировать свои продажи, улучшать управление запасами и оптимизировать свои бизнес-процессы.
* Хранение данных о поставках и партнерах. База данных может использоваться для хранения информации о поставках ингредиентов, а также о партнерах, которые поставляют эти ингредиенты. Это позволяет ресторану контролировать свои затраты на ингредиенты и улучшать управление своими поставками.
* Хранение данных о сотрудниках. База данных может использоваться для хранения информации о сотрудниках ресторана, включая их имена, контактные данные, должности, зарплаты, даты начала работы и т.д. Это позволяет ресторану контролировать свои затраты на персонал, улучшать управление своими ресурсами и оптимизировать свои бизнес-процессы. Создание отчетов и аналитика.

База данных может использоваться для создания отчетов и аналитики, которые помогают ресторану понимать свою деятельность и принимать управленческие решения на основе данных. Например, база данных может использоваться для создания отчетов о продажах, запасах, затратах и прибыли, а также для анализа трендов и паттернов в деятельности ресторана.

## **1.3 Возможные пользователи базы данных и сценарии их взаимодействия с базой данных**

Для нашей базы данных возможными пользователями могут быть администраторы, менеджеры компании, руководители компании, клиенты и сотрудники компании. Администраторы базы данных могут создавать и настраивать базу данных, настраивать права доступа, обеспечивать безопасность базы данных, резервное копирование и восстановление базы данных, а также мониторить ее производительность. Менеджеры компании могут просматривать информацию о клиентах, заказах и продажах, а также добавлять, изменять и удалять информацию о них. Менеджер по работе с клиентами может управлять информацией о клиентах, менеджер по обработке заказов может управлять информацией о заказах, а менеджер по продажам - информацией о продажах. Руководители компании могут просматривать общую информацию о компании, отчеты о продажах, заказах и клиентах, а также анализировать производительность компании. Клиенты могут просматривать меню блюд, оформлять заказы и просматривать информацию о своих заказах и скидках. Сотрудники компании могут управлять информацией о блюдах и ингредиентах, заказах и поставках, клиентах и партнерах, а также просматривать информацию о своей зарплате и должности.

Краткое описание доступных функций для каждого из возможных клиентов:

1. Администраторы базы данных: Создание и настройка базы данных. Настройка прав доступа к базе данных. Обеспечение безопасности базы данных. Резервное копирование и восстановление базы данных. Мониторинг производительности базы данных.
2. Менеджеры компании:

i. Менеджер по работе с клиентами: Просмотр информации о клиентах. Добавление, изменение и удаление информации о клиентах.

ii. Менеджер по обработке заказов: Просмотр информации о заказах. Добавление, изменение и удаление информации о заказах.

iii. Менеджер по продажам: Просмотр информации о продажах. Добавление, изменение и удаление информации о продажах.

1. Руководители компании: Просмотр общей информации о компании. Просмотр отчетов о продажах, заказах и клиентах. Анализ производительности компании.
2. Клиенты: Просмотр меню блюд. Оформление заказов. Просмотр информации о своих заказах и скидках.
3. Сотрудники: Просмотр информации о своей зарплате и должности. Добавление, изменение и удаление информации о блюдах и ингредиентах. Добавление, изменение и удаление информации о заказах и поставках. Добавление, изменение и удаление информации о клиентах и партнерах.

## **1.4 Основные сущности, их атрибуты и связи между ними**

* Dishes - блюда, которые готовятся в компании. Атрибуты: id, name, sale\_price.
* Ingredient - ингредиенты, используемые для приготовления блюд. Атрибуты: id, name, price, in\_stock.
* Client - клиенты компании. Атрибуты: id, name, sex, discount.
* Department - отделы компании. Атрибуты: id, adress.
* Partner - партнеры компании. Атрибуты: id, name, description.
* Contract - контракты с партнерами. Атрибуты: id, date, information, partner\_id.
* Employee - сотрудники компании. Атрибуты: id, name, phone\_number, position, salary, start\_date, dept\_id.
* Order - заказы клиентов. Атрибуты: id, date, tips, comment, employee\_id, client\_id.
* Supply - поставки ингредиентов от партнеров. Атрибуты: id, date, partner\_id, dept\_id.

Связи между сущностями:

1. Dishes и Ingredient - связь многие ко многим через таблицу contains.
2. Client и Order - связь один ко многим.
3. Department и Employee - связь один ко многим.
4. Partner и Contract - связь один ко многим.
5. Partner и Supply - связь один ко многим.
6. Employee и Order - связь один ко многим.
7. Dishes и Order - связь многие ко многим через таблицу includes.
8. Supply и Ingredient - связь многие ко многим через таблицу delivered.

## **1.5. Алгоритмы обработки данных, используемые в сценариях**

Для создания полезной и понятной базы данных необходимо использовать алгоритмы обработки данных, такие как CRUD-операции, триггеры, хранимые процедуры, индексы, оптимизация запросов, транзакции, агрегирующие функции и внешние ключи. Эти алгоритмы помогут выполнить различные операции, оптимизировать запросы и вычислить статистические метрики:

1. CRUD-операции — это основные операции, которые выполняются с данными в базе данных: создание (Create), чтение (Read), обновление (Update) и удаление (Delete) записей. Для реализации этих операций можно использовать язык SQL.
2. Триггеры — это специальные хранимые процедуры, которые автоматически вызываются при определенных событиях (например, при вставке, обновлении или удалении записей). Триггеры могут использоваться для автоматизации определенных задач, таких как проверка целостности данных или запись изменений в журнал.
3. Хранимые процедуры — это набор инструкций SQL, которые могут быть вызваны из приложения. Хранимые процедуры могут использоваться для упрощения сложных операций, таких как обработка больших объемов данных или выполнение нескольких операций одновременно.
4. Индексы — это структуры данных, которые ускоряют поиск и сортировку данных в таблицах. Индексы могут быть созданы на одном или нескольких столбцах таблицы, что ускоряет выполнение запросов, которые относятся к этим столбцам.
5. Оптимизация запросов — это процесс оптимизации запросов к базе данных для ускорения их выполнения. Этот процесс может включать в себя выбор правильных индексов, оптимизацию структуры таблицы и т.д.
6. Транзакции — это группы операций, которые выполняются как единое целое. Транзакции могут использоваться для обеспечения целостности данных и предотвращения конфликтов при параллельном доступе к данным.
7. Агрегирующие функции — это функции, которые позволяют вычислять статистические данные, такие как среднее значение, максимальное и минимальное значение, сумму и т.д. Агрегирующие функции могут быть использованы для анализа данных и вычисления различных метрик.
8. Внешние ключи — это ограничения, которые определяют связи между таблицами в базе данных. Внешние ключи могут использоваться для поддержания целостности данных и предотвращения удаления или изменения записей, на которые ссылаются другие таблицы.

## **1.6.** **Перечень отчетных форм**

Конечной целью проекта является создание нескольких отчетных форм, которые будут содержать информацию о продажах и запасах товаров, а также динамику продаж и наиболее востребованные типы товаров. Одним из примеров отчетной формы может быть информационная панель в программе Excel с диаграммами, построенными на основании сводных таблиц.

