ОГЛАВЛЕНИЕ

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:

a) 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.

b) 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.

c) Руководители компании: Просмотр общей информации о компании. Просмотр отчетов о продажах, заказах и клиентах. Анализ производительности компании.

d) Клиенты: Просмотр меню блюд. Оформление заказов. Просмотр информации о своих заказах и скидках.

e) 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

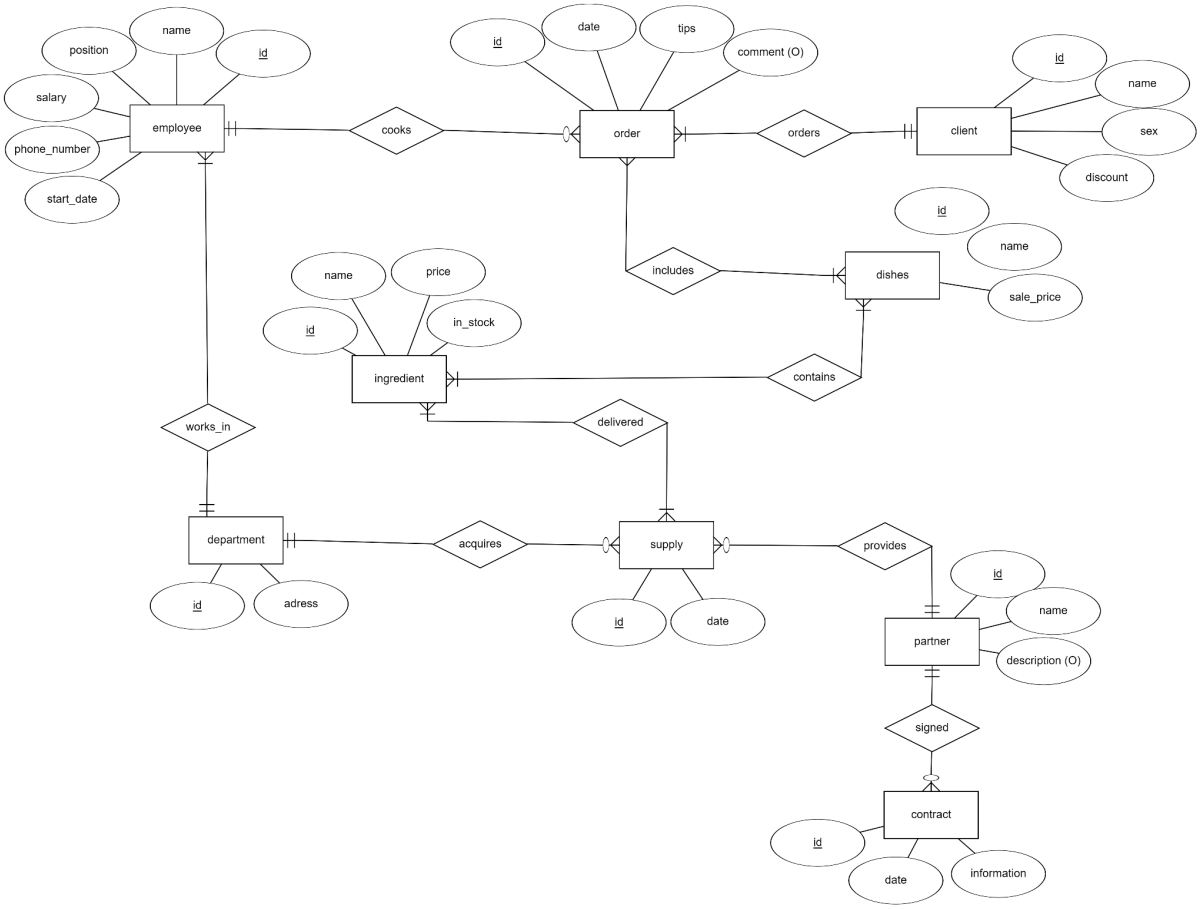


Рисунок 1. Схема инфологической модели предметной области

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

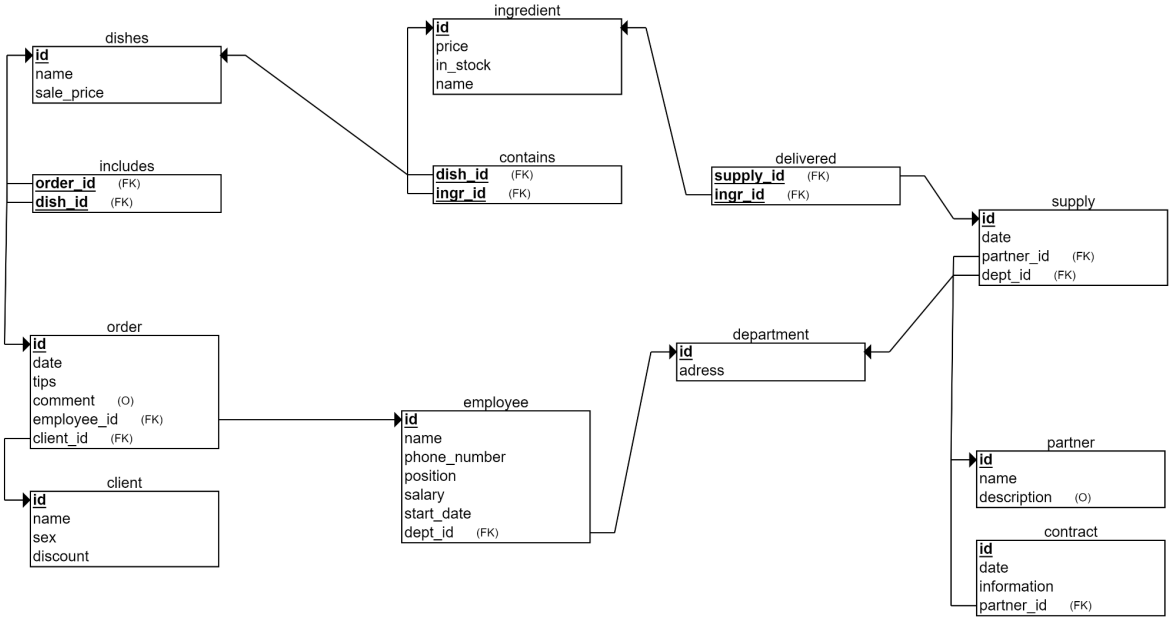


Рисунок 2. Схема даталогической модели базы данных

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

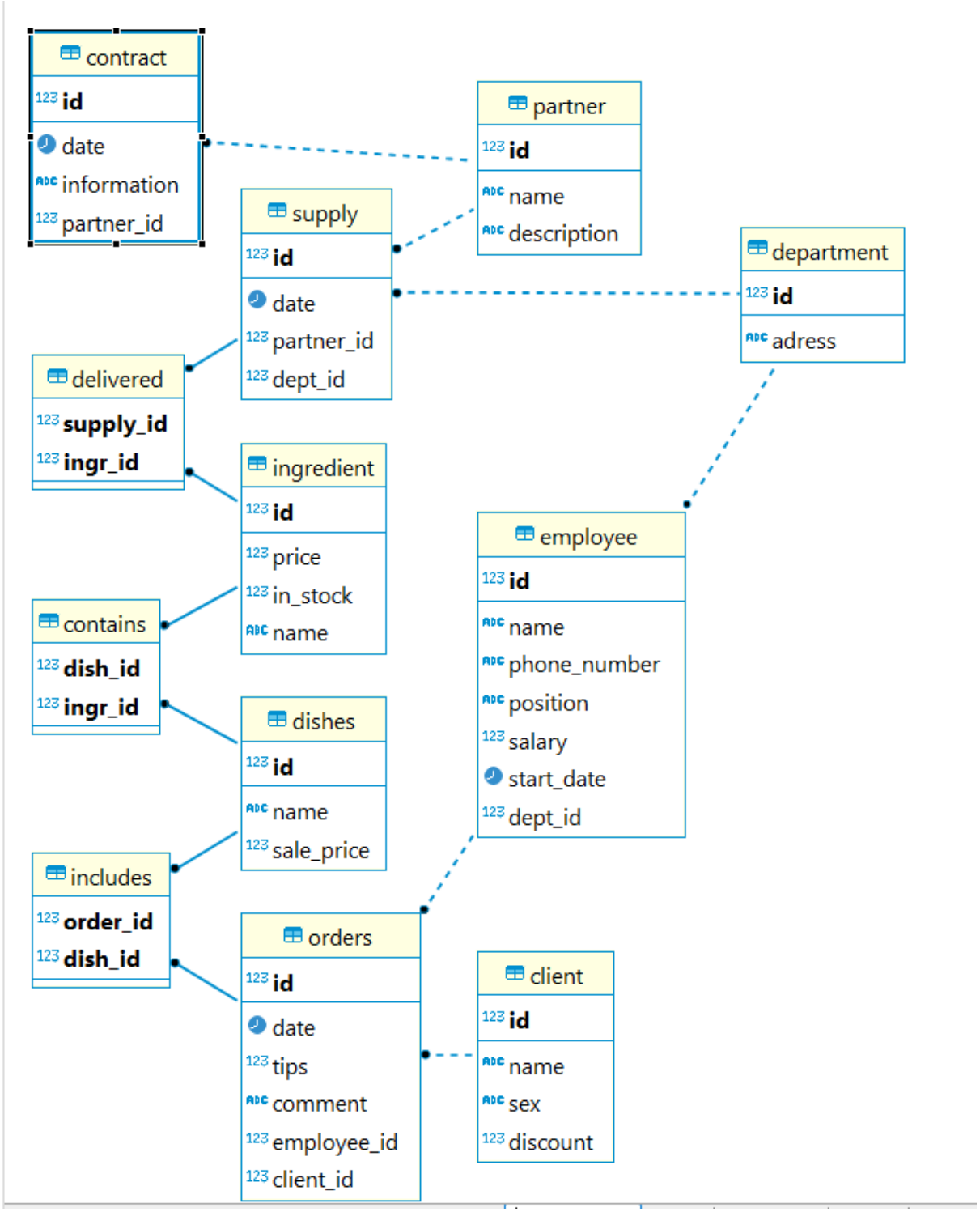


Рисунок 3. Схема модели базы данных, реализованной в СУБД

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

2) 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

3) 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

4) 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

5) Таблица *Department*, хранящая в себе адреса всех филиалов

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

6) 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

7) 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.

8) 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.

9) 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

10) 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

11) 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

12) 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.

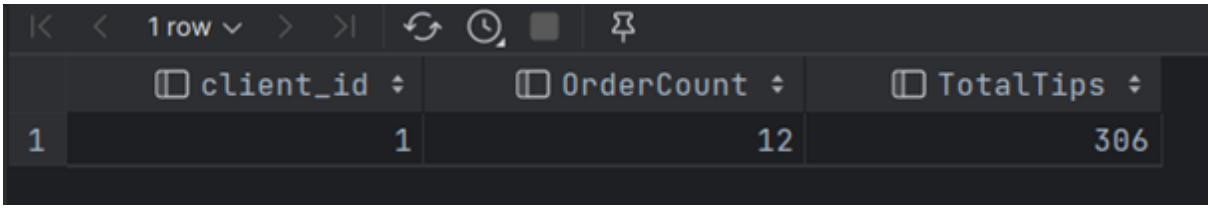
