Министерство науки и образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

МЕТОДИЧЕСКИЕ УКАЗАНИЯ К ЛАБОРАТОРНЫМ РАБОТАМ ПО КУРСУ «БАЗЫ ДАННЫХ»

Лабораторная работа №1
«Введение в SQL»

Авторы:
Скворцова М.А., magavrilova@bmstu.ru
Лапшин А.В.

Москва, 2022

Общие сведения

Сокращения

SQL– Structured Query Language («язык структурированных запросов»)

БД – База данных

СУБД – Система управления базами данных

РБД – Реляционная база данных

Краткая информация о СУБД PostgreSQL

PostgreSQL – программа, которая относится к классу систем управления базами данных. Когда эта программа выполняется, мы называем ее сервером PostgreSQL, или экземпляром сервера.

Данные, которыми управляет PostgreSQL, хранятся в базах данных. Один экземпляр PostgreSQL одновременно работает с несколькими базами, которые вместе называются кластером баз данных.

Чтобы кластер можно было использовать, его необходимо инициализировать (создать). Каталог, в котором размещаются все файлы, относящиеся к кластеру, обычно называют, словом, **PGDATA**, по имени переменной окружения, указывающей на этот каталог.

Установка программного обеспечения и ссылки на источники

Установка СУБД PostgreSQL 11.5

Для выполнения заданий практикума предлагается использовать СУБД PostgreSQL 11.5 и новее, дистрибутив которой можно скачать по ссылке: https://www.enterprisedb.com/downloads/postgres-postgresql-downloads

По ссылке выше нужно будет выбрать операционную систему и скачать .exe-установщик, который предложит установить нужные компоненты для работы с PostgreSQL на рисунке 1. Главные из которых первые две, сам PostgreSQL Server и pgAdmin 4, остальное не является обязательным.

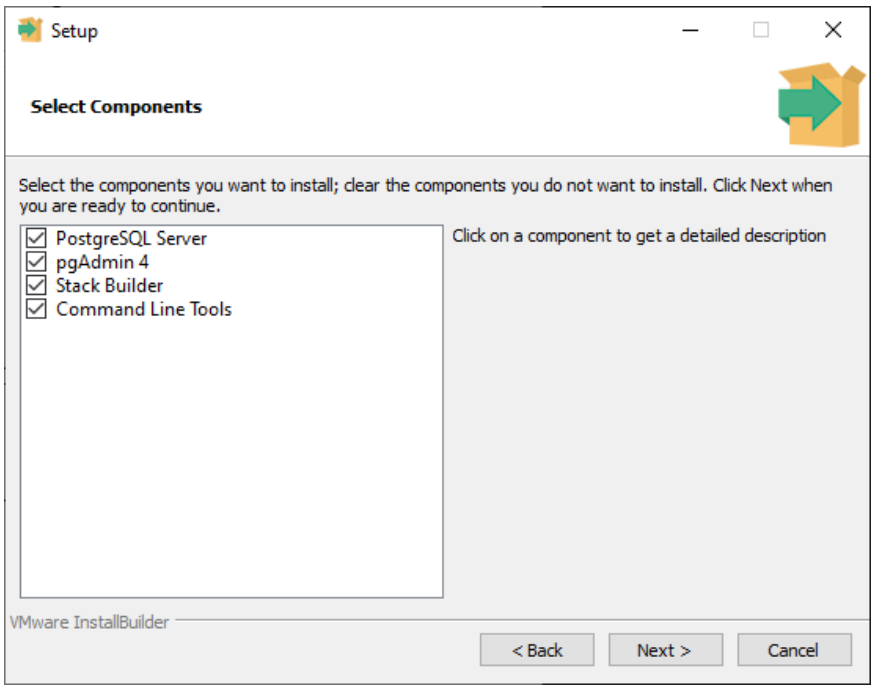

Рисунок 1 – Компоненты

Следующим выбором будет путь, где планируется хранить ваши базы данных на рисунке 2.

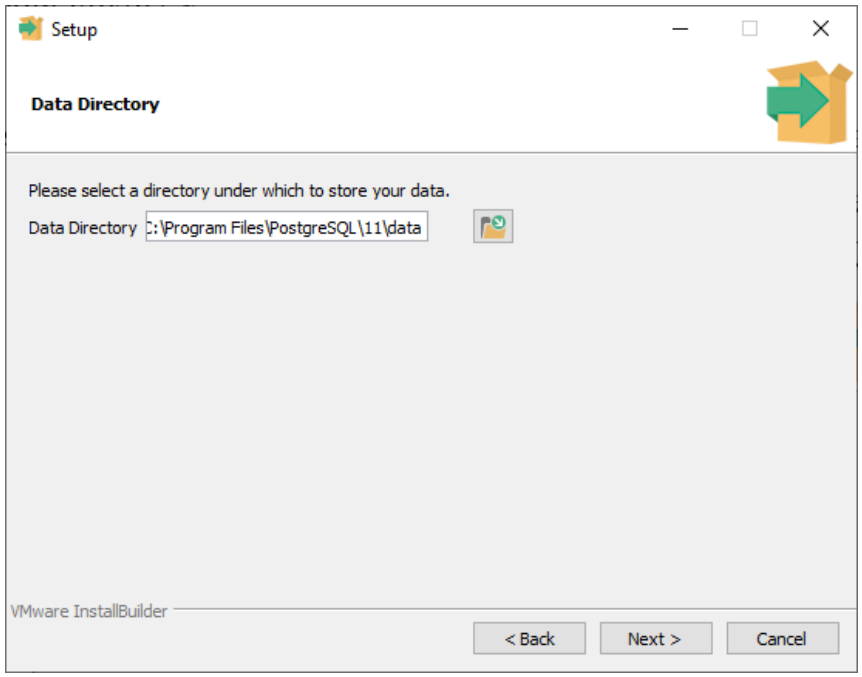

Рисунок 3 – Путь

Далее нужно будет установить пароль для вашей базы данных на рисунке 3. (лучше где-нибудь записать, на случай если забудете)

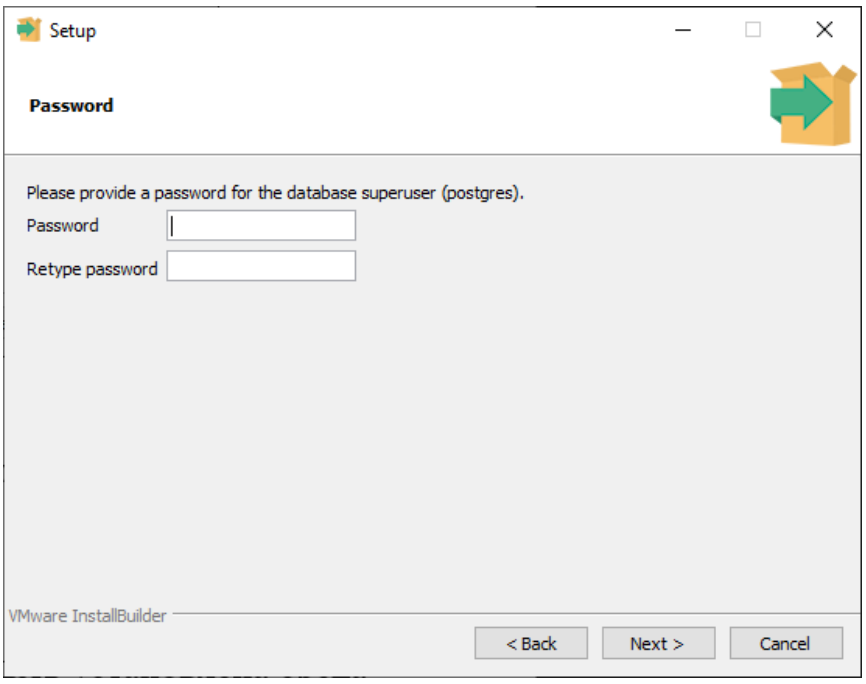

Рисунок 3 – Установка пароля

Так как база данных является сервером, к которому нужно будет обращаться, нам предлагают выбрать порт, по которому будет идти соединение на рисунке 4. (лучше всего оставить по умолчанию)