## **1.7. Архитектура программного продукта**

Архитектура программного продукта для нашего ресторана представляет собой программное обеспечение, использующее базу данных, содержащую информацию о блюдах, ингредиентах, клиентах, заказах, поставках и партнерах, для работы функций программного продукта. Программный продукт делится на две части: клиентскую и серверную.

Клиентская часть должна обеспечивать удобный доступ пользователя к необходимой информации и содержать следующие функции:

* Каталог блюд;
* Возможность добавления блюд в корзину;
* Возможность редактирования корзины;
* Корзину заказов;
* Историю заказов;
* Личный кабинет пользователя;
* Возможность внесения изменений в личные данные пользователя;
* Отображение важной информации о сделанном заказе;
* Возможность связаться со службой поддержки.

Клиентская часть должна также обладать возможностями расширения функционала. Серверная часть должна обеспечивать простой доступ администратору к информации о пользователях, заказах, поставках, партнерах, блюдах и ингредиентах, а также предоставлять функционал редактирования данных без необходимости применения СУБД.

Список необходимых для реализации в серверной части программного продукта функций:

* Функция редактирования пользовательских данных;
* Функция добавления новых блюд;
* Функция удаления блюд;
* Функция изменения описания блюд;
* Функция внесения изменений в заказ;
* Функция контроля заказа;
* Функция учета оставшегося на складе количества ингредиентов и продуктов;
* Функция автоматического формирования гибкой отчетности.

# **ГЛАВА 2. ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ**

## **2.1. Логическая (инфологическая) модель в нотации Чена**

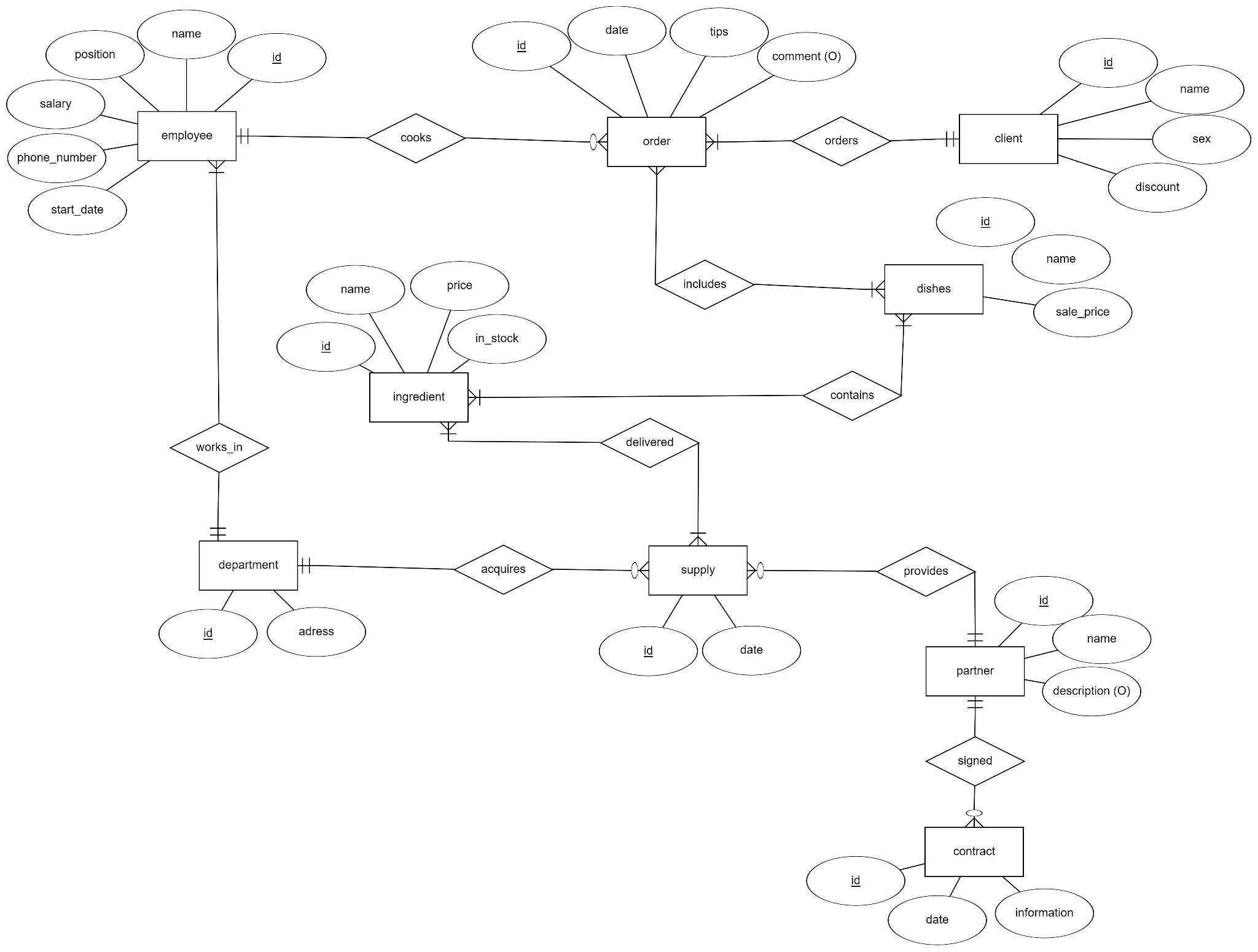


Рисунок 1. Схема инфологической модели предметной области

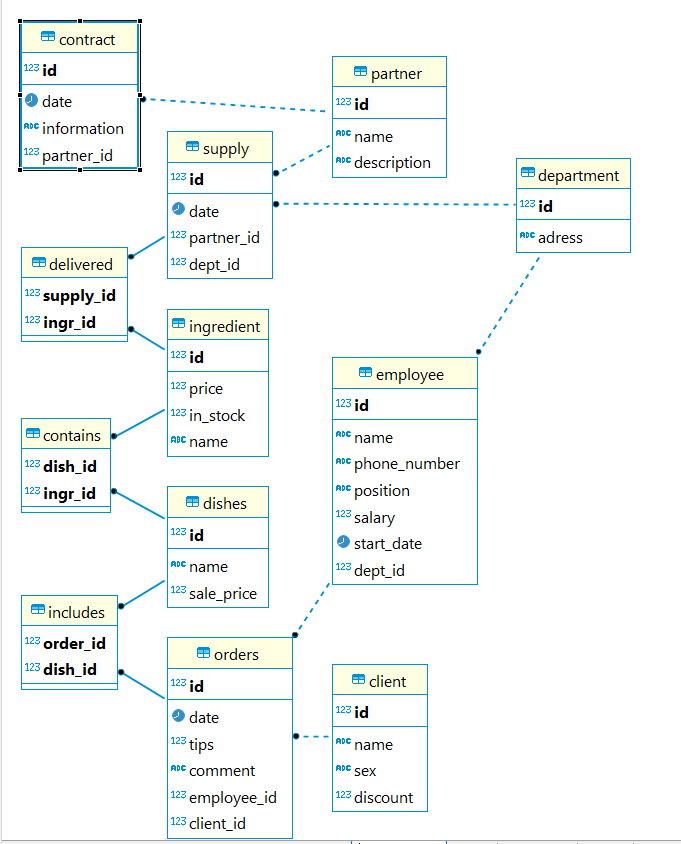
На первом рисунке изображена инфологическая модель, которая основана на описании предметной области и учитывает потребности пользователей базы данных. Модель выполнена в нотации Чена и создана при помощи ресурса "ERDplus", который предоставляет широкий спектр возможностей для моделирования и редактирования инфологических моделей.

