

**Министерство транспорта Российской Федерации
Федеральное государственное автономное образовательное
учреждение высшего образования
«Российский университет транспорта»
(МИИТ)**

Институт транспортной техники и систем управления

Кафедра «Управление и защита информации»

М. А. Васильева

К. М. Филипченко

В. А. Пугачёв

**«Фильтрация набор данных».
Рекомендации по выполнению работы и
перечень типовых заданий**

Учебно-методическое пособие

Москва 2024

Васильева М. А., Филипченко К. М., Пугачёв В. А. Фильтрация набора данных. Рекомендации по выполнению работы и перечень типовых заданий: Учебно-методическое пособие. Издание третье, исправленное и дополненное. Москва–2024, 133 с.

Учебно-методическое пособие предназначено для обучающихся по направлению «Управление в технических системах» профиля «Управление и информатика в технических системах» (27.03.04), а также обучающихся по специальности «Компьютерная безопасность» специализации «Информационная безопасность объектов информатизации на базе компьютерных систем» (10.05.01).

В данном издании приведены краткие сведения об операторах SQL диалекта СУБД Postgres Pro, необходимых для фильтрации данных. Операторы и предикаты, рассмотренные в данном пособии, являются начальными и необходимыми для дальнейшего изучения SQL, а также для выполнения и сдачи лабораторных работ. В пособии приведены рекомендации по выполнению лабораторной работы «Фильтрация набора данных» с применением системы контроля версий Git, содержится перечень вопросов по вариантам. Использование системы контроля версий Git является необходимым для выполнения лабораторных работ по дисциплинам «Информационное обеспечение систем управления» для бакалавров и «Основы построения защищенных баз данных» для специалистов.

Рецензент:

к. т. н., доцент кафедры «Автоматика, телемеханика и связь на железнодорожном транспорте» РУТ (МИИТ) Ваньшин А. Е.

© Васильева М. А., 2024
© Филипченко К. М., 2024
© Пугачёв В. А., 2024

ОГЛАВЛЕНИЕ

Цель работы	6
Требования к работе:	8
Установка СУБД PostgreSQL	8
Установка DBeaver	10
Установка базы данных AdventureWorksPostgresPro.....	10
Подключение к базе данных в программе DBeaver.....	12
Краткие сведения об операторах SQL.....	19
Выбор полей	19
Фильтрация данных	24
Предикат LIKE.....	26
Регулярные выражения SIMILAR TO.....	29
Регулярные выражения POSIX	30
Предикат IN.....	33
Предикат BETWEEN	35
Преобразование типов данных	37
Работа с NULL значениями	41
Предложение DISTINCT	47
Сортировка результирующего набора данных	49
Функция SUBSTRING.....	52
Рекомендации по выполнению работы:.....	53
Создание нового локального репозитория	53

Создание нового репозитория	53
Выполнение лабораторной работы	55
Рекомендации по исправлению кодировки в имеющемся файле	57
Порядок действий при исправлении ошибок.....	59
Последовательность действий при разработке отчета.....	60
Задание на лабораторную работу	65
ВАРИАНТ 1	65
ВАРИАНТ 2.....	67
ВАРИАНТ 3.....	69
ВАРИАНТ 4.....	71
ВАРИАНТ 5.....	73
ВАРИАНТ 6.....	75
ВАРИАНТ 7.....	77
ВАРИАНТ 8.....	79
ВАРИАНТ 9.....	81
ВАРИАНТ 10.....	83
ВАРИАНТ 11.....	85
ВАРИАНТ 12.....	87
ВАРИАНТ 13.....	89
ВАРИАНТ 14.....	91
ВАРИАНТ 15.....	93
ВАРИАНТ 16.....	95
ВАРИАНТ 17.....	97
ВАРИАНТ 18.....	99

ВАРИАНТ 19.....	101
ВАРИАНТ 20.....	103
ВАРИАНТ 21.....	105
ВАРИАНТ 22.....	107
ВАРИАНТ 23.....	109
ВАРИАНТ 24.....	111
ВАРИАНТ 25.....	113
Список использованных источников	115
ПРИЛОЖЕНИЕ А.....	119
ПРИЛОЖЕНИЕ Б.....	120

ЦЕЛЬ РАБОТЫ

Изучить операторы SQL на примере диалекта СУБД Postgres Pro [1], необходимые для фильтрации набора данных. Научиться создавать простые запросы на фильтрацию данных. Подготавливать отчет, выполненный согласно требованиям ГОСТ 7.32–2017 [2].

ВВЕДЕНИЕ

В соответствии с Федеральным государственным общеобразовательным стандартом выпускники, освоившие программу специалитета по специальности 10.05.01 Компьютерная безопасность, и программу бакалавриата по направлению подготовки 27.03.04 Управление в технических системах, должны в том числе освоить общепрофессиональные компетенции, связанные со способностью проектировать базы данных [3], [4]. Для освоения вышеуказанных компетенций в учебном плане имеются соответствующие дисциплины. В рамках преподавания данных дисциплин студенты изучают в том числе язык структурированных запросов SQL (Structured Query Language). Для выполнения лабораторных, практических и курсовой работ требуется наличие системы управления базами данных (СУБД).

Преподавание данных дисциплин начиналось в 2000-х годах с использованием СУБД Paradox for Window [5]. Был поставлен цикл лабораторных работ, разработаны индивидуальные задания и написаны методические указания [6], [7], [8], [9], [10], [11]. Выбор данной СУБД обоснован тем, что курс, связанный с преподаванием программирования, также читаемый авторами, изучался с использованием языка программирования Delphi [12]. СУБД Paradox являлась на тот момент embedded in Delphi application.

Начиная с 2011 года, проведение занятий по данным дисциплинам авторами осуществлялось с использованием СУБД MS SQL Server [13]. Для

изучения SQL, компания Microsoft предлагает учебную базу данных (БД) AdventureWorks, заполненную демонстрационными данными. Для проведения лабораторных и практических работ студентам были предложены задания, разработанные в компании Microsoft специально для этой БД. Частично вопросы, которые относились к соответствующей теме, вошли в пособие [14].

После проведения учебного процесса в дистанционном формате авторам пришлось пересмотреть лабораторные работы и подход к их выполнению и сдаче. Сдача скриптов работ, а затем отчетов происходит с применением системы контроля версий [15], [16], [17]. В соавторстве со студентами написаны индивидуальные задания для трех лабораторных работ. В 2021–2022 учебном году эти вопросы были апробированы, переформулированы, написаны тесты на задания. Методические указания по трем лабораторным работам и рекомендации по выполнению курсового проектирования с использованием СУБД MS SQL Server были готовы к печати, но по технически причинам, были сданы только в январе 2023 года [18], [19], [20], [21]. Также подготовленный материал был переработан в учебник [22].

С началом СВО многие иностранные производители программного обеспечения ушли с российского рынка, и никто не гарантирует стабильную работу оставшихся иностранных продуктов [23]. В связи со сложившейся обстановкой принято решение использовать российскую СУБД с открытым исходным кодом Postgres Pro (<https://postgrespro.ru/>).

Так как авторы разработали значительное количество вопросов для учебной базы данных AdventureWorks, то было принято решение о перенесении части описания предметной области вышеупомянутой БД, переориентировав её на СУБД Postgres Pro [24].

Авторами разработаны скрипты на создание необходимых таблиц. В результате проделанной работы задания, ориентированные на БД

AdventureWorks, могут быть использованы для БД под управлением СУБД Postgres Pro. Для распространения учебной БД был сформирован файл, который можно скачать со страницы авторов (<https://gitflic.ru/project/docent-paley/adventureworkspostgrespro>).

ТРЕБОВАНИЯ К РАБОТЕ:

Работа производится с демонстрационной базой данных для СУБД AdventureWorksPostgresPro. Все запросы и итоговый отчет должны разрабатываться с применением системы контроля версий Git. Подробное описание работы с системой контроля версий Git изложено в учебно-методическом пособии «Введение в систему контроля версий Git» [15].

УСТАНОВКА СУБД POSTGRESQL

Для выполнения лабораторных работ и курсового проектирования можно скачать бесплатную последнюю версию СУБД PostgreSQL с сайта <https://www.postgresql.org/download/>. Или скачать текущую версию СУБД Postgres Pro в тестовом режиме с сайта <https://postgrespro.ru/>.

Российская СУБД Postgres Pro обладает значительным преимуществом перед СУБД PostgreSQL в оптимизации запросов и т. д [1]. Данные преимущества не понадобятся при изучении языка SQL в рамках данного курса.

Для установки СУБД PostgreSQL зайдите на главную страницу (Рисунок 1) и выберите версию для своей операционной системы.

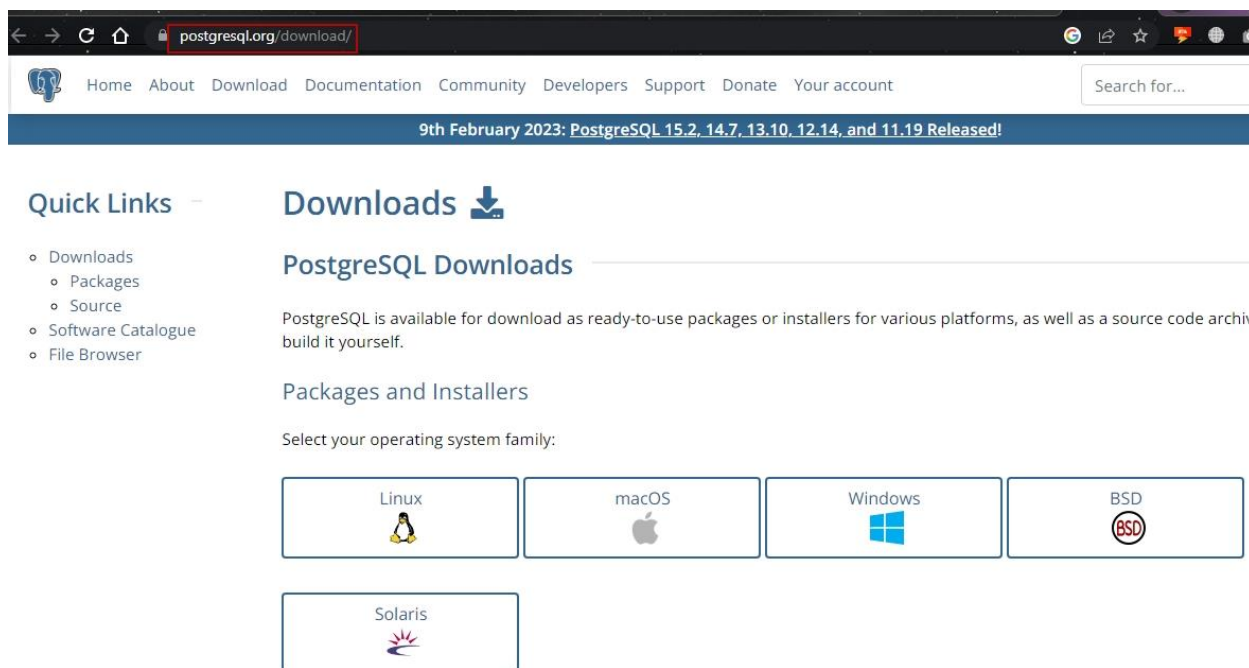


Рисунок 1 – Главная страница загрузки СУБД PostgreSQL

После выбора способа загрузки (Package and Installers) откроется окно, представленное ниже (Рисунок 2).

Windows installers

Interactive installer by EDB

[Download the installer](#) certified by EDB for all supported PostgreSQL versions.

Note! This installer is hosted by EDB and not on the PostgreSQL community servers. If you have issues with the website it's hosted on, please contact webmaster@enterprisedb.com.

This installer includes the PostgreSQL server, pgAdmin; a graphical tool for managing and developing your databases, and StackBuilder; a package manager that can be used to download and install additional PostgreSQL tools and drivers. Stackbuilder includes management, integration, migration, replication, geospatial, connectors and other tools.

Рисунок 2 – Окно перехода для установки (пример ОС Windows)

Нажмите на гиперссылку [Download the installer](#). Данная установка включает в себя PostgreSQL сервер и графический интерфейс для управления базами данных pgAdmin. В результате откроется окно, представленное ниже (Рисунок 3).

The installers are tested by EDB on the following platforms. They can generally be expected to run on other comparable versions, for example, desktop releases of Windows:

PostgreSQL Version	64 Bit Windows Platforms	32 Bit Windows Platforms
16	2022, 2019	
15	2019, 2016	
14	2019, 2016	
13	2019, 2016	
12	2019, 2016, 2012 R2	
11	2019, 2016, 2012 R2	
10	2016, 2012 R2 & R1, 7, 8, 10	2008 R1, 7, 8, 10

Рисунок 3 – Выберите версию PostgreSQL для своей ОС

Установите СУБД PostgreSQL.

ВАЖНО! При установке СУБД попросит вас ввести пароль. Запомните его!

УСТАНОВКА DBEAVER

DBeaver – это приложение, предназначенное для управления базами данных. Взаимодействовать с реляционными базами данных ему помогает интерфейс JDBC. Редактор DBeaver позволяет применять большое количество дополнительных плагинов и дает подсказки по заполнению кода, подсвечивая синтаксис [25]. По мнению авторов, в данном приложении удобнее работать с базами данных, чем в pgAdmin. Подробная инструкция по установке приложения представлена в ПРИЛОЖЕНИИ Б.

УСТАНОВКА БАЗЫ ДАННЫХ ADVENTUREWORKSPRO

Перейдите на страницу с репозиторием учебной базы (<https://gitflic.ru/project/docent-paley/adventureworkspostgrespro>). Рисунок с репозиторием представлен ниже (Рисунок 4).

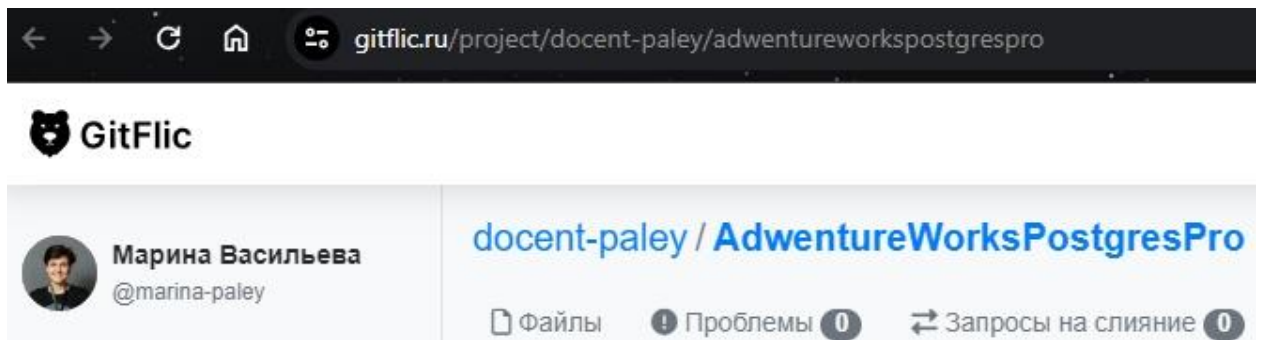


Рисунок 4 – Репозиторий

На странице репозитория выберите вкладку **Релизы** (Рисунок 5) и нажмите на *Архив БД*. В открывшемся окне выберите файл для скачивания (Рисунок 6).