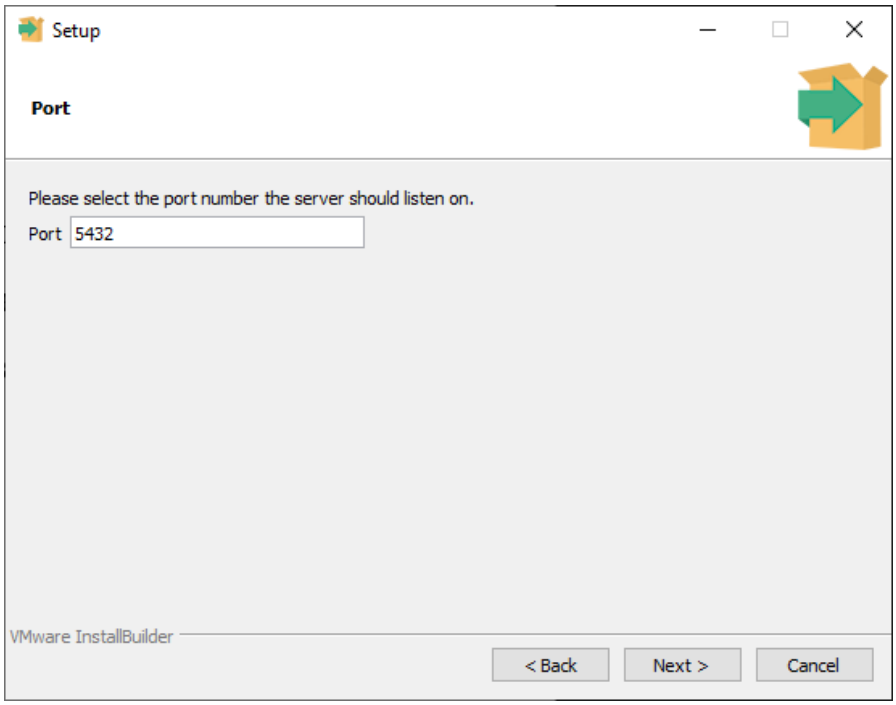

Рисунок 4 – Порт

Последним действием можно выбрать кодировку (язык) данных, его так же можно оставить по умолчанию. (Default locale – будет опираться на настройки системы)

После установки при желании (галочка) в последнем окне предлагается установить дополнительные приложения для СУБД. Этот пункт можно пропустить, а в дальнейшем если понадобится что-то установить просто набрать в поиске windows название установщика и дополнить вашу среду. (Application Stack Builder)

Установка и настройка среды pgAdmin 4

В результате установки СУБД PostgreSQL должна быть установлена среда pgAdmin 4 для подключения к СУБД и выполнения SQL-запросов. Если этого не произошло, то ее необходимо установить отдельно, скачав по ссылке: https://www.pgadmin.org/download/

После установки, программу можно найти с помощью строки поиска (pgAdmin4). С запуском среды она предложит придумать мастер пароль для вашей системы на рисунке 5. Служит этот пароль для разблокировки возможности сохранять пароли от серверов и других привилегий. Устанавливать его необязательно.

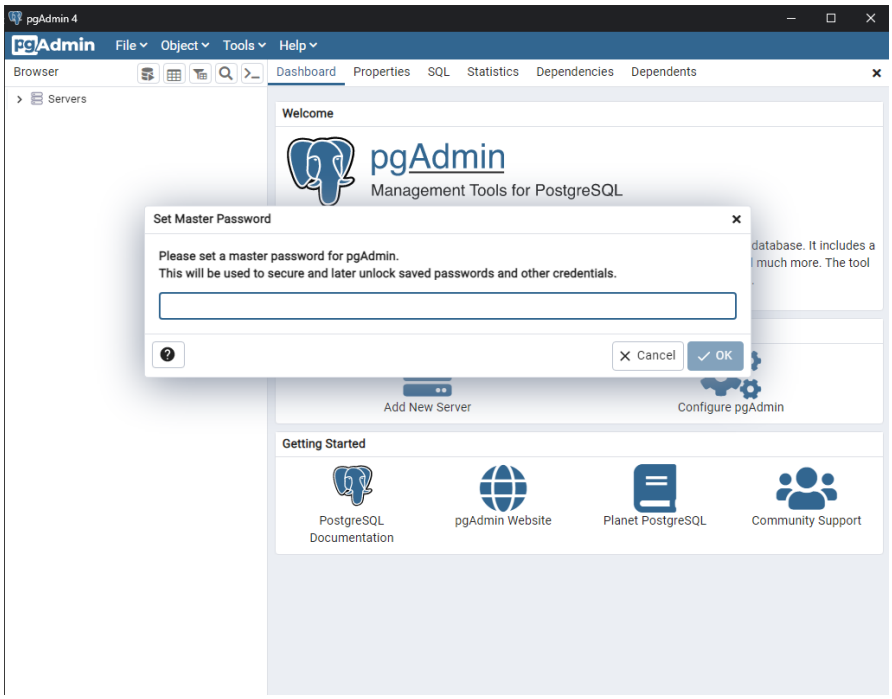

Рисунок 5 – Мастер пароль

Далее если вам нужно поменять тему или язык (либо что-то ещё) можно зайти в настройки на рисунке 6.

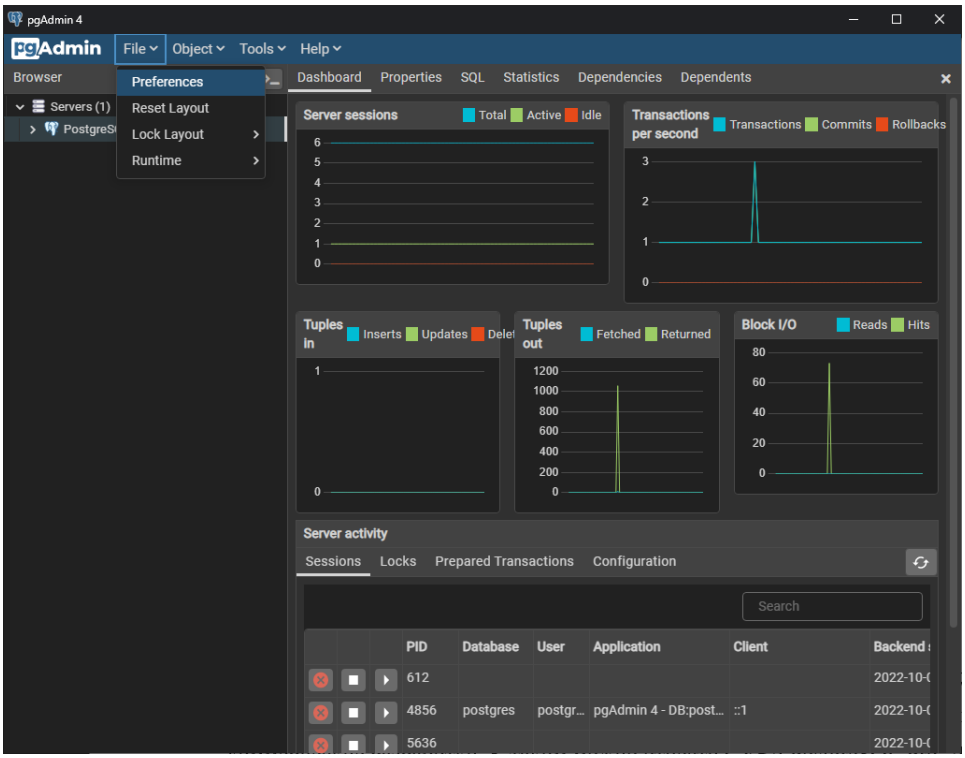

Рисунок 6 – Настройки

Коммерческая версия PostgreSQL Pro

СУБД PostgreSQL является бесплатным и свободно распространяющимся ПО, однако существуют также производные коммерческие разработки. К таким можно отнести СУБД PostgreSQL Pro. Для нее существует достаточно подробная документация на русском языке, которая также актуальна для СУБД PostgreSQL, так как СУБД PostgreSQL Pro является ее расширением и наследует всю функциональность: https://postgrespro.ru/docs/postgrespro/11/index

Англоязычная документация

Также можно воспользоваться англоязычной документацией для СУБД PostgreSQL: https://www.postgresql.org/docs/11/index.html

Установка СУБД PostgreSQL в Docker

Скачайте Docker с официального сайта и установите: https://www.docker.com/products/docker-desktop/

Настоятельно рекомендуется использовать актуальную версию ПО.

Официальные образы PostgreSQL расположены тут: https://hub.docker.com/_/postgres

На этой же странице описан алгоритм установки образа в Docker.

Зачем добавлять PostgreSQL в контейнер? Главное преимущество контейнеризации — сохранение и быстрое развертывание зависимостей.

Альтернатива PgAdmin

Универсальный инструмент для баз данных DBeaver. Это Бесплатный многоплатформенный инструмент для работы с базами данных для разработчиков, администраторов баз данных, аналитиков и всех, кому необходимо работать с базами данных. Поддерживает все популярные базы данных: MySQL, PostgreSQL, SQLite, Oracle, DB2, SQL Server, Sybase, MS Access, Teradata, Firebird, Apache Hive, Phoenix, Presto и др.

Скачать и установить можно на официальном сайте: https://dbeaver.io/

Порядок сдачи и сроки

Практические задания предполагают предварительную теоретическую и практическую подготовку студента, которую он демонстрирует в процессе сдачи задания. Кроме того, в процессе сдачи задания студент может получить дополнительные вопросы, ответы на которые необходимо дать до окончания защиты лабораторной работы.

Сроки сдачи каждой лабораторной не более 3х недель. После истечения данного срока, за сдачу лабораторной работы будут сниматься баллы. Готовый отчет необходимо загрузить на страницу курса «Базы данных. Лабораторные работы и КР». Защита лабораторных работ будет проходить на неделе следующей за дедлайном по лабораторной работе №4.