## **2.2. Логико–физическая (даталогическая) модель**Рисунок 2. Схема даталогической модели базы данных

На основе инфологической модели была построена даталогическая модель физического уровня. Как можно заметить, был осуществлён переход модели из состояния сущностей в состояние таблиц. Модель построена в нотации IDEF1x.

# **ГЛАВА 3. СТРУКТУРА БАЗЫ ДАННЫХ**

## **3.1 Таблицы базы данных в СУБД**

Рисунок 3. Схема модели базы данных, реализованной в СУБД

После построения даталогической физической модели базы данных был выполнен её перенос в СУБД Microsoft SQL Server при помощи приложения администрирования DBeaver..

На рисунке 3 изображена схема базы данных, состоящая из 12 таблиц.

1. Таблица *Client,* предназначенная для хранения информации о клиентах ресторана

| **Название атрибута** | **Тип данных** | **Описание** |
| --- | --- | --- |
| id | int | ID клиента, первичный ключ |
| name | nvarchar | Имя клиента |
| sex | nvarchar | Пол клиента |
| discount | int | Размер скидки клиента в процентах |

1. Таблица *Contains*, в которой хранятся данные о том, какие ингредиенты для каких блюд используются

| **Название атрибута** | **Тип данных** | **Описание** |
| --- | --- | --- |
| dish\_id | int | ID блюда |
| ingr\_id | int | ID ингредиента |

1. Таблица *Contract*, содержащая информацию о контрактах на поставку товаров от партнеров

| **Название атрибута** | **Тип данных** | **Описание** |
| --- | --- | --- |
| id | int | ID контракта |
| date | date | Дата заключения контракта |
| partner\_id | int | ID поставщика |
| information | nvarchar | Описание деталей поставки |

1. Таблица *Delivered*, в которой хранится информация о том, в каких поставках какие ингредиенты поставлялись

| **Название атрибута** | **Тип данных** | **Описание** |
| --- | --- | --- |
| supply\_id | int | ID поставки |
| ingr\_id | int | ID ингредиента |

1. Таблица *Department*, хранящая в себе адреса всех филиалов

| **Название атрибута** | **Тип данных** | **Описание** |
| --- | --- | --- |
| id | int | ID филиала |
| adress | nvarchar | Адрес филиала |

1. Таблица Dishes, содержащая информацию о блюдах, доступных к конкретных филиалах и их цене

| **Название атрибута** | **Тип данных** | **Описание** |
| --- | --- | --- |
| id | int | Первичный ключ |
| name | nvarchar | Название блюда |
| sale\_price | float | Цена блюда |

1. Таблица *Employee,* предназначенная для хранения информации о работниках ресторана

| **Название атрибута** | **Тип данных** | **Описание** |
| --- | --- | --- |
| name | nvarchar | Имя работника |
| phone\_number | nvarchar | Номер телефона работника |
| position | nvarchar | Должность работника |
| salary | float | Размер зарплаты работника |
| start\_date | date | Дата приступления к должности |
| id | int | ID работника, первичный ключ |
| dept\_id | int | ID департамента, к которому привязан работник. |

1. Таблица *Includes,* хранящая в себе информацию о блюдах в заказе

| **Название атрибута** | **Тип данных** | **Описание** |
| --- | --- | --- |
| order\_id | int | ID заказа, первичный ключ |
| dish\_id | int | ID блюда, который привязан к определенному заказу. |

1. Таблица *Ingredient* содержащая информацию об ингредиентах для блюд

| **Название атрибута** | **Тип данных** | **Описание** |
| --- | --- | --- |
| price | float | Цена за ед. ингредиента |
| in\_stock | float | Количество ингредиентов, доступных на складе |
| id | int | ID ингредиента, первичный ключ |
| name | nvarchar | Название ингредиента |

1. Таблица *Orders*, содержащая информацию о заказах клиентов

| **Название атрибута** | **Тип данных** | **Описание** |
| --- | --- | --- |
| date | date | Дата создания заказа |
| tips | float | Сумма оставленных клиентом чаевых |
| id | int | ID заказа |
| comment | nvarchar | Комментарий к заказу |
| employee\_id | int | Работник, который приготовил заказ |
| client\_id | int | Клиент, сделавший заказ |

1. Таблица *Partner*, содержащая информацию о партнерах (поставщиках) ресторана

| **Название атрибута** | **Тип данных** | **Описание** |
| --- | --- | --- |
| id | int | ID партнера |
| name | nvarchar | Имя партнера ресторана |
| description | nvarchar | Описание партнера |

1. Таблица *Supply*, содержащая информацию о поставке товаров от партнеров

| **Название атрибута** | **Тип данных** | **Описание** |
| --- | --- | --- |
| id | int | ID поставки, первичный ключ |
| date | date | Дата поставки |
| partner\_id | int | ID партнера, который поставляет товар. |
| dept\_id | int | ID департамента, который принял поставку |

## **3.2 Соответствие базы данных 3 нормальной форме**

База данных соответствует первой нормальной форме, потому все поля таблиц являются атомарными и все кортежи каждой из таблиц различны. База данных соответствует второй нормальной форме, потому что она соответствует первой нормальной форме и все неключевые атрибуты каждой из таблиц неприводимо функционально зависят от первичного ключа, то есть никакой неключевой атрибут не выводится из другого неключевого атрибута. База данных находится в третьей нормальной форме, потому что она находится во второй нормальной форме и исключены транзитивные функциональные зависимости неключевых полей от первичного ключа во всех таблицах.

## **3.3 Заполнение таблицы**

База данных заполнялась автоматически сгенерированными тестовыми данными при помощи языка программирования Python. Таблица Orders содержит

1000 строк данных по заказам; Таблица Client содержит 100 строк данных;

# **ГЛАВА 4. РАЗРАБОТКА**

## **4.1. Хранимые процедуры**

1. GetClientsOrders - процедура с использованием ветвления и содержащая не

менее 3 запросов.

Описание: процедура предназначена для получения количества заказов и

общей стоимости для определённого пользователя.

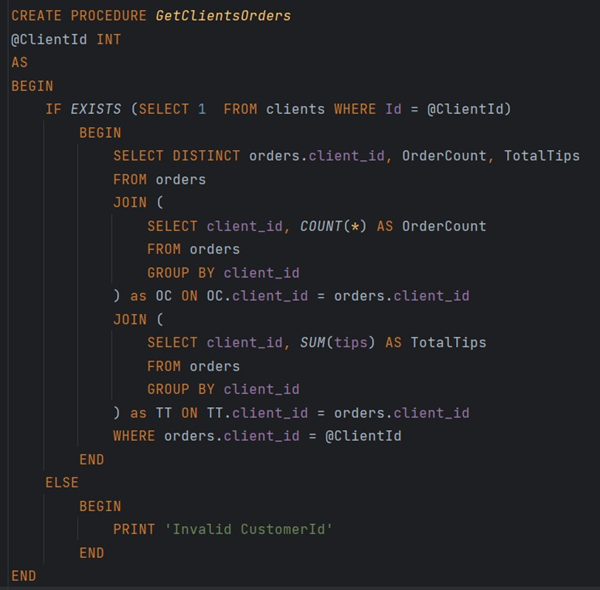


Рисунок 4. Процедура GetClientsOrders



Рисунок 5. Скрипт выполнения процедуры GetClientsOrders

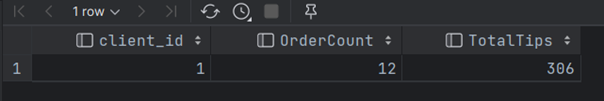


Рисунок 6. Результат выполнения процедуры GetClientsOrders

1. AddOrder - процедура, включающая в себя обработку ошибок TRY CATCH, а также работу с транзакциями.

Описание: процедура добавления нового заказа. Если добавление прошло успешно, транзакция коммитится, иначе - происходит роллбек и в консоль выводится описание ошибки.