Описание: процедура предназначена для получения количества заказов и общей стоимости для определённого пользователя.

```
CREATE PROCEDURE GetClientsOrders
@ClientId INT
AS
BEGIN
    IF EXISTS (SELECT 1  FROM clients WHERE Id = @ClientId)
        BEGIN
            SELECT DISTINCT orders.client_id, OrderCount, TotalTips
            FROM orders
            JOIN (
                SELECT client_id, COUNT(*) AS OrderCount
                FROM orders
                GROUP BY client_id
            ) as OC ON OC.client_id = orders.client_id
            JOIN (
                SELECT client_id, SUM(tips) AS TotalTips
                FROM orders
                GROUP BY client_id
            ) as TT ON TT.client_id = orders.client_id
            WHERE orders.client_id = @ClientId
        END
    ELSE
        BEGIN
            PRINT 'Invalid CustomerId'
        END
END
```

Рисунок 4. Процедура GetClientsOrders

```
EXECUTE dbo.GetClientsOrders @ClientId = 1;
```

Рисунок 5. Скрипт выполнения процедуры GetClientsOrders



	client_id	OrderCount	TotalTips
1	1	12	306

Рисунок 6. Результат выполнения процедуры GetClientsOrders

2) 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.

```
CREATE PROCEDURE AddOrder
@date DATETIME,
@tips INT,
@id INT,
@comment VARCHAR(300),
@employee_id INT,
@client_id INT
AS
BEGIN
    BEGIN TRY
        BEGIN TRANSACTION
            INSERT INTO orders (date, tips, id, comment, employee_id, client_id)
            VALUES (@date, @tips, @id, @comment, @employee_id, @client_id)
        COMMIT
    END TRY
    BEGIN CATCH
        ROLLBACK
        PRINT 'An error occurred: ' + ERROR_MESSAGE()
    END CATCH
END
```

Рисунок 7. Процедура AddOrder

```
EXECUTE AddOrder '2019-01-01', 100, 123321, 'comment', 1, 1;
```