На защиту вы приносите только титульный лист, сам отчет, содержащий в себе результат выполнения всех четырех лабораторных работ в формате ФИО_Группа_Вариант_Название курса.pdf загружаете также на соответствующую страницу курса.

Практические задания выполняются по индивидуальным вариантам. Вариант содержит в себе предметную область для лабораторных заданий 2.*, 3.*, 4*: http://sp.cs.msu.ru/prak3/prak3_tasks.pdf

Узнать актуальный номер своего варианта можно в ЭУ: https://eu.bmstu.ru

Лабораторная работа №1. Основы языка SQL. SELECT запросы

Цель:

Первое практическое задание заключается в знакомстве со средой pgAdmin и написании SQL-запросов с использованием оператора SELECT.

Задачи:

- Получить опыт взаимодействия с консолью psql (Shell)
- Ознакомится со средой pgAdmin 4
- Изучение основ языка SQL

Теоретическая часть

Подготовка

Если будете повторять приведённые примеры и выполнять задания в модулях, то стоит посмотреть, как добавить БД из sql файла в части «Задание».

После проделанных манипуляций в pgAdmin должна появиться БД с название demo, как на рисунке 7.

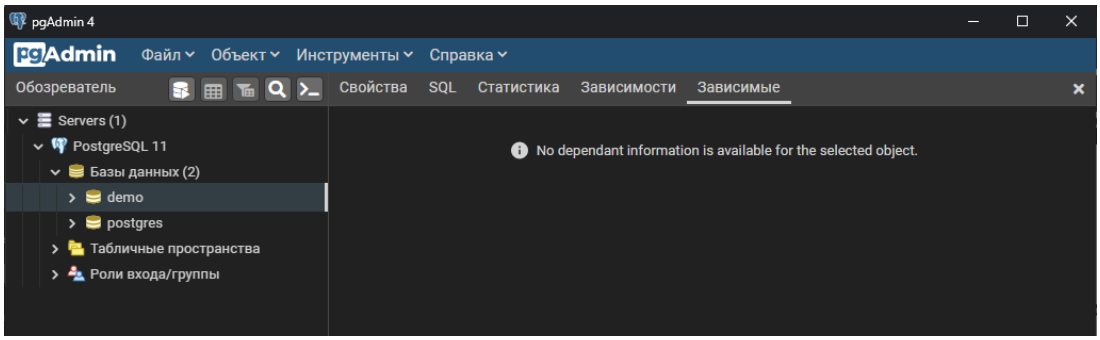

Рисунок 7 – Главное меню

Что такое SQL?

На основе этой базой данных будут приводиться примеры основных возможностей языка SQL. Но для начала надо разобраться что данный язык из себя представляет. SQL является инструментом, предназначенным для организации, управления, выборки и обработки информации, содержащейся в базе данных. Изначально в IBM этому языку было дано имя Structured English Query Language, сокращенно SEQUEL. В настоящее время аббревиатура SQL читается либо как "сиквел", либо как "эс-кю-эль", причем корректны оба произношения, хотя предпочтительно второе. Как следует из названия, SQL является языком программирования, который применяется для организации взаимодействия пользователя с базой данных.

На самом деле SQL работает только с базами данных определенного типа, называемыми реляционными базами данных, которые представляют собой основной способ организации данных в широком диапазоне приложений.

На рисунке 8 показана схема работы SQL с СУБД. Согласно этой схеме, в вычислительной системе имеется база данных, в которой хранится важная

информация. Если вычислительная система относится к сфере бизнеса, то в базе данных могут содержаться сведения о материальных ценностях, выпускаемой продукции, объемах продаж и зарплате. В базе данных на персональном компьютере может находиться информация о выписанных чеках, телефонах и адресах или информация, извлеченная из более крупной вычислительной системы. Компьютерная программа, которая управляет базой данных, называется системой управления базой данных, или же СУБД.

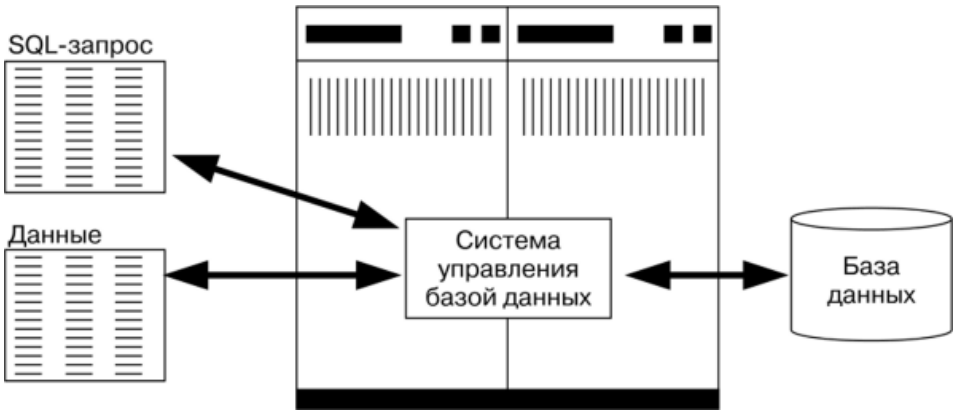

Рисунок 8 – Применение SQL для обращения к базе данных

SQL позволяет выполнять такие действия как:

- получать доступ к данным в системах управления РБД;
- описывать данные (их структуру);
- определять данные в БД и управлять ими;
- взаимодействовать с другими языками через модули SQL, библиотеки и предварительные компиляторы;
- создавать и удалять БД и таблицы;
- создавать представления, хранимые процедуры (stored procedures) и функции в БД;
- устанавливать разрешения на доступ к таблицам, процедурам и представлениям.

Основные команды SQL

Стандартными командами для взаимодействия с РБД являются CREATE, SELECT, INSERT, UPDATE, DELETE и DROP. Эти команды могут

быть классифицированы на три вида, но в данной лабораторной рассмотрим только одну команду из одной группы это SELECT (DML).

SELECT –выбрать данные из указанных столбцов и (если необходимо) выполнить перед выводом их преобразование в соответствии с указанными выражениями и (или) функциями.

Так же рассмотрим команды самих операторов, основные из них, которые будут использованы в данной работе перечислены в таблице 1.

Таблица 1 – Вспомогательные команды

Команда	**Описание**
FROM	Перечисляет используемые в запросе таблицы из базы данных
WHERE	Это условный оператор, который используется для ограничения строк по какому-либо условию
GROUP BY	Используется для группировки строк
ORDER BY	Используется для сортировки. У него есть два параметра:
ASC	(По умолчанию) используется для сортировки по возрастанию
DESC	По убыванию
INTO	Служит как вспомогательная команда для расширения возможностей основных операторов
AS	Задает новое имя полям или таблицам при выборке из базы (так же, как и INTO, необязателен)
VALUES	определяет строку значений, которые будут вставлены в таблицу или представление
LIMIT	Команда LIMI задает ограничение на количество записей, выбираемых из базы данных.

Стоит обратить внимание и на агрегатные функции выполняют вычисления над значениями в наборе строк. В T-SQL имеются следующие агрегатные функции:

- AVG: находит среднее значение
- SUM: находит сумму значений
- MIN: находит наименьшее значение
- MAX: находит наибольшее значение
- COUNT: находит количество строк в запросе

Синтаксис SQL

Синтаксис — это уникальный набор правил и рекомендаций. Все инструкции SQL должны начинаться с ключевого слова, такого как SELECT, INSERT, UPDATE, DELETE, ALTER, DROP, CREATE, USE, SHOW и т.п. и заканчиваться точкой с запятой (;) (точка с запятой не входит в синтаксис SQL, но ее наличия, как правило, требуют консольные клиенты СУБД для обозначения окончания ввода команды). SQL не чувствителен к регистру, т.е. SELECT, select и SeLeCt являются идентичными инструкциями. Исключением из этого правила является MySQL, где учитывается регистр в названии таблицы.

Таблицы типов данных

Таблица 2 – Точные числовые

Тип данных	От	До
bigint	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
int	-2,147,483,648	2,147,483,647
smallint	-32,768	32,767
tinyint	0	255
bit	0	1
decimal	-10^38 +1	10^38 -1
numeric	-10^38 +1	10^38 -1
money	-922,337,203,685,477.5808	+922,337,203,685,477.5807
smallmoney	-214,748.3648	+214,748.3647

Таблица 3 – Приблизительные числовые

Тип данных	От	До
float	-1.79E + 308	1.79E + 308
real	-3.40E + 38	3.40E + 38

Таблица 4 – Дата и время

Тип данных	От	До
datetime	Jan 1, 1753	Dec 31, 9999
smalldatetime	Jan 1, 1900	Jun 6, 2079
date	Дата сохраняется в виде June 30, 1991	
time	Время сохраняется в виде 12:30 P.M.	