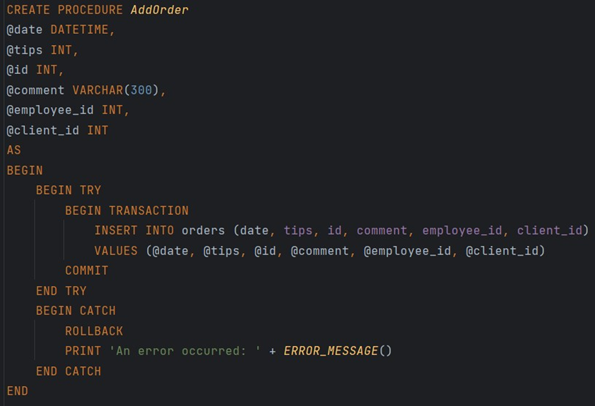


Рисунок 7. Процедура AddOrder

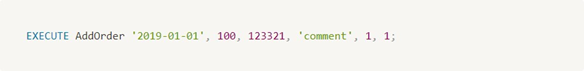


Рисунок 8. Скрипт вызова процедуры AddOrder

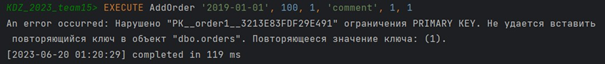


Рисунок 9. Вывод консоли при попытке добавить заказ с существующим ID





Рисунок 10, 11. Вывод консоли при уникальном ID заказа и результат скрипта

1. UpdateOrderComment - процедура, выполняющая обновление на основе результатов другого запроса.

Описание: обновление комментария к заказу только для заказов с существующим ID.

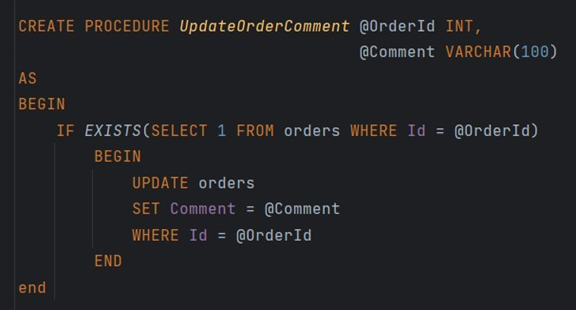


Рисунок 12. Процедура UpdateOrderComment



Рисунок 13. Скрипт вызова процедуры UpdateOrderComment



Рисунок 14. Состояние таблицы до выполнения процедуры UpdateOrderComment.



Рисунок 15. Состояние таблицы после выполнения процедуры UpdateOrderComment.

1. ChangePhoneNumber - процедура обновления номера работника.

Описание: процедура изменяет номер телефона работника на указанный с проверкой удовлетворения шаблону x-(xxx)-(xx)-(xx).

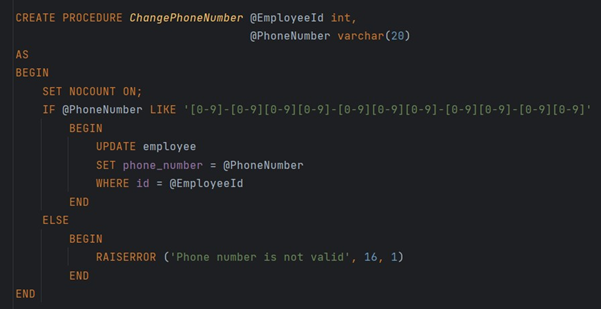


Рисунок 16. Процедура ChangePhoneNumber

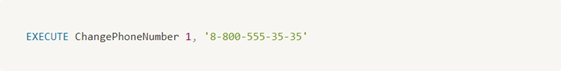


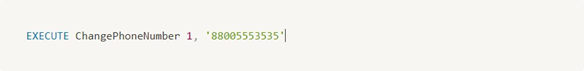
Рисунок 17. Скрипт вызова процедуры ChangePhoneNumber



Рисунок 18. Состояние таблицы employee до выполнения скрипта



Рисунок 19. Состояние таблицы employee после выполнения скрипта



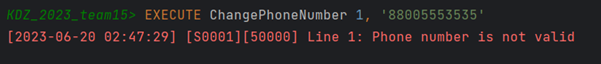


Рисунок 20, 21. Вывод консоли при попытке запуска скрипта

1. AddOrderWithCurrentDate - процедура добавления заказа с текущей датой.

Описание: при необходимости внести данные о заказе с текущей датой используется процедура AddOrderWithCurrentDate с переданными параметрами заказа, которая внутри себя вызывается ранее написанную процедуру AddOrder с соответственными параметрами и в качестве даты кладёт текущую.

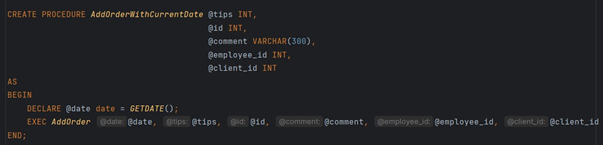


Рисунок 22. Процедура AddOrderWithCurrentDate



Рисунок 23. Скрипт запуска процедуры AddOrderWithCurrentDate

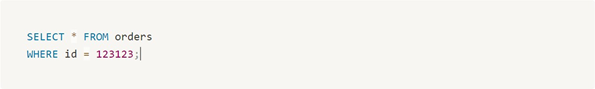




Рисунок 23, 24. Состояние таблицы после выполнения скрипта

## **4.2. Триггер**

Описание: при добавлении в таблицу delivered записи, в таблице ingredient количество ингредиента in\_stock увеличивается на единицу для ингредиента с соответствующим ingr\_id.