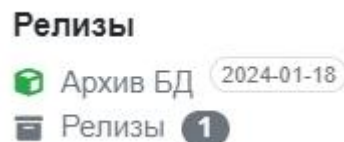


Рисунок 5 – Вкладка Релизы

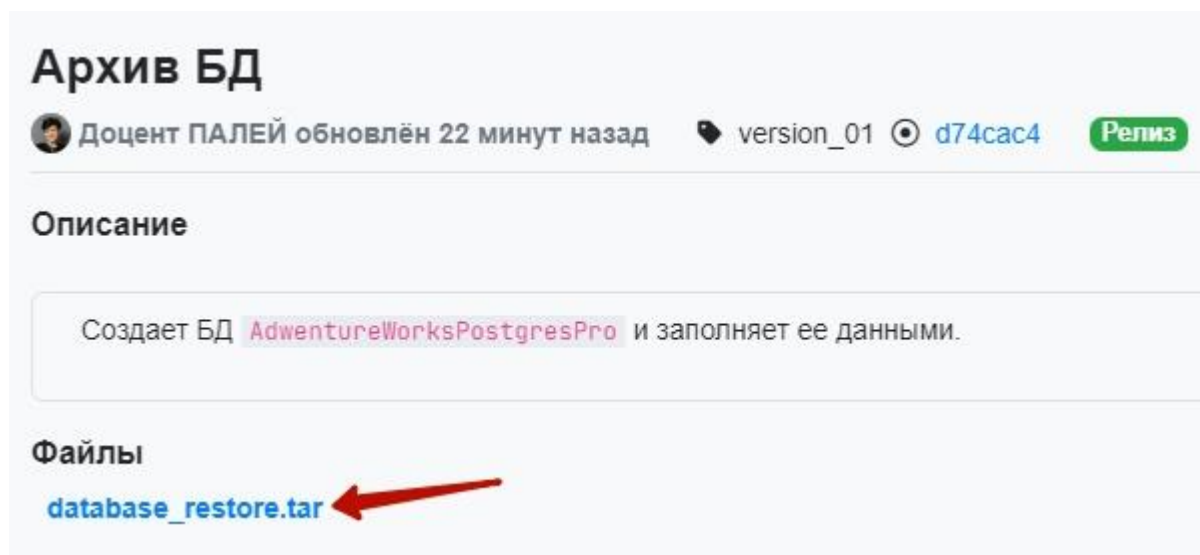


Рисунок 6 – Скачивание необходимого файла

Скачайте файл `database_restore.tar`.

ПОДКЛЮЧЕНИЕ К БАЗЕ ДАННЫХ В ПРОГРАММЕ DBEAVER

Для развертывания учебной базы необходимо создать пустую базу данных. Для этого нажмите правой кнопкой мыши на вкладку Database и в открывшемся меню выберите пункт Create New Database (Рисунок 7).

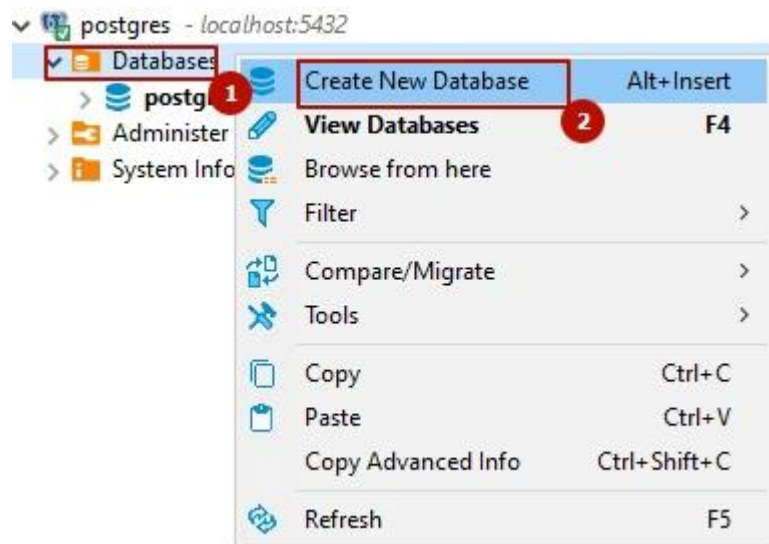
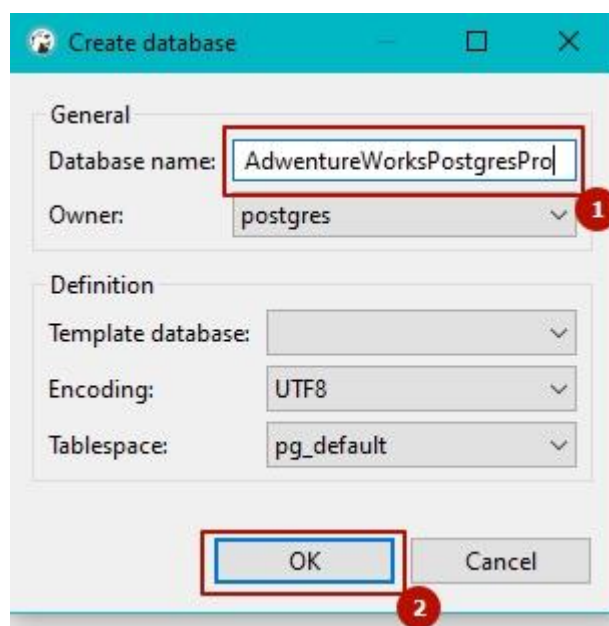


Рисунок 7 – Создание новой базы данных

В открывшемся окне напишите имя БД AdventureWorksPostgresPro (Рисунок 8).



**Рисунок 8 – Создание новой базы данных
AdventureWorksPostgresPro**

Выберите пункт меню Tools/ Execute script (Рисунок 9), нажав правой кнопкой мыши на созданную БД.

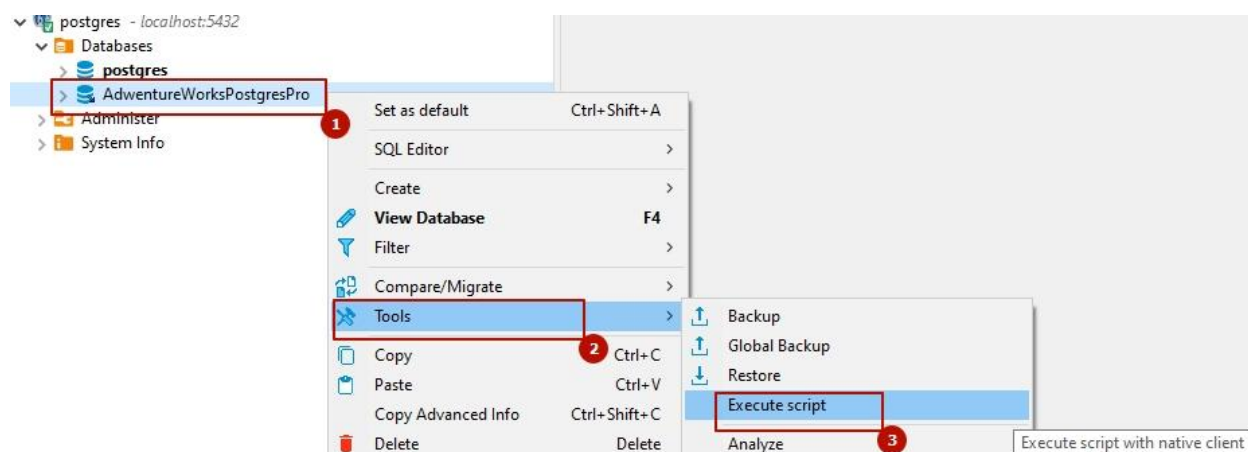


Рисунок 9 –Пункт выполнения скрипта

В открывшемся окне (Рисунок 10) выберите скачанный файл (Рисунок 10).

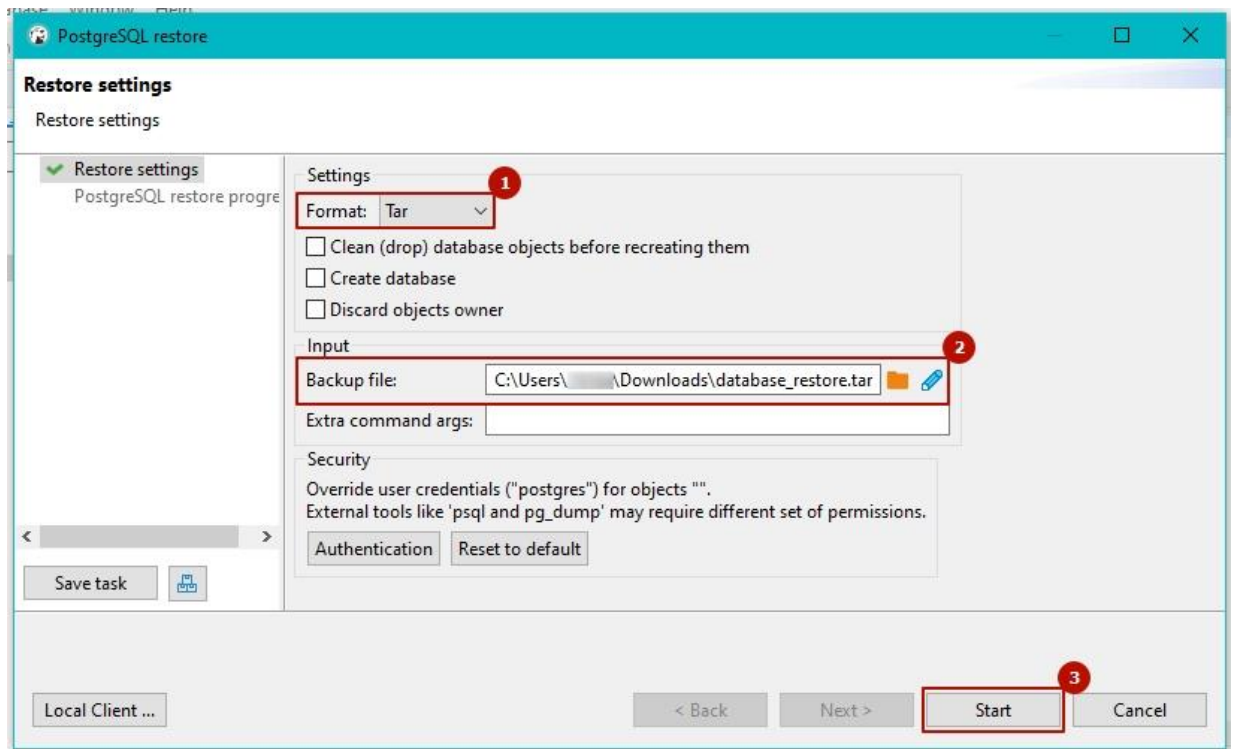


Рисунок 10 – Выбор скрипта на исполнение

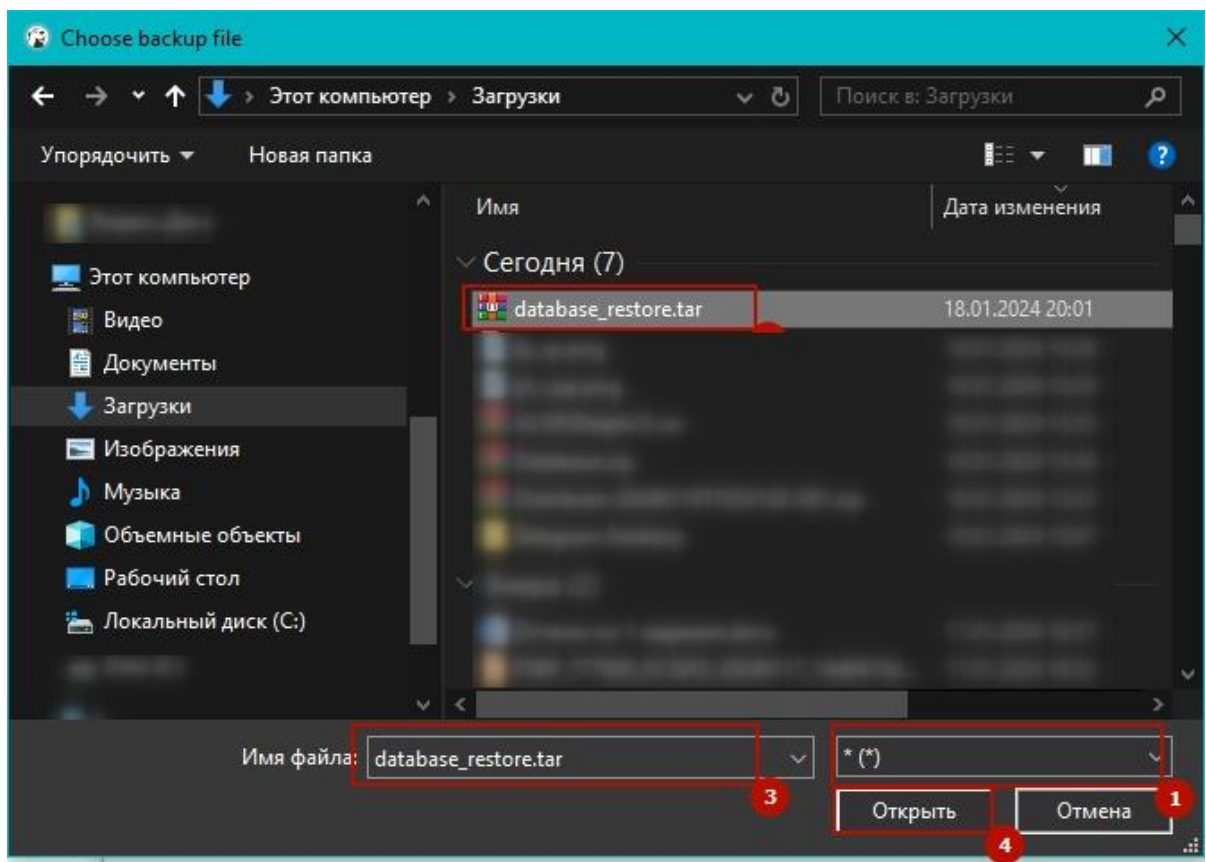


Рисунок 11 – Выбор файла для восстановления БД

Нажмите на кнопку Start (Рисунок 10) и подтвердите обновление БД (Рисунок 12).

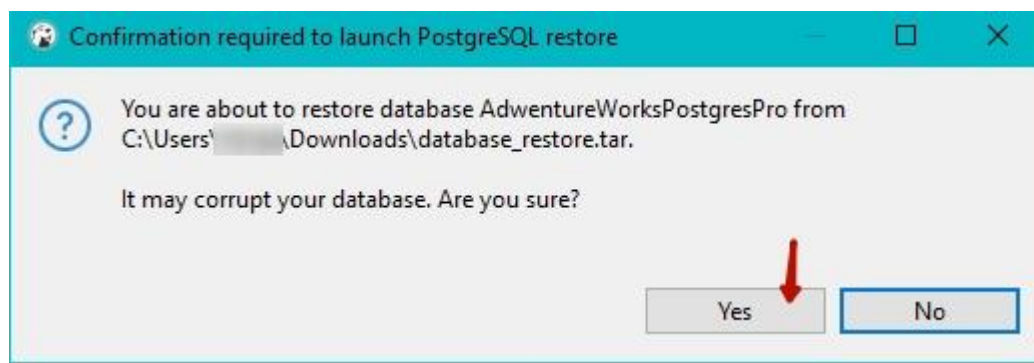


Рисунок 12 – Подтверждение изменения БД

Дождитесь окончания выполнения скрипта (Рисунок 13) и закройте ОКНО.

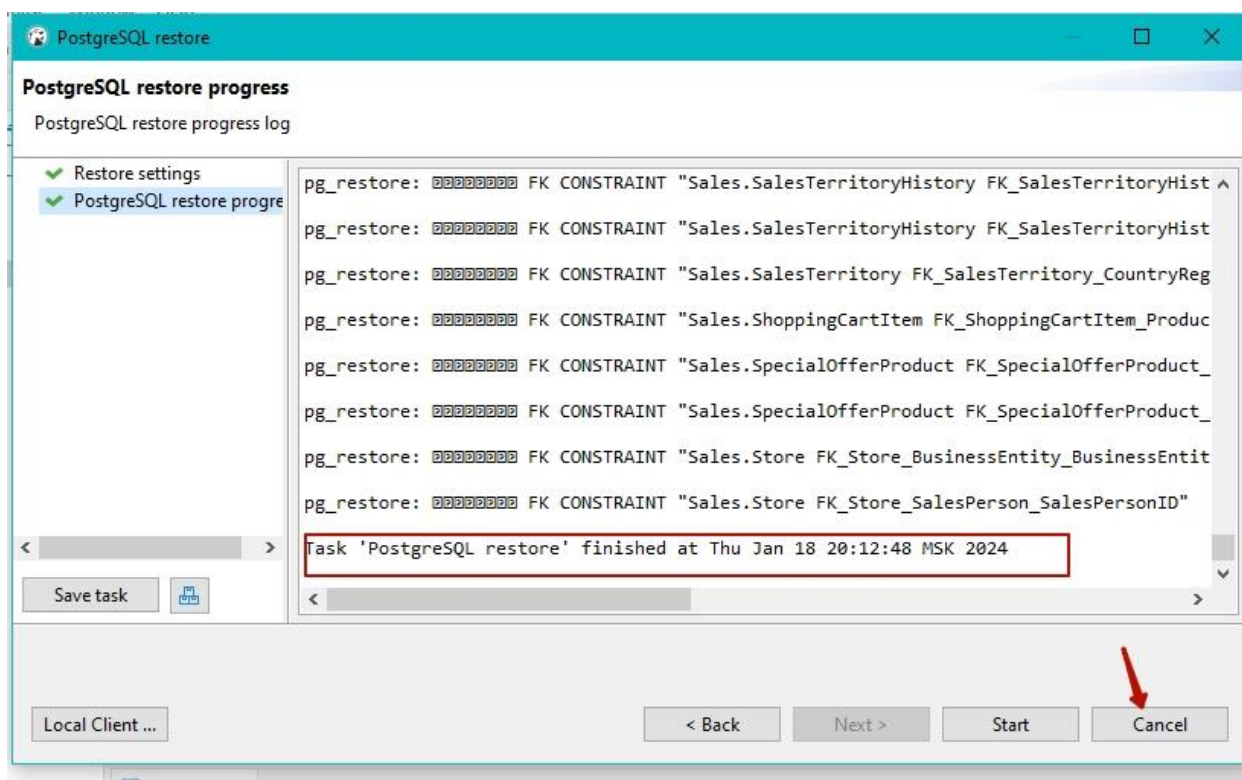


Рисунок 13 – Скрипт выполнен успешно

Обновите базу данных (Рисунок 14) и установите ее как используемую по умолчанию (Рисунок 15).