Таблица 5 – Строковые символьные

Тип данных	Описание
char	Строка длиной до 8,000 символов (не-юникод символы, фиксированной длины)
varchar	Строка длиной до 8,000 символов (не-юникод символы, переменной длины)
text	Не-юникод данные переменной длины, длиной до 2,147,483,647 символов

Таблица 6 – Строковые символьные (юникод)

Тип данных	Описание
nchar	Строка длиной до 4,000 символов (юникод символы, фиксированной длины)
nvarchar	Строка длиной до 4,000 символов (юникод символы, переменной длины)
ntext	Юникод данные переменной длины, длиной до 1,073,741,823 символов

Таблица 7 – Бинарные

Тип данных	Описание
binary	Данные размером до 8,000 байт (фиксированной длины)
varbinary	Данные размером до 8,000 байт (переменной длины)
image	Данные размером до 2,147,483,647 байт (переменной длины)

Таблица 8 – Смешанные

Тип данных	Описание
timestamp	Уникальные числа, обновляющиеся при каждом изменении строки
uniqueidentifier	Глобально-уникальный идентификатор (GUID)
cursor	Объект курсора
table	Промежуточный результат, предназначенный для дальнейшей обработки

Таблицы логических операторов

Таблица 9 – Арифметические

Оператор	Описание	Пример
+ (сложение)	Сложение значений	a + b = 30
— (вычитание)	Вычитание правого операнда из левого	b — a = 10
* (умножение)	Умножение значений	a * b = 200
/ (деление)	Деление левого операнда на правый	b / a = 2
% (деление с остатком/по модулю)	Деление левого операнда на правый с остатком (возвращается остаток)	b % a = 0

Таблица 10 – Операторы сравнения

Оператор	Описание	Пример
=	Определяет равенство значений	a = b -> false
!=	Определяет НЕравенство значений	a != b -> true
<>	Определяет НЕравенство значений	a <> b -> true
>	Значение левого операнда больше значения правого операнда?	a > b -> false
<	Значение левого операнда меньше значения правого операнда?	a < b -> true
>=	Значение левого операнда больше или равно значению правого операнда?	a >= b -> false
<=	Значение левого операнда меньше или равно значению правого операнда?	a <= b -> true

!<	Значение левого операнда НЕ меньше значения правого операнда?	a !< b -> false
!>	Значение левого операнда НЕ больше значения правого операнда?	a !> b -> true

Таблица 11 – Логические операторы

Оператор	Описание
ALL	Сравнивает все значения
AND	Объединяет условия (все условия должны совпадать)
ANY	Сравнивает одно значение с другим, если последнее совпадает с условием
BETWEEN	Проверяет вхождение значения в диапазон от минимального до максимального
EXISTS	Определяет наличие строки, соответствующей определенному критерию
IN	Выполняет поиск значения в списке значений
LIKE	Сравнивает значение с похожими с помощью операторов подстановки
NOT	Инвертирует (меняет на противоположное) смысл других логических операторов, например, NOT EXISTS, NOT IN и т.д.
OR	Комбинирует условия (одно из условий должно совпадать)
IS NULL	Определяет, является ли значение нулевым
UNIQUE	Определяет уникальность строки

Термины

Чаще всего придётся писать запросы для выборки какой-либо информации из БД взаимодействуя с основными типами данных в ней, сущностями и их атрибутами.

Сущность – это базовые типы информации, которые хранятся в БД (в реляционной БД каждой сущности назначается таблица). К сущностям могут относиться в нашем случае: самолеты, аэропорты, билеты и т.д. Экземпляр

сущности и тип сущности - это разные понятия. К примеру, **типом сущности** таблицы «самолеты», является собственно самолет. А набор данных о конкретном самолете (Туполев Ту-134) называется **экземпляром сущности**.

Атрибут – это свойство сущности в предметной области. Его наименование должно быть уникальным для конкретного типа сущности. Например, для сущности самолет могут быть использованы следующие атрибуты: модель, расстояние, кол-во посадочных мест, и т.д. В реляционной БД атрибуты хранятся в полях таблиц.

Как писать запросы в среде pgAdmin4

Переходя к практике, начнём с самого простого, а именно получение данных с таблицы без каких-либо условий с помощью операторов SELECT и FROM. Для того чтобы открыть среду написания запросов нужно нажать на базы данных demo и сверху загорится нужная кнопка «Запросник», как на рисунке 9.

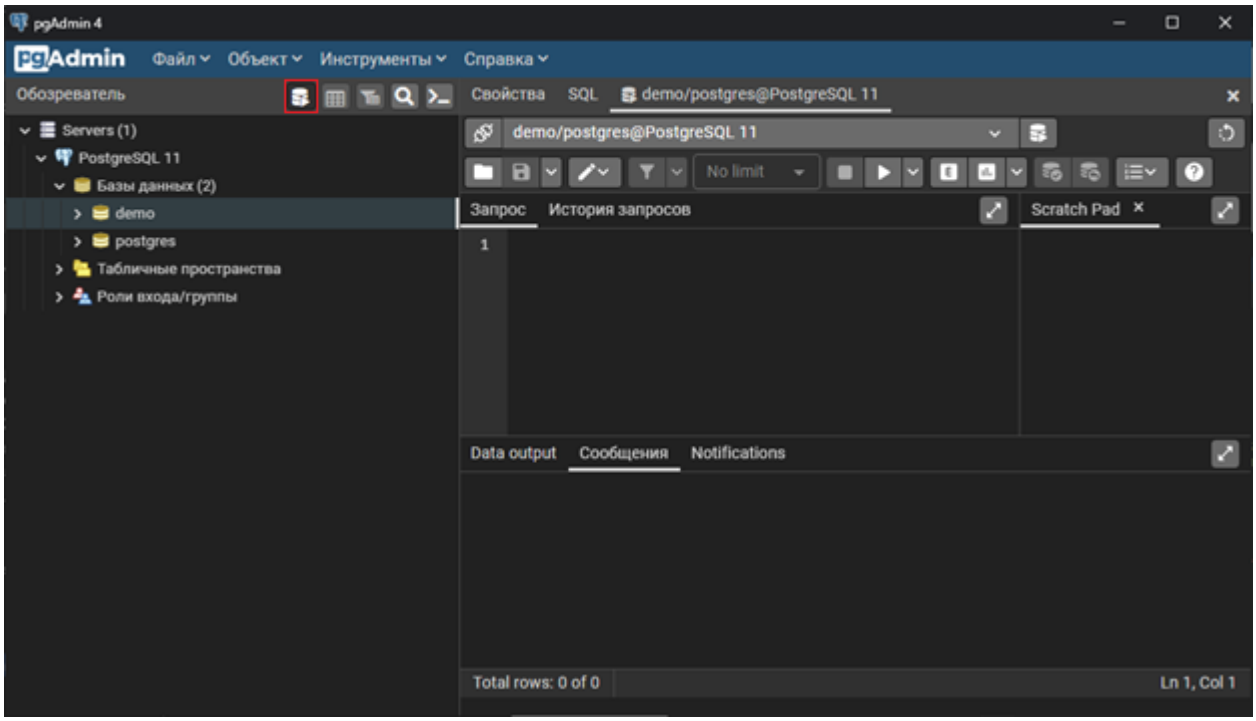

Рисунок 9 – Запросник

Чтобы понять к чему обращаться проанализируем структуру БД demo, начнем с главного для этой лабораторной работы, контентно со схем. Схемы нужны для того, чтобы разграничивать таблицы для упорядочивания их

использования и доступа разным пользователям. По умолчанию при создании базы данных создается схема «public», так же каждая создаваемая таблица, у которой не указана привязка к какой-либо схеме, будет автоматически в данной. В нашей же базе есть отдельная схема «booking» с ней и будут связанны большинство запросов, а именно с её таблицами на рисунке 10.

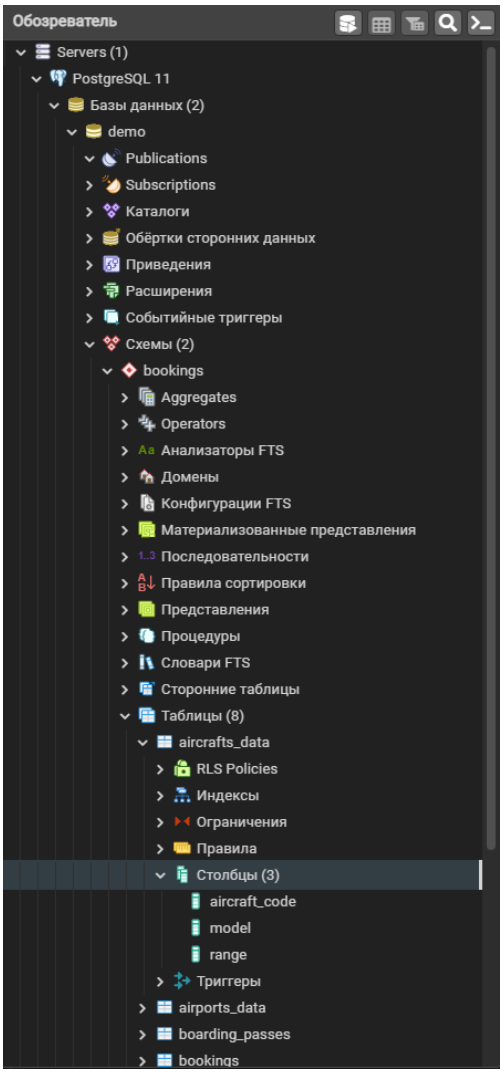

Рисунок 10 – Структура БД

SELECT и FROM, LIMIT, ROUND, AS

Оператор SELECT, как и говорилось выше предназначен для извлечения записей из таблиц, для его работы нужен с FROM, но для самого простого запроса с SELECT в SQL можно обойтись и без него. Записав, к примеру арифметическое выражение SELECT (10*5)/3 или же просто вывести цифру

SELECT 2, который выдаст результат в целочисленной форме. Это потому, что стоит помнить следующее, что в SQL работает следующая логика:

- Целое / Целое = Целое (т.е. в данном случае происходит целочисленное деление)
- Вещественное / Целое = Вещественное
- Целое / Вещественное = Вещественное

То есть если команду SELECT (10*5)/3 не слегка изменить на SELECT (10.0 * 5)/3, то результат будет вещественным.