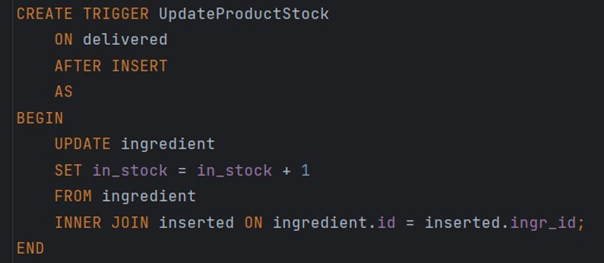


Рисунок 25. Добавлении в таблицу delivered записи

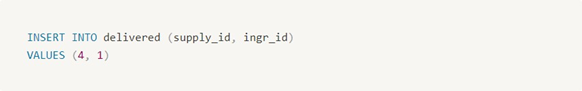


Рисунок 26. Добавление в таблицу delivered запись с ingr\_id = 1 (то есть с name = яйца)

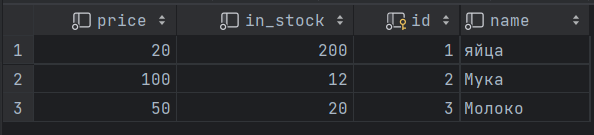


Рисунок 27. Состояние таблицы ingredient до срабатывания триггера (до добавления записи в таблицу delivered)

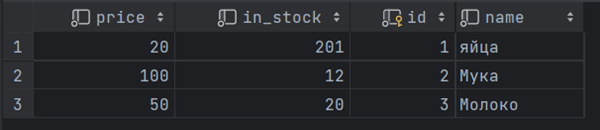


Рисунок 28. Состояние таблицы ingredient после срабатывания триггера (после добавления записи в таблицу delivered)

## 

## **4.3. Функции**

1. DishIngredients - функция определения ингредиентов для блюда по его ID. Вывод название блюда и имена необходимых для приготовления ингредиентов.

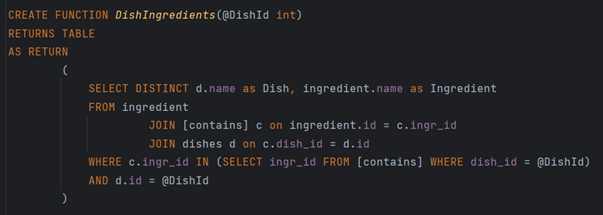
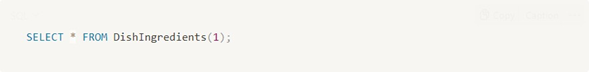


Рисунок 29. Вызов функции DishIngredients



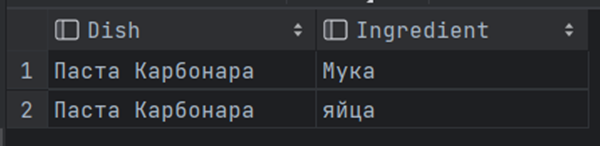


Рисунок 30, 31. Вызов функции и результат

1. GetOrderDishNameByClientId - функция определения списка всех заказанных блюд определённого клиента.

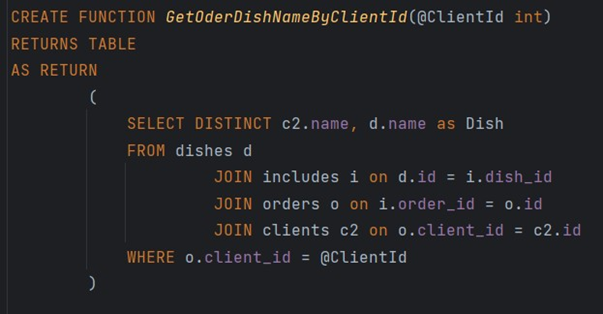
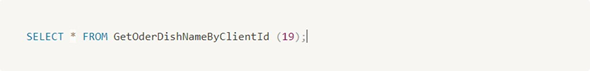


Рисунок 32. Вызов функции GetOrderDishNameByClientId



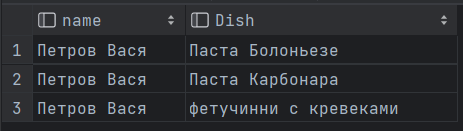
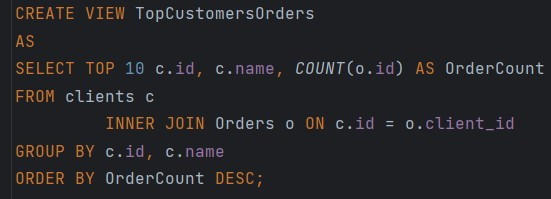


Рисунок 33, 34. Вызов функции и результат

## **4.4. Представления**

1. TopCustomersOrders - ID, имена и количество заказов для 10 пользователей с наибольшим количеством заказов. Это представление может быть полезным для анализа и идентификации самых активных клиентов.



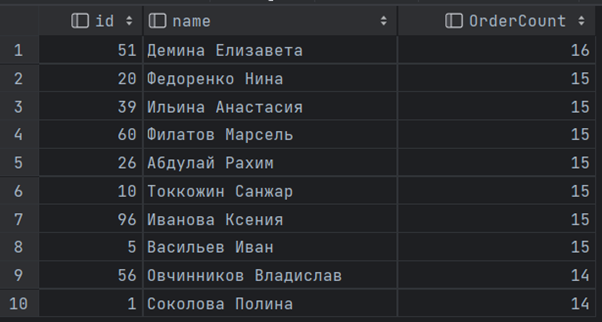
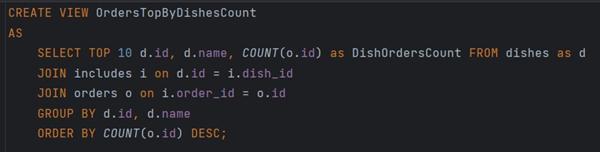


Рисунок 35, 36. Представление TopCustomersOrders

1. OrdersTopByDishesCount - ID, названия и количество заказов с наибольшим количеством заказов отдельного блюда. Это представление может быть полезным для анализа и отслеживания продаж по каждому блюду.



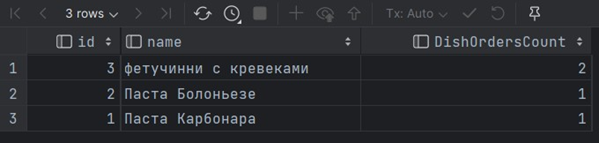


Рисунок 37, 38. Представление OrdersTopByDishesCount

## **4.5. Запросы**

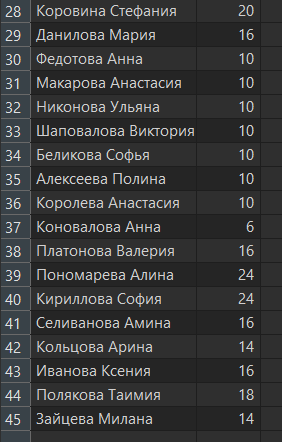
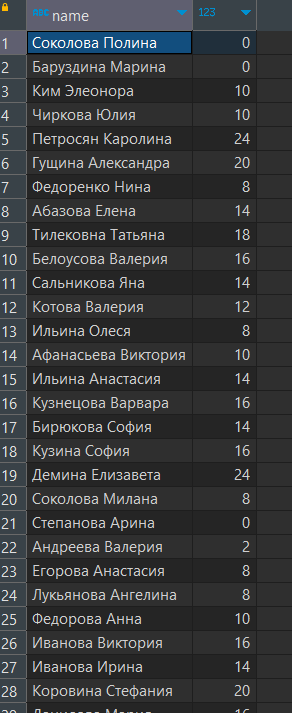
* Простой запрос с условием и формулами в SELECT – 2;

а) запрос на скидки(discount) женщинам с учетом повышения скидки в 2 раза

**select** name, discount \* 2

**from** clients

**where** sex = 'G'



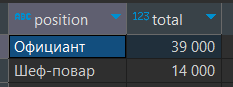
б) запрос на то, кому нужно повысить зарплату на 10%(люди, которые работают больше 1-ого года получают повышение). Вывести траты на повышение зп.