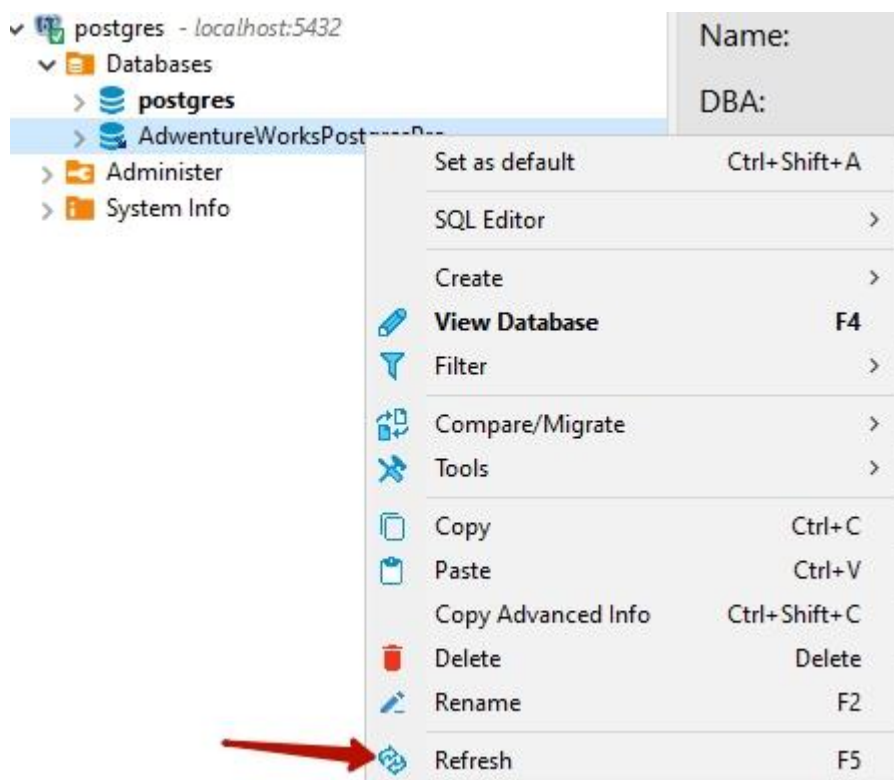


Рисунок 14 – Обновление базы данных

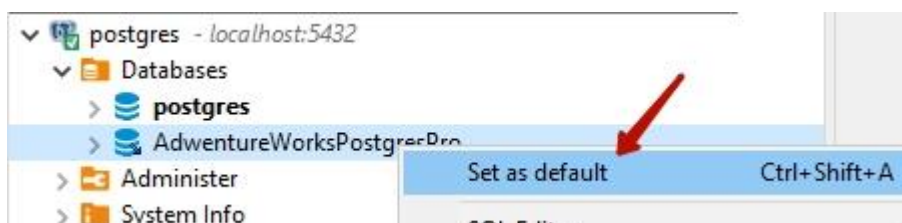


Рисунок 15 – Установка базы данных по умолчанию

Проверьте, что БД развернута правильно. Откройте ее, убедитесь, что присутствуют все необходимые схемы, таблицы имеют данные (Рисунок 16).

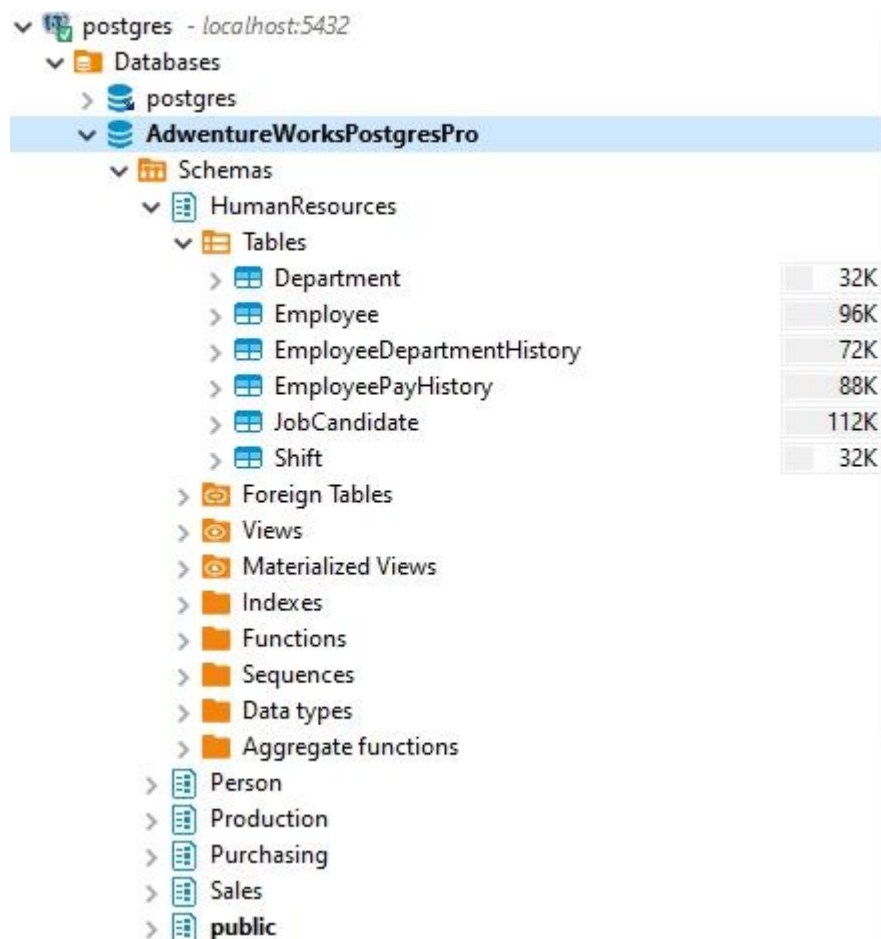


Рисунок 16 – БД развернута правильно

Для того, чтобы DBeaver отражал все имеющиеся БД, выберите соответствующие опции, представленные на рисунке ниже (Рисунок 17).

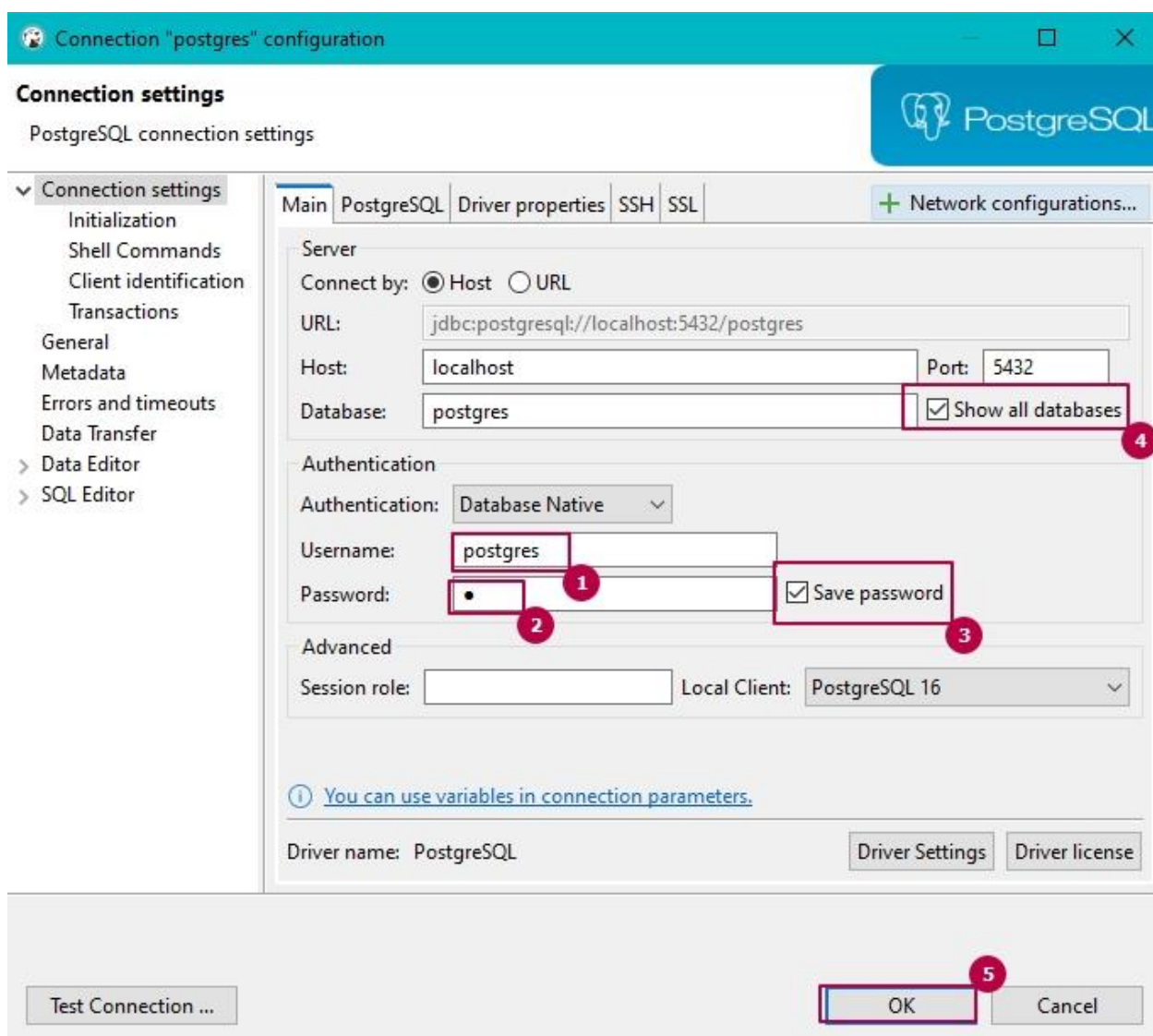


Рисунок 17 – Изменение соединения с БД

КРАТКИЕ СВЕДЕНИЯ ОБ ОПЕРАТОРАХ SQL

Structured Query Language (SQL) – язык структурированных запросов можно разделить на несколько подразделов [14], [20], [22]:

Data Definition Language (DDL) – язык определения данных,

Data Manipulation Language (DML) – язык манипулирования данными,

Data Query Language (DQL) – язык запросов.

Начнем изучение операторов DQL с последнего подраздела.

Выбор полей

Любой запрос начинается с оператора SELECT (выбрать), после которого через запятую следует перечислить список полей (выражений), значения которых требуется выбирать. Имя таблицы или поля, учитывая регистр, следует экранировать, заключив в кавычки.

```
SELECT
    выражение
    , выражение2
    , ...;
```

В предложении SELECT можно записать следующее выражение:

```
SELECT 1;
```

Выберите опцию New SQL script (Рисунок 18) и в новом окне запишите запрос.

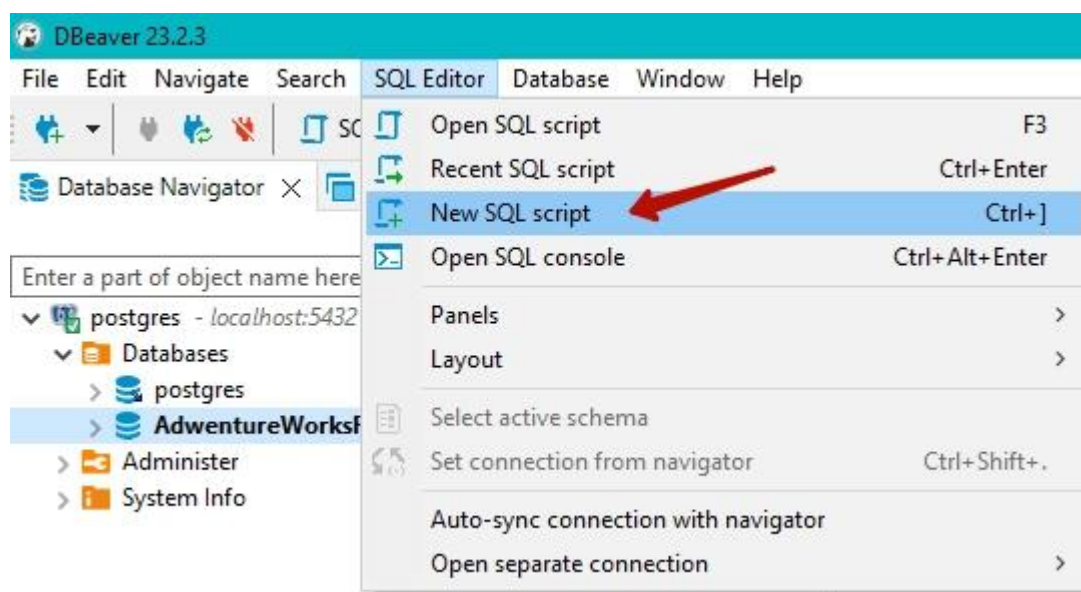


Рисунок 18 – Опция выбора нового скрипта

В результате выполнения запроса будет выведено «1» (Рисунок 19) в колонке без названия.

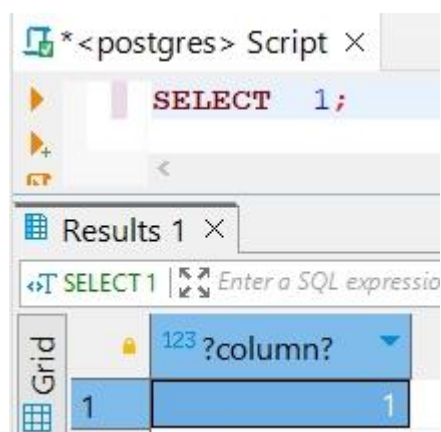


Рисунок 19 – Результат выполнения запроса

Следующим оператором является FROM (из), после которого следует название источника (таблицы), поля которой были указаны в предложении SELECT. Запрос заканчивается оператором «точка с запятой»:

```

SELECT
    "TargetTable"."Field1"
    , "TargetTable"."Field2"
    , ...
    , "TargetTable"."FieldN"
FROM "TargetTable";

```

Для сокращения записи можно использовать вместо имени таблицы её псевдоним, имя которого может быть любым соответствующим правилу именования полей, т. е. начинаться с буквы и не иметь в своем составе других символов, кроме букв и цифр:

```

SELECT
    T."Field1"
    , T."Field2"
    , ...
    , T."FieldN"
FROM "TargetTable" AS1 T;

```

Так как все поля принадлежат только таблице TargetTable, то в предложении SELECT имя таблицы или псевдоним можно не указывать:

```

SELECT
    "Field1"
    , "Field2"
    , ...
    , "FieldN"
FROM "TargetTable";

```

Если требуется выбирать все имеющиеся в таблице поля, то используется предикат астериск («звездочка»):

```

SELECT *
FROM "TargetTable";

```

¹ Предикат AS можно опустить

Пример 1

Показать все поля из таблицы `Person.AddressType`.

Ниже (Рисунок 20) показаны все поля таблицы `Person.AddressType`.