Для чего может пригодиться такие действия? К примеру, нужно посчитать скидку 30% на билет.

Запрос:

```
SELECT amount, ROUND ((amount / 100), 2) * 30 as sale FROM ticket_flights LIMIT 100;
```

Последовательность запроса такова:

- С помощью SELECT и разделителем в виде запятой выбирается какие данные поместить в итоговую таблицу, в этом запросе запрошены данные о полной стоимости билета, (обращаясь по имени столбца, указанного в таблице) и подсчёт скидки (используя столбец amount) вкладывая функцию подсчета в функцию самого SQL ROUND для сокращения цифр после запятой до двух. (синтаксис прост, ROUND (значение, кол-во цифр после запятой))
- С помощью ключевого слова «as» даем имя будущему столбцу в таблице результатов запроса на «sale»
- С помощью FROM выбираем таблицу откуда будем брать данные в нашем случае «ticket_flights», из данной таблицы мы и берем значения столбца «amount»
- Для быстроты запроса добавим команду LIMIT 100 (так как в таблицу записей много)

Как это выглядит в самой среде можно посмотреть рисунке 11 (можно писать команды и вертикально, чтобы лучше читалось).

Для запуска кода нужно нажать кнопку «Play», на рисунке выделено синей рамкой!

А для сохранения запроса нужно нажать кнопку с «дискетой», на рисунке красной рамкой!

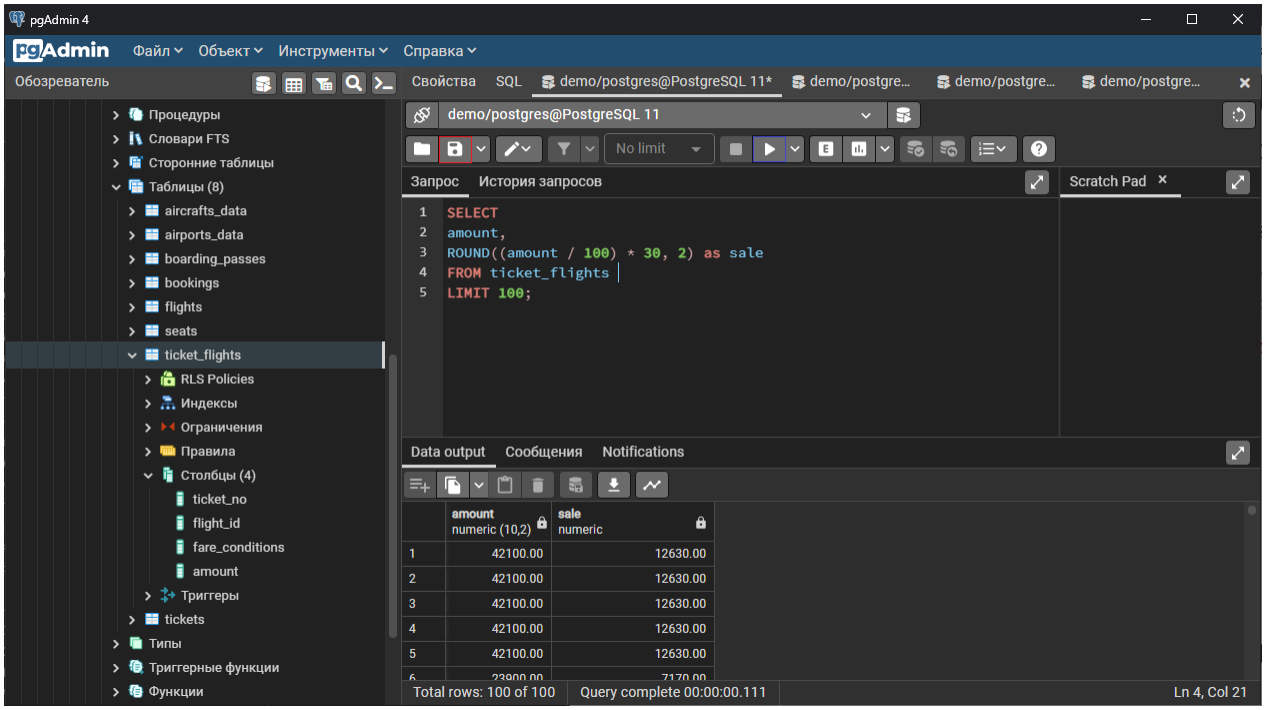

Рисунок 11 – Запрос №1

Задание для практики

Добавить ещё один столбец с итоговой ценой билета со скидкой 30%.

ORDER BY, GROUP BY, ASC, DESC, WHERE

WHERE

При выборке записей из таблицы практически всегда требуется задавать определённые условия, по которым мы определяем, какие записи нам нужны, а какие нет. И именно эти условия можно задавать с помощью конструкции WHERE в SQL. Для примера рассмотрим туже таблицу с билетами напишем запрос сортировке по цене в диапазоне от 3000 до 9000 и класс билета (Economy).

Запрос:

```
SELECT * FROM ticket_flights WHERE BETWEEN ‘3000’ AND ‘9000’ AND fare_conditions = ‘Economy’; (Важно! в одинарных кавычках)
```

Последовательность запроса такова:

- С помощью SELECT отбирается информация, которая будет отображена в итоговой таблице. В этом запросе это вся (для этого после SELECT нужно поставить символ умножения «*») информация о билете, который прошел фильтр.
- В блоке WHERE пишется информация, по которой будет проходить выборка из таблицы. Как и писалось выше это стоимость в диапазоне от 3000 до 9000 с помощью команды BETWEEN и эконом класс. С помощью логического AND (И) указывается какие ещё параметры должны быть учтены при выборке.

Как это выглядит в самой среде можно посмотреть рисунке 12.