**select** [position], **SUM**(salary \* 0.1) **as** total

**from** employee

**WHERE** **DATEDIFF**(**year**, start\_date, '2023-06-19') >= 1

**GROUP** **BY** [position]



* Запрос с коррелированным подзапросом в SELECT – 2;

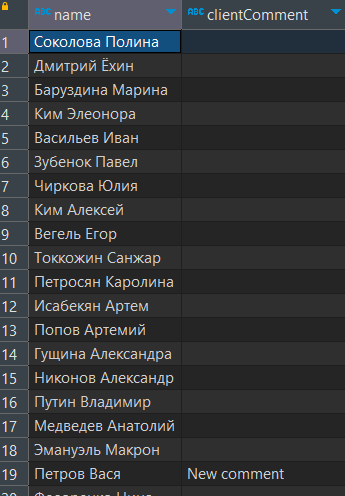
Запрос с коррелированным подзапросом в select:

1. смотрим имя и комментарии клиентов

select name,

(select top 1 o.comment from orders o where [c.id](https://vk.com/away.php?to=http%3A%2F%2Fc.id&cc_key=) = o.client\_id) as clientComment

from clients c

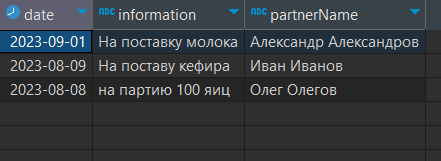


2. Показывает какой партнер что поставляет и в какую дату

select c.[date], c.information,

(select [p.name](https://vk.com/away.php?to=http%3A%2F%2Fp.name&cc_key=) from partner p where [p.id](https://vk.com/away.php?to=http%3A%2F%2Fp.id&cc_key=) = [c.id](https://vk.com/away.php?to=http%3A%2F%2Fc.id&cc_key=)) as partnerName

from contract c

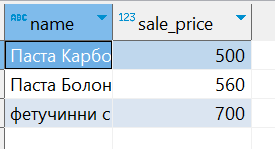


* Запрос с подзапросом в FROM – 2;

1. Этот запрос выбирает названия и цены блюд, которые содержат ингредиенты с ценой менее 100из таблицы "dishes" и связывает их с таблицей "contains" по идентификатору блюда.

**SELECT** name, sale\_price **FROM** (**SELECT** \* **FROM** dishes) **AS** d **JOIN** [contains] c **ON** d.id = c.dish\_id **WHERE** c.ingr\_id **IN**

(**SELECT** id **FROM** ingredient **WHERE** price < 100);



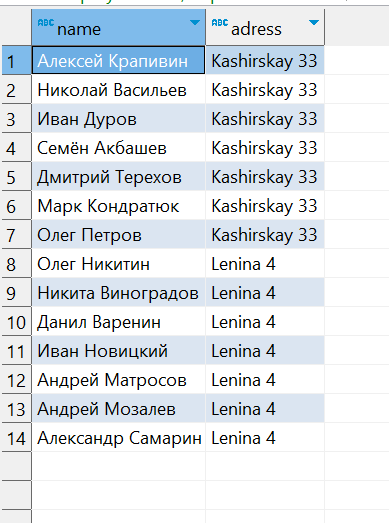
1. Этот запрос выводит имена и адреса отделов компании, в которых работает не менее 5 сотрудников, исходя из данных в таблицах "employee" и "department"

**SELECT** employee.name, department.adress **FROM** (**SELECT** dept\_id, **COUNT**(\*) **as** num\_employees

**FROM** employee **GROUP** **BY** dept\_id) **as** subquery **INNER** **JOIN** employee **ON** subquery.dept\_id = employee.dept\_id

**INNER** **JOIN** department **ON** subquery.dept\_id = department.id

**WHERE** subquery.num\_employees >= 5;



* Запрос с подзапросом в FROM, агрегированием, группировкой и сортировкой – 1;

В запросе выбраны название блюда и общая стоимость всех его ингредиентов. Использованы таблицы dishes, contains и ingredient, связанные через внешние ключи. Выборка производится по равенству id блюда и dish\_id в таблице contains, присоединении ингредиента по ingr\_id. Далее выбираются только те блюда, у которых больше или равно 2 ингредиентов и суммируются их стоимости. Результат сортируется сначала по убыванию суммарной стоимости и потом по возрастанию названий блюд.

SELECT d.name, SUM(i.price) as total\_cost

FROM dishes d

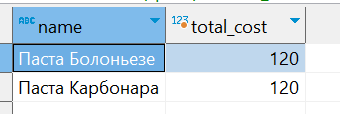
JOIN [contains] c ON d.id = c.dish\_id

JOIN ingredient i ON c.ingr\_id = i.id

GROUP BY d.id, d.name

HAVING COUNT(c.ingr\_id) >= 2

ORDER BY total\_cost DESC, d.name ASC;



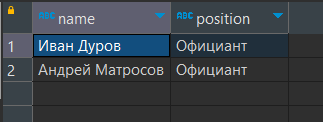
* Запрос с коррелированным подзапросом в WHERE – 2;

1. Вывести имя, должность, который работал в определенный день

select [e.name](https://vk.com/away.php?to=http%3A%2F%2Fe.name&cc_key=), e.[position] from employee e

where '2021-12-10' in

(Select o.[date] from orders o where o.employee\_id = [e.id](https://vk.com/away.php?to=http%3A%2F%2Fe.id&cc_key=))

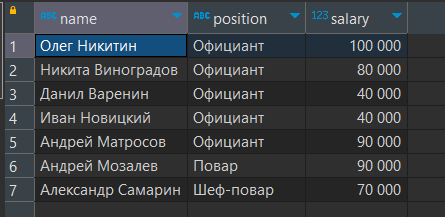


2. Вывести имя, должность и зарплату сотрудника который работал в ресторане по адресу ленина 4

select [e.name](https://vk.com/away.php?to=http%3A%2F%2Fe.name&cc_key=), e.[position], e.salary from employee e

where 'Lenina 4' in

(Select d.adress from department d where [d.id](https://vk.com/away.php?to=http%3A%2F%2Fd.id&cc_key=) = e.dept\_id)



* Запрос, использующий оконную функцию LAG или LEAD для выполнения сравнения данных в разных периодах – 1;

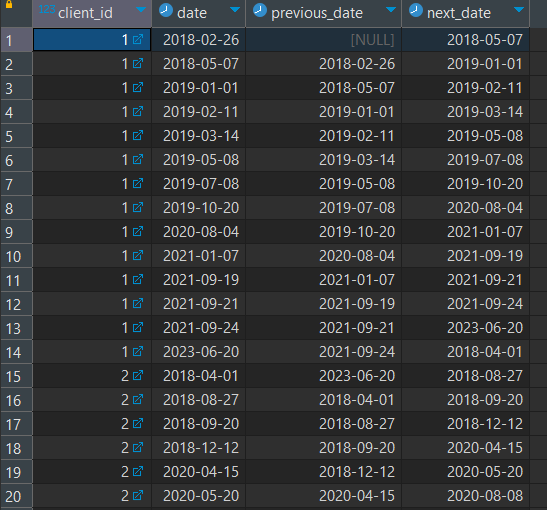
запрос будет полезен при анализе того как часто клиент посещает ресторан, какие перерывы делает между посещениями