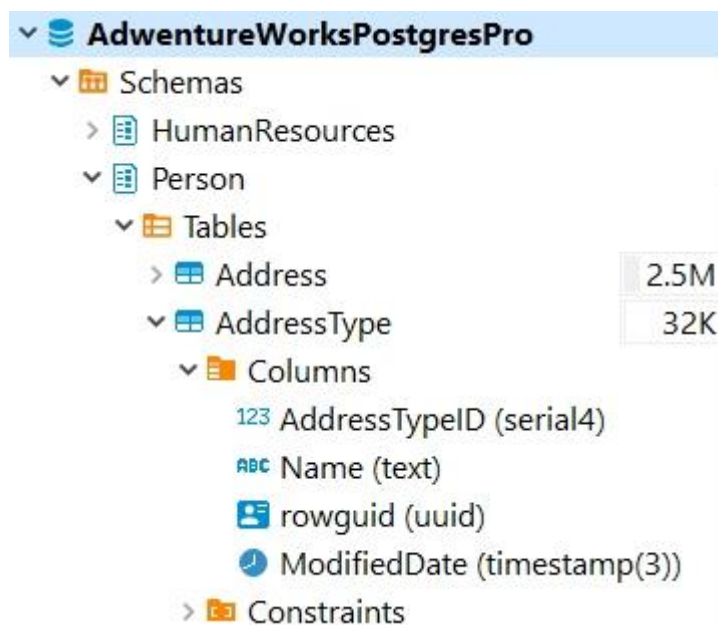


Рисунок 20 – Структура таблицы `Person.AddressType`

Напишем скрипт запроса:

```
SELECT *  
FROM "Person"."AddressType";
```

Выполните запрос, нажав на кнопку «Выполнить» (Execute SQL query). Результат выполнения данного запроса представлен ниже (Рисунок 21). Как видно, на рисунке отображены все имеющиеся у таблицы поля.

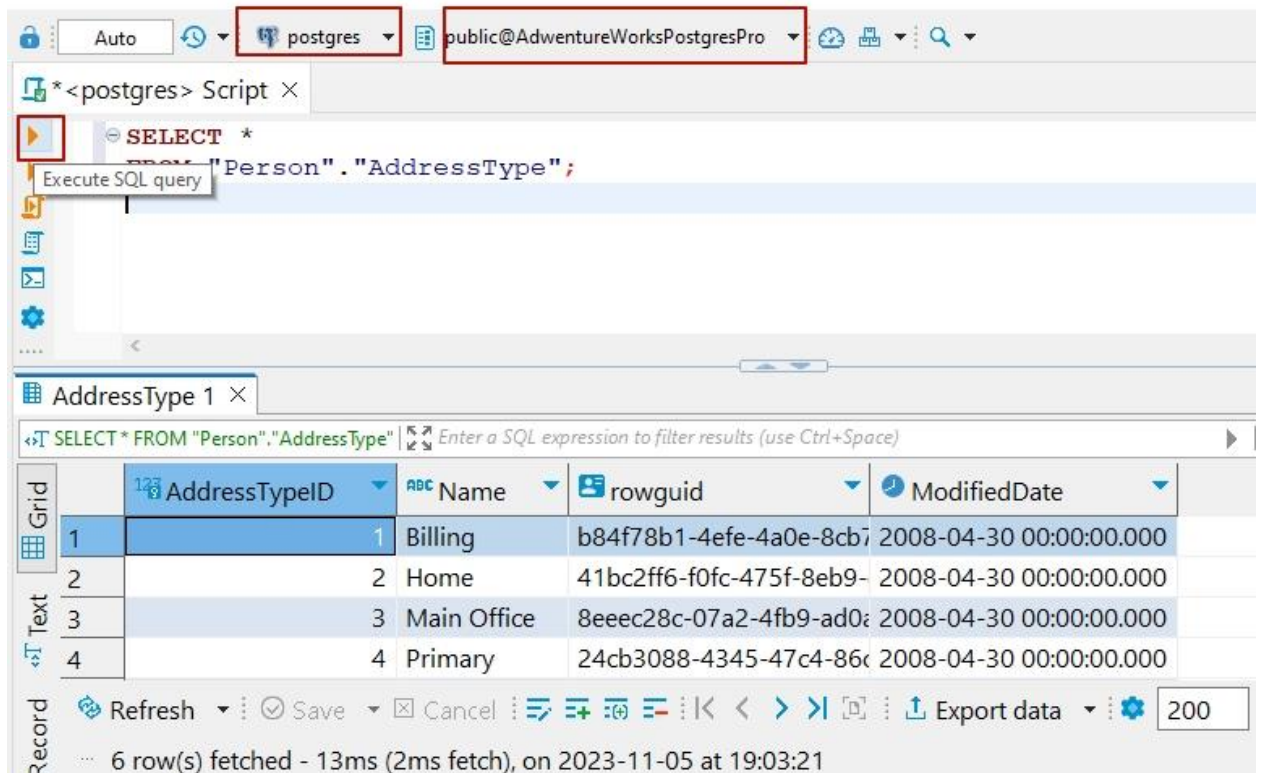


Рисунок 21 – Результат выполнения запроса (Пример 1)

Пример 2

Из таблицы *Production.Product* БД *AdventureWorksPostgresPro* показать наименование продукта, его цвет и размер (выбрать значения полей *Name*, *Color* и *Size*).

Создайте новый запрос. Выберите БД, к которой собираетесь обращаться (Рисунок 21). Наберите текст запроса:

```
SELECT
    "Name"
    , "Color"
    , "Size"
FROM "Production"."Product";
```

Результат выполнения запроса представлен ниже (Рисунок 22).

Product 1 X			
SELECT "Name", "Color", "Size" FROM Enter a SQL expression to filter r			
Grid		ABC Name ▼	ABC Color ▼
		ABC Size ▼	
	283	Mountain-100 Black, 4	Black 48
	284	Mountain-200 Silver, 3	Silver 38
Text	285	Mountain-200 Silver, 4	Silver 42
	286	Mountain-200 Silver, 4	Silver 46

Рисунок 22 – Результат выполнения запроса (Пример 2)

Фильтрация данных

Для фильтрации данных из выбранных полей используется оператор WHERE (где), после которого следует множество предикатов. Все записи из таблицы, значения полей которых при подстановке в предикат, дают значение «истина», попадут в результирующий набор данных.

```
SELECT
    "Field1"
    , "Field2"
    , ...
    , "FieldN"
FROM "TargetTable"
WHERE <Условие 1> AND (OR) <Условие 2> ...;
```

Условий может быть несколько, сочетаться они могут при помощи различных комбинаций булевых операторов AND, OR, NOT² (И, ИЛИ, НЕ, соответственно).

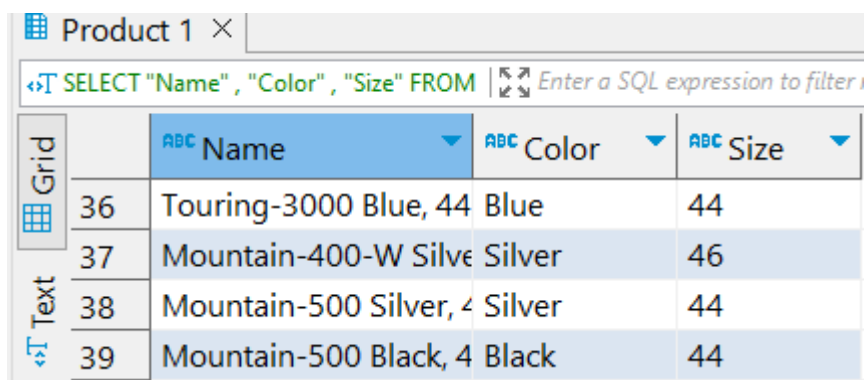
² Старайтесь строить запрос, не используя условие отрицания, так как на его отработку уходит больше времени. Если нельзя переписать условие на положительный исход, то для записи условия можно использовать как союз NOT, так и знак «<>» или сочетание « != ».

Пример 3

Выбрать из таблицы *Production.Product* товары, размер которых 44 или 46 (размер имеет символьный тип).

```
SELECT
    "Name"
  , "Color"
  , "Size"
FROM "Production"."Product"
WHERE ("Size" = '44' OR "Size" = '46');
```

Результат выполнения запроса представлен ниже (Рисунок 23). Как видно, размер имеет только значение 44 или 46.



	ABC Name	ABC Color	ABC Size
36	Touring-3000 Blue, 44	Blue	44
37	Mountain-400-W Silver	Silver	46
38	Mountain-500 Silver, 4	Silver	44
39	Mountain-500 Black, 4	Black	44

Рисунок 23 – Результат выполнения запроса (Пример 3)

Дополнительных Условий может быть несколько, сочетаться они могут при помощи различных комбинаций булевых операторов AND, OR, NOT (И, ИЛИ, НЕ, соответственно).

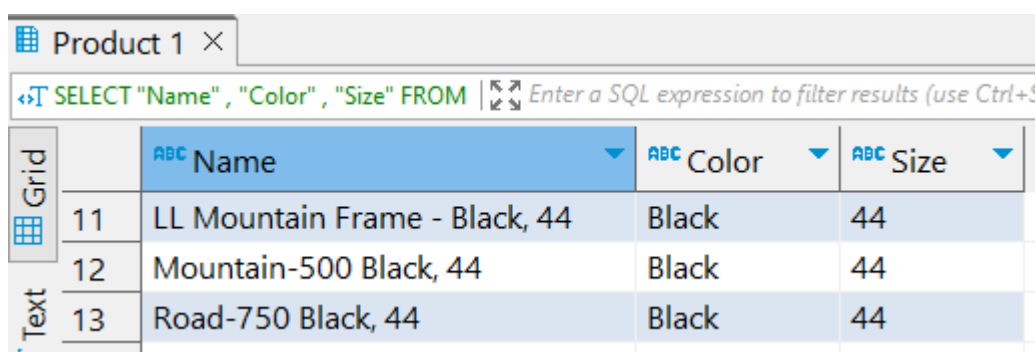
Пример 3 ещё одним условием.

Пример 4

Выбрать из таблицы *Production.Product* товары чёрного цвета, размер которых 44 или 46 (размер имеет символьный тип).

```
SELECT
    "Name"
  , "Color"
  , "Size"
FROM "Production"."Product"
WHERE ("Size" = '44' OR "Size" = '46')
        AND "Color" = 'Black';
```

Результат запроса представлен ниже (Рисунок 24).



	ABC Name	ABC Color	ABC Size
11	LL Mountain Frame - Black, 44	Black	44
12	Mountain-500 Black, 44	Black	44
13	Road-750 Black, 44	Black	44

Рисунок 24 – Результат выполнения запроса (Пример 4)

Как видно (Рисунок 24), в результирующем множестве представлены только товары чёрного цвета (Black).

Предикат LIKE

Для фильтрации строковых данных на полное совпадение можно использовать знак равенства. Однако обычно для строковых переменных поиск по полному совпадению является частным случаем. Чаще используется частичное совпадение. Для таких целей применяется предикат LIKE. Формат данного предиката такой:

```

SELECT
    "Fields1"
    , "Fields2"
    , ...
    , "FieldsN"
FROM "TableName"
WHERE "FieldK" LIKE 'example';
-- Последняя строка эквивалентна
-- WHERE "FieldK" = 'example';

```

Предикат LIKE может принимать в качестве аргумента не только строку, но и шаблон. Например, для частичного совпадения можно использовать управляющие последовательности (escape-sequences):

% (знак процента) – заменяет любое количество любых символов;
 _ (нижнее подчеркивание) – заменяет один символ;

Пример 5

Выбрать из таблицы Person.Person (полные) имена всех сотрудников, фамилия которых начинается на буквосочетание 'Smi'.

```

SELECT
    "LastName"
    , "FirstName"
    , "MiddleName"
FROM "Person"."Person"
WHERE "LastName" LIKE 'Smi%';

```

Как видно (Рисунок 25), в результирующий набор данных попали сотрудники с фамилией Smith и Smith-Bates.

Person 1 ×

SQL: SELECT "LastName", "FirstName", "MiddleName" FROM "Person"."Person" WHERE "LastName" LIKE '_____' ; --7 СИМВОЛОВ

	Grid	Text	Record	LastName	FirstName	MiddleName
	10			Smith	Samantha	[NULL]
	11			Smith-Bates	Lorrin	G.
	12			Smith	Mahesh	[NULL]
	13			Smith	Ben	[NULL]
	14			Smith	Frank	[NULL]
	15			Smith	Ben	[NULL]
	16			Smith-Bates	Lorrin	[NULL]
	17			Smith	Summer	K
	18			Smith	Adriana	[NULL]

Рисунок 25 – Результат применения предиката LIKE и управляющей последовательности «%»

Пример 6

Выбрать из таблицы Person.Person (полные) имена всех сотрудников, фамилия которых состоит из 7 символов.

```
SELECT
    "LastName"
    , "FirstName"
    , "MiddleName"
FROM "Person"."Person"
WHERE "LastName" LIKE '_____' ; --7 СИМВОЛОВ
«подчеркивание»;
```

Ниже (Рисунок 26) представлен результирующий набор данных. Фамилии сотрудников состоят только из 7 букв.

Person 1 ×

SELECT "LastName", "FirstName", "MiddleName" | Enter a SQL expression to filter results (use Ctrl+F)

	ABC LastName	ABC FirstName	ABC MiddleName
10	Kearney	Bonnie	
11	Maxwell	Taylor	R
12	Tibbott	Diane	H
13	Masters	Steve	F
14	Philips	Carol	M
15	Creasey	Jack	T
16	Stadick	Betsy	A
17	Cetinok	Baris	F
18	Gubbels	Eric	[NULL]
19	Reedell	Anthony	C

**Рисунок 26 – Результат применения предиката LIKE
и управляющей последовательности «_»**

Вместо LIKE можно использовать ключевое слово ILIKE, чтобы поиск был регистр-независимым с учётом текущей языковой среды. Этот оператор не описан в стандарте SQL; это расширение PostgreSQL [26].

Регулярные выражения SIMILAR TO

Как и LIKE, условие SIMILAR TO истинно, только если шаблон соответствует всей строке; это отличается от условий с регулярными выражениями, в которых шаблон может соответствовать любой части строки. Также подобно LIKE, SIMILAR TO воспринимает символы _ и % как знаки подстановки, подменяющие любой один символ или любую подстроку, соответственно [26].

Регулярные выражения POSIX³

Регулярное выражение — это последовательность символов, представляющая собой краткое определение набора строк (регулярное множество). Строка считается соответствующей регулярному выражению, если она является членом регулярного множества, описываемого регулярным выражением. Как и для LIKE, символы шаблона непосредственно соответствуют символам строки, за исключением специальных символов языка регулярных выражений. При этом спецсимволы регулярных выражений отличается от спецсимволов LIKE. В отличие от шаблонов LIKE, регулярное выражение может совпадать с любой частью строки, если только оно не привязано явно к началу и/или концу строки [26].

Внимание

Хотя чаще всего поиск по регулярному выражению бывает очень быстрым, регулярные выражения бывают и настолько сложными, что их обработка может занять приличное время и объём памяти. Поэтому опасайтесь шаблонов регулярных выражений, поступающих из недоверенных источников. Если у вас нет другого выхода, рекомендуется ввести тайм-аут для операторов [26].

Поиск с LIKE гораздо проще, поэтому его безопаснее использовать с недоверенными источниками шаблонов поиска [26].

³ POSIX (англ. Portable Operating System Interface — переносимый интерфейс операционных систем) — набор стандартов, описывающих интерфейсы между операционной системой и прикладной программой (системный API), библиотеку языка C и набор приложений и их интерфейсов. Стандарт создан для обеспечения совместимости различных UNIX-подобных операционных систем и переносимости прикладных программ на уровне исходного кода, но может быть использован и для не-Unix систем.

Пример 7

Выбрать из таблицы `Person.Person` (полные) имена всех сотрудников, фамилия которых начинается на один из символов «А», «С», «Е», «I», «Z».

