|  |
| --- |
|  |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования  **"МИРЭА — Российский технологический университет"**  **РТУ МИРЭА**   |  | | --- | | Институт информационных технологий (ИТ) | | Кафедра промышленной информатики (ПИ) | |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **ОТЧЕТ ПО ПРАКТИЧЕСКИМ РАБОТАМ** | | | | |
| **по дисциплине** | | | | |
| «Разработка баз данных» | | | | |
| **Студент группы** | ИКБО-36-22 Утенков Ю. Ю. |  |  |
|  | *(учебная группа, фамилия, имя, отчество студента)* |  | *(подпись студента)* |
| **Преподаватель** | ассистент Баев И. Б. |  |  |
| *(должность, ученая степень, звание, фамилия, имя, отчество преподавателя) (подпись преподавателя)* | | | |

|  |  |
| --- | --- |
| Работа выполнена | «20» сентября 2024 г. |
| «Зачтено» | «\_\_\_» \_\_\_\_\_\_\_\_ 2024 г. |
|  |  |

Москва 2024

Содержание

[Цель работы 3](#_Toc177767485)

[Ход работы 4](#_Toc177767486)

[Глава 1 4](#_Toc177767487)

[Глава 2 6](#_Toc177767488)

[Глава 3 11](#_Toc177767489)

[Глава 4 20](#_Toc177767490)

[Базовые SQL запросы с дополнениями 20](#_Toc177767491)

[Агрегатные функции 28](#_Toc177767492)

[Алгебра множеств 32](#_Toc177767493)

[Глава 5 41](#_Toc177767494)

[Триггер №1 41](#_Toc177767495)

[Триггер №2 42](#_Toc177767496)

[Триггер №3 45](#_Toc177767497)

[Триггер №4 46](#_Toc177767498)

[Триггер №5 48](#_Toc177767499)

[Глава 6 50](#_Toc177767500)

[Вывод 57](#_Toc177767501)

# Цель работы

1. Научится формировать модель БД с помощью инструментов СУБД (MySQL Workbench, dbForge Studio, PostgreSQL – по выбору студента) по своей теме.
2. Научится осуществлять перенос своей БД на другой сервер.
3. Изучить команды модификации данных (DML)
4. Осуществить выборку данных по своей теме с помощью различных операторов.
5. Изучить и применить к своей БД хранимые процедуры, функции и триггеры.

За время практикума каждый студент должен разработать собственное приложение баз данных, которое обязательно должно включать следующие элементы:

1. Создание логической модели базы данных. Описание ER-модели, генерация на ее основе реляционной модели данных.
2. Реализация модели в СУБД. В качестве СУБД могут быть выбраны: MS SQL Server, MySQL или PostgreSQL или иное серверное СУБД.
3. Заполнение базы данных.
4. Создание различных запросов на получение данных (для формирования навыков работы с реализацией различных операций реляционной алгебры). Для каждой из операций (исключая деление) нужно показать минимум три запроса (хотя один и тот же запрос может демонстрировать выполнение нескольких операций).
5. Создание хранимых процедур и триггеров для обеспечения серверной части работы с данными.

# Ход работы

## **Глава 1**

Тема проекта: «Производство чая».

Средствами PostgreSQL была сделана физическая модель базы данных.

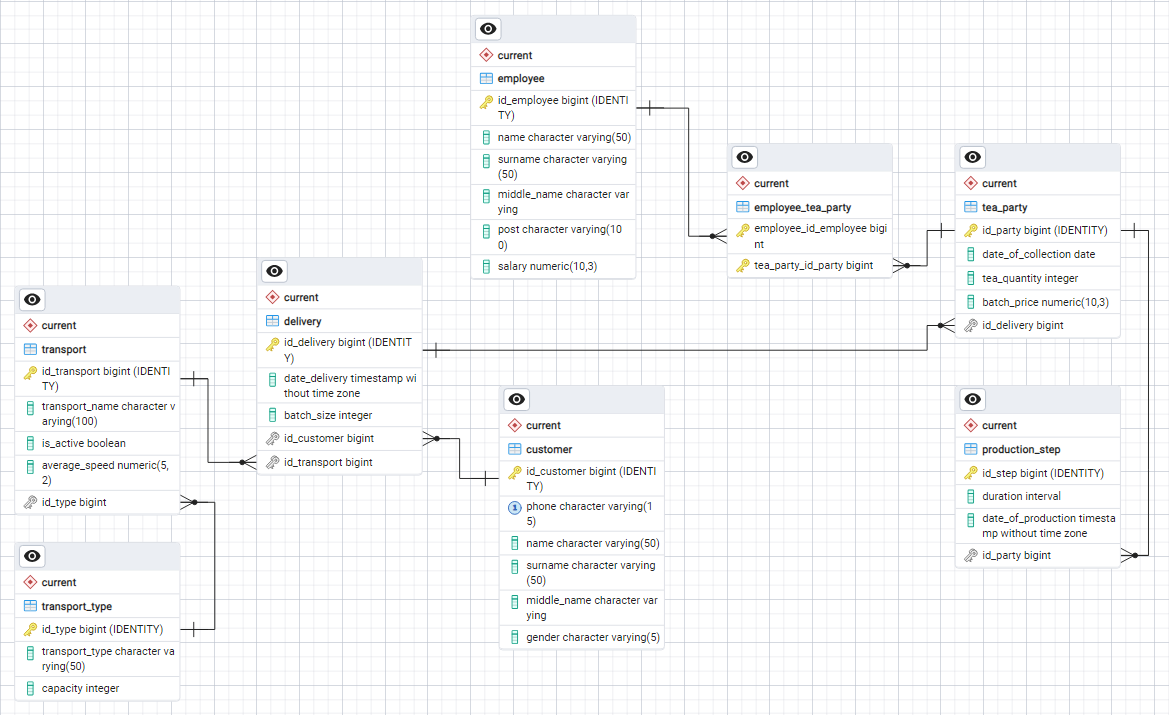


Рисунок 1 – Физическая модель

Физическая модель состоит из следующих таблиц:

**1:** Таблица Транспорт (transport): (Идентификатор, Имя, Активность, Средняя скорость).

Данная таблица связана с таблицей delivery через неидентифицирущую связь один-ко-многим.

Данная таблица необходима для сохранения информации о транспорте.

**2:** Таблица Вид транспорта (transport\_type): (Идентификатор, Тип транспорта, Грузоподъёмность).

Эта таблица содержит тип транспорта для доставки партий чая, а также грузоподъёмность самого транспорта.

Таблица связана с таблицей транспорт неидентифицирущей связью один-ко-многим.

**3:** Таблица Развоза (delivery): (Идентификатор, Дата развоза, Размер перевозимой партии).

Данная таблица содержит данные для развоза и доставки партий чая в определённые места.

**4:** Таблица Заказщик (Customer): (Идентификатор, Телефон, Имя, Фамилия, Отчество, Пол).

Таблица связана с таблицей Развоза как неидентифицирующая связь Один-К-Многим.

Эта таблица содержит всю информацию о заказщиках готовой партии чая.

**5:** Таблица Партии чая (tea\_party): (Идентификатор, Дата получения сырья, количество листьев чая, стоимость готовой партии, номер телефона заказщика данной партии).

Таблица связана с таблицей Сотрудники как неидентифицирующая связь Многие-К-Многим и с таблицей Производственный шаг как связь Один-К-Многим.

Данная таблица содержит данные обо всех заказанных партиях чая с информацией о них, а также с информацией, для кого данная партия предназначена.

**6:** Таблица Производственного шага (production\_step): (Идентификатор, Время производства чая из сырья, Дата отправки партии).

Данная таблица содержит информацию об производстве чая из сырья.

**7:** Таблица Сотрудников (Employee): (*Идентификатор*, Имя, Фамилия, Отчество, Должность, Зарплата).

Таблица связана с таблицей Партия чая как неидентифицирующая связь Многие-Ко-Многим.

Данная таблица содержит информацию о сотрудниках, которые работают с конкретной партией чая.

## **Глава 2**

Из ER-диаграммы формируем базу данных (Рисунки 2-12).