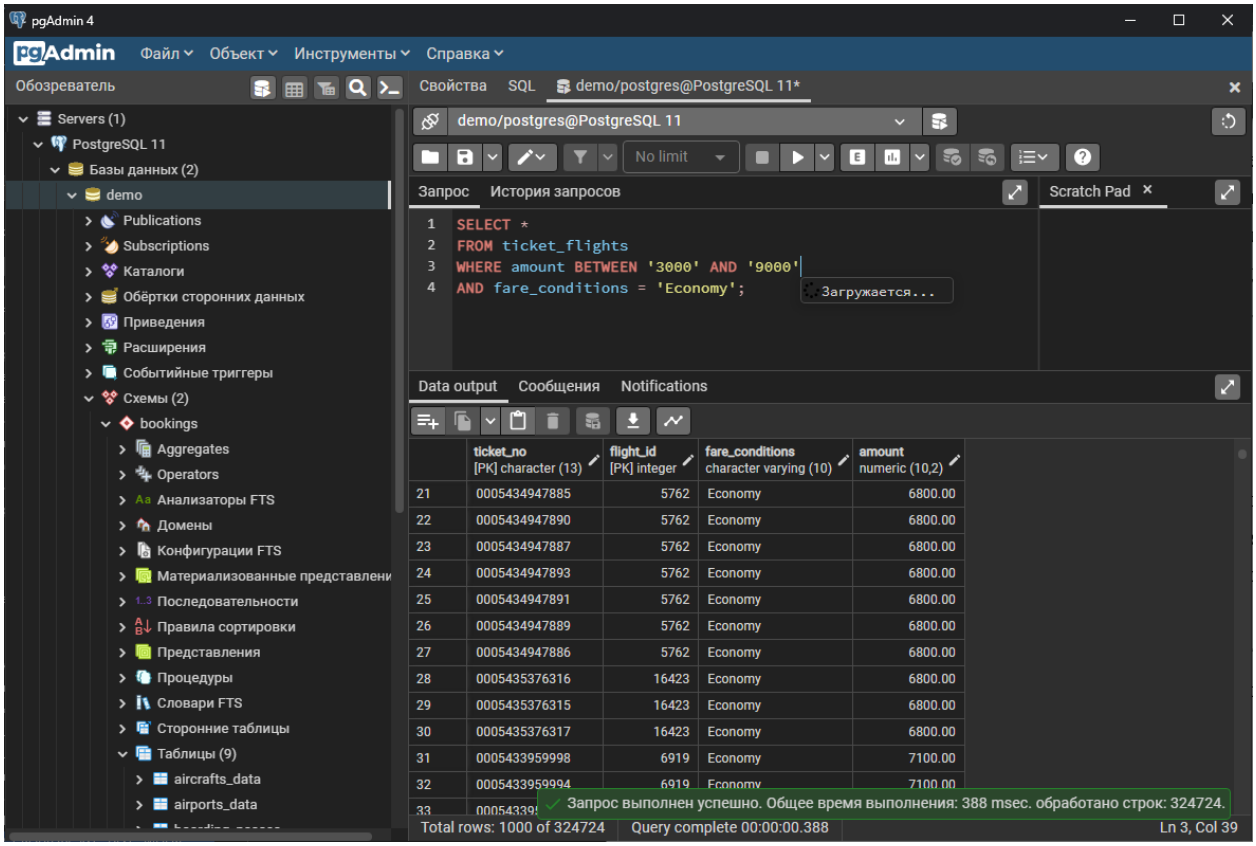

Рисунок 12 – Запрос №2

ORDER BY, ASC и DESC

При выполнении SELECT запроса, строки по умолчанию возвращаются в неопределенном порядке. Фактический порядок строк в этом случае зависит от плана соединения и сканирования, а также от порядка расположения данных на диске, поэтому полагаться на него нельзя. Для упорядочивания записей используется конструкция ORDER BY. Для примера возьмем предыдущий запрос с WHERE и модифицируем его, чтобы сортировать по цене.

Запрос:

SELECT * FROM ticket_flights WHERE amount < 5000 AND fare_conditions = ‘Economy’ ORDER BY amount DESC\ASC; (если не будет указано как упорядочить, то по умолчанию ASC)

Последовательность запроса такова:

- С помощью SELECT отбирается информация, которая будет отображена в итоговой таблице. В этом запросе это вся (для этого после SELECT нужно поставить символ умножения «*») информация о билете, который прошел фильтр.
- В блоке WHERE пишется информация, по которой будет проходить выборка из таблицы. Как и писалось выше это стоимость не выше 5000 и эконом класс.
- Далее в строке после ORDER BY выбирается по какому полю будет идти сортировка в нашем случае указываем стоимость, и ставим ключевое слово DESC что будет означать сортировку по убыванию

Как выглядит вариация с DESC в среде pgAdmin можно посмотреть рисунке 13.

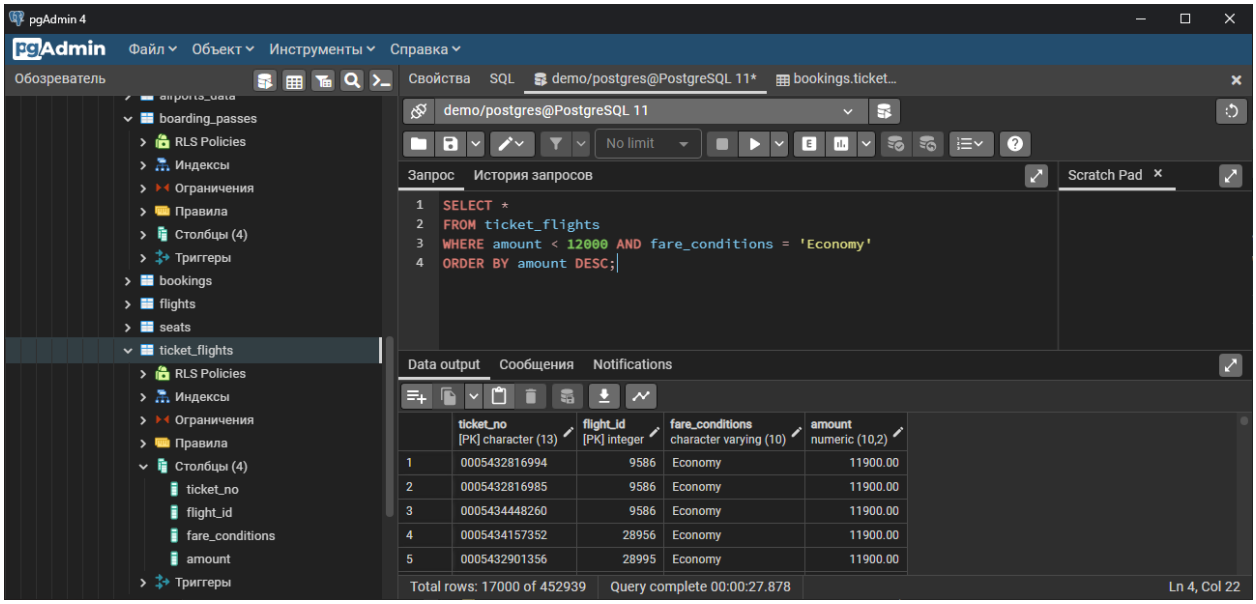

Рисунок 13 – Запрос №3

GROUP BY

Команда GROUP BY позволяет группировать результаты при выборке из базы данных. Группировку лучше всего использовать в месте с агрегатными функциями по типу: COUNT (подсчёт кол-ва строк), SUM (сумма значений), MAX (максимальное значение в строках), MIN (минимальное значение строках), AVG (среднее значение). Пример такого запроса может звучать так,

нужно сгруппировать таблицу seats по классу места (Economy, Comfort Business) и посчитать количество таких мест.

Функция COUNT принимает один из нескольких параметров:

- «*» – означает то, что функция COUNT возвращает все записи в таблице;
- column_name – функция COUNT возвращает количество записей конкретного столбца (только NOT NULL);
- DISTINCT column_name – функция COUNT возвращает количество только разных записей конкретного столбца (только NOT NULL).

Запрос:

```
SELECT fare_conditions, COUNT(*) FROM seats GROUP BY fare_conditions;
```

Последовательность запроса такова:

- С помощью SELECT отбирается информация, которая будет отображена в итоговой таблице. В этом запросе это названия групп (обязательное поле, когда используется GROUP BY) и подсчёт сгруппированных строк с помощью функции COUNT так же даем этому столбцу название «seats» (не путать с названием таблицы)
- В строке с FROM выбирается таблица для группировки
- Далее в строке после GROUP BY пишем название столбца, по которому будет проведена группировка

Как выглядит в среде pgAdmin можно посмотреть рисунке 14.

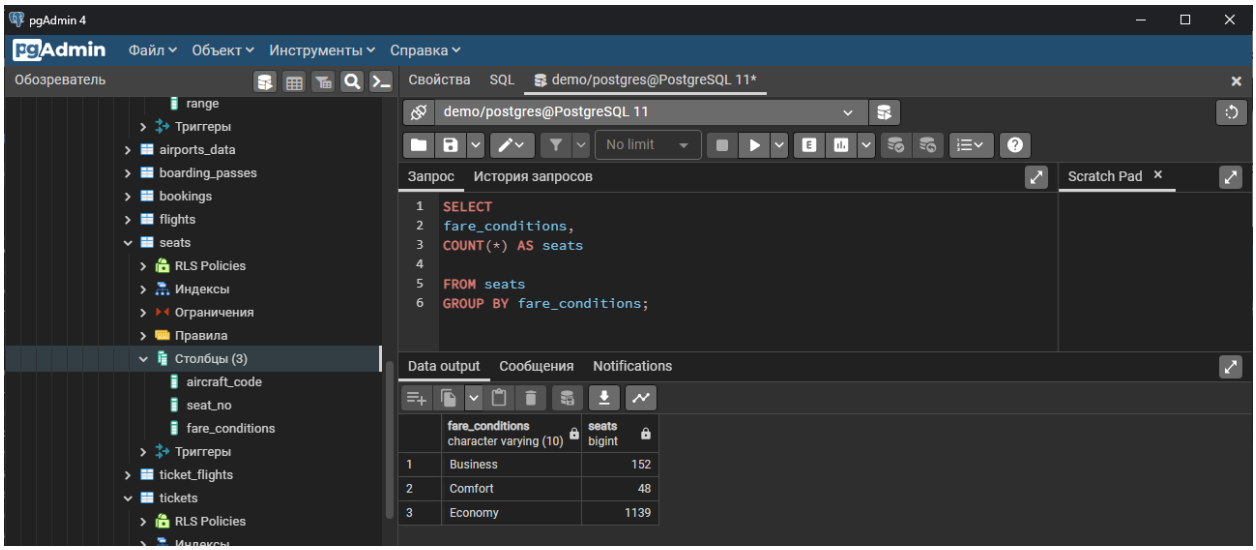

Рисунок 14 – Запрос №4

Оконные функции (OVER)

У вас может возникнуть вопрос – Что значит оконные? При обычном запросе все множество строк обрабатывается как бы единым «цельным куском», для которого считаются агрегаты. А при использовании оконных функций, запрос делится на части (окна) и уже для каждой из отдельных частей считаются свои агрегаты, на рисунке 15.