Решение 1. Применение регулярного выражения `SIMILAR TO`:

```
SELECT
    "LastName"
  , "FirstName"
  , "MiddleName"
FROM "Person"."Person"
WHERE "LastName" SIMILAR TO '[A, C, E, I, Z] %';
-- тот же результат:
--WHERE "LastName" SIMILAR TO '(A|C|E|I|Z) %';
```

Решение 2. Применение регулярного выражения `POSIX`:

```
SELECT
    "LastName"
  , "FirstName"
  , "MiddleName"
FROM "Person"."Person"
WHERE "LastName" ~ '^ (A|C|E|I|Z) %';
```

Оператор «~» ищет совпадение с шаблоном с учетом регистра символов. Символ «^» в начале регулярного выражения означает, что поиск совпадения будет привязан к началу строки. Если же требуется проверить наличие такого символа в составе строки, то перед ним нужно поставить символ обратной косой черты «\». Выражение в круглых скобках означает альтернативный выбор между значениями, разделяемыми символом «|». Поэтому в выборку попадут значения, начинающиеся либо на «А», либо на «С», либо на «Е» и т. д. [27].

Результаты выполнения данных запросов совпадают и представлены ниже (Рисунок 27). Как видно из рисунка, после фамилии, начинающейся на символ «А», следует фамилия, которая начинается на символ «С».

SQL Editor: `SELECT "LastName", "FirstName", "MiddleName" FROM "Person"."Person"` Enter a SQL expression to filter results (use Ctrl+F)

	Grid	ABC LastName	ABC FirstName	ABC MiddleName
	910	Avalos	Robert	A.
	911	Ayers	Stephen	M.
	912	Cabello	Albert	S
	913	Cai	Curtis	[NULL]

Рисунок 27 – Результат выполнения запроса (Пример 7)

Пример 8

*Выбрать из таблицы `Person.Person` (полные) имена всех сотрудников, фамилия которых **не** начинается на перечисленные символы «A», «C», «E», «I», «Z».*

Решение 1. Применение регулярного выражения `SIMILAR TO`:

```
SELECT
    "LastName"
    , "FirstName"
    , "MiddleName"
FROM "Person"."Person"
WHERE "LastName" SIMILAR TO '^[^A, ^C, ^E, ^I, ^Z]%';
-- тот же результат:
--WHERE "LastName" NOT SIMILAR TO '(A|C|E|I|Z)%';
```

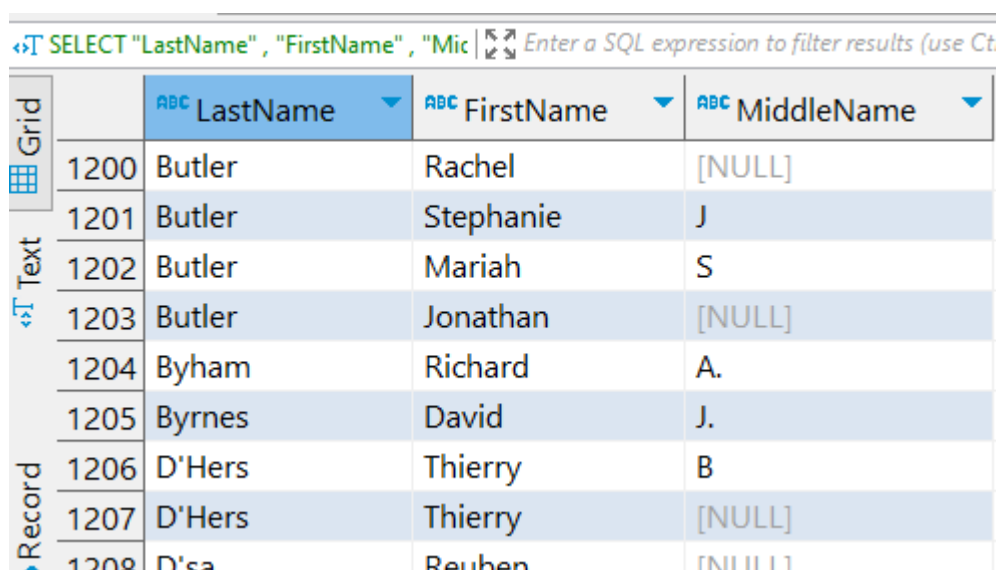
Решение 2. Применение регулярного выражения `POSIX`:

```
SELECT
    "LastName"
    , "FirstName"
    , "MiddleName"
FROM "Person"."Person"
WHERE "LastName" !~ '^(A|C|E|I|Z)';
```

Для инвертирования смысла оператора «~» нужно передним добавить знак«!». В регулярном выражении символ «\$» означает привязку поискового шаблона к концу строки. Если же требуется проверить наличие такого символа

в составе строки, то передним нужно поставить символ обратной косой черты [27]. Использование регулярных выражений подробно рассматривается в разделе документации 9.7.3 «Регулярные выражения POSIX» [26].

Результат выполнения скриптов представлен ниже (Рисунок 28). Из рисунка видно, что в результирующем наборе представлены фамилии, которые начинаются на символ «B», а затем на символ «D». Фамилии, которые начинаются на символ «E» пропущены.



	LastName	FirstName	MiddleName
1200	Butler	Rachel	[NULL]
1201	Butler	Stephanie	J
1202	Butler	Mariah	S
1203	Butler	Jonathan	[NULL]
1204	Byham	Richard	A.
1205	Byrnes	David	J.
1206	D'Hers	Thierry	B
1207	D'Hers	Thierry	[NULL]
1208	D'sa	Reuben	[NULL]

Рисунок 28 – Результат выполнения запроса (Пример 8)

Предикат IN

Если необходимо выбрать записи, значения которых принадлежат некоторому множеству, то запрос выглядит следующим образом [20]:

```
SELECT
    "Field1"
    , ...
FROM "TargetTable"
WHERE "Field1" = <Значение1>
    OR "Field1" = <Значение2>
    OR ...
    OR "Field1" = <ЗначениеN>;
```

последнее предложение можно заменить, используя предикат IN:

```
WHERE "Field1" IN (<Значение1>, <Значение2>, ..., <ЗначениеN>);
```

Пример 9

В таблице Production.Product найти товары, идентификационные номера которых равны 1, 316 или 329.

Решение 1:

```
SELECT
    "ProductID"
    , "Name"
    , "Color"
    , "Size"
FROM "Production"."Product"
WHERE "ProductID" = 1 OR "ProductID" = 316 OR
"ProductID" = 329;
```

Решение 2. Использование предиката IN:

```
SELECT
    "ProductID"
    , "Name"
    , "Color"
    , "Size"
FROM "Production"."Product"
WHERE "ProductID" IN (1, 316, 329);
```

При выполнении и решения 1, и решения 2, получим одинаковые результаты (Рисунок 29).

SQL Query: `SELECT "ProductID", "Name", "Color",` | Enter a SQL expression to filter results (use Ctrl+Sp)

Grid	ProductID	Name	Color	Size
1	1	Adjustable Race	[NULL]	[NULL]
2	316	Blade	[NULL]	[NULL]
3	329	Road End Caps	[NULL]	[NULL]

Рисунок 29 – Результат выполнения запроса (Пример 9)

Пример 10

Показать код и название валюты (*CurrencyCode* и *Name*) из таблицы *Sales.Currency*. Показать только валюты из списка ('Kroon', 'Armenian Dram', 'US Dollar').

```
SELECT
    "CurrencyCode"
    , "Name"
FROM "Sales"."Currency"
WHERE "Name" IN ('Kroon', 'Armenian Dram',
                'US Dollar');
```

Результат выполнения запроса представлен ниже (Рисунок 30).

SQL Query: `SELECT "CurrencyCode", "Name" FROM` | Enter a SQL expres:

Grid	CurrencyCode	Name
1	AMD	Armenian Dram
2	EEK	Kroon
3	USD	US Dollar

Рисунок 30 – Результат выполнения скрипта (Пример 10)

Предикат BETWEEN

Если требуется выбрать записи, входящие в некоторый диапазон значений, то запрос выглядит следующим образом [22]:

```

SELECT "Field1"
      , ...
FROM "TargetTable"
WHERE "Field1" >= <Значение1>
      AND "Field1" <= <Значение2>;

```

последнее предложение можно заменить, используя предикат BETWEEN:

```

WHERE "Field1" BETWEEN <Значение1> AND <Значение2>;

```

Пример 11

В таблице Production.Product найти товары, дата продажи которых находится в диапазоне от 2002-06-01 до 2007-06-01.

Решение 1:

```

SELECT
    "ProductID"
  , "Name"
  , "Color"
  , "Size"
  , "SellEndDate"
FROM "Production"."Product"
WHERE "SellEndDate" >= '2012-05-01'
      AND "SellEndDate" <= '2013-04-01';

```

Решение 2. Использование предиката BETWEEN:

```

SELECT
    "ProductID"
  , "Name"
  , "Color"
  , "Size"
  , "SellEndDate"
FROM "Production"."Product"
WHERE "SellEndDate" BETWEEN '2012-05-01' AND '2013-04-01';

```

Результат выполнения скриптов приведен ниже (Рисунок 31).

SELECT "ProductID", "Name", "Color", | Enter a SQL expression to filter results (use Ctrl+Space)

	ProductID	Name	Color	Size	SellEndDate
1	709	Mountain Bike Socks, M	White	M	2012-05-29 00:00:00.C
2	710	Mountain Bike Socks, L	White	L	2012-05-29 00:00:00.C
3	731	ML Road Frame - Red, 44	Red	44	2012-05-29 00:00:00.C
4	732	ML Road Frame - Red, 48	Red	48	2012-05-29 00:00:00.C
5	733	ML Road Frame - Red, 52	Red	52	2012-05-29 00:00:00.C
6	734	ML Road Frame - Red, 58	Red	58	2012-05-29 00:00:00.C

Рисунок 31 – Результат использования предиката BETWEEN

Преобразование типов данных

Неявное преобразование типов

SQL – язык со строгой типизацией. То есть каждый элемент данных в нём имеет некоторый тип, определяющий его поведение и допустимое использование. PostgreSQL наделён расширяемой системой типов, более универсальной и гибкой по сравнению с другими реализациями SQL [1].

Тип данных **целое число**, **число с плавающей точкой** и тип **дата** можно неявно преобразовать к символьному типу. В следующих примерах используем конкатенацию (оператор «||») двух значений: преобразуемого типа и символьного. Результаты выполнения запросов представлены ниже (Рисунок 32, Рисунок 33, Рисунок 34).

SELECT 1 || '2' AS text; -- целое число и символ

	text
1	12

**Рисунок 32 –Неявное преобразование типов
(целое число в строку)**

```
SELECT 0.2 || '0.2' AS text;
```

--число с плавающей точкой и символ

	ABC text
1	0.20.2

**Рисунок 33 – Неявное преобразование типов
(число с плавающей точкой в строку)**

```
SELECT 'Today is ' || now() AS "Today";--строка и дата
```

	ABC Today
1	Today is 2023-11-06 16:32:24.71628+03

**Рисунок 34 – Неявное преобразование типов
(тип дата в строку)**

Символьный тип данных также может быть неявно преобразован к числовому типу данных, если значение содержит допустимый для преобразуемого типа символы (только цифры для целого типа данных, и знак⁴ отделения целой части от дробной – для числа с плавающей точкой). Результаты выполнения примеров приведены ниже (Рисунок 35, Рисунок 36).

```
SELECT '1' + 2 AS INT;--символ в целое число
```

⁴ Точка или запятая. Зависит от языковой настройки среды выполнения.

123	int
1	3

**Рисунок 35 – Неявное преобразование типов
(строка в целое число)**

```
SELECT '0.2' + 0.2 AS float;
```

--СИМВОЛ В ЧИСЛО С ПЛАВАЮЩЕЙ ТОЧКОЙ

123	float
1	0.4

**Рисунок 36– Неявное преобразование типов
(строка в число с плавающей точкой)**

Если значения из предыдущих примеров будут содержать нецифровые символы, то при попытке выполнить данный запрос будет выведено сообщение об ошибке преобразования.

Явное преобразование типов

Для явного преобразования типов используется функция CAST (). При невозможности преобразования будет выведено сообщение об ошибке.

CAST(преобразуемое выражение AS тип данных, в который нужно преобразовать);

```
SELECT CAST(now() AS DATE) AS Date;
```

	🔒 date
1	2023-11-06

**Рисунок 37 – Явное преобразование типов
(тип дата-время в дата)**

```
SELECT CAST ('1.2345' AS NUMERIC (6, 4) ) AS float;
```

	🔒 123 float
1	1.2345

**Рисунок 38 – Явное преобразование типов
(строка в число с плавающей точкой)**

NUMERIC (*n*, *m*) – число с плавающей точкой, на которое отводится *n* символов, из них *m* знаков после запятой, на символ отделения целой части от дробной место не нужно.

Если будет указано меньшее количество знаков после запятой, чем было в числе, то число будет усечено до указанного количества знаков (Рисунок 39).

```
SELECT CAST ('1.2345' AS NUMERIC (5, 3) ) AS float;
```

	🔒 123 float
1	1.235

**Рисунок 39 – Явное преобразование типа с усечением
до трех знаков после запятой**

При ошибочном указании формата типа будет выведена ошибка (Рисунок 40).

```
SELECT CAST ('1.2345' AS NUMERIC (4, 4) ) AS float;
```


SELECT CAST('1.2345' AS NUMERIC(4,4)) Enter a SQL expression to filter results (use Ctrl+Space)



SQL Error [22003]: ОШИБКА: переполнение поля numeric

Detail: Поле с точностью 4, порядком 4 должно округляться до абсолютного значения меньше чем 1.

**Рисунок 40 – Ошибка при попытке преобразовать
в число с плавающей точкой с одинаковым количеством символов
на число и количеством знаков после запятой**

Работа с NULL значениями

СУБД поддерживают значение пусто (NULL) для полей, значение которых неизвестно (UNKNOWN). К таким значениям применяется трехзначная логика (Таблица 1 – Таблица 3) [14], [20], [22].

Таблица 1

AND (A, B)		A		
		TRUE	FALSE	UNKNOWN
B	TRUE	TRUE	FALSE	UNKNOWN
	FALSE	FALSE	FALSE	UNKNOWN
	UNKNOWN	UNKNOWN	UNKNOWN	UNKNOWN

Таблица 2

OR (A, B)		A		
		TRUE	FALSE	UNKNOWN
B	TRUE	TRUE	TRUE	TRUE
	FALSE	TRUE	FALSE	UNKNOWN
	UNKNOWN	TRUE	UNKNOWN	UNKNOWN

Таблица 3

NOT (A)	A		
	TRUE	FALSE	UNKNOWN
	FALSE	TRUE	UNKNOWN

При необходимости показывать только известные значения полей в предложении WHERE применяется предикат IS NOT NULL (и, наоборот, для выборки всех неизвестных значений, применяется предикат IS NULL).

```
SELECT
    "Field1"
    , "Field2"
    , ...
FROM "TargetTable"
WHERE Field1 IS NOT NULL;
```

Пример 12

Выбрать наименование продукта и его цену, для которого известно значение в линейке продуктов (ProductLine):

```
SELECT
    "Name"
    , "ListPrice"
    , "ProductLine"
FROM "Production"."Product"
WHERE "ProductLine" IS NOT NULL;
```

Результат выполнения скрипта представлен ниже (Рисунок 41).

	ABC Name	123 ListPrice	ABC ProductLine
1	HL Road Frame - Black, 58	1,431.5	R
2	HL Road Frame - Red, 58	1,431.5	R
3	Sport-100 Helmet, Red	34.99	S
4	Sport-100 Helmet, Black	34.99	S
5	Mountain Bike Socks, M	9.5	M
6	Mountain Bike Socks, L	9.5	M
7	Sport-100 Helmet, Blue	34.99	S
8	AWC Logo Cap	8.99	S
9	Long-Sleeve Logo Jersey, S	49.99	S
10	Long-Sleeve Logo Jersey, M	49.99	S

Рисунок 41 – Фильтрация пустых значений в поле ProductLine

Для работы с NULL значениями в PostgreSQL предусмотрены несколько функций.

```
NULLIF(выражение1, выражение2)
```

Функция NULLIF() используется для сравнения двух выражений. В случае их равенства функция возвращает значение NULL, иначе – значение первого выражения.

Пример 13

Показать наименование продукта, его цену и значение в линейке продуктов (ProductLine). Показать только те значения, которые известны, а значения, которые равны «R» заменить на NULL.

```
SELECT
    "Name"
  , "ListPrice"
  , "ProductLine"
  , NULLIF("ProductLine", 'R') AS "NullIfR"
FROM "Production"."Product"
WHERE "ProductLine" IS NOT NULL;
```

Результат выполнения данного скрипта представлен ниже (Рисунок 42). Из результирующего набора видно, что данные в поле NullIfR содержат значение NULL (замененные значения «R» поля ProductLine).