**SELECT** client\_id, [date],

lag([date])**over** (**order** **by** client\_id, [date]) **as** previous\_date,

lead ([date])**over** (**order** **by** client\_id, [date]) **as** next\_date

**FROM** orders



* Запрос с агрегированием и выражением JOIN, включающим не менее 2 таблиц – 3;

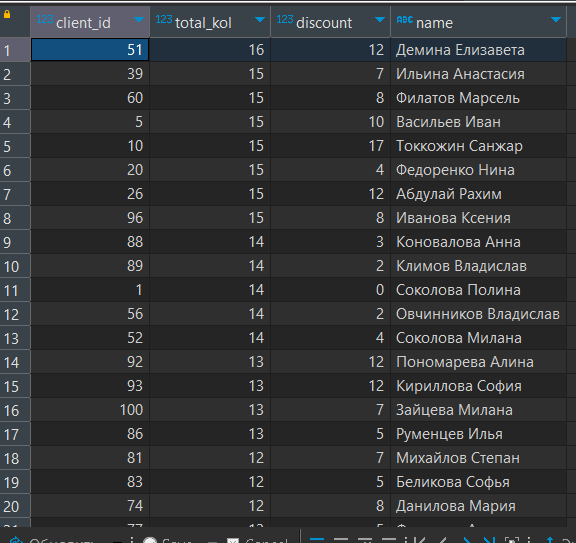
а) Запрос показывает с какими скидочными картами люди чаще посещают ресторан

**SELECT** client\_id, **COUNT**(\*) **as** total\_kol, discount, name

**FROM** orders o **join** clients c **on** o.client\_id = c.id

**GROUP** **BY** client\_id , discount, name

**ORDER** **BY** total\_kol **DESC**



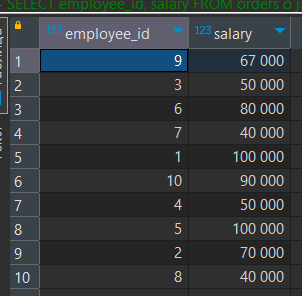
б)

**SELECT** employee\_id, salary

**FROM** orders o **join** employee e **on** o.employee\_id = e.id

**GROUP** **BY** employee\_id, salary

**HAVING** **SUM**(tips) > **AVG**(o.tips)



в) Запрос показывает какую часть дохода официант получает чаевыми(в зависимости от зп).

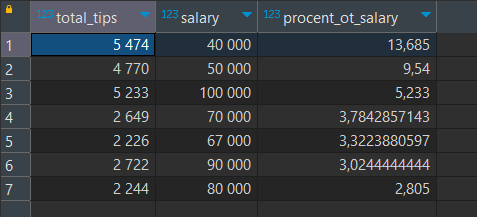
Можно заметить что наибольший процент у официантов с наименьшей зп.

**SELECT SUM(tips) as total\_tips, salary, SUM(tips)/salary \* 100 as procent\_ot\_salary**

**FROM orders o join employee e on o.employee\_id = e.id**

**GROUP BY salary**

**ORDER BY procent\_ot\_salary DESC**



* Запрос с EXISTS – 1;

Запрос позволяет узнать какие сотрудники привязаны к департаменту Ленина 4

**SELECT** id, name

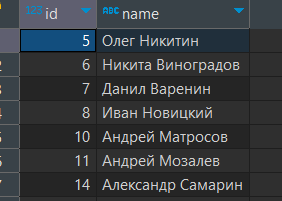
**FROM** employee e

**WHERE** **EXISTS** (

**SELECT** 1

**FROM** department d

**WHERE** e.dept\_id = d.id **AND** adress = 'Lenina 4')



* Запрос, использующий манипуляции с множествами – 1;

Запрос выводит сотрудников работающих в указанную неделю

**SELECT** id

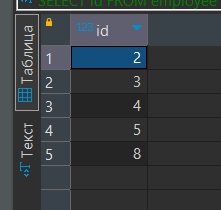
**FROM** employee e

**INTERSECT**

**SELECT** employee\_id

**FROM** orders o

**WHERE** [date] > '2018-01-07' **AND** [date] < '2018-01-14'



* Запрос с внешним соединением и проверкой на наличие NULL – 1;

В запросе смотрим из-за чего и какие клиенты оставили мало чаевых(читаем отзывы).

**SELECT** \*

**FROM** clients c

**LEFT** **OUTER** **JOIN** orders **ON** c.id = orders.client\_id

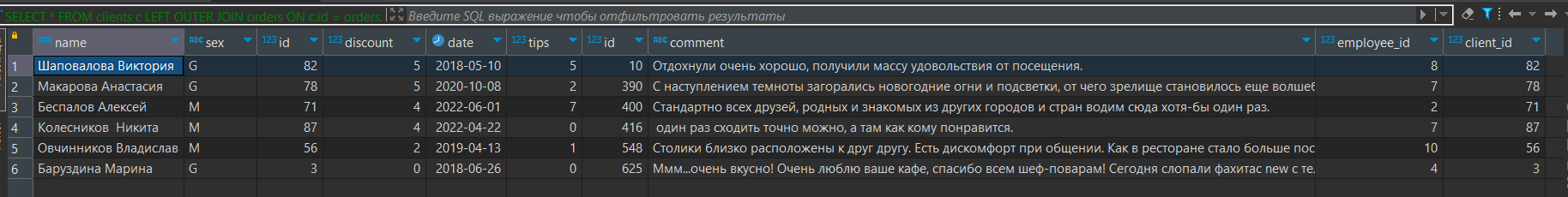
**WHERE** orders.id **IS** **NOT** **NULL** **and** comment != '' **and** tips < 10

**SELECT** \*

**FROM** clients c

**LEFT** **OUTER** **JOIN** orders **ON** c.id = orders.client\_id

**WHERE** orders.id **IS** **NOT** **NULL** **and** comment != '' **and** tips < 10

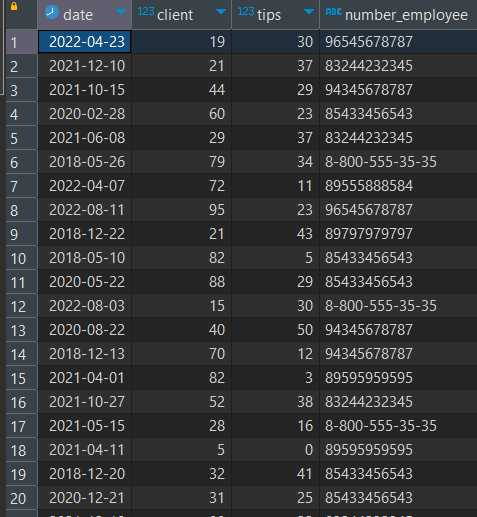


* Запрос с агрегированием и выражением JOIN, включающим не менее 3 таблиц/выражений – 1;

Данный запрос полезен при отправке чаевых на номер официанту(в запросе клиент. сумма чаевых, номер официанта, дата заказа)

**SELECT** o.[date] ,c.id **as** client, o.tips, e.phone\_number **as** number\_employee

**FROM** clients c **join** orders o **on** o.client\_id = c.id **join** employee e **on** e.id =o.employee\_id



* Запрос с CASE (IIF) и агрегированием – 1;