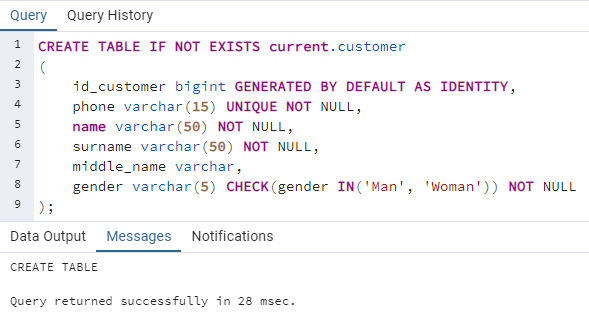


Рисунок 2 – Создание таблицы customer

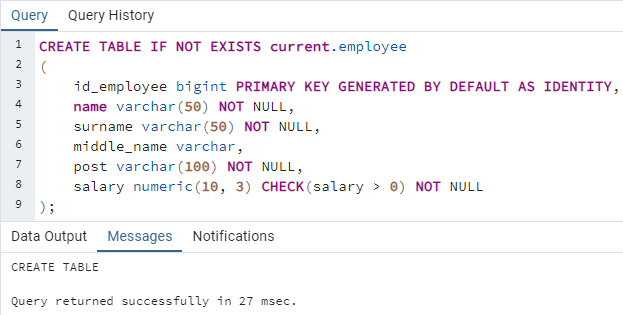


Рисунок 3 – Создание таблицы employee

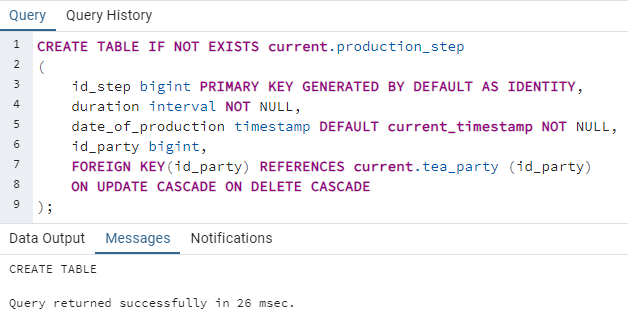


Рисунок 4 – Создание таблицы production\_step

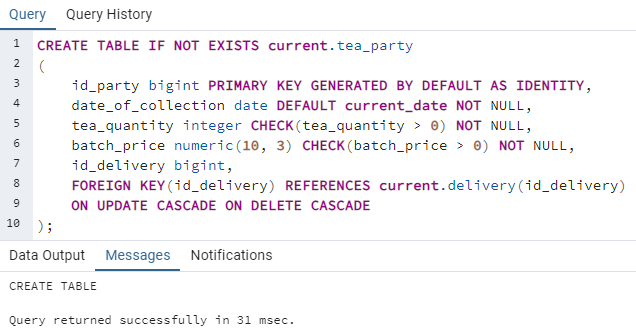


Рисунок 5 – Создание таблицы tea\_party

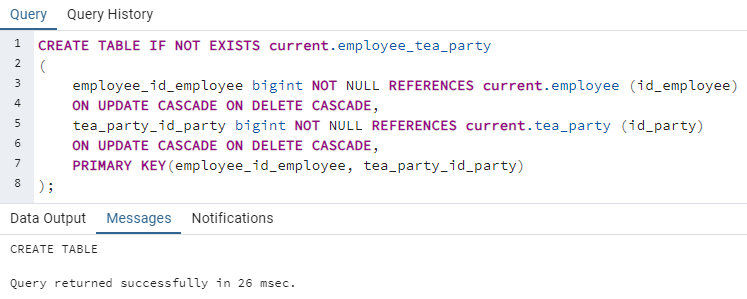


Рисунок 6 – Создание таблицы employee\_tea\_party

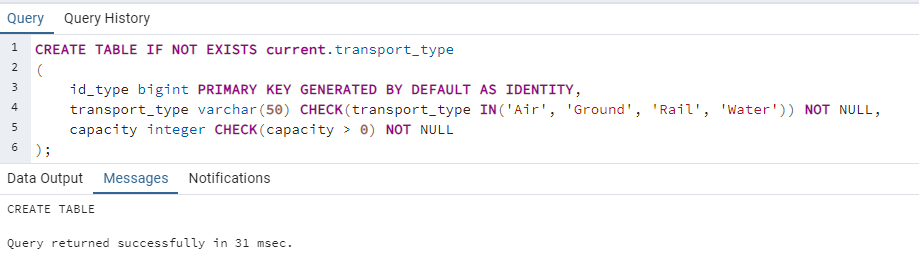


Рисунок 7 – Создание таблицы transport\_type

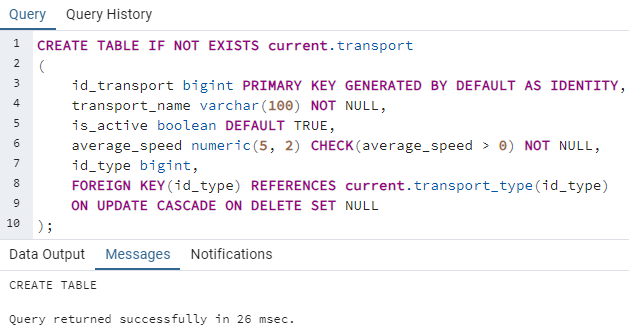


Рисунок 8 – Создание таблицы transport

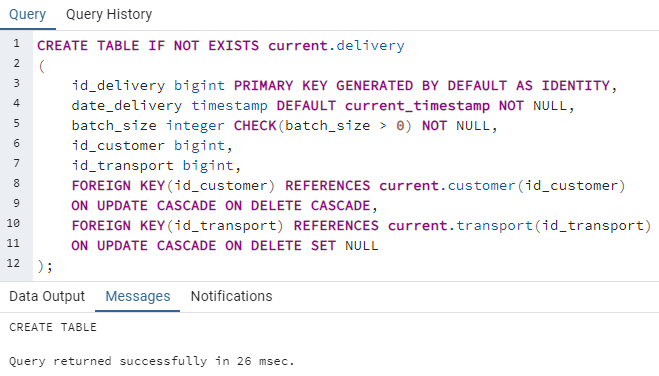


Рисунок 9 – Создание таблицы delivery

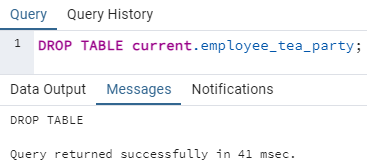


Рисунок 10 – Удаление таблицы employee\_tea\_party

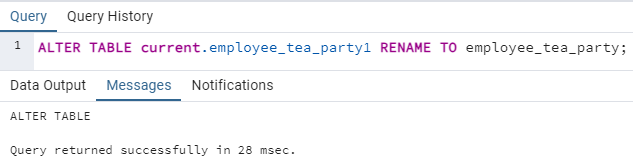


Рисунок 11 – Изменение таблицы employee\_tea\_party (переименование)

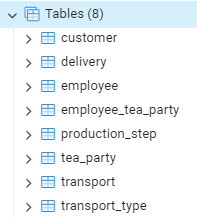


Рисунок 12 – Созданные таблицы в базе данных

## **Глава 3**

Внесение данных в базу данных доступно через SQL запрос INSERT (рисунки 10-15).



Рисунок 13 – Внесение данных в таблицу customer

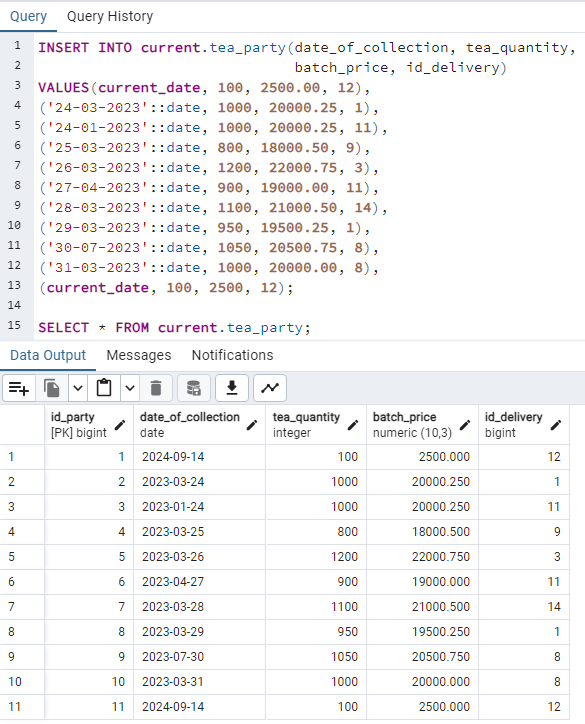


Рисунок 14 – Внесение данных в таблицу tea\_party с соблюдением корректности внешнего ключа