	ABC Name	123 ListPrice	ABC ProductLine	ABC NullIfR
1	HL Road Frame - Black, 58	1,431.5	R	[NULL]
2	HL Road Frame - Red, 58	1,431.5	R	[NULL]
3	Sport-100 Helmet, Red	34.99	S	S
4	Sport-100 Helmet, Black	34.99	S	S
5	Mountain Bike Socks, M	9.5	M	M
6	Mountain Bike Socks, L	9.5	M	M
7	Sport-100 Helmet, Blue	34.99	S	S

Рисунок 42 – Результат выполнения функции NULLIF ()

COALESCE (выражение1, выражение2, ..., выражение по умолчанию)

Функция COALESCE () применяется в том случае, когда необходимо показать первое известное значение. Сначала проверяется выражение1, если его значение неизвестно, то проверяется значение2, и так далее. Значение по умолчанию будет выведено в том случае, если все значения в предыдущих выражениях были неизвестны. **Все аргументы функции должны быть одного типа.** Если это не так, то необходимо применить функции преобразования типов.

Пример 14

Показать наименование продукта, его цену и одно из известных значений из набора цвет или размер продукта. В случае если ни один из параметров продукта неизвестен, вывести значение «неизвестно» (UNKNOWN) в отдельное поле с названием «измерение» (Measurement).
Показать товар с ценой, большей нуля.

```
SELECT
    "Name"
  , "ListPrice"
  , COALESCE("Color", "Size", 'UNKNOWN') AS
      Meauserement
FROM "Production"."Product"
WHERE "ListPrice" > 0;
```

Результат выполнения текста запроса представлен ниже (Рисунок 43).
Из результирующего набора видно, что те значения, которые содержат NULL в полях заменены на UNKNOWN.

	ABC Name	123 ListPrice	ABC meauserement
7	LL Touring Seat Assembly	133.34	UNKNOWN
8	ML Touring Seat Assembly	147.14	UNKNOWN
9	HL Touring Seat Assembly	196.92	UNKNOWN
10	HL Road Frame - Black, 58	1,431.5	Black
11	HL Road Frame - Red, 58	1,431.5	Red
12	Sport-100 Helmet, Red	34.99	Red
13	Sport-100 Helmet, Black	34.99	Black

Рисунок 43 – Результат выполнения запроса (Пример 14)

Пример 15

Показать наименование продукта, его цену и одно из известных значений из набора цвет, или размер, **или вес продукта**. В случае если ни один из параметров продукта неизвестен, вывести значение «позвонить» (CALL). Показать товар с ценой, большей нуля.

В отличие от предыдущего примера (Пример 14), необходимо значения в поле Weight привести к символьному типу данных. Это можно сделать с помощью функции `CAST("Weight" AS VARCHAR(10))`.

```
SELECT
    "Name"
  , "ListPrice"
  , COALESCE (
        "Color"
      , "Size"
      , CAST ("Weight" AS VARCHAR (10) )
      , 'CALL') AS Meauserement
FROM "Production"."Product"
WHERE "ListPrice" > 0;
```

Результат выполнения данного скрипта представлен ниже (Рисунок 44).

	ABC Name	123 ListPrice	ABC meauserement
8	ML Touring Seat Assembly	147.14	CALL
9	HL Touring Seat Assembly	196.92	CALL
10	HL Road Frame - Black, 58	1,431.5	Black
11	HL Road Frame - Red, 58	1,431.5	Red
12	Sport-100 Helmet, Red	34.99	Red
13	Sport-100 Helmet, Black	34.99	Black
14	Mountain Bike Socks, M	9.5	White
15	Mountain Bike Socks, L	9.5	White

Рисунок 44 – Результат выполнения функции `COALESCE ()`

Пример 16

Из таблицы *Production.Document* показать поля *DocumentNode*, *Title* и *FileExtension*. Все пустые значения поля *FileExtension* заменить на *NULL*, а затем все *NULL* этого же поля заменить на *' .txt '*. Названия полей оставить без изменений.

```
SELECT
    "DocumentNode"
    , "Title"
    , COALESCE (NULLIF ("FileExtension", ''), '.txt')
FileExtension
FROM "Production"."Document";
```

Результат выполнения данного скрипта представлен ниже (Рисунок 45).

	DocumentNode	Title	fileextension
1	1	Documents	.txt
2	2	Overview	.txt
3	3	Introduction 1	.doc
4	7	Lubrication Maintenance	.doc
5	8	Assembly	.txt
6	9	Front Reflector Bracket and Reflector Assembly 3	.doc
7	10	Front Reflector Bracket Installation	.doc

Рисунок 45 – Результат выполнения запроса (Пример 16)

Предложение **DISTINCT**

Для выборки множества значений применяется предложение **DISTINCT**.

Сравним результаты выполнения двух запросов:

Показать значения столбца *Field1* из таблицы *TargetTable* и показать уникальные значения столбца *Field1* из таблицы *TargetTable*.

Таблица 4

<pre>SELECT "Field1" FROM "TargetTable";</pre>	<pre>SELECT DISTINCT "Field1" FROM "TargetTable";</pre>
Результат выполнения запроса – все записи	Результат выполнения запроса – множество значений

Пример 17

Показать значения столбца цвет.

```
SELECT "Color"
FROM "Production"."Product";
```

В результате выполнения данного запроса выведено 504 строки (**все строки**) (Рисунок 46).

Grid	Color	
3	[NULL]	
4	[NULL]	
5	[NULL]	
6	Black	
7	Black	
8	Black	
9	Silver	
10	Silver	

Refr... Sa...
... 504 row(s) fetched - 1

Рисунок 46 – Все записи. Результат выполнения запроса

Пример 18

Показать все имеющиеся цвета товара.

```
SELECT DISTINCT "Color"  
FROM "Production"."Product";
```

В результате показано 10 строк (Рисунок 47).



	Color
3	Yellow
4	Red
5	Grey
6	Multi
7	White
8	Black
9	Blue
10	Silver/Black

Рисунок 47 – Результат применения предложения DISTINCT

Сортировка результирующего набора данных

Если необходимо результат выполнения запроса отсортировать по какому-либо полю, то применяется предложение ORDER BY. По умолчанию (ASC), сортировка происходит по возрастанию значений в указанном поле.

```
SELECT  
    "Field1"  
    , "Field2"  
    , ...  
    , "FieldN"  
FROM "TargetTable"  
ORDER BY "Field1";
```

В случае обратного порядка применяется указатель DESC.

```
SELECT
    "Field1"
    , "Field2"
    , ...
    , "FieldN"
FROM "TargetTable"
ORDER BY Field1 DESC;
```

Не всегда известно имя поля (например, в случае применения одной из вышеперечисленных функций работы с NULL значениями, столбец не имеет имени). В этом случае можно не указывать имя поля, а указать его номер в операторе SELECT:

```
SELECT
    "Field1"
    , "Field2"
    , ...
    , "FieldN"
FROM "TargetTable"
ORDER BY 1;
```

Пример 19

Показать наименование товара, его цену и цвет. Отсортировать по возрастанию цены (показать только товар, со значением цены, отличной от нуля).

```
SELECT
    "Name"
    , "ListPrice"
    , "Color"
FROM "Production"."Product"
WHERE "ListPrice" > 0
ORDER BY "ListPrice";
```

	ABC Name ▼	123 ListPrice ▼	ABC Color ▼
1	Patch Kit/8 Patches	2.29	[NULL]
2	Road Tire Tube	3.99	[NULL]
3	Water Bottle - 30 oz.	4.99	[NULL]
4	Touring Tire Tube	4.99	[NULL]
5	Mountain Tire Tube	4.99	[NULL]
6	Bike Wash - Dissolver	7.95	[NULL]
7	Racing Socks, M	8.99	White
8	Racing Socks, L	8.99	White

Рисунок 48 – Результат запроса с применением предиката сортировки

Пример 20

Показать наименование товара, его цену и цвет. Отсортировать по убыванию цены (показать только товар, со значением цены, отличной от нуля).

```

SELECT
    "Name"
    , "ListPrice"
    , "Color"
FROM "Production"."Product"
WHERE "ListPrice" > 0
ORDER BY "ListPrice" DESC;

```

Результат выполнения скрипта представлен ниже (Рисунок 49).

	ABC Name ▼	123 ListPrice ▼	ABC Color ▼
1	Road-150 Red, 62	3,578.27	Red
2	Road-150 Red, 44	3,578.27	Red
3	Road-150 Red, 56	3,578.27	Red
4	Road-150 Red, 52	3,578.27	Red
5	Road-150 Red, 48	3,578.27	Red
6	Mountain-100 Silver, 3	3,399.99	Silver
7	Mountain-100 Silver, 4	3,399.99	Silver
8	Mountain-100 Silver, 4	3,399.99	Silver

Рисунок 49 – Результат выполнения запроса (Пример 20)

Функция SUBSTRING

Функция SUBSTRING (выражение, начальная позиция, количество символов) применяется, когда необходимо отфильтровать данные по некоторому шаблону [14], [20], [22].

Пример 21

Найти товар с ценой, отличной от нуля, который начинается на «Рас». Показать его наименование, цену и цвет.

```

SELECT
    "Name"
  , "ListPrice"
  , "Color"
FROM "Production"."Product"
WHERE "ListPrice" > 0
      AND SUBSTRING ("Name", 1, 3) = 'Рас'
ORDER BY "ListPrice" DESC;

```

Результат выполнения скрипта показан ниже (Рисунок 50).

ABC Name ▼	123 ListPrice ▼	ABC Color ▼
Racing Socks, M	8.99	White
Racing Socks, L	8.99	White

Рисунок 50 – Применение предиката SUBSTRING ()

Такого же результата можно добиться, применяя известный ранее поиск по шаблону, с использованием предиката LIKE:

```
SELECT
    "Name"
  , "ListPrice"
  , "Color"
FROM "Production"."Product"
WHERE "ListPrice" > 0 AND "Name" LIKE 'Rac%'
ORDER BY "ListPrice" DESC;
```

Результат выполнения данного скрипта идентичен результату, представленному выше (Рисунок 50), однако производительность запроса с оператором SUBTRING () выше.

РЕКОМЕНДАЦИИ ПО ВЫПОЛНЕНИЮ РАБОТЫ:

Создание нового локального репозитория

Данная лабораторная работа является первой в цикле работ по курсу, связанному с базами данных. Настоятельно рекомендуется создать новый локальный репозиторий для лабораторных работ [15]. Каждую лабораторную работу следует делать в отдельной папке.

Создание нового репозитория

В веб-хостинге GitHub создать новый внешний репозиторий назвать его DataBaseCourse, добавить описание к репозиторию, в котором

обязательно указать цель создания репозитория, фамилию, имя и отчество владельца, группу обучающегося (Рисунок 51) [20].

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Required fields are marked with an asterisk (*).

Owner *	Repository name *
 	<input type="text" value="DataBaseCourse"/>
 DataBaseCourse is available.	

Great repository names are short and memorable. Need inspiration? How about [symmetrical-tribble](#) ?

Description (optional)

Репозиторий для работ по дисциплине...

- ☐  **Public**
Anyone on the internet can see this repository. You choose who can commit.
- ☒  **Private**
You choose who can see and commit to this repository.

Рисунок 51 – Новый репозиторий для выполнения работ по курсу

Создайте файл `.gitignore` (Рисунок 52), игнорирующий временные файлы MS Office (для отчета), временные и служебный файлы SQL (Visual Studio), служебные файлы операционной системы (Windows) и добавьте его в главную ветку (master).

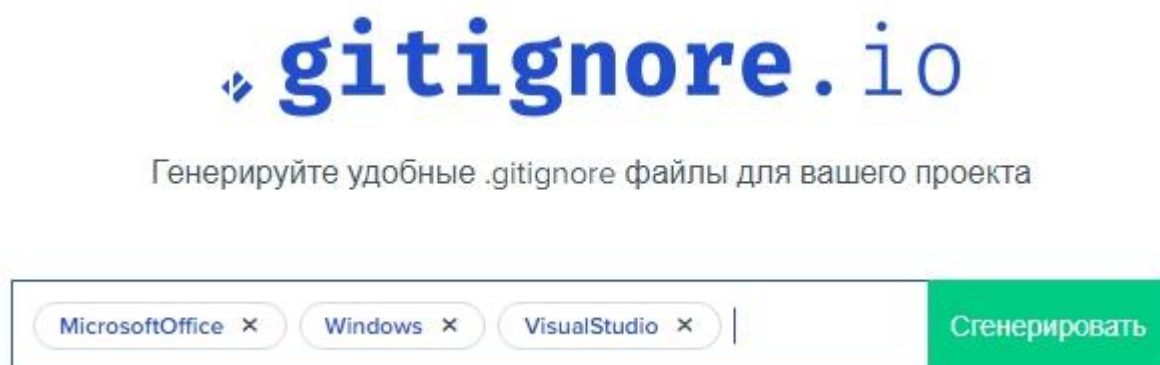


Рисунок 52 – Генерация файла `.gitignore`

В GitHub создайте новый `issue`, в котором запишите задание «Лабораторная работа №1» (Рисунок 53) и назначьте его выполнение себе. В тело `issue` вставьте задание, соответствующее вашему варианту.

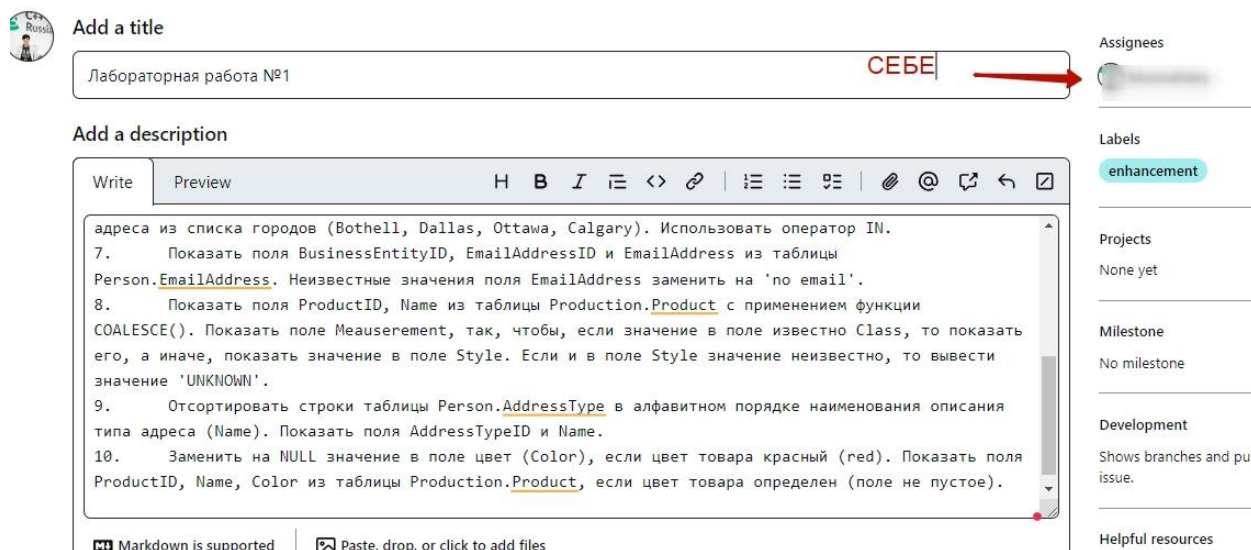


Рисунок 53 – Создание нового `issue`

Для выполнения текущего `issue` на локальном компьютере создайте новую ветку.

Выполнение лабораторной работы

Каждое задание для лабораторной работы необходимо выполнять в отдельном скрипте. Наименование скрипта – Task N. В начале файла добавьте комментарий с формулировкой текущего задания (Рисунок 54).

```
--Отсортировать строки таблицы Person.AddressType
--в алфавитном порядке наименования описания типа адреса (Name).
--Показать поля AddressTypeID и Name.
SELECT
    "Name"
    , "AddressTypeID"
FROM "Person"."AddressType"
ORDER BY "Name" DESC;
```

Рисунок 54 – Пример оформления текущего задания

После того, как все задания лабораторной работы выполнены, создайте новый pull request, добавьте рецензента (reviewer). В качестве рецензента, принимающего решение, должен быть преподаватель по данному курсу.

Дождитесь проверки лабораторной работы преподавателем. Если преподаватель вынес решение requested changes (Рисунок 55), откройте задание, которое требует изменения, исправьте ошибки и сохраните файл.