Рисунок 8. Скрипт вызова процедуры AddOrder

```
KDZ_2023_team15> EXECUTE AddOrder '2019-01-01', 100, 1, 'comment', 1, 1
An error occurred: Нарушено "PK__order1__3213E83FDF29E491" ограничения PRIMARY KEY. Не удается вставить
 повторяющийся ключ в объект "dbo.orders". Повторяющееся значение ключа: (1).
[2023-06-20 01:20:29] completed in 119 ms
```

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

```
KDZ_2023_team15> EXECUTE AddOrder '2019-01-01', 100, 123321, 'comment', 1, 1
[2023-06-20 01:21:19] 1 row affected in 106 ms
```

	date	tips	id	comment	employee_id	client_id
1	2019-01-01	100	123321	comment	1	1

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

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

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

```
CREATE PROCEDURE UpdateOrderComment @OrderId INT,
                                    @Comment VARCHAR(100)
AS
BEGIN
    IF EXISTS(SELECT 1 FROM orders WHERE Id = @OrderId)
        BEGIN
            UPDATE orders
            SET Comment = @Comment
            WHERE Id = @OrderId
        END
end
```

Рисунок 12. Процедура UpdateOrderComment

```
EXECUTE UpdateOrderComment 1, 'New comments';
```

Рисунок 13. Скрипт вызова процедуры UpdateOrderComment

	date	tips	id	comment	employee_id	client_id
1	2022-04-23	30	1		3	19

Рисунок 14. Состояние таблицы до выполнения процедуры UpdateOrderComment.

	date	tips	id	comment	employee_id	client_id
1	2022-04-23	30	1	New comment	3	19

Рисунок 15. Состояние таблицы после выполнения процедуры UpdateOrderComment.

4) ChangePhoneNumber - процедура обновления номера работника.

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

```
CREATE PROCEDURE ChangePhoneNumber @EmployeeId int,
                                   @PhoneNumber varchar(20)
AS
BEGIN
    SET NOCOUNT ON;
    IF @PhoneNumber LIKE '[0-9]-[0-9][0-9][0-9]-[0-9][0-9][0-9]-[0-9][0-9]-[0-9][0-9]'
        BEGIN
            UPDATE employee
            SET phone_number = @PhoneNumber
            WHERE id = @EmployeeId
        END
    ELSE
        BEGIN
            RAISERROR ('Phone number is not valid', 16, 1)
        END
END
```

Рисунок 16. Процедура ChangePhoneNumber

```
EXECUTE ChangePhoneNumber 1, '8-800-555-35-35'
```

Рисунок 17. Скрипт вызова процедуры ChangePhoneNumber

	name	phone_number	position	salary	start_date	id	dept_id
1	Алексей Крапивин	89876545677	Официант	100000	2006-02-21	1	1

Рисунок 18. Состояние таблицы employee до выполнения скрипта

	name	phone_number	position	salary	start_date	id	dept_id
1	Алексей Крапивин	8-800-555-35-35	Официант	100000	2006-02-21	1	1

Рисунок 19. Состояние таблицы employee после выполнения скрипта

```
EXECUTE ChangePhoneNumber 1, '88005553535'
```

```
KDZ_2023_team15> EXECUTE ChangePhoneNumber 1, '88005553535'
[2023-06-20 02:47:29] [S0001][50000] Line 1: Phone number is not valid
```

Рисунок 20, 21. Вывод консоли при попытке запуска скрипта

5) 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.

```
CREATE PROCEDURE AddOrderWithCurrentDate @tips INT,
                                         @id INT,
                                         @comment VARCHAR(300),
                                         @employee_id INT,
                                         @client_id INT
AS
BEGIN
    DECLARE @date date = GETDATE();
    EXEC AddOrder @date: @date, @tips: @tips, @id: @id, @comment: @comment, @employee_id: @employee_id, @client_id: @client_id
END;
```

Рисунок 22. Процедура AddOrderWithCurrentDate

```
EXECUTE AddOrderWithCurrentDate  100, 123123, 'comment', 1, 1;
```

Рисунок 23. Скрипт запуска процедуры AddOrderWithCurrentDate

```
SELECT * FROM orders
WHERE id = 123123;
```

	date	tips	id	comment	employee_id	client_id
1	2023-06-20	100	123123	comment	1	1

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.

```
CREATE TRIGGER UpdateProductStock
    ON delivered
    AFTER INSERT
    AS
BEGIN
    UPDATE ingredient
    SET in_stock = in_stock + 1
    FROM ingredient
    INNER JOIN inserted ON ingredient.id = inserted.ingr_id;
END
```

Рисунок 25. Добавлении в таблицу delivered записи

```
INSERT INTO delivered (supply_id, ingr_id)
VALUES (4, 1)
```

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

	price	in_stock	id	name
1	20	200	1	яйца
2	100	12	2	Мука
3	50	20	3	Молоко

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

	price	in_stock	id	name
1	20	201	1	яйца
2	100	12	2	Мука
3	50	20	3	Молоко

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.

```
CREATE FUNCTION DishIngredients(@DishId int)
RETURNS TABLE
AS RETURN
        (
            SELECT DISTINCT d.name as Dish, ingredient.name as Ingredient
            FROM ingredient
                     JOIN [contains] c on ingredient.id = c.ingr_id
                     JOIN dishes d on c.dish_id = d.id
            WHERE c.ingr_id IN (SELECT ingr_id FROM [contains] WHERE dish_id = @DishId)
            AND d.id = @DishId
        )
```

Рисунок 29. Вызов функции DishIngredients

```
SELECT * FROM DishIngredients(1);
```

	Dish	Ingredient
1	Паста Карбонара	Мука
2	Паста Карбонара	яйца

Рисунок 30, 31. Вызов функции и результат

2) GetOrderDishNameByClientId - функция определения списка всех заказанных блюд определённого клиента.

```
CREATE FUNCTION GetOderDishNameByClientId(@ClientId int)
RETURNS TABLE
AS RETURN
        (
            SELECT DISTINCT c2.name, d.name as Dish
            FROM dishes d
                     JOIN includes i on d.id = i.dish_id
                     JOIN orders o on i.order_id = o.id
                     JOIN clients c2 on o.client_id = c2.id
            WHERE o.client_id = @ClientId
        )
```

Рисунок 32. Вызов функции GetOrderDishNameByClientId

```
SELECT * FROM GetOderDishNameByClientId (19);
```

	name	Dish
1	Петров Вася	Паста Болоньезе
2	Петров Вася	Паста Карбонара
3	Петров Вася	фетучинни с кревеками

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.