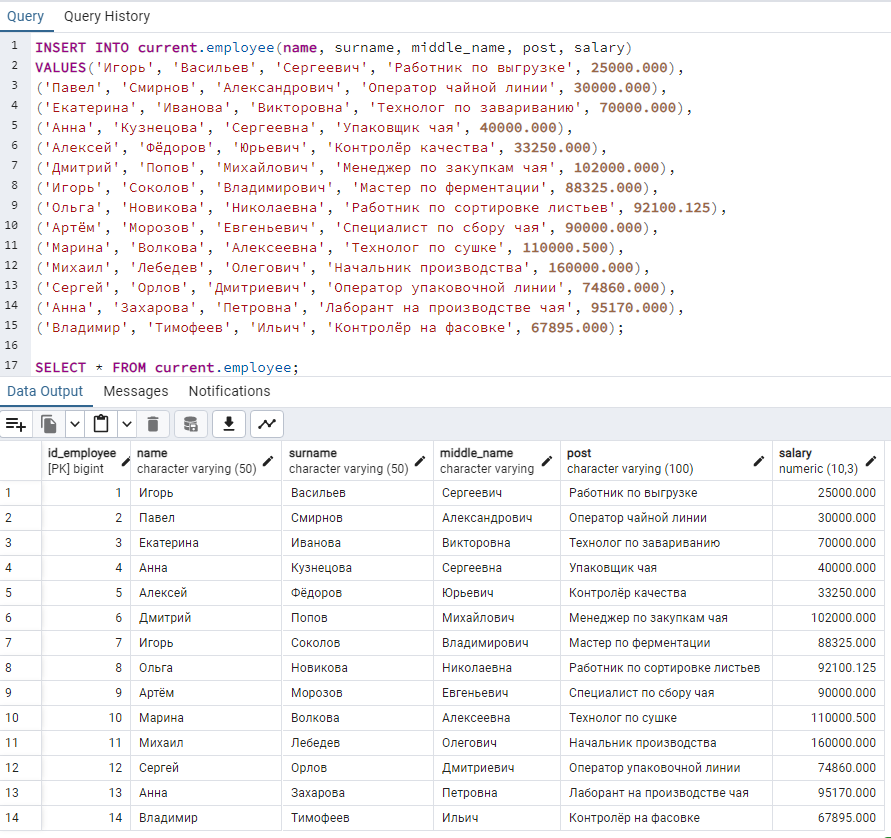


Рисунок 15 – Внесение данных в таблицу employee

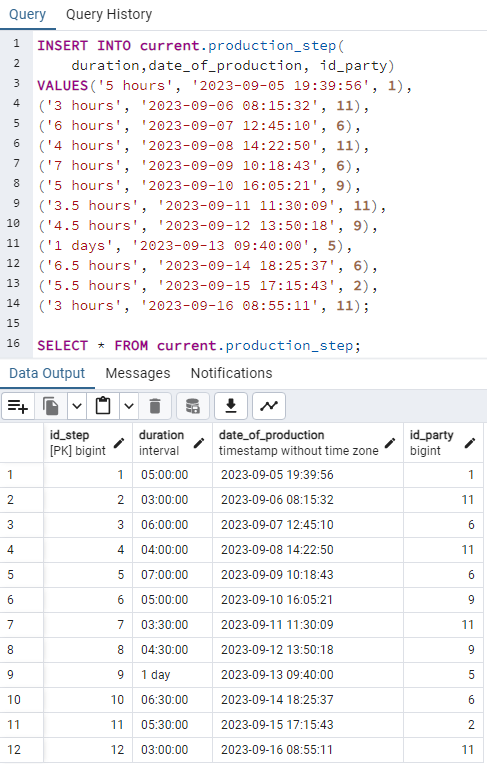


Рисунок 16 – Внесение данных в таблицу production\_step

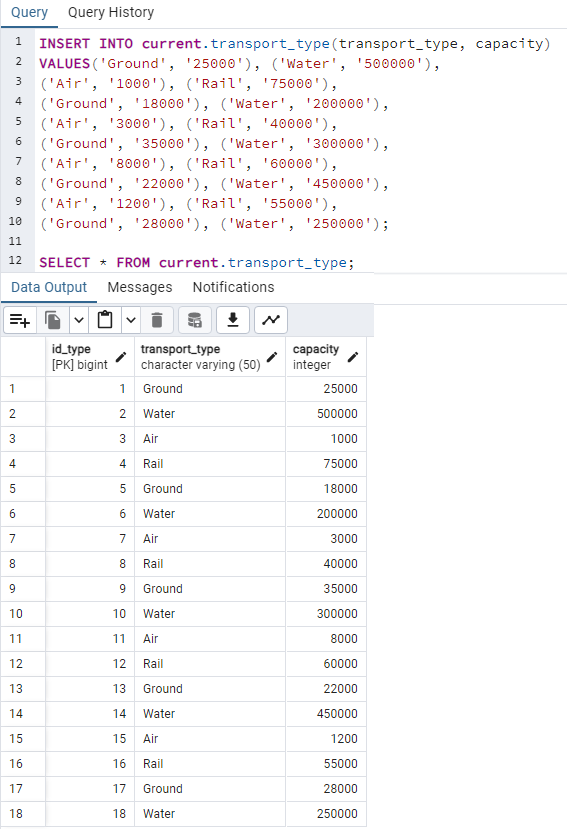


Рисунок 17 – Внесение данных в таблицу transport\_type

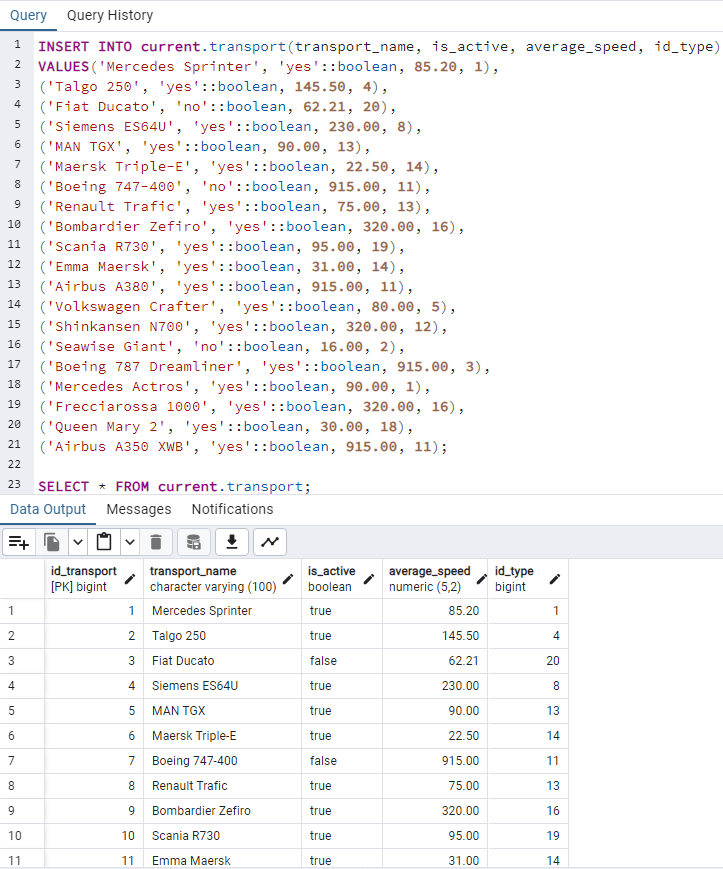


Рисунок 18 – Внесение данных в таблицу transport

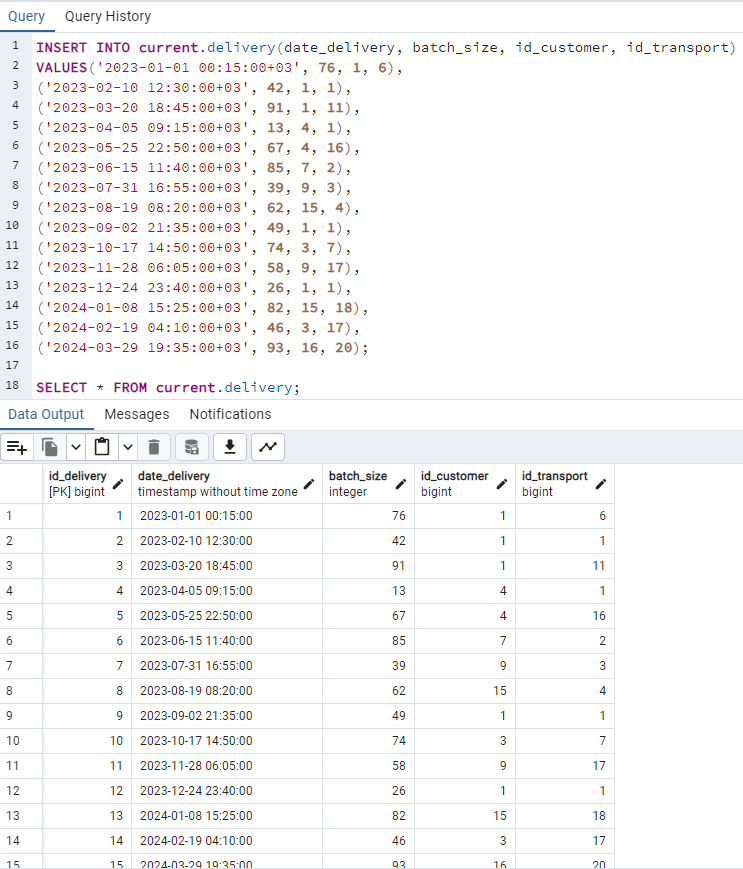


Рисунок 19 – Внесение данных в таблицу delivery

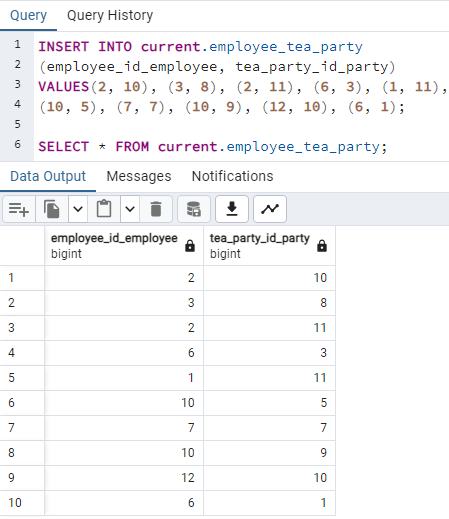


Рисунок 20 – Внесение данных в таблицу employee\_tea\_party

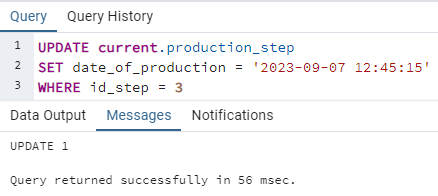


Рисунок 21 – Обновление строки таблицы production\_step по условию

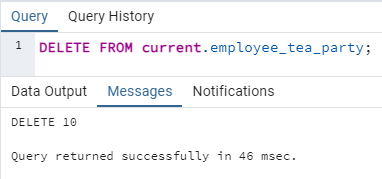


Рисунок 22 – Удаление всех строк из таблицы employee\_tea\_party

## **Глава 4**

Изучить и создать выборку и сортировку данных. Изучить и применить операторы для изменения данных в таблицах. Применение базовых SQL операций (рисунки 23-29)

### **Базовые SQL запросы с дополнениями**

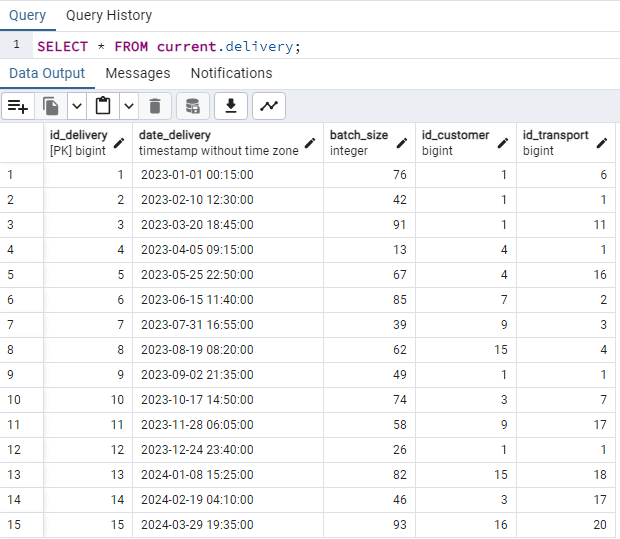


Рисунок 23 – Применение операции SELECT для всех колонок таблицы delivery

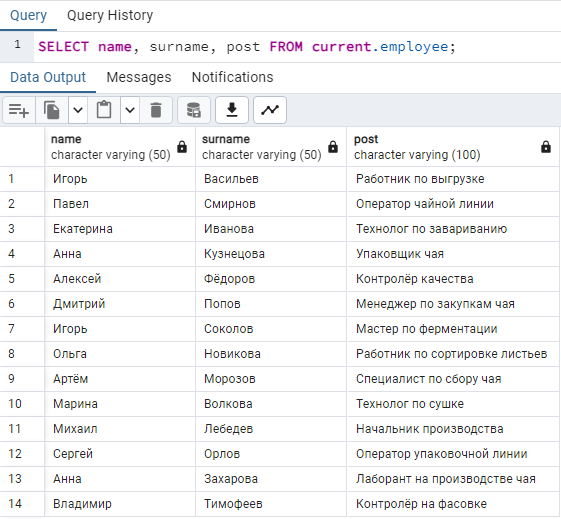


Рисунок 24 – Применение операции SELECT для колонок name, surname и post таблицы employee

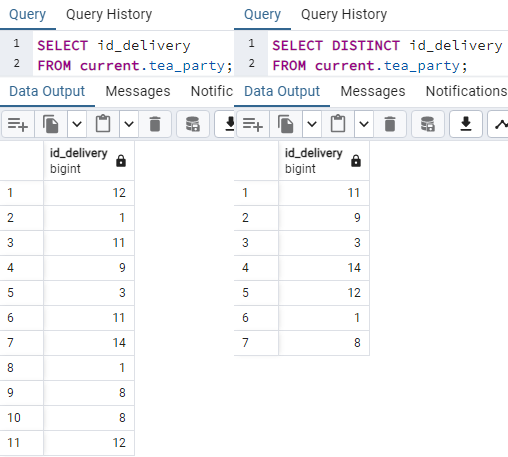


Рисунок 25 – Применение операции SELECT с оператором DISTINCT для удаления дубликатов и без него