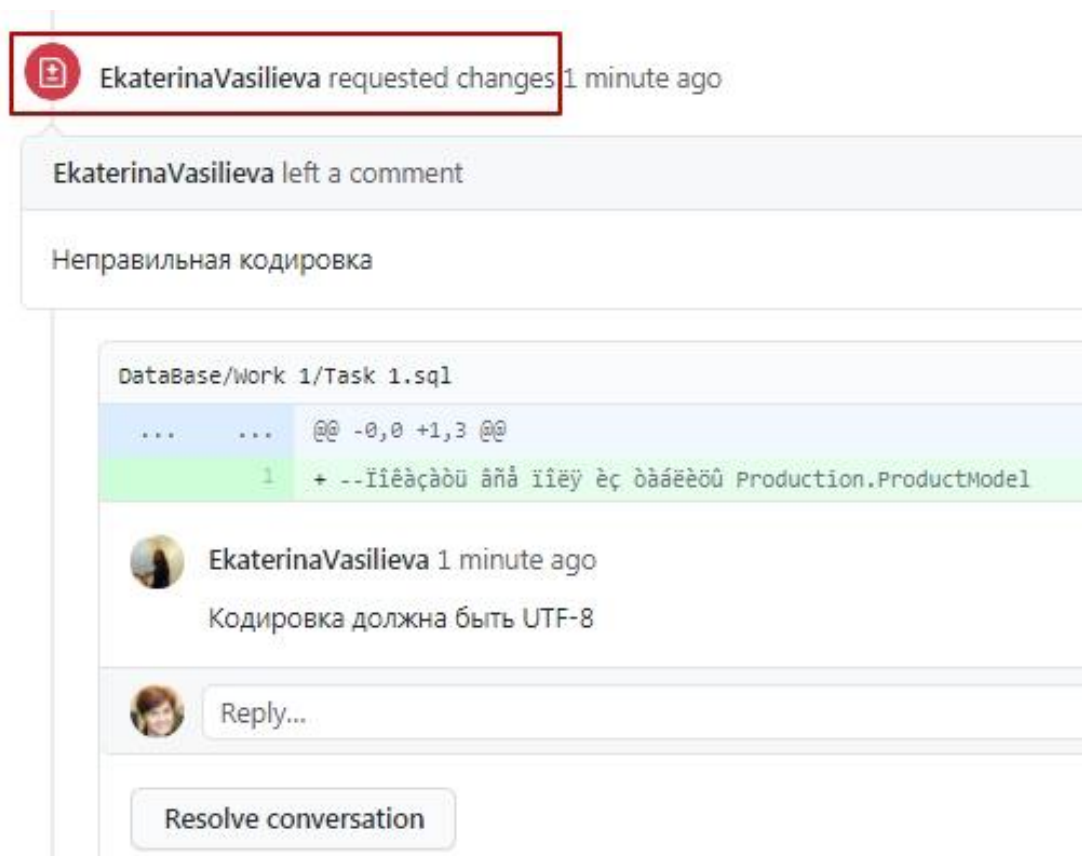


Рисунок 55 – Ответ преподавателя `requested changes`

Для исправления ошибок НЕ НУЖНО создавать новых веток, новых файлов, удалять текущие файлы или сохранять исправляемый скрипт под новым именем!

Рекомендации по исправлению кодировки в имеющемся файле

В данном примере файлы были сохранены не в кодировке UTF-8, поэтому задание, написанное кириллицей, не читается. Для исправления такой ситуации откройте файлы в Notepad++ (Рисунок 56). Выберите нужную кодировку (Рисунок 57). Сохраните файл (Рисунок 58).

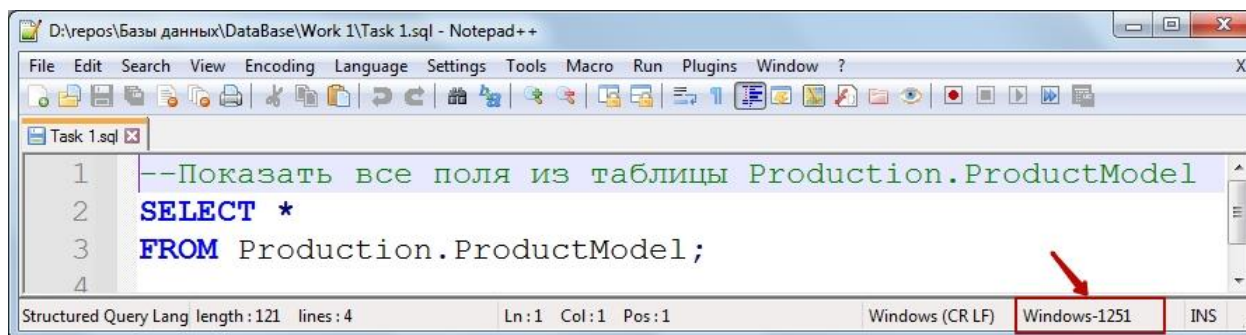


Рисунок 56 – Исправление неправильной кодировки

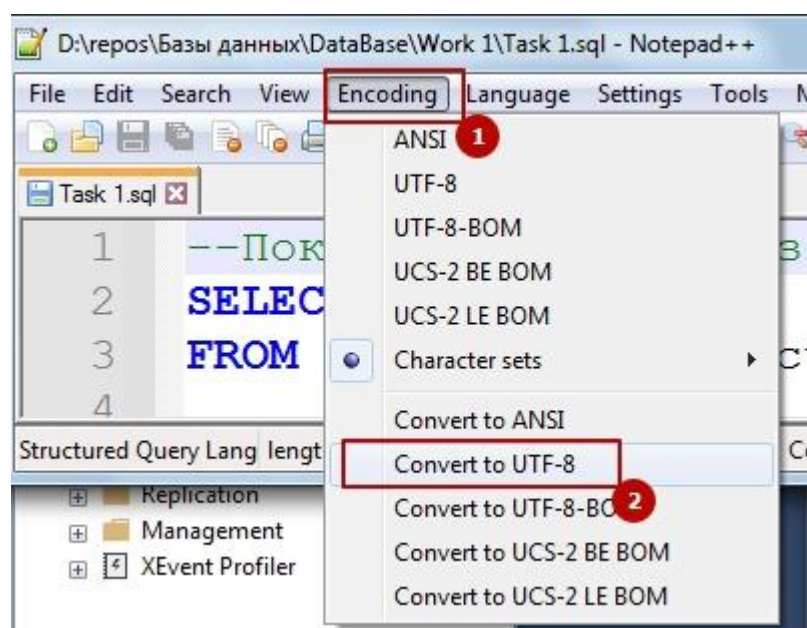


Рисунок 57 – Конвертация кодировки в UTF-8

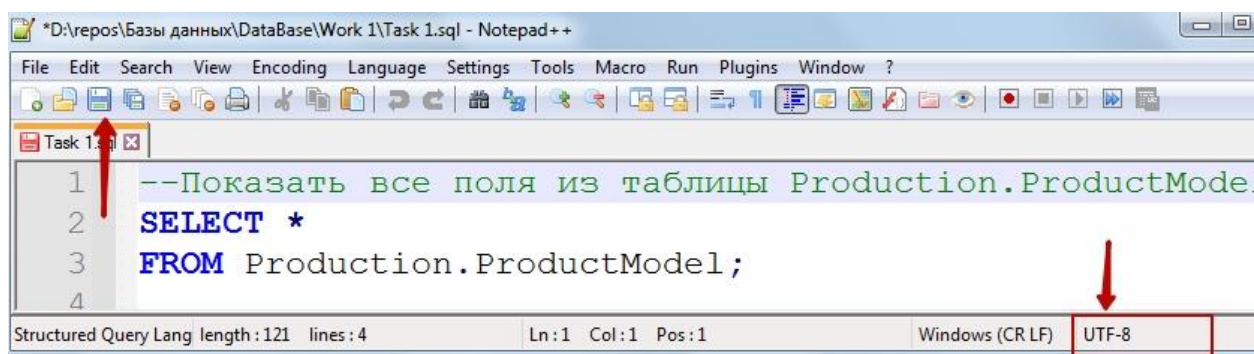


Рисунок 58 – Сохранение текущего файла

Порядок действий при исправлении ошибок

После исправления **всех** ошибок, указанных преподавателем, сохраните (push) работу (Рисунок 59) и **явно перезапросите** рецензию, нажав на соответствующую кнопку (голубые стрелки) в pull request (Рисунок 60), иначе преподаватель никак не узнает, о вашем желании проверить работу [20], [15].

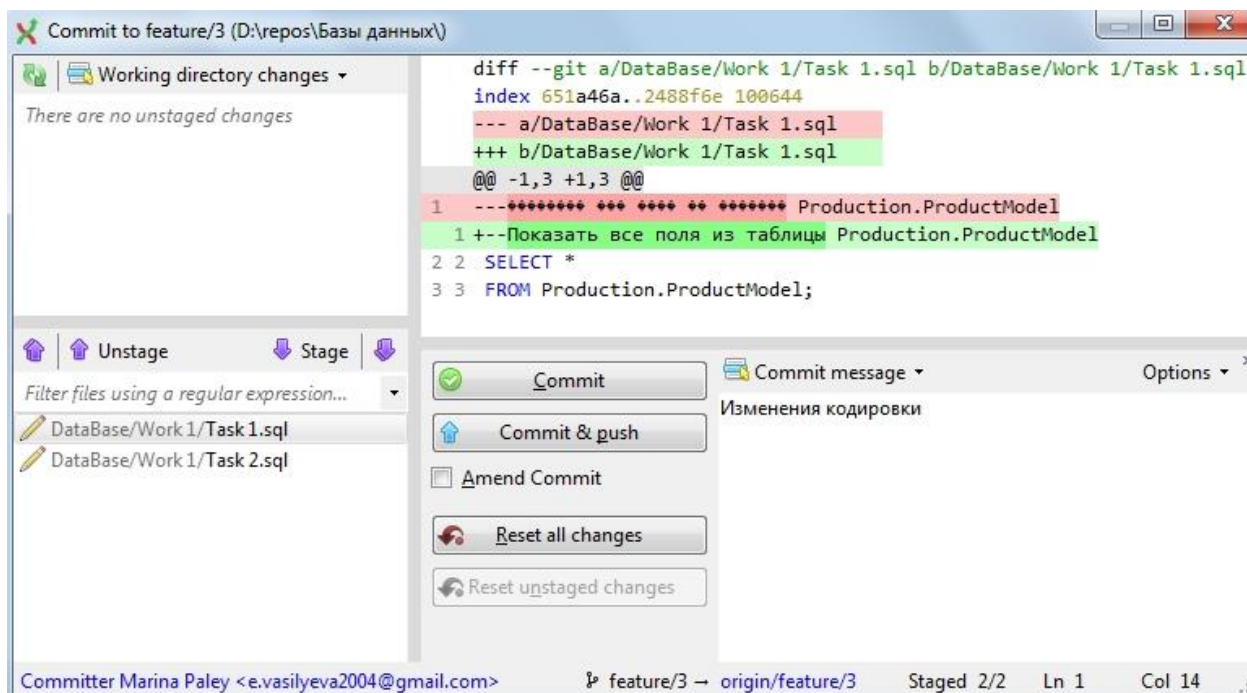


Рисунок 59 – Commit&push измененных файлов



Рисунок 60 – Перезапрос на проверку от преподавателя

Все задания по текущей лабораторной работе выполняйте в одной ветке.

Дождитесь утверждения (approve) работы преподавателем, затем слейте (merge) изменения с главной веткой.

Последовательность действий при разработке отчета

Работу над отчетом нужно начинать после утверждения всех заданий по текущей лабораторной работе. Будьте при этом аккуратны и внимательны, так как **работа над отчетом должна выполняться в отдельной ветке**, а сам отчет так же, как и выполненные задания, должен быть **утвержден преподавателем** (поставлен approve в GitHub).

Заведите новый issue в GitHub. На локальном компьютере создайте новую ветку для работы над созданным issue.

Титульный лист [2] представлен в приложении (ПРИЛОЖЕНИЕ А). Текст отчета должен быть набран шрифтом Times New Roman размером не ниже 12 пунктов (пт) с междустрочным интервалом 1.5 пт с отступом для красной строки (1,25 см) и выравниванием по ширине [2] (Рисунок 61).

Для того, чтобы выделить имена таблиц, столбцов или других надписей на английском языке, применяется моноширинный шрифт, например, Courier New. Все заголовки должны быть выделены специальным стилем,

например, **Заголовок 2** (Heading 2), основанным на том же шрифте (Times New Roman) черного цвета (Рисунок 61).

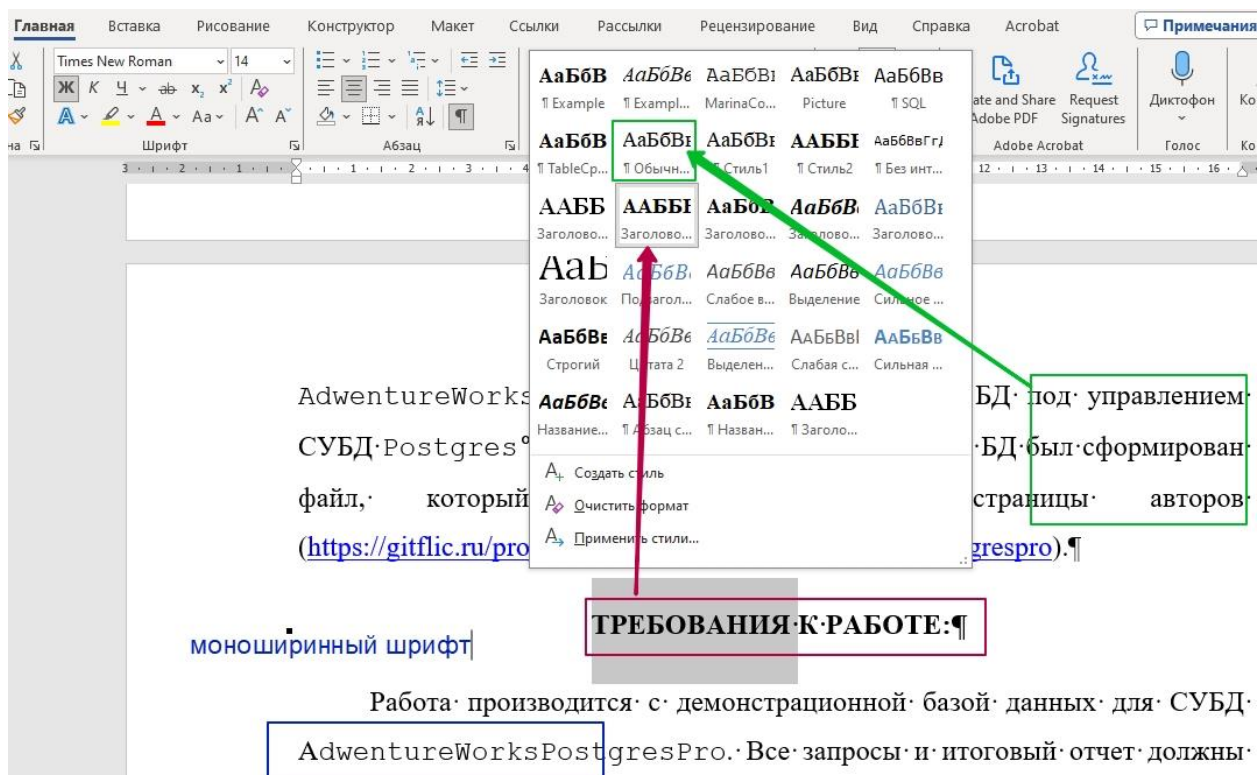


Рисунок 61 – Выбор стиля для форматирования документа

Текста скрипта должен быть вставлен моноширинным шрифтом, каким был написан в среде DBeaver (Consolas), размер можно увеличить до 12 пт, но междустрочный интервал следует оставить одинарным. Для отделения скрипта от основного текста, до и после скрипта добавьте по 14 пт (Рисунок 62).

Для того, чтобы параграф «не рвался», используйте «мягкий перенос» (Рисунок 62). Для этого используйте сочетание клавиш Enter+Shift [28].