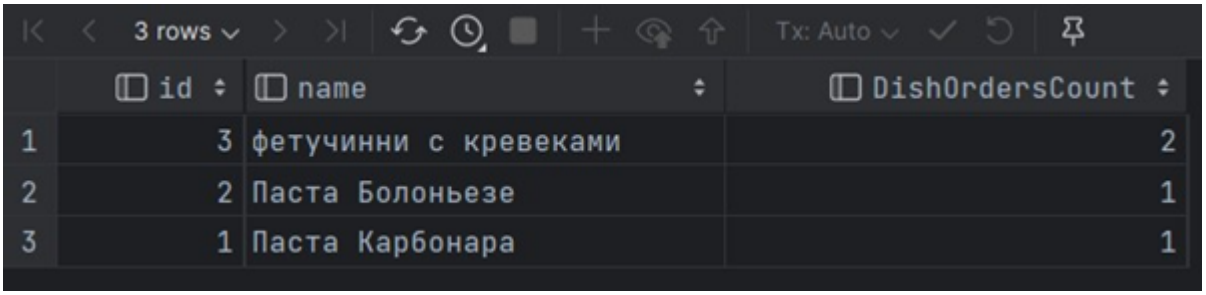
```sql
CREATE VIEW TopCustomersOrders
AS
SELECT TOP 10 c.id, c.name, COUNT(o.id) AS OrderCount
FROM clients c
         INNER JOIN Orders o ON c.id = o.client_id
GROUP BY c.id, c.name
ORDER BY OrderCount DESC;
```

	id	name	OrderCount
1	51	Демина Елизавета	16
2	20	Федоренко Нина	15
3	39	Ильина Анастасия	15
4	60	Филатов Марсель	15
5	26	Абдулай Рахим	15
6	10	Токкожин Санжар	15
7	96	Иванова Ксения	15
8	5	Васильев Иван	15
9	56	Овчинников Владислав	14
10	1	Соколова Полина	14

Рисунок 35, 36. Представление TopCustomersOrders

2) 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.

```sql
CREATE VIEW OrdersTopByDishesCount
AS
    SELECT TOP 10 d.id, d.name, COUNT(o.id) as DishOrdersCount FROM dishes as d
    JOIN includes i on d.id = i.dish_id
    JOIN orders o on i.order_id = o.id
    GROUP BY d.id, d.name
    ORDER BY COUNT(o.id) DESC;
```



	id	name	DishOrdersCount
1	3	фетучинни с кревеками	2
2	2	Паста Болоньезе	1
3	1	Паста Карбонара	1

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'
```

	name	123
1	Соколова Полина	0
2	Баруздина Марина	0
3	Ким Элеонора	10
4	Чиркова Юлия	10
5	Петросян Каролина	24
6	Гущина Александра	20
7	Федоренко Нина	8
8	Абазова Елена	14
9	Тилековна Татьяна	18
10	Белоусова Валерия	16
11	Сальникова Яна	14
12	Котова Валерия	12
13	Ильина Олеся	8
14	Афанасьева Виктория	10
15	Ильина Анастасия	14
16	Кузнецова Варвара	16
17	Бирюкова София	14
18	Кузина София	16
19	Демина Елизавета	24
20	Соколова Милана	8
21	Степанова Арина	0
22	Андреева Валерия	2
23	Егорова Анастасия	8
24	Лукьянова Ангелина	8
25	Федорова Анна	10
26	Иванова Виктория	16
27	Иванова Ирина	14
28	Коровина Стефания	20

28	Коровина Стефания	20
29	Данилова Мария	16
30	Федотова Анна	10
31	Макарова Анастасия	10
32	Никонова Ульяна	10
33	Шаповалова Виктория	10
34	Беликова Софья	10
35	Алексеева Полина	10
36	Королева Анастасия	10
37	Коновалова Анна	6
38	Платонова Валерия	16
39	Пономарева Алина	24
40	Кириллова София	24
41	Селиванова Амина	16
42	Кольцова Арина	14
43	Иванова Ксения	16
44	Полякова Таимия	18
45	Зайцева Милана	14

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]
```

position	total
Официант	39 000
Шеф-повар	14 000

- 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 = o.client_id) as clientComment
from clients c
```

	name	clientComment
1	Соколова Полина	
2	Дмитрий Ёхин	
3	Баруздина Марина	
4	Ким Элеонора	
5	Васильев Иван	
6	Зубенок Павел	
7	Чиркова Юлия	
8	Ким Алексей	
9	Вегель Егор	
10	Токкожин Санжар	
11	Петросян Каролина	
12	Исабекян Артем	
13	Попов Артемий	
14	Гущина Александра	
15	Никонов Александр	
16	Путин Владимир	
17	Медведев Анатолий	
18	Эмануэль Макрон	
19	Петров Вася	New comment

2. Показывает какой партнер что поставляет и в какую дату

```
select c.[date], c.information,
(select p.name from partner p where p.id = c.id) as partnerName
```

from contract c

date	information	partnerName
2023-09-01	На поставку молока	Александр Александров
2023-08-09	На поставу кефира	Иван Иванов
2023-08-08	на партию 100 яиц	Олег Олегов

- 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);
```

name	sale_price
Паста Карбо	500
Паста Болон	560
фетучинни с	700

  2. 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;
```

	name	adress
1	Алексей Крапивин	Kashirskay 33
2	Николай Васильев	Kashirskay 33
3	Иван Дуров	Kashirskay 33
4	Семён Акбашев	Kashirskay 33
5	Дмитрий Терехов	Kashirskay 33
6	Марк Кондратюк	Kashirskay 33
7	Олег Петров	Kashirskay 33
8	Олег Никитин	Lenina 4
9	Никита Виноградов	Lenina 4
10	Данил Варенин	Lenina 4
11	Иван Новицкий	Lenina 4
12	Андрей Матросов	Lenina 4
13	Андрей Мозалев	Lenina 4
14	Александр Самарин	Lenina 4

- 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;
```

name	total_cost
Паста Болоньезе	120
Паста Карбонара	120

- Запрос с коррелированным подзапросом в WHERE – 2;

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

```
select e.name, e.[position] from employee e
where '2021-12-10' in
(Select o.[date] from orders o where o.employee_id = e.id)
```

	name	position
1	Иван Дуров	Официант
2	Андрей Матросов	Официант

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

```
select e.name, e.[position], e.salary from employee e
where 'Lenina 4' in
(Select d.adress from department d where d.id = e.dept_id)
```

	name	position	salary
1	Олег Никитин	Официант	100 000
2	Никита Виноградов	Официант	80 000
3	Данил Варенин	Официант	40 000
4	Иван Новицкий	Официант	40 000
5	Андрей Матросов	Официант	90 000
6	Андрей Мозалев	Повар	90 000
7	Александр Самарин	Шеф-повар	70 000