Запрос показывает кому надо вручить премию(лучший сотрудник по чаевым за месяц)

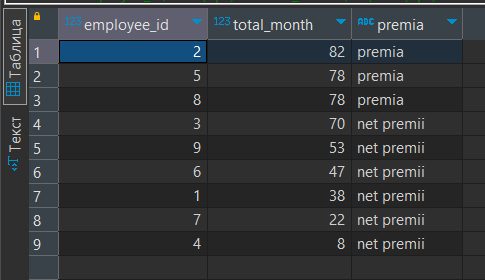
**SELECT** employee\_id, **SUM**(tips) **AS** total\_month , IIF(**SUM**(tips) > 70, 'premia', 'net premii') **AS** premia

**FROM** orders

**WHERE** [date] > '2018-01-01' **AND** [date] < '2018-01-31'

**GROUP** **BY** employee\_id

**ORDER** **BY** **SUM**(tips) **DESC**



* Запрос с HAVING и агрегированием – 1;

Официанты, которые заработали чаевых меньше 2500

**SELECT** name

**FROM** employee

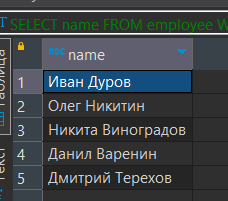
**WHERE** id **IN**(

**SELECT** employee\_id

**FROM** orders

**GROUP** **BY** employee\_id

**HAVING** **SUM**(tips) < 2500)



* Запрос SELECT INTO для подготовки выгрузки – 1.

Запрос показывает зарплатную ведомость, зп и чаевые(создает новую таблицу)

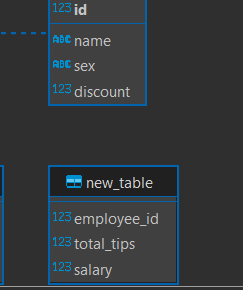
**SELECT** employee\_id, **SUM**(tips) **AS** total\_tips, (**SELECT** salary **FROM** employee e **WHERE** e.id = orders.employee\_id) **AS** salary

**INTO** new\_table

**FROM** orders

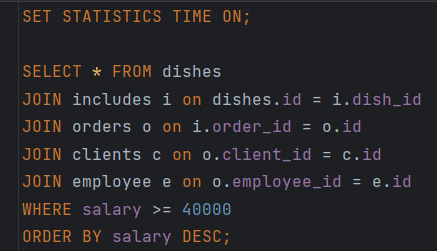
**GROUP** **BY** employee\_id

**ORDER** **BY** employee\_id



## **4.6. Индексы**

SQL-запрос, для которого будем измерять метрики до и после добавления индекса:



Скрипт создания индекса:

CREATE NONCLUSTERED INDEX IX\_Salary ON employee (salary) INCLUDE (name, phone\_number);

Результат SQL-запроса до добавления индекса:



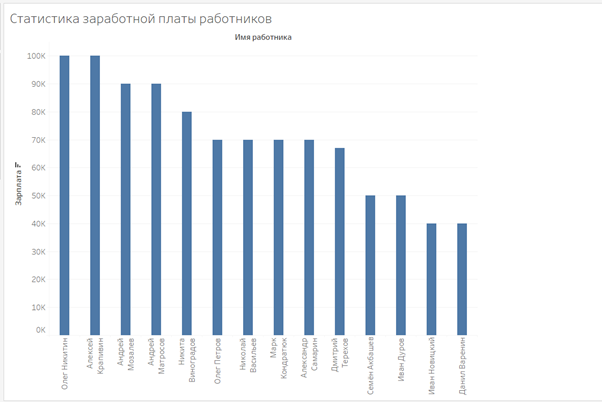
Результат SQL-запроса после добавления индекса:



# **ГЛАВА 5. ОТЧЁТНЫЕ ФОРМЫ**

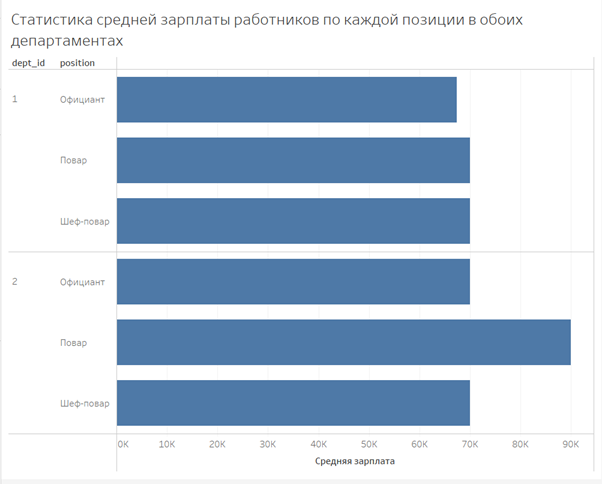
## **5.1. Основные отчеты**

Основной отчёт представлен статистикой заработной платы работников, начиная с наибольших значений. График помогает визуализировать и анализировать данные по оплате труда, что способствует принятию информированных решений и обеспечению справедливости внутри организации.

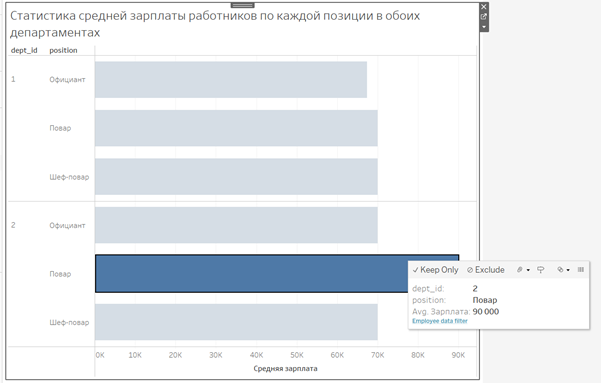


## **5.2. DrillDown**

Drilldown отчёт представлен статистикой средней заработной платы работников в зависимости от их департамента и занимаемой должности. Имеет большое значение для анализа различий в показателях для работников, занимаемых одинаковые должности, но находящихся в разных департаментах. В частности, из статистики ниже видно, что Повар во втором департаменте в среднем имеет заработную плату на порядок выше, чем такой же Повар в первом департаменте, что является сигналом о пересмотре заработной платы для работников данной должности.



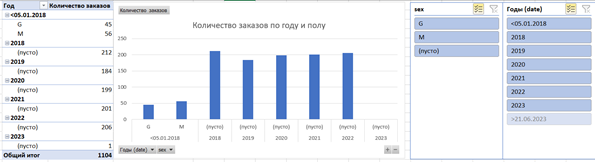
При нажатии на показатель раскрывается подробная статистика по работнику, в частности его департамент и занимаемая должность.



## **5.3. Excel**

В качестве Excel отчёта была выбрана статистика количества заказов в зависимости от года и пола клиента с подробным отображением количества заказов по каждому критерию.

Позволяет определить сезонные колебания в спросе на продукты а также предпочтения клиентов.



# **СПИСОК ИСТОЧНИКОВ**

1. Tableau // URL: <https://www.tableau.com/>
2. Database administration tool DBeaver // DBeaver Corp., US. 2022. URL: <https://dbeaver.com/>
3. Microsoft Excel // URL: https://www.microsoft.com/ru-ru/microsoft-365/excel