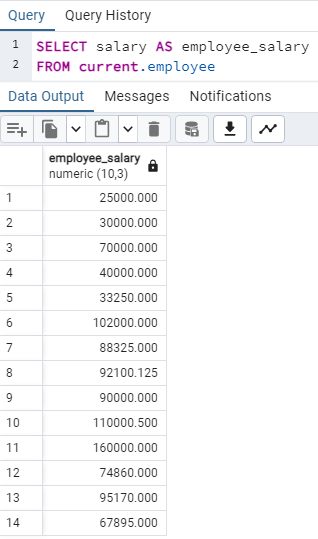


Рисунок 26 – Применение операции SELECT с использованием псевдонима для колонки salary таблицы employee

Применение запросов операций селекции (рисунки 27-31)

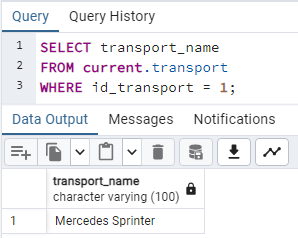


Рисунок 27 – Применение операции SELECT с оператором WHERE и условием равенства

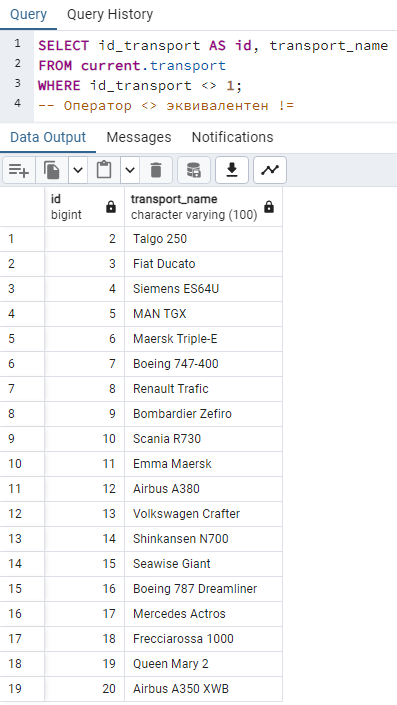


Рисунок 28 – Применение операции SELECT с оператором WHERE и условием неравенства

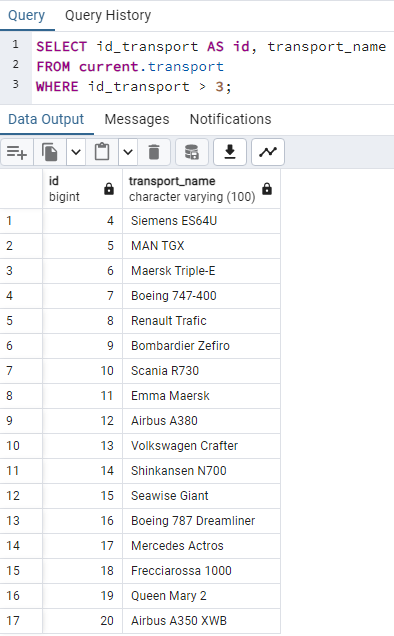


Рисунок 29 – Применение операции SELECT с оператором WHERE и условием больше

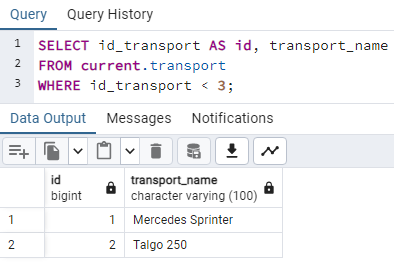


Рисунок 30 – Применение операции SELECT с оператором WHERE и условием меньше

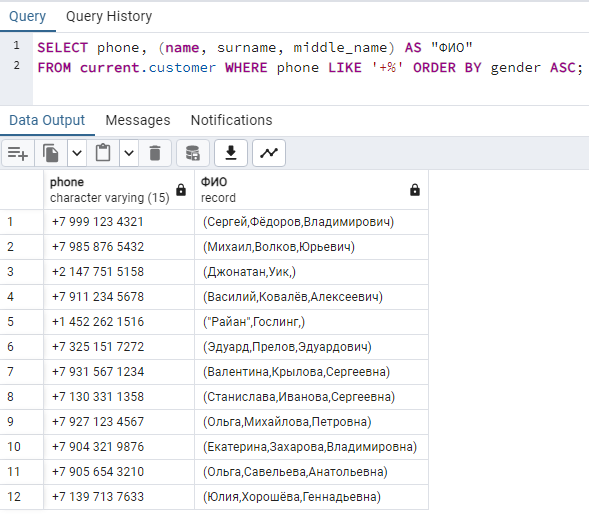


Рисунок 31 – Применение операции SELECT с оператором LIKE и сортировкой по возрастанию по колонке gender

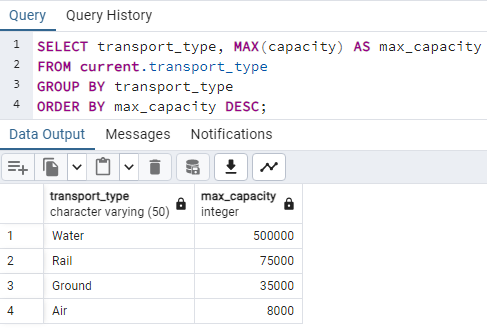


Рисунок 32 – Применение операции SELECT с оператором GROUP BY и ORDER BY с сортировкой по убыванию

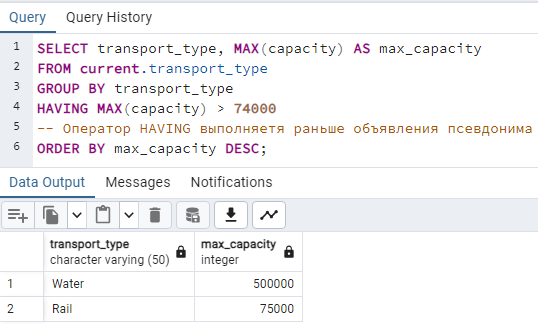


Рисунок 33 – Применение операции SELECT с оператором GROUP BY и HAVING с сортировкой по убыванию

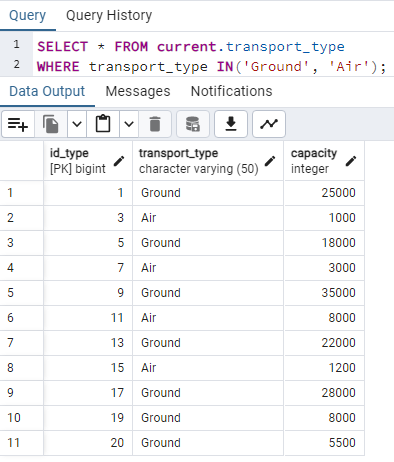


Рисунок 34 – Применение операции SELECT с функцией IN(), проверяющей, что данная колонка содержит все значения параметров функции IN()

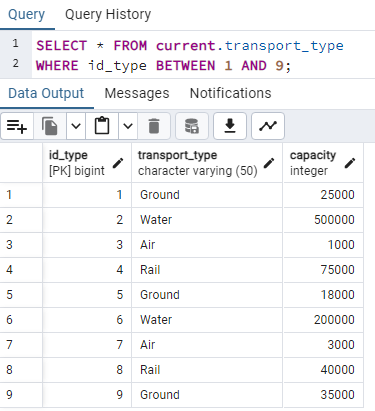


Рисунок 35 – Применение операции SELECT с BETWEEN для задания диапазона поиска

### **Агрегатные функции**

Агрегатные функции – специальные функции, которые на вход получают набор каких-либо данных, а результатом работы возвращается конкретное значение, т.е. конкретное число. В PostgreSQL в основном используют 5 видов таких функций. На рисунках 36 – 39 представлены результаты их работы.

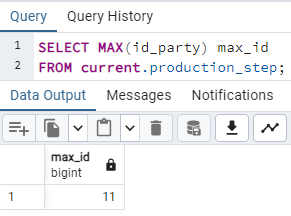


Рисунок 36 – Применение функции MAX() для подсчёта максимального значения колонки id\_party

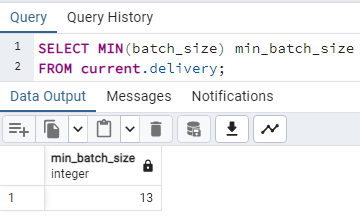


Рисунок 37 – Применение функции MIN() для подсчёта минимального значения колонки batch\_size

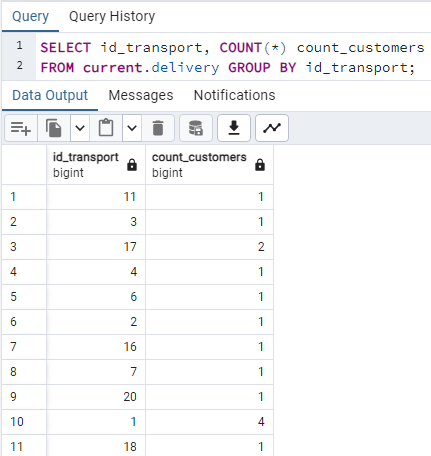


Рисунок 38 – Применение функции COUNT() для подсчёта количества строк в таблице delivery с последующей группировкой

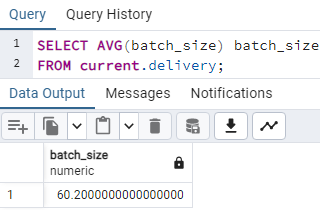


Рисунок 33 – Применение функции AVG() для подсчёта среднего значения колонки batch\_size

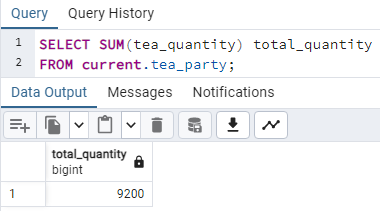


Рисунок 39 – Применение функции SUM() для подсчёта суммы всех значений в колонке tea\_quantity

На рисунках 40 – 42 представлены результаты применения агрегатных функций вместе с подзапросами основных запросов SQL.

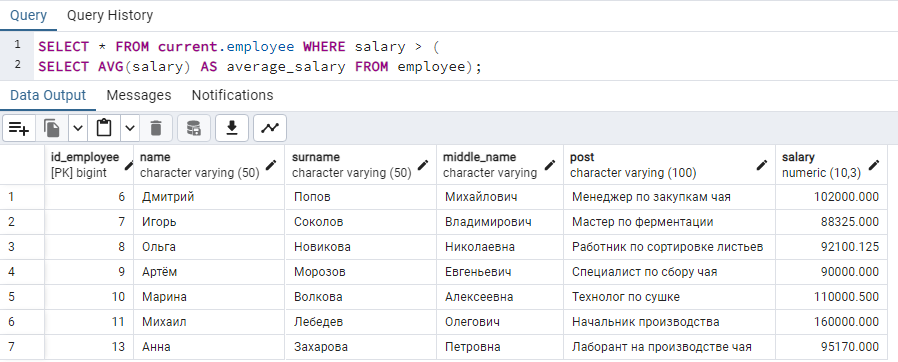


Рисунок 40 – Применение агрегатной функции AVG() вместе с подзапросом для получения выборки всех сотрудников, кто получает зарплату выше среднего



Рисунок 41 – Применение агрегатной функции SUM() вместе с подзапросом для получения выборки всех партий чая, где их цена выше, чем сумма размера всех партий

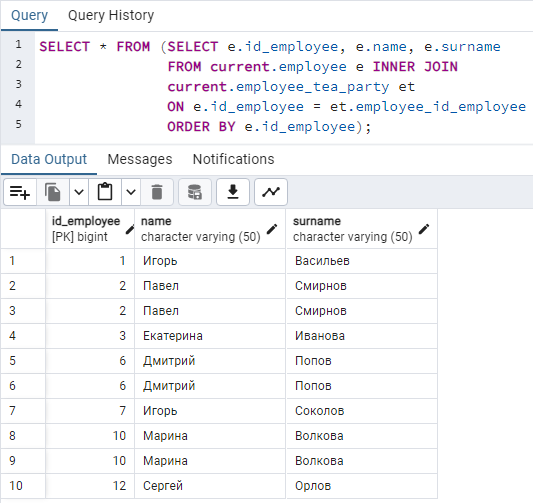


Рисунок 42 – Применение INNER JOIN в подзапросе для получения выборки сотрудников, которые работают над какими-либо партиями

### **Алгебра множеств**

Здесь будут использоваться элементы алгебры множеств, ведь таблица представляет собой множество строк, по которым можно производить соединение, пересечение, разность, а также выполнять различные JOIN’ы двух и более таблиц. Будут рассмотрены следующие операции: UNION, INTERSECT, EXCEPT, INNER JOIN, LEFT/RIGHT JOIN, FULL OUTER JOIN.

Применение операций соединения, пересечения, разности (рисунки 39-42).

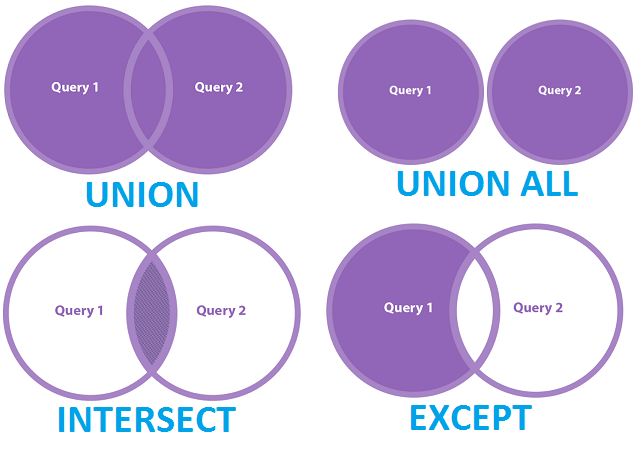


Рисунок 43 – Пример работы различных операторов для работы с множествами, группой данных

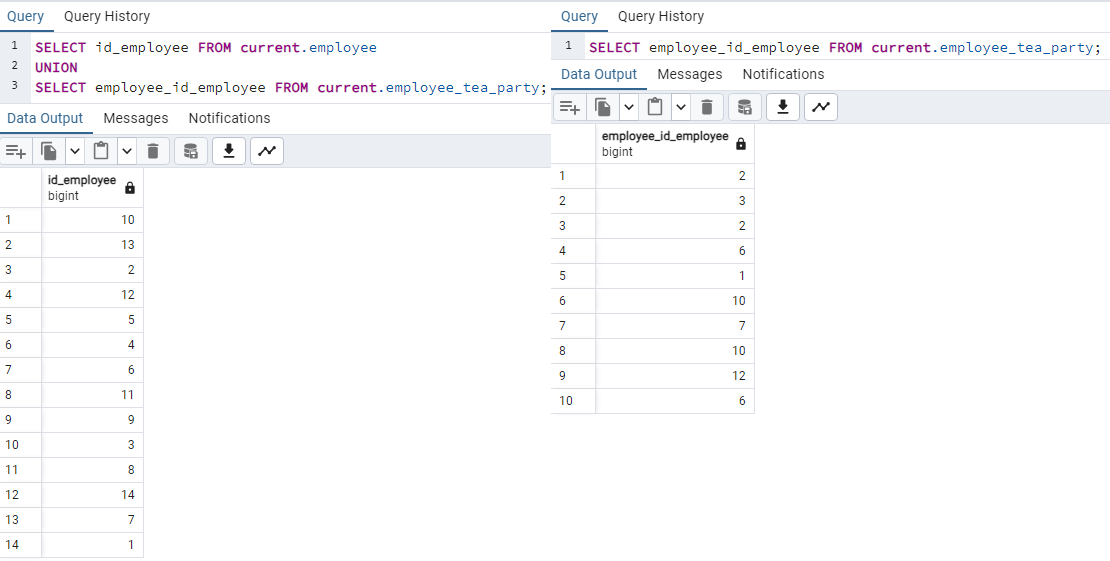


Рисунок 44 – Применение операции соединения UNION для таблиц employee и employee\_tea\_party

Данная операция не включает в итоговую выборку дубликаты строк для того, чтобы добавить дубликаты в выборку можно использовать конструкцию UNION ALL, пример выполнения операции показан на рисунке 39.

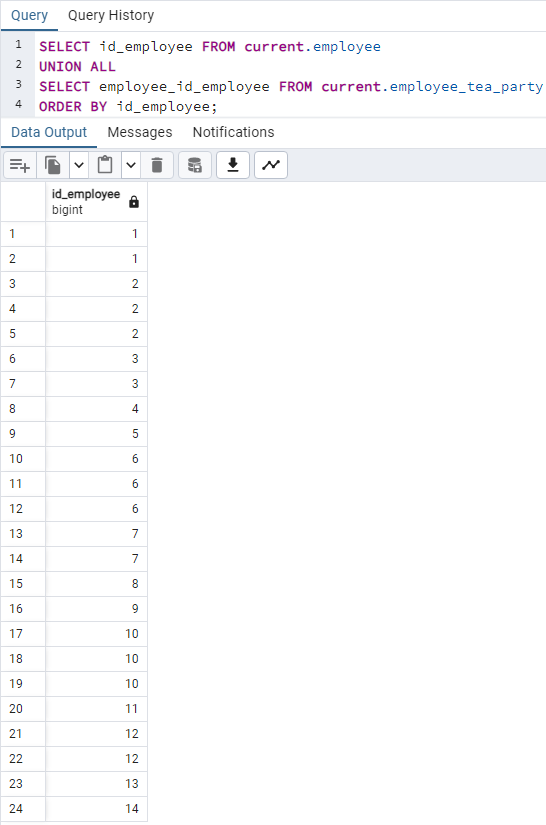


Рисунок 45 – Применение операции соединения UNION ALL для таблиц transport и delivery

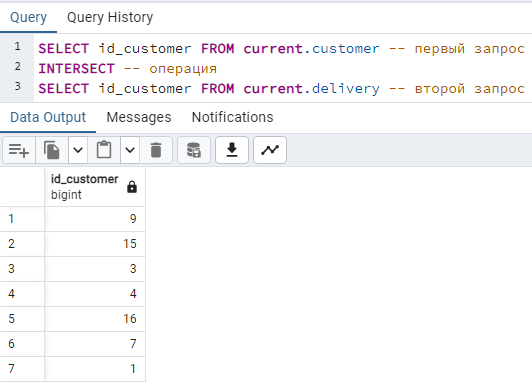


Рисунок 46 – Применение операции пересечения INTERSECT для таблиц customer и delivery для определения, к какому заказщику производится доставка партии

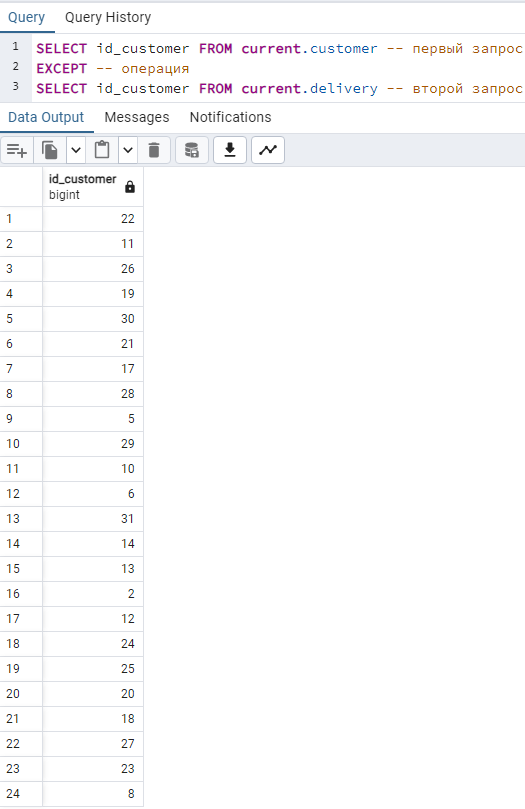


Рисунок 47 – Применение операции разности EXCEPT для таблиц customer и delivery для определения, какие заказщики не оформили доставку партии

Применение JOIN операций (рисунки 48-51).

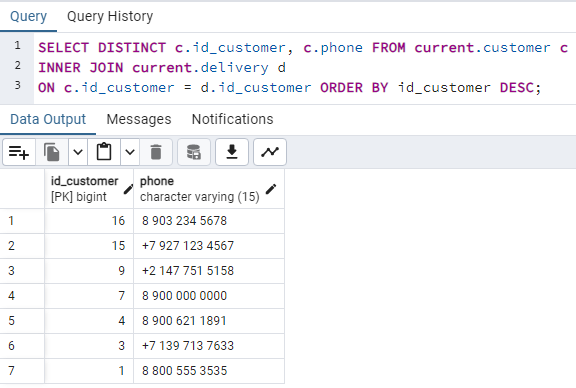


Рисунок 48 – Inner Join двух таблиц по колонке id\_customer

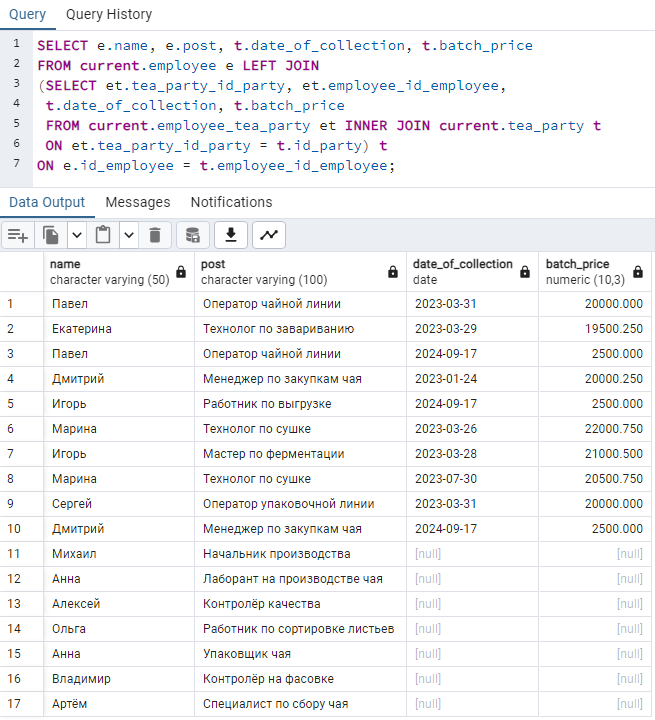


Рисунок 49 – Left Join c Inner Join двух таблиц по колонкам id\_employee и id\_party

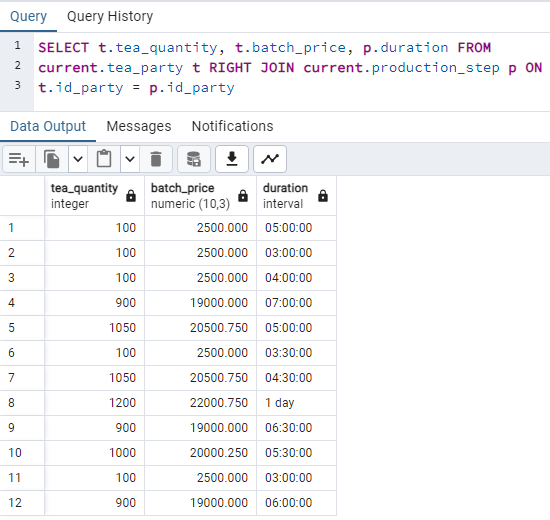


Рисунок 50 – Right Join двух таблиц по колонке id\_party

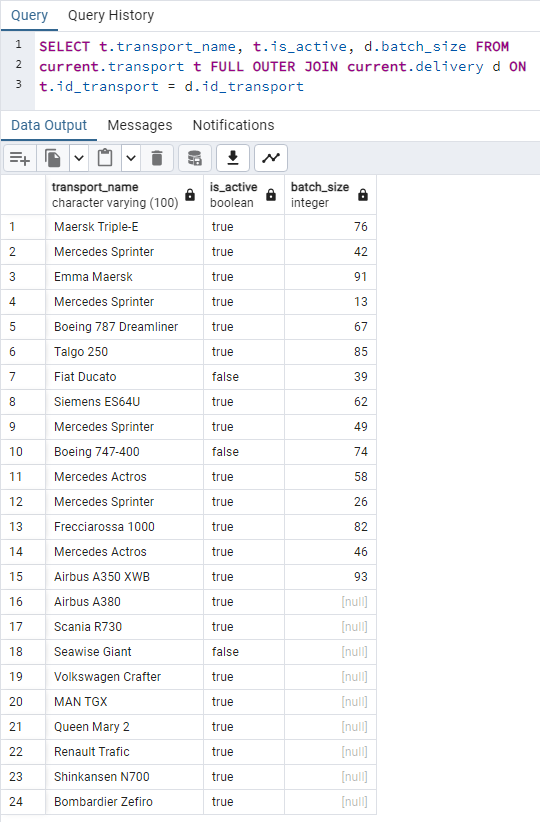


Рисунок 51 – Full Outer Join двух таблиц по колонке id\_transport

## **Глава 5**

В данной главе будут рассмотрены триггеры и триггерные функции для конкретных таблиц. Триггеры позволяют выполнять какие-либо действия при вставке, удалении, изменении данных из таблицы.

### **Триггер №1**

Данный триггер был создан для вычисления средней зарплаты всех сотрудников, он срабатывает после добавления нового сотрудника в таблицу employee. Таблица для ведения информации о средней зарплате состоит из primary key, самой средней зарплате, и дате последнего проведения замера средней зарплаты.

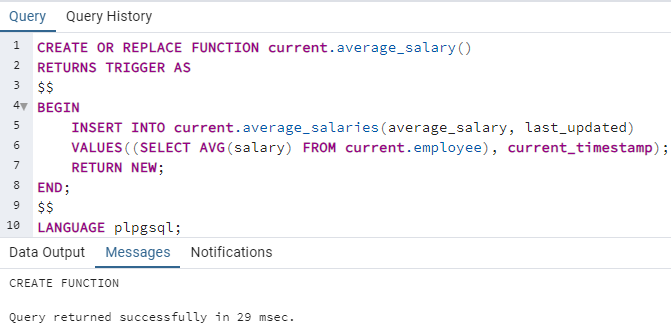


Рисунок 52 – Добавление триггерной функции

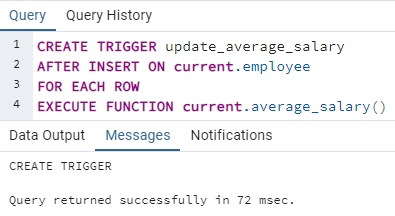


Рисунок 53 – Добавление триггера на таблицу employee и связывание с триггерной функцией average\_salary().

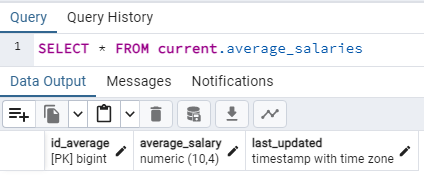


Рисунок 54 – Изначально таблица пустая



Рисунок 55 – Пример добавления данных

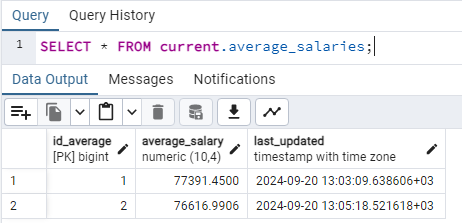


Рисунок 56 – При каждом новом добавлении сотрудника в таблицу employee средняя зарплата меняется и заносится в другую таблицу

### **Триггер №2**

Данный триггер позволяет при удалении особо важных компонентов базы данных (были определены, как таблицы customer и tea\_party) сохранять их в таблицы в специальной схеме temp, которая создаётся внутри тела триггерной функции для разделения спецификации воспринимаемой информации. Триггерная функция saving\_deleted\_customers() проверяет наличие схемы, и если её нет, создаёт её в текущей базе данных, далее такая же процедура происходит с таблицей, в которую будут занесены удалённые сотрудники для аварийных случаев. В случае чего, всю схему temp можно удалить, если она будет не нужна.

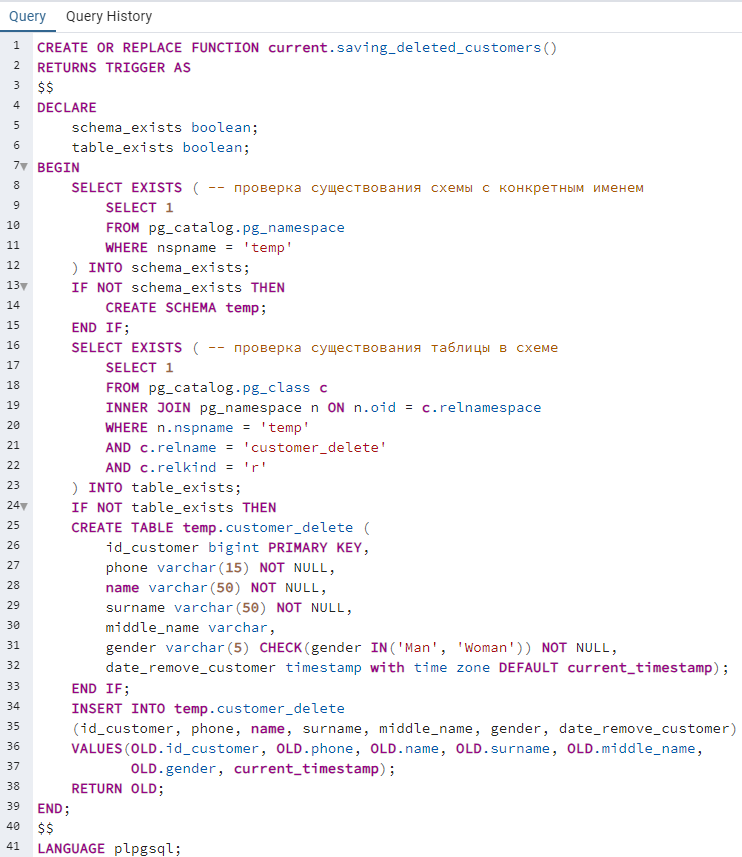


Рисунок 57 – Добавление триггерной функции



Рисунок 58 – Добавление триггера на таблицу customer и связывание с триггерной функцией saving\_deleted\_customers().



Рисунок 59 – Пример удаления заказщика для срабатывания триггера



Рисунок 60 – Результат работы триггера (до/после)

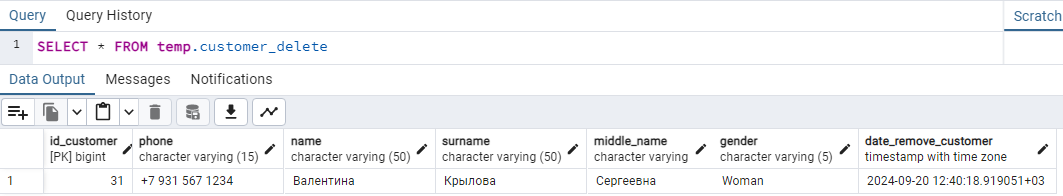


Рисунок 61 – Результат работы триггера (данные удалённого заказщика в новой таблице и схеме)

### **Триггер №3**

Данный триггер делает тоже самое, что и предыдущий, но с таблицей tea\_party.

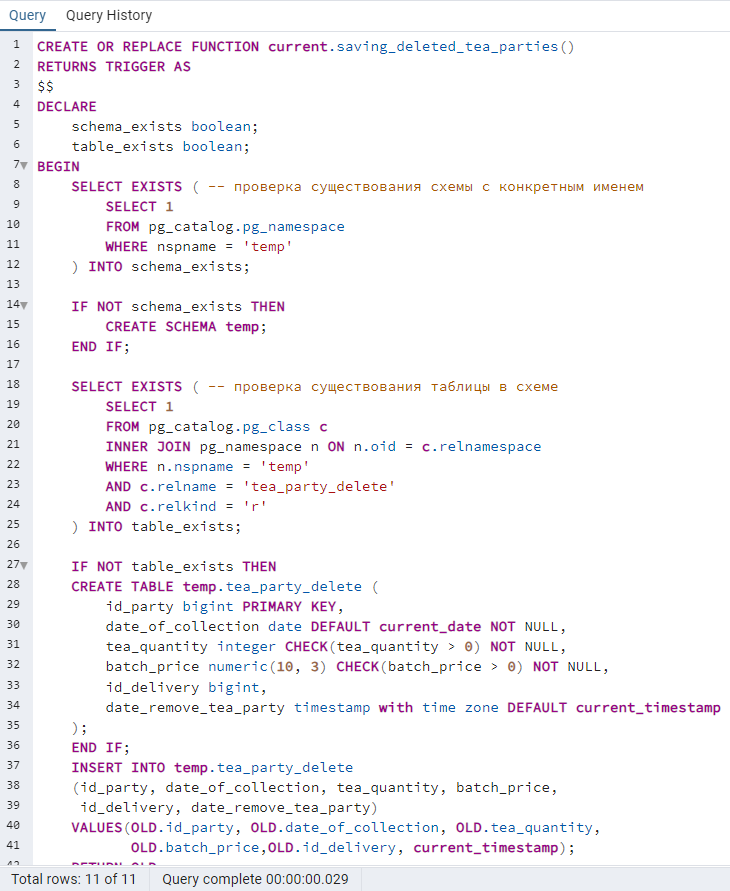
****

Рисунок 62 – Добавление триггерной функции

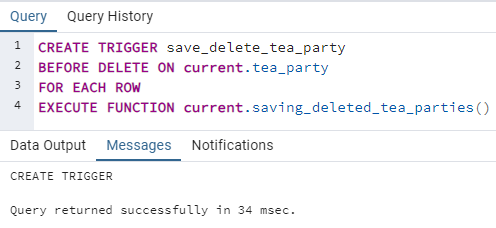
****

Рисунок 63 – Добавление триггера на таблицу tea\_party и связывание с триггерной функцией saving\_deleted\_ tea\_party().

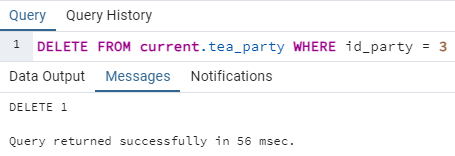
****

Рисунок 64 – Пример удаления партии чая для срабатывания триггера

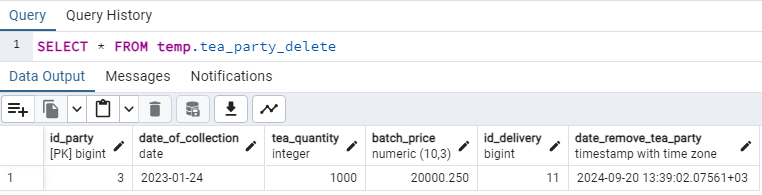
****

Рисунок 65 – Результат работы триггера (данные удалённой партии чая в новой таблице и схеме)

### **Триггер №4**

Данный триггер предназначен для отслеживания количества добавленных кортежей в таблицу customer\_delete, если количество добавляемых строк будет больше 10, то из таблицы (как подразумевается) удаляется самая ранняя запись, поскольку она считается уже не нужной.

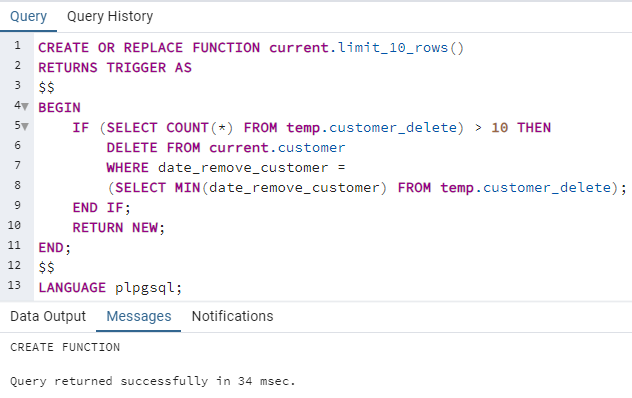


Рисунок 66 – Добавление триггерной функции



Рисунок 67 – Добавление триггера на таблицу customer\_delete и связывание с триггерной функцией limit\_10\_rows()

### **Триггер №5**

Данный триггер служит для проверки обновляемых данных в таблице employee, который срабатывает перед добавлением нового кортежа в таблицу employee, выводя сообщение об ошибке, если новое поле полностью совпадает по значению с предыдущим. В случае, если вставляемые данные различаются с предыдущими, происходит успешное изменение данных в таблице, и триггер выводит сообщение об успешности операции.

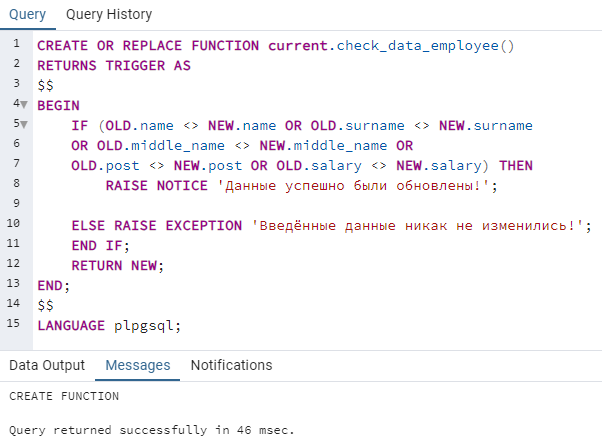


Рисунок 68 – Добавление триггерной функции

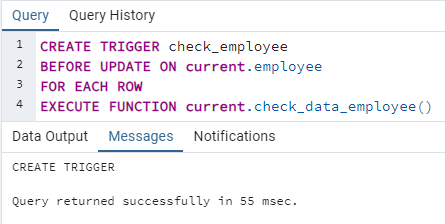


Рисунок 69 – Добавление триггера на таблицу employee и связывание с триггерной функцией check\_data\_employee()

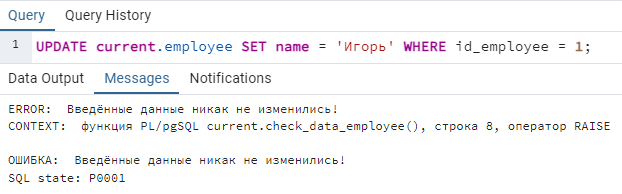


Рисунок 70 – Попытка изменить данные на те же самые

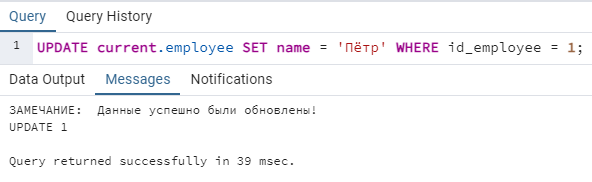


Рисунок 71 – Попытка изменить данные на другие

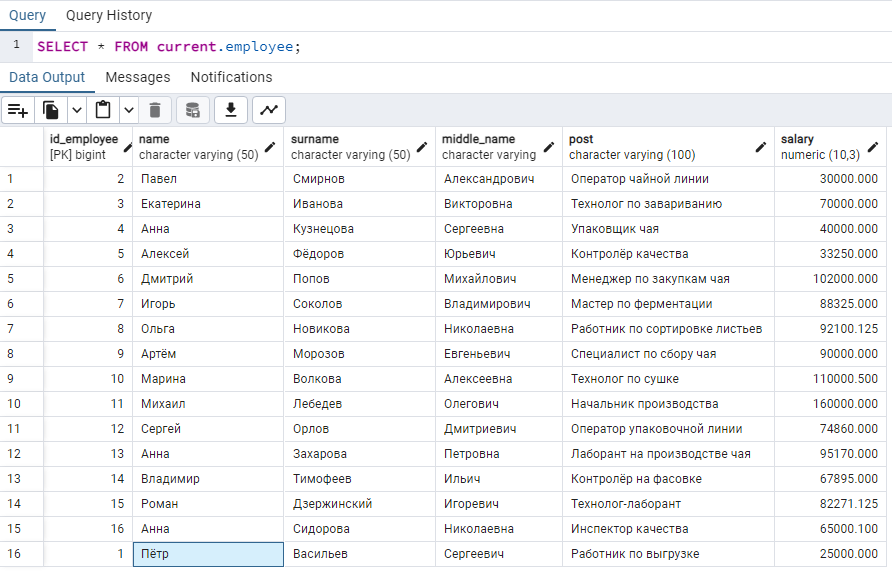


Рисунок 72 – Изменённые данные успешно обновились в исходной таблице

## **Глава 6**

В данной главе буду рассмотрены оконные функции, они позволяют работать не со всей таблицей целиком, а только с конкретными типами (окнами) данных, чем-то напоминает группировку данных GROUP BY. На рисунках Х-Х представлена работа с оконными функциями.

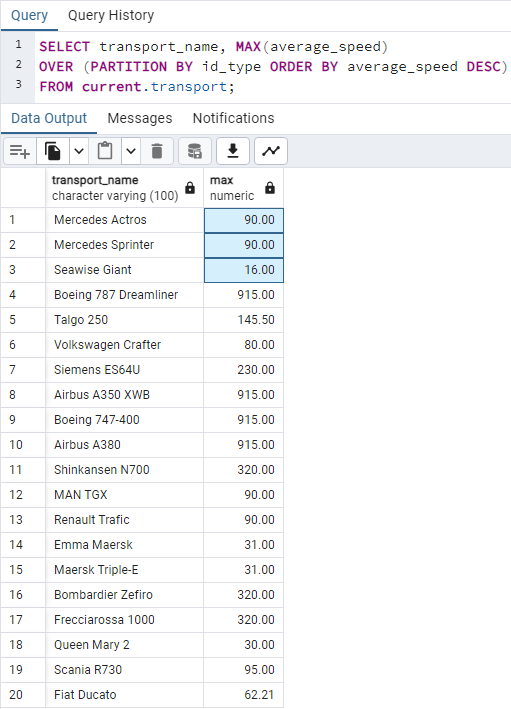


Рисунок 73 – Использование оконной функции MAX()

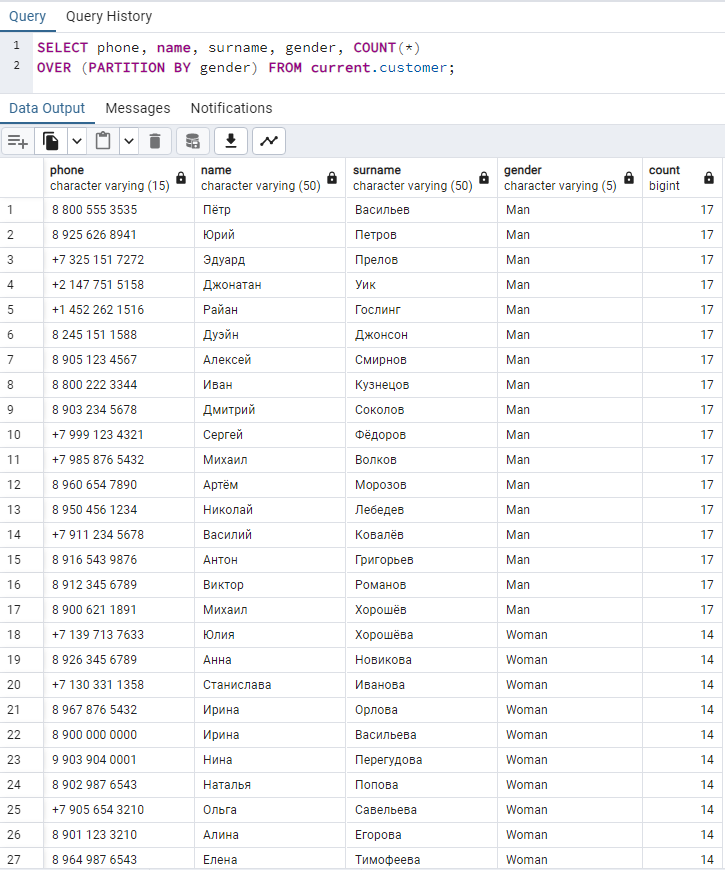


Рисунок 74 – Использование оконной функции COUNT()

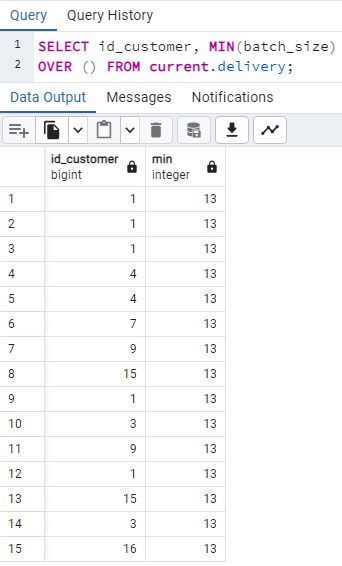


Рисунок 75 – Использование оконной функции MIN() без условия деления на партиции

В таком случае окном функции будет вся таблица, и минимальный объём партии будет искаться среди всей таблицы.

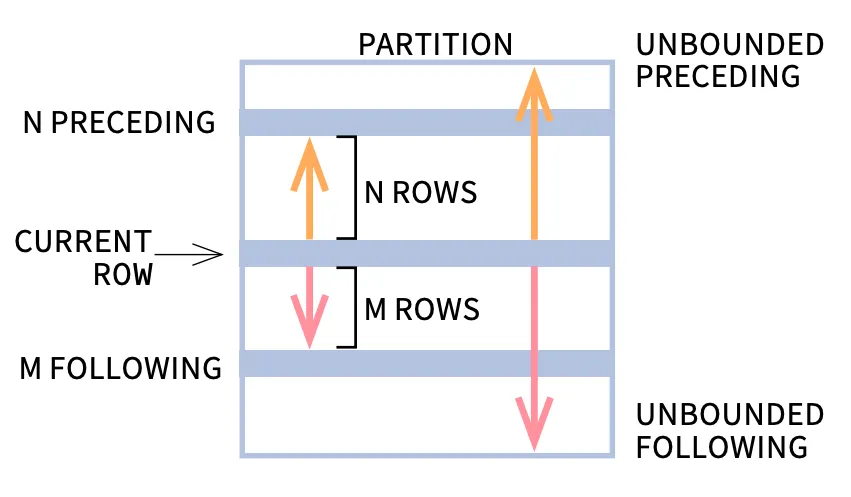


Рисунок 76 – Использование различных операций с оконными функциями

Образец: ROWS BETWEEN *нижняя граница* AND *верхняя граница*

Границами могут служить любые их следующих пяти вариантов:

* UNBOUNDED PRECEDING - все строки выше текущей строки.
* *n* PRECEDING - n строк выше текущей строки.
* CURRENT ROW - только текущая строка.
* *n* FOLLOWING - n строк ниже текущей строки.
* UNBOUNDED FOLLOWING - все строки ниже текущей строки.

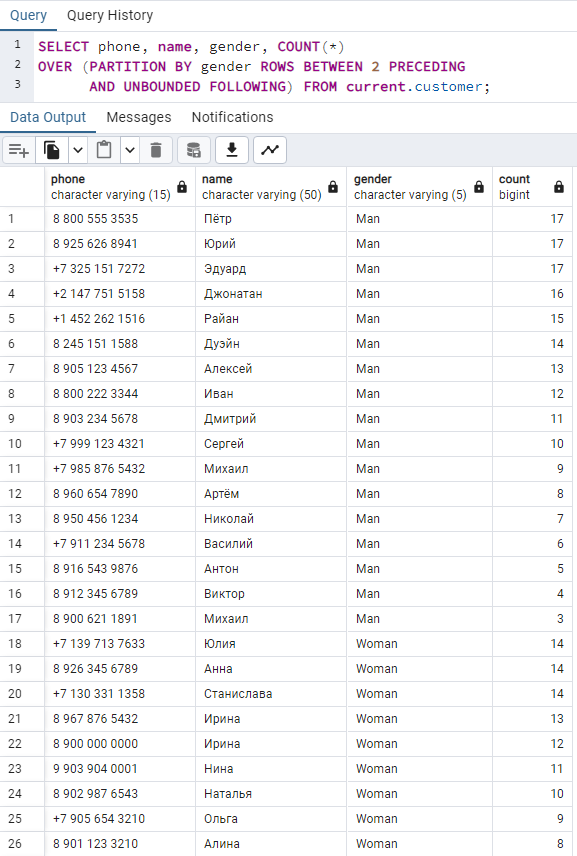


Рисунок 77 – Использование оконной функции COUNT() с диапазоном в партициях

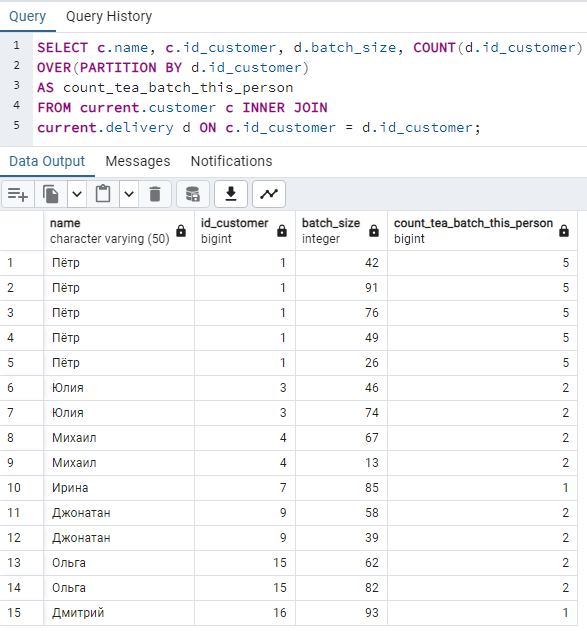


Рисунок 78 – Использование оконной функции COUNT()

На рисунке выше выполняется запрос, который отображает всех людей, кто заказал партии чая, а также сколько партий заказал конкретный человек.

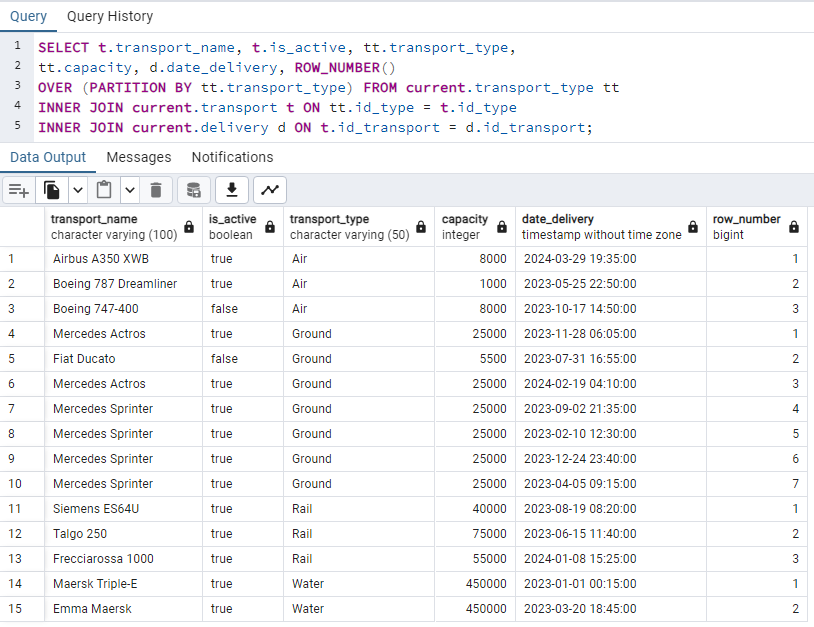


Рисунок 79 – Использование оконной функции COUNT()

На рисунке выше отображён запрос, который показывает весь транспорт, который записан на доставку партий чая. Отображён как тип транспорта, так и его имя и грузоподъёмность из разных таблиц, путём JOIN’a их. Каждому активному транспорту присвоен номер строки в зависимости от его типа.

# Вывод

После выполнения данных операций над базой данных было расширено понимание устройства баз данных, работы СУБД PostgreSQL. Получено понимание работы оконных функций и работы триггерных функций и триггеров у таблиц. Была создана диаграмма базы данных, на основе которой была создана база данных, где производились выборки, запросы, а также были созданы процедурные функции и триггеры для автоматизации работы системы.