- 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

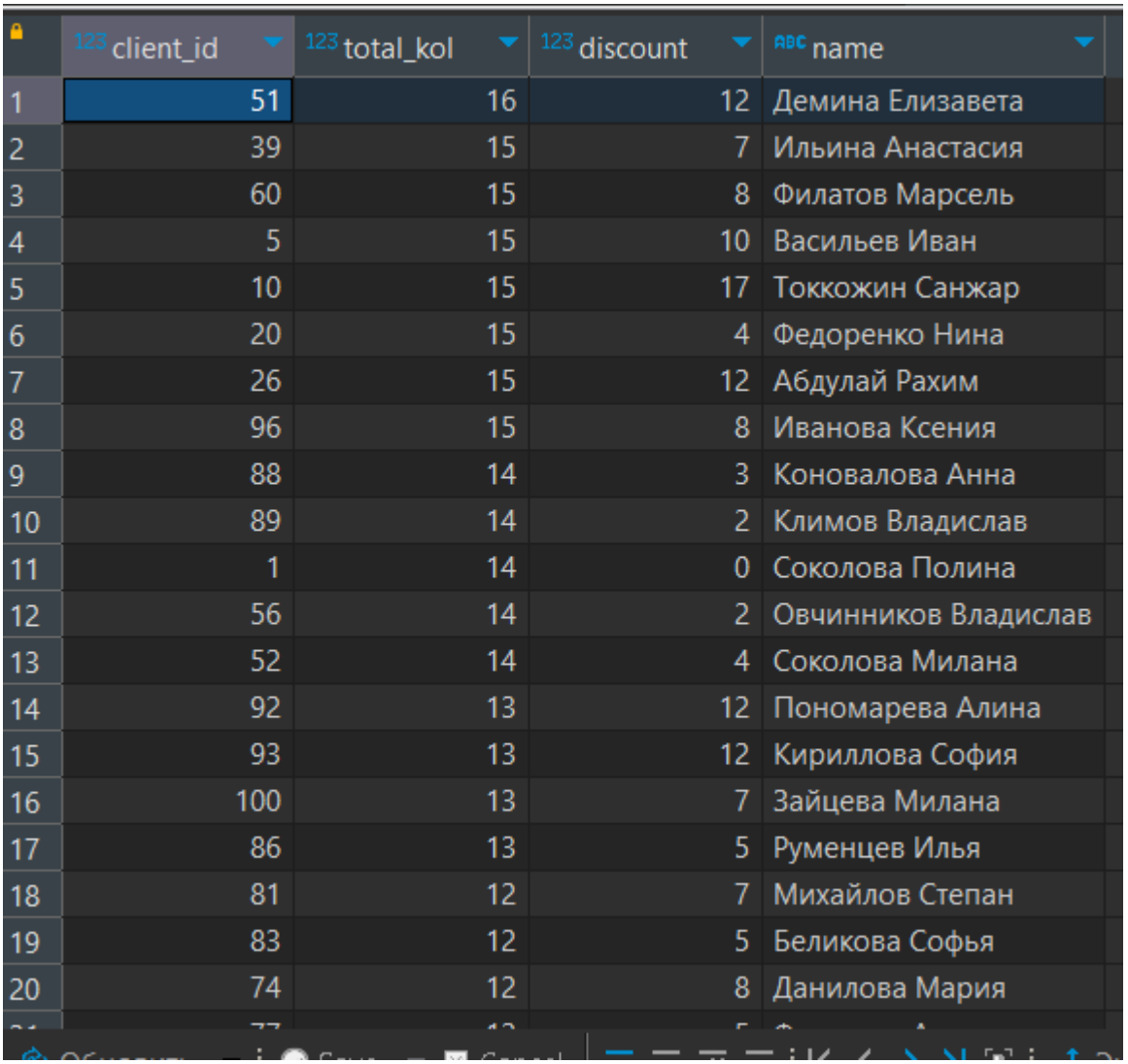
```
SELECT client_id, [date],
lag([date])over (order by client_id, [date]) as previous_date,
lead ([date])over (order by client_id, [date]) as next_date
FROM orders
```

	client_id	date	previous_date	next_date
1	1	2018-02-26	[NULL]	2018-05-07
2	1	2018-05-07	2018-02-26	2019-01-01
3	1	2019-01-01	2018-05-07	2019-02-11
4	1	2019-02-11	2019-01-01	2019-03-14
5	1	2019-03-14	2019-02-11	2019-05-08
6	1	2019-05-08	2019-03-14	2019-07-08
7	1	2019-07-08	2019-05-08	2019-10-20
8	1	2019-10-20	2019-07-08	2020-08-04
9	1	2020-08-04	2019-10-20	2021-01-07
10	1	2021-01-07	2020-08-04	2021-09-19
11	1	2021-09-19	2021-01-07	2021-09-21
12	1	2021-09-21	2021-09-19	2021-09-24
13	1	2021-09-24	2021-09-21	2023-06-20
14	1	2023-06-20	2021-09-24	2018-04-01
15	2	2018-04-01	2023-06-20	2018-08-27
16	2	2018-08-27	2018-04-01	2018-09-20
17	2	2018-09-20	2018-08-27	2018-12-12
18	2	2018-12-12	2018-09-20	2020-04-15
19	2	2020-04-15	2018-12-12	2020-05-20
20	2	2020-05-20	2020-04-15	2020-08-08

- 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
```



	client_id	total_kol	discount	name
1	51	16	12	Демина Елизавета
2	39	15	7	Ильина Анастасия
3	60	15	8	Филатов Марсель
4	5	15	10	Васильев Иван
5	10	15	17	Токкожин Санжар
6	20	15	4	Федоренко Нина
7	26	15	12	Абдулай Рахим
8	96	15	8	Иванова Ксения
9	88	14	3	Коновалова Анна
10	89	14	2	Климов Владислав
11	1	14	0	Соколова Полина
12	56	14	2	Овчинников Владислав
13	52	14	4	Соколова Милана
14	92	13	12	Пономарева Алина
15	93	13	12	Кириллова София
16	100	13	7	Зайцева Милана
17	86	13	5	Руменцев Илья
18	81	12	7	Михайлов Степан
19	83	12	5	Беликова Софья
20	74	12	8	Данилова Мария

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)
```

SELECT employee_id, salary FROM orders o j

	employee_id	salary
1	9	67 000
2	3	50 000
3	6	80 000
4	7	40 000
5	1	100 000
6	10	90 000
7	4	50 000
8	5	100 000
9	2	70 000
10	8	40 000

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

Можно заметить что наибольший процент у официантов с наименьшей зп.

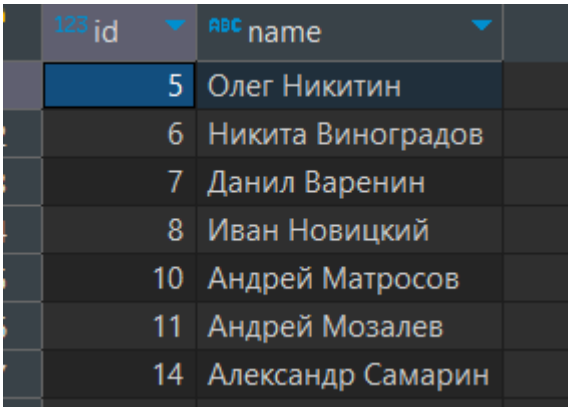
```
SELECT SUM(tips) as total_tips, salary, SUM(tips)/salary * 100 as procent_ot_salary
FROM orders o join employee e on o.employee_id = e.id
GROUP BY salary
ORDER BY procent_ot_salary DESC
```

	total_tips	salary	procent_ot_salary
1	5 474	40 000	13,685
2	4 770	50 000	9,54
3	5 233	100 000	5,233
4	2 649	70 000	3,7842857143
5	2 226	67 000	3,3223880597
6	2 722	90 000	3,0244444444
7	2 244	80 000	2,805

- 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')
```

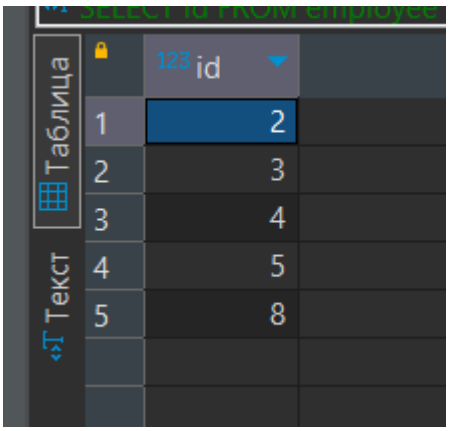


id	name
5	Олег Никитин
6	Никита Виноградов
7	Данил Варенин
8	Иван Новицкий
10	Андрей Матросов
11	Андрей Мозалев
14	Александр Самарин

- Запрос, использующий манипуляции с множествами – 1;

Запрос выводит сотрудников работающих в указанную неделю

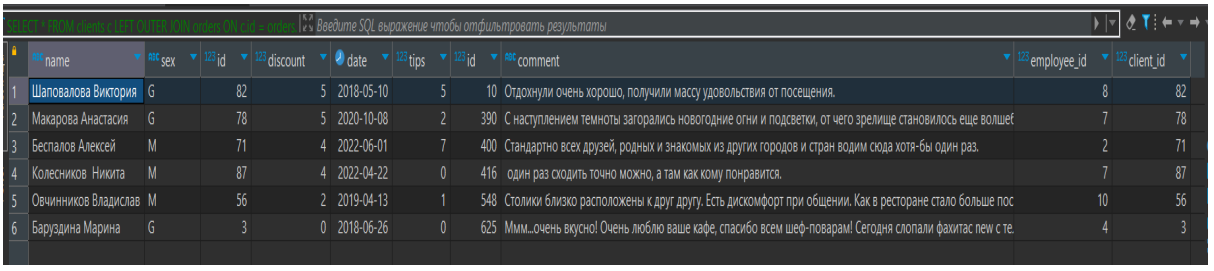
```
SELECT id
FROM employee e
INTERSECT
SELECT employee_id
FROM orders o
WHERE [date] > '2018-01-07' AND [date] < '2018-01-14'
```



- 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
```

	date	client	tips	number_employee
1	2022-04-23	19	30	96545678787
2	2021-12-10	21	37	83244232345
3	2021-10-15	44	29	94345678787
4	2020-02-28	60	23	85433456543
5	2021-06-08	29	37	83244232345
6	2018-05-26	79	34	8-800-555-35-35
7	2022-04-07	72	11	89555888584
8	2022-08-11	95	23	96545678787
9	2018-12-22	21	43	89797979797
10	2018-05-10	82	5	85433456543
11	2020-05-22	88	29	85433456543
12	2022-08-03	15	30	8-800-555-35-35
13	2020-08-22	40	50	94345678787
14	2018-12-13	70	12	94345678787
15	2021-04-01	82	3	89595959595
16	2021-10-27	52	38	83244232345
17	2021-05-15	28	16	8-800-555-35-35
18	2021-04-11	5	0	89595959595
19	2018-12-20	32	41	85433456543
20	2020-12-21	31	25	85433456543

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

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

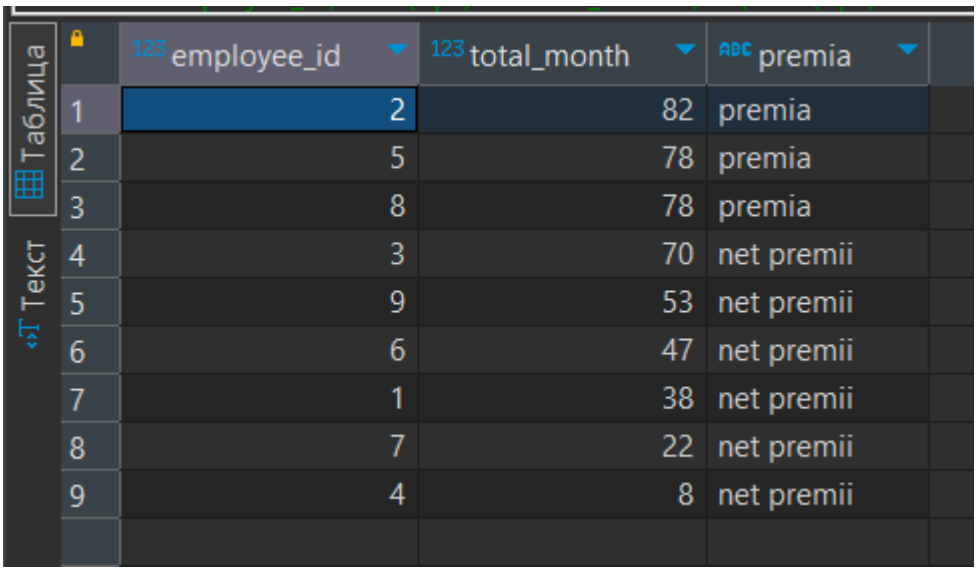
```
SELECT employee_id, SUM(tips) AS total_month , IIF(SUM(tips) > 70, 'premia', 'net premii') AS premia

FROM orders

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

GROUP BY employee_id

ORDER BY SUM(tips) DESC
```

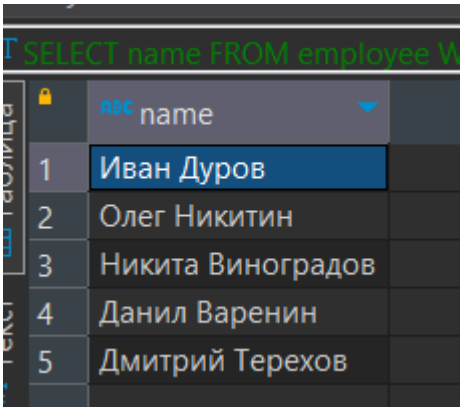


	employee_id	total_month	premia
1	2	82	premia
2	5	78	premia
3	8	78	premia
4	3	70	net premii
5	9	53	net premii
6	6	47	net premii
7	1	38	net premii
8	7	22	net premii
9	4	8	net premii

- 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)
```



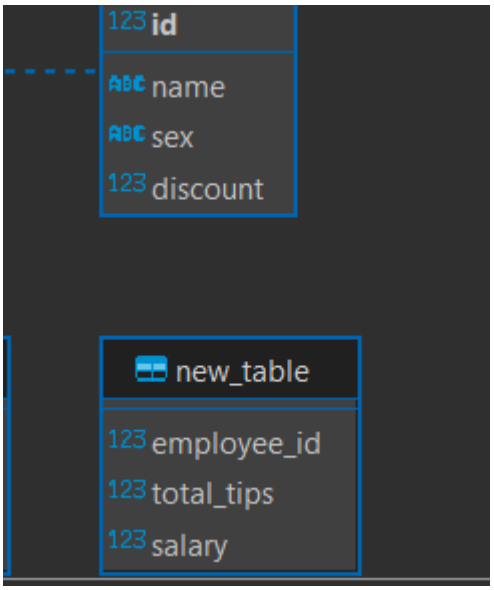
	name
1	Иван Дуров
2	Олег Никитин
3	Никита Виноградов
4	Данил Варенин
5	Дмитрий Терехов

- 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:

```
SET STATISTICS TIME ON;

SELECT * FROM dishes
JOIN includes i on dishes.id = i.dish_id
JOIN orders o on i.order_id = o.id
JOIN clients c on o.client_id = c.id
JOIN employee e on o.employee_id = e.id
WHERE salary >= 40000
ORDER BY salary DESC;
```

Скрипт создания индекса:

CREATE NONCLUSTERED INDEX IX_Salary ON employee (salary) INCLUDE (name, phone_number);

Результат SQL-запроса до добавления индекса:

```
[2023-06-20 13:58:46] 4 rows retrieved starting from 1 in 106 ms (execution: 57 ms, fetching: 49 ms)
```

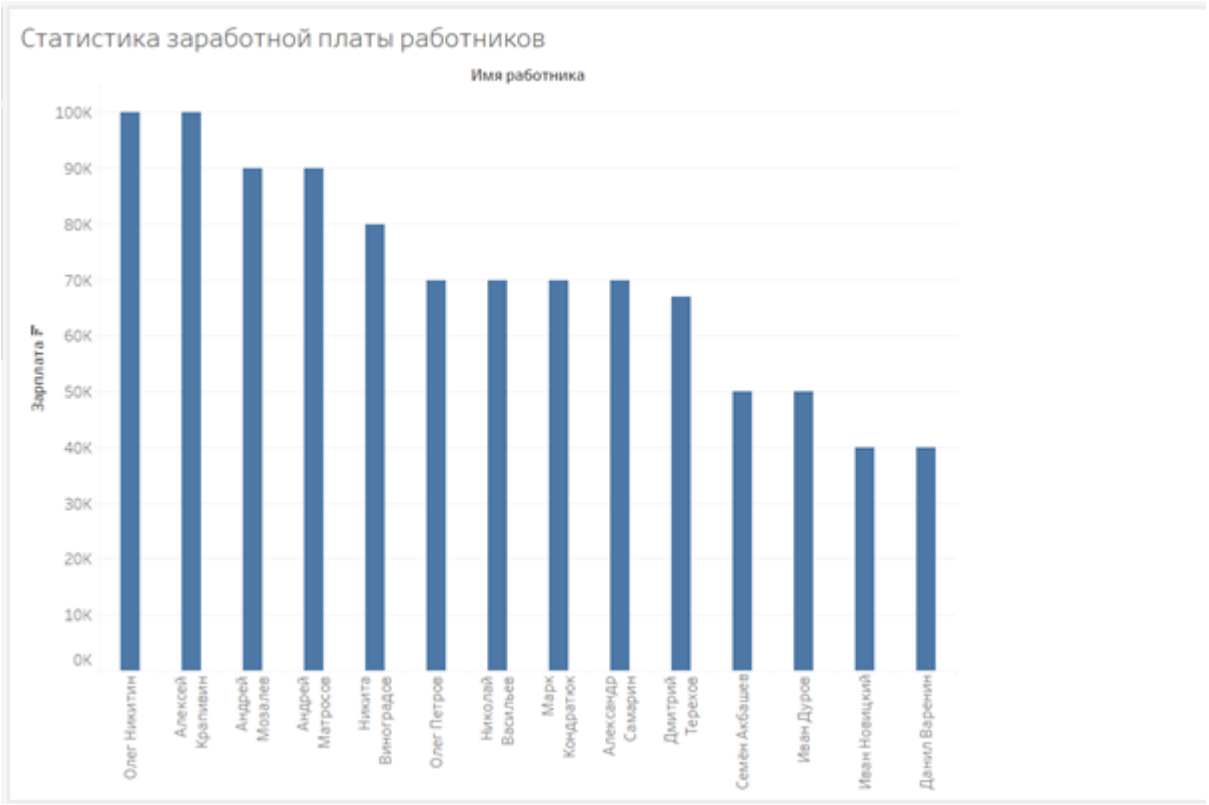
Результат SQL-запроса после добавления индекса:

```
[2023-06-20 13:59:41] 4 rows retrieved starting from 1 in 83 ms (execution: 25 ms, fetching: 58 ms)
```

ГЛАВА 5. ОТЧЁТНЫЕ ФОРМЫ

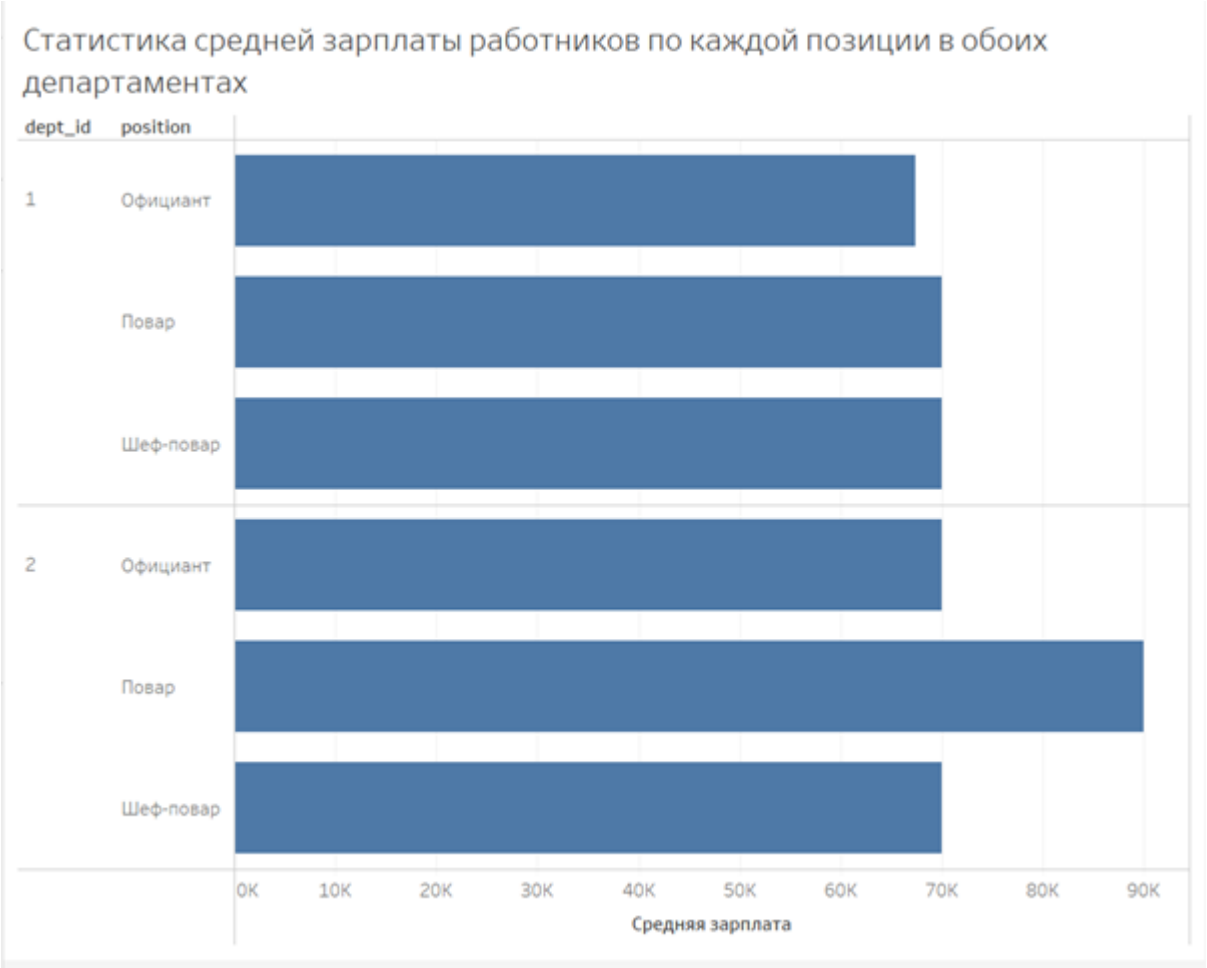
5.1. Основные отчеты

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.

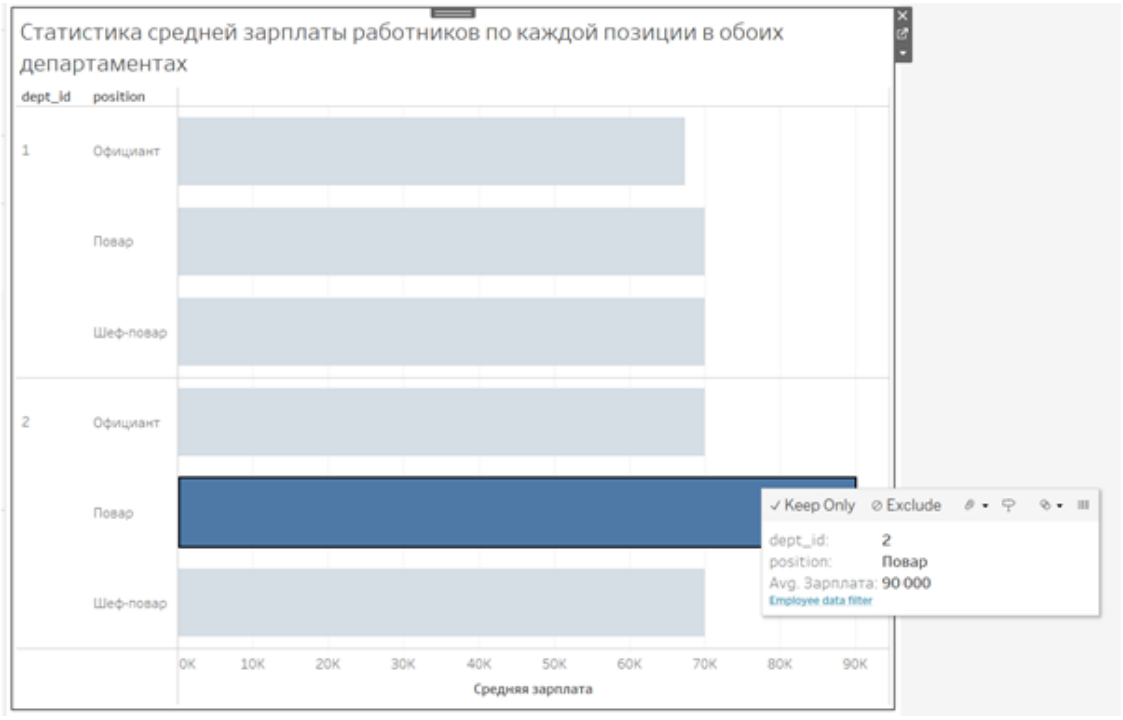


5.2. DrillDown

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.



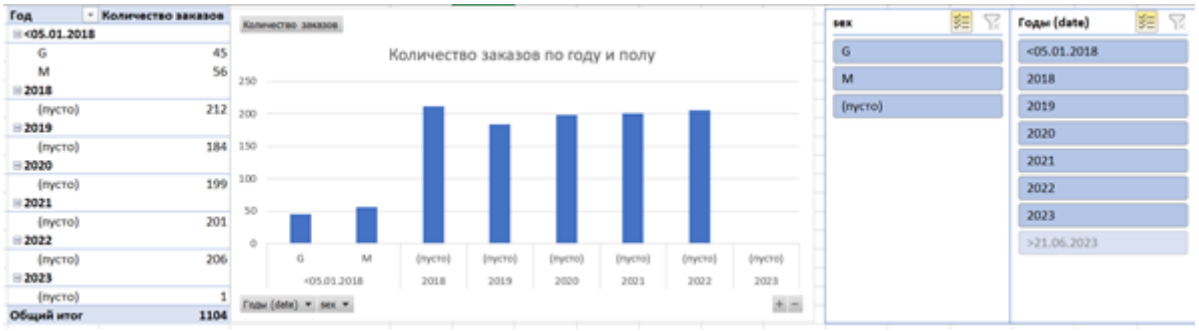
При нажатии на показатель раскрывается подробная статистика по работнику, в частности его департамент и занимаемая должность.



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