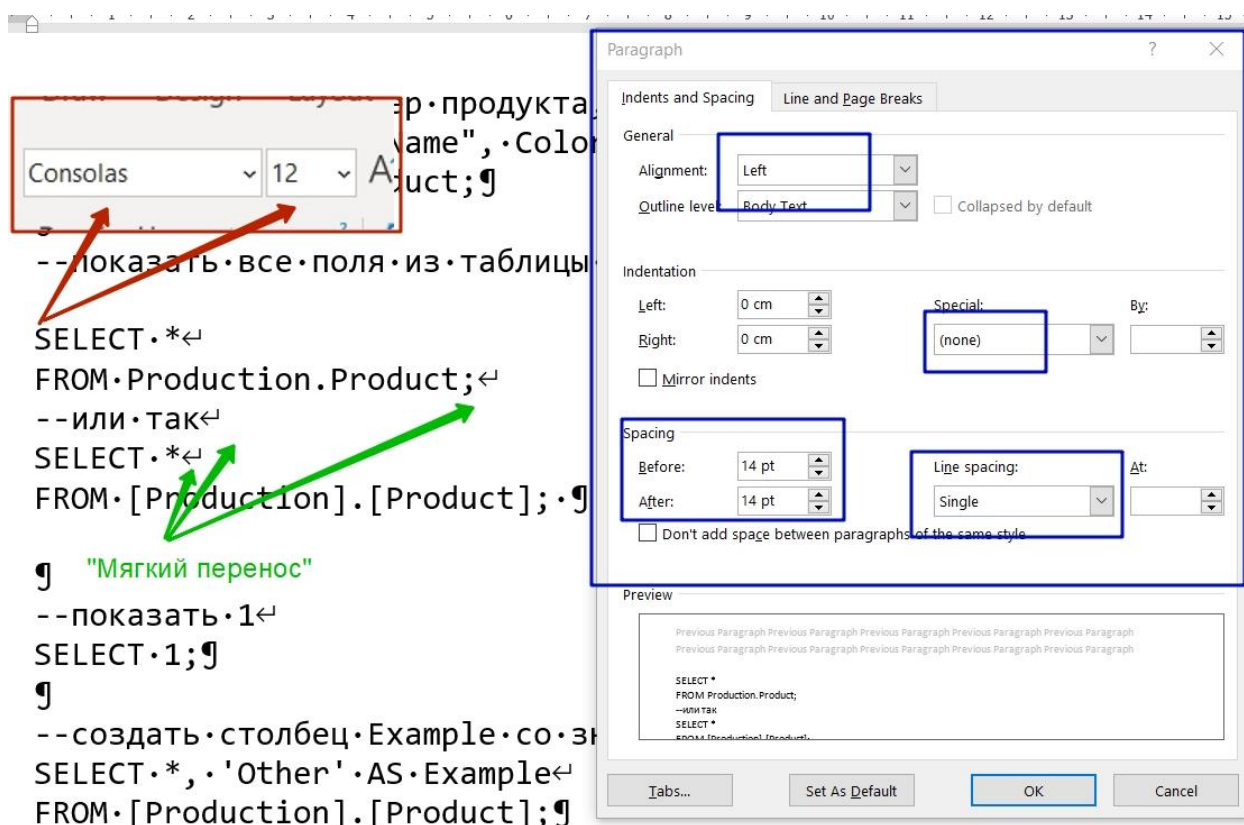
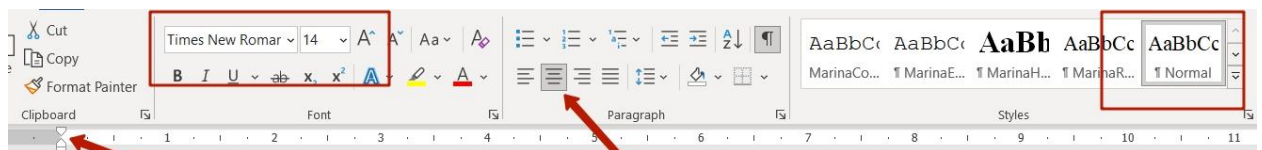


Рисунок 62 – Работа с текстом скрипта

Структура отчета должна быть следующая: цель работы, формулировка задания, текст скрипта, рисунок с результатом выполнения данного скрипта. В конце отчета приложите снимок экрана, на котором отчетливо виден approve и автор репозитория, которому данный approve поставлен.

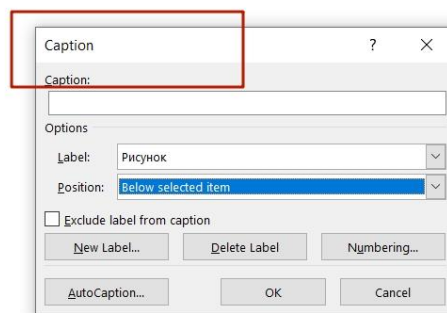
После получения approve в GitHub по данной работе, запустите каждый скрипт лабораторной работы и сделайте снимок экрана, который максимально поможет вам при защите данной работы. Все рисунки, вставленные в отчет, должны хорошо читаться, иметь подпись, расположенную под рисунком [2]. Рисунок и подпись (caption) должны располагаться по центру без отступа красной строки [2] (Рисунок 63). Подпись рисунка нужно вставить, выделив рисунок и нажав правую кнопку мыши. Подпись (Label) «Рисунок» необходимо написать целиком без сокращений [2]. Номер рисунка и заголовок (caption) рисунка нужно отделять тире (En Dash – Ctrl+Num- [28]), обрамляя неразрывными

пробелами (Nonbreaking Space – Ctrl+Shift+Space [28]). В конце подписи точка не ставится. На все рисунки должны быть ссылки в тексте [2].

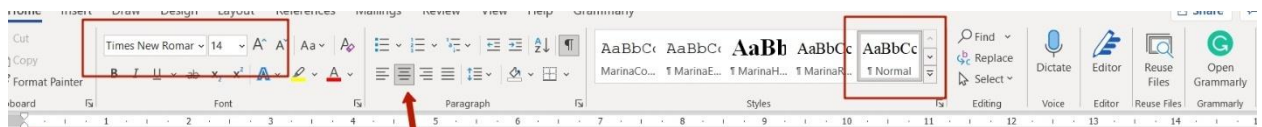


--создать·столбец·Example·со·значениями·"Other"¶

```
SELECT·*,·'Other'·AS·Example←
FROM·[Production].[Product];¶
```



Результаты	
AvgWeight	
1	30.127361



--создать·столбец·Example·со·значениями·"Other"¶

```
SELECT·*,·'Other'·AS·Example←
FROM·[Production].[Product];¶
```

Результаты	
AvgWeight	
1	30.127361

Рисунок¹⁰–Результат·выполнения·скрипта·для·Задания¹⁰¶

Рисунок 63 – Расположение рисунка в отчете

Работа с редактором Word представлена на странице (<https://gist.github.com/MarinaPaley/1f83a9a15ae0a36f740eefe573a86f46>).

После того, как отчет выполнен, отправьте (push) все изменения в удаленный (remote) репозиторий и создайте pull request. Дождитесь утверждения (approve) отчета.

Если в отчете имеются ошибки, исправьте их, работая именно с данным документом, не удаляя или пересохраняя его под другими именами!

После утверждения отчета влейте (merge) рабочую ветку (feature/...) в главную (master), удалите рабочую ветку (feature/...). В локальном репозитории также обновите (pull) главную ветку (master), рабочую ветку (feature/...), созданную для выполнения отчета, удалите.

Пример отчета представлен на странице авторов (<https://gist.github.com/MarinaPaley/54786bf18e96d0441632aec2aa391902>).

ЗАДАНИЕ НА ЛАБОРАТОРНУЮ РАБОТУ

ВАРИАНТ 1

1. Показать все поля из таблицы `Production.Document`.
2. Показать поля `ProductID`, `DocumentNode` из таблицы `Production.ProductDocument`.
3. Показать поля `ProductID`, `BusinessEntityID`, `AverageLeadTime`, `StandardPrice` из таблицы `Purchasing.ProductVendor`, где средний промежуток времени (в днях) между размещением заказа у продавца и получением приобретенного продукта (`AverageLeadTime`) больше 16, а обычная отпускная цена продавца (`StandardPrice`) находится в диапазоне от \$50 до \$60. Использовать оператор `BETWEEN`.
4. Показать поля `PurchaseOrderID`, `RevisionNumber`, `Status`, `ShipDate` из таблицы `Purchasing.PurchaseOrderHeader`, для которых известна ориентировочная дата отгрузки от поставщика (`ShipDate`) и текущее состояние заказа (`Status`) "Ожидание" (= 1).
5. Показать стандартный код ISO для стран и регионов (`CountryRegionCode`) и название страны или региона (`Name`), имеющие отношение к Конго (Congo) из таблицы `Person.CountryRegion`.
6. Показать поля `AddressID`, `AddressLine1`, `City` из таблицы `Person.Address`. Показать только адреса из списка городов (Bothell, Dallas, Ottawa, Calgary). Использовать оператор `IN`.

7. Показать поля `BusinessEntityID`, `EmailAddressID` и `EmailAddress` из таблицы `Person.EmailAddress`. Неизвестные значения поля `EmailAddress` заменить на `'no email'`.
8. Показать поля `ProductID`, `Name` из таблицы `Production.Product` с применением функции `COALESCE()`. Показать поле `Meauserement`, так, чтобы, если значение в поле известно `Class`, то показать его, а иначе, показать значение в поле `Style`. Если и в поле `Style` значение неизвестно, то вывести значение `'UNKNOWN'`.
9. Отсортировать строки таблицы `Person.AddressType` в алфавитном порядке наименования описания типа адреса (`Name`). Показать поля `AddressTypeID` и `Name`.
10. Заменить на `NULL` значение в поле цвет (`Color`), если цвет товара красный (`red`). Показать поля `ProductID`, `Name`, `Color` из таблицы `Production.Product`, если цвет товара определен (поле не пустое).

ВАРИАНТ 2

1. Показать все поля из таблицы `Production.ProductInventory`.
2. Показать поля `ProductID`, `StartDate`, `EndDate` из таблицы `Production.ProductCostHistory`.
3. Показать поля `ProductID`, `BusinessEntityID`, `AverageLeadTime`, `StandardPrice` из таблицы `Purchasing.ProductVendor`, для товаров, имеющих цену продажи (поле не пустое) при последней покупке (`LastReceiptCost`).
4. Показать уникальные названия городов (`City`) из таблицы `Person.Address`, у которых известен (`"AddressLine2"`).
5. Показать `ID (ProductID)`, название (`Name`) и цвет (`Color`) товаров из таблицы `Production.Product`, содержащих в названии слово `'Chainring'` (звездочка).
6. Показать `ID (StateProvinceID)` и название (`Name`) регионов или штатов, а также `ID страны`, на территории которой они располагаются (`TerritoryID`) из таблицы `Person.StateProvince`. Показать только регионы из списка (`'Alaska'`, `'Alabama'`, `'Colorado'`, `'Georgia'`, `'Iowa'`), используя оператор `IN`.
7. Показать поля `PersonType`, `NameStyle`, `Title`, `FirstName`, `MiddleName`, `LastName` из таблицы `Person.Person`. Неизвестные значения поля `Title` заменить на `'Dear'` (уважаемый). Названия полей оставить без изменений.
8. Показать поля `ProductID`, `Name` из таблицы `Production.Product` с применением функции `COALESCE()`. Показать поле `Measurement`, так, чтобы, если значение в поле `Class` известно, то показать его, а

иначе, показать значение в поле `Color`. Если и в поле `Color` значение неизвестно, то вывести значение `'UNKNOWN'`.

9. Показать ID департамента (`DepartmentID`), название отдела (`GroupName`) и вид деятельности данного отдела (`Name`) из таблицы `HumanResources.Department` и отсортировать строки таблицы в алфавитном порядке вида деятельности (`Name`).
10. Показать название фирмы (`Name`) из таблицы `Purchasing.Vendor`. Если фирма называется `'Advanced Bicycles'`, заменить значение поля `Name` на `NULL`. Названия полей оставить без изменений.

ВАРИАНТ 3

1. Показать все поля из таблицы `Production.WorkOrder`.
2. Показать поля `ProductID`, `StartDate`, `EndDate`, `ListPrice` из таблицы `Production.ProductListPriceHistory`.
3. Показать информацию о товаре (`ProductID`, `StartDate`, `EndDate`, `StandardCost`) из таблицы `Production.ProductCostHistory`, где стандартная стоимость товара (`StandardCost`) варьируется от \$12 до \$17.
4. Из таблицы `Production.WorkOrder` показать поля `WorkOrderID`, `ProductID`, `OrderQty`, `StockedQty`, `ScrapReasonID`. Показать значения поля `ScrapReasonID`, не равные `NULL`.
5. Показать поля `Name`, `ShipBase`, `ShipRate` из таблицы `Purchasing.ShipMethod`. Из поля `Name` показать только те значения, которые содержат слово 'Fast' (Быстрый).
6. Из таблицы `Sales.SalesTerritory` показать название и код страны или её региона (`Name`, `CountryRegionCode`), и название части света (`Group`). Показать только города или регионы из списка ('Central', 'Canada', 'Germany', 'Australia', 'United Kingdom'). Использовать оператор `IN`.
7. Показать поля `DepartmentID`, `ShiftID`, `StartDate`, `EndDate` из таблицы `HumanResources.EmployeeDepartmentHistory`. Если дата окончания (`EndDate`) неизвестна, то показать текущую дату. Названия полей оставить без изменений.
8. Показать поля `ProductID`, `Name` из таблицы `Production.Product` с применением функции `COALESCE()`. Показать поле `Meauserement`,

так, чтобы, если значение в поле `Size` известно, то показать его, а иначе, показать значение в поле `Color`. Если и в поле `Color` значение неизвестно, то вывести значение `'UNKNOWN'`.

9. Из таблицы `Person.StateProvince` показать `ID` (`StateProvinceID`) и название (`Name`) регионов или штатов, а также `ID` страны, на территории которой они располагаются (`TerritoryID`). Отсортировать строки таблицы в обратном алфавитном порядке в соответствии с названием регионов или штатов (`Name`).
10. Показать поле `Name` из таблицы `Person.CountryRegion`. Заменить название региона (`Name`) на `NULL`, если регион называется `American Samoa`. Названия полей оставить без изменений.

ВАРИАНТ 4

1. Показать все поля из таблицы `Production.Location`.
2. Показать поля `ProductModelID`, `IllustrationID` из таблицы `Production.ProductModelIllustration`.
3. Показать поля `PurchaseOrderID`, `Status`, `SubTotal` из таблицы `Purchasing.PurchaseOrderHeader`. Показать только те записи, для которых итог по заказу на покупку (`SubTotal`) больше \$2000 и меньше \$3000. Использовать оператор `BETWEEN`.
4. Из таблицы `Purchasing.Vendor` показать название поставщика (`Name`), кредитный рейтинг (`CreditRating`) и страницу вебсайта (`PurchasingWebServiceURL`). Показать только те товары, у которых есть страница вебсайта.
5. Показать только те ID фотографии товара (`ProductPhotoID`) и названия файлов, содержащие миниатюру фотографии товара (`ThumbnailPhotoFileName`) из таблицы `Production.ProductPhoto`, у которых в поле `ThumbnailPhotoFileName` содержится значение «черный» ('black').
6. Показать поля `ProductSubcategoryID` и `Name` из таблицы `Production.ProductSubcategory`. Показать только названия товаров (`Name`) из списка ("Тормоза", "Кепки", "Гарнитуры", "Седла") ('Brakes', 'Caps', 'Headsets', 'Saddles'). Использовать оператор `IN`.
7. Из таблицы `Purchasing.ProductVendor` показать ID продукта (`ProductID`), ID сотрудника (`BusinessEntityID`), среднее время

выполнения заказа (AverageLeadTime) и количество имеющихся заказов (OnOrderQty) из таблицы Purchasing.ProductVendor. Несуществующие значения поля OnOrderQty заменить на 0. Названия полей оставить без изменений.

8. Показать поля ProductID, Name из таблицы Production.Product с применением функции COALESCE(). Показать поле Meauserement, так, чтобы, если значение в поле Class известно, то показать его, а иначе, показать значение в поле Size. Если и в поле Size значение неизвестно, то вывести значение 'UNKNOWN'.
9. Отсортировать значения полей BusinessEntityID, AccountNumber и Name из таблицы Purchasing.Vendor в обратном алфавитном порядке относительно поля Name.
10. Из таблицы Production.Product показать поля Name и Color. Показать только те товары, которые имеют цвет. Заменить на NULL значение в поле цвет (Color), если цвет товара красный (red). Названия полей оставить без изменений.

ВАРИАНТ 5

1. Показать все