Рисунок 15 – Различая обычного запроса с оконным

Простым языком – это внутреннее деление на блоки. Более подробно рассмотрим на простом примере с таблицей tickets_data.

Запрос:

```
SELECT *, max(amount) OVER(PARTITION BY fare_conditions ORDER BY amount) as Data FROM ticket_flights;
```

Последовательность запроса такова:

С помощью SELECT отбираем все столбцы таблицы и дополняем это оконной функцией, которая добавит столбец максимального значения стоимости (так как оконные функции работает только с агрегатными функциями), однако благодаря этому элементы таблицы сгруппируются по столбцу после ключевого слова PARTITION BY (fare_conditions) и от сортируются по столбцу после ORDER BY (amount). (как параметра сортировки, так и группировки может и не быть вовсе) Так же называем новый столбец с помощью AS и берем данные из таблицы с помощью FROM.

Как выглядит в среде pgAdmin можно посмотреть рисунке 16.

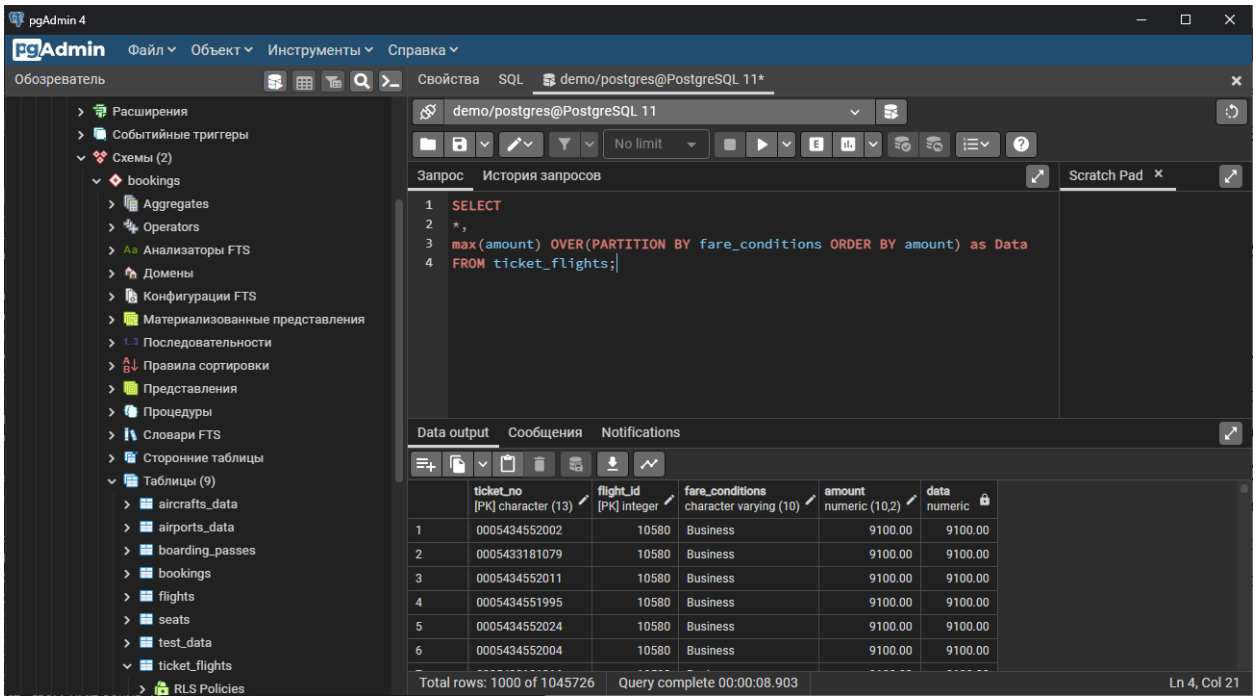

Рисунок 16 – Запрос №5

Задание для практики

Создать запрос с другими параметрами OVER и агрегатными функциями.

Оператор WITH и (иерархические) рекурсивные запросы

Рекурсивность в SQL построена на том, что таблица вызывает саму себя тем самым выстраивая иерархию вызов, по которому в итоге будет выведен результат запроса. Данный процесс реализуется с помощью ключевого слова WITH. Пример будет основан на первых запросах с подсчетом скидки 30%, разделим эту операции на часть с основным запросом (подсчет цены со скидкой) и дополнительным (подсчет скидки).

Запрос:

```
--Дополнительный
WITH sales AS (
 SELECT
ticket_no as number,
amount as price,
Round((amount / 100) * 30, 2) as sale
FROM ticket_flights)

--Основной запрос
SELECT
number,
price,
sale,
price - sale as with_sale
FROM sales
LIMIT 1000;
```

Последовательность запроса такова:

- С начало ключевое слово WITH потом название временной таблицы далее, как пишем AS и уже после пишем наш подзапрос для подсчета скидки для этого используем полученные раннее навыки. (Изменяя названия столбцов оригинальной таблицы во временной, далее нужно будет обращаться по изменённому имени)
- После переходим к основному запросу, где выбираем с помощью SELECT нужные нам столбцы и подчитывает цены со скидкой при это через FROM обращаемся уже к временной таблице. (Так поставлен лимит на количество строк в 1000 для быстроты запроса)

Как выглядит в среде pgAdmin можно посмотреть рисунке 17.

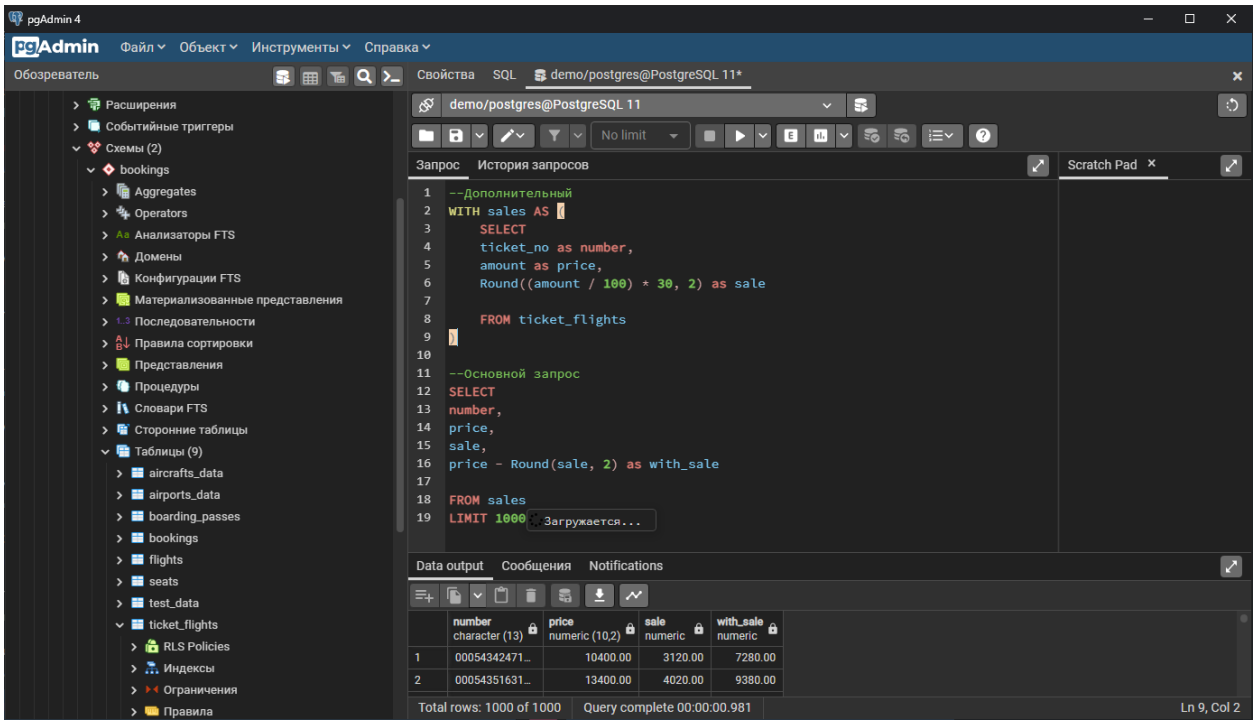

Рисунок 17 – Запрос №6

Задание для практики

Придумать свой запрос с имеющемся в базе таблицами.

Практическая часть

Задание для выполнения лабораторной работы

Первое практическое задание заключается в знакомстве со средой pgAdmin и написании SQL-запросов с использованием оператора SELECT.

Для модельной базы данных должны быть составлены 4 произвольных SELECT-запроса, демонстрирующие полученные знания. Запросы должны охватывать проработанные темы. После составления запросов следует убедиться в их правильности при помощи более простых запросов.

Дополнительные вопросы могут заключаться в построении более сложных запросов или объяснении работы подготовленных заданий.

Инструкция по добавлению сторонней базы данных

Скрипт для создания и заполнения модельной базы данных и её описание: https://postgrespro.ru/education/demodb

Для данной задачи нам понадобиться psql (shell), чтобы прочитать .sql файл. Программа psql – это терминальный клиент для работы с PostgreSQL. Она позволяет интерактивно вводить запросы, передавать их в PostgreSQL и видеть результаты. Также запросы могут быть получены из файла или из аргументов командной строки. Кроме того, psql предоставляет ряд мета команд и различные возможности, подобные тем, что имеются у командных оболочек, для облегчения написания скриптов и автоматизации широкого спектра задач.

Как только вы откроете консоль вам сразу же предложат ввести ip сервера, названия базы данных, порт, имя пользователя. В процессе установки PostgreSQL был создан сервер и БД со значениями по умолчанию, они и показаны в квадратных скобках, поэтому можно просто нажать несколько раз на Enter до момента ввода пароля на рисунке 18. И, собственно, ввести тот пароль, который вы вводили при установке. (P.S символы при вводе пароля могут не отображаться, но они записываются в строку пароля)

Рисунок 18 – Ввод данных

После если у вас наблюдаются ошибки с кодировкой, то это можно исправить с помощью команды: psql \! chcp 1251 (но это не обязательно, можно работать и без исправления этого предупреждения)

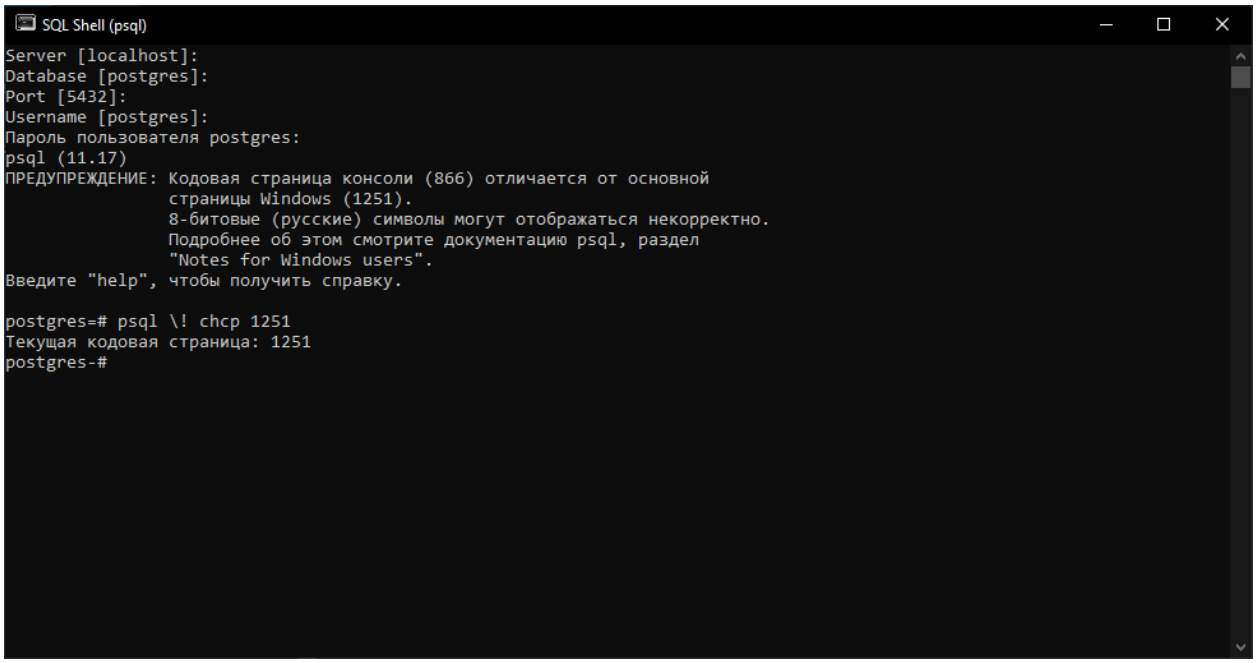

Рисунок 19 – Изменение кодировки

Далее приступим не посредственно к добавлению БД с помощью команды

\i ‘C:\\DataLab\\demo-small.sql’ где нужно указать команду \i (include) и путь к читаемому файлу в одинарных кавычках как на рисунке *. Как только вы нажмете Enter начнется считывание файла, которое займет некоторое время и после завершения в БД должна быть добавлена на ваш сервер.

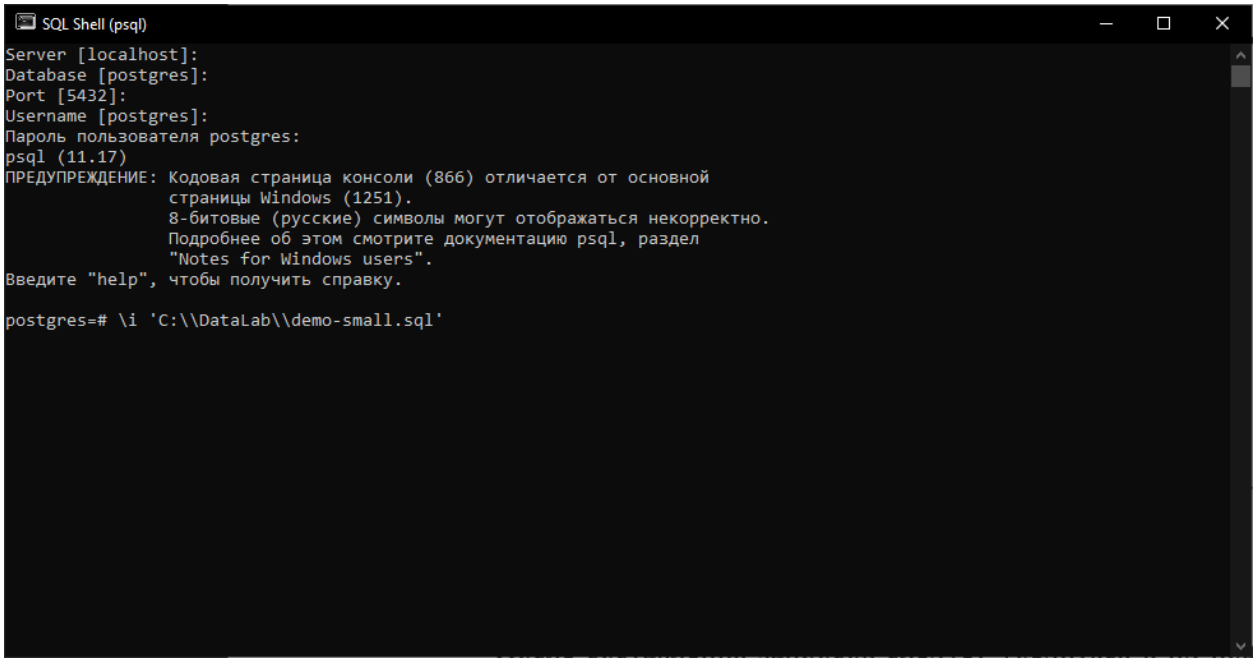

Рисунок 20 – Чтение sql-файла

Список ресурсов для самостоятельной проработки изученного синтаксиса

1. Синтаксис SQL: https://postgrespro.ru/docs/postgrespro/11/sql-syntax
2. SELECT-запросы: https://postgrespro.ru/docs/postgrespro/11/queries
3. Функции и операторы: https://postgrespro.ru/docs/postgrespro/11/functions
4. Оконные функции: https://postgrespro.ru/docs/postgrespro/9.5/tutorial-window
5. Полное описание синтаксиса встретившихся команд: https://postgrespro.ru/docs/postgrespro/11/sql-commands
6. Оператор With и рекурсивные запросы: https://postgrespro.ru/docs/postgrespro/11/queries-with
7. Работа в среде pgAdmin: https://metanit.com/sql/postgresql/1.1.php

Вопросы для самостоятельной проработки

1. Объяснить, как работают написанные запросы.
2. Рассказать про операцию соединения (JOIN) и различные её разновидности.
3. Рассказать про агрегатные функции, предложения GROUP BY и HAVING.
4. Как выбрать только уникальные значения какого-либо столбца?
5. Как осуществить сортировку по возрастанию/убыванию по значению какого-либо столбца?
6. Как агрегатные функции ведут себя по отношению к неопределённым значениям?
7. Рассказать о теоретико-множественных операциях в SQL.
8. Чем отличаются UNION и UNION ALL?
9. Чем отличаются COUNT(*) и COUNT(field)?
10. Как подсчитать количество уникальных значений столбца?
11. Как можно осуществить проверку на неопределенное значение?
12. Рассказать про предикат LIKE.
13. Как можно выбрать только определенное количество строк?
14. Чем SQL-таблица отличается от отношения?
15. Исправить неверно работающий запрос (запросы).
16. Упростить один или несколько запросов.
17. Округлить результирующее значение до 3 знаков после точки.
18. Округлить вещественное число до целого без нулей после точки.
19. Переписать запрос, не используя функцию MAX (MIN).
20. Изменить формат вывода данных (например, формат даты и времени).
21. Написать или модифицировать запрос по сформулированному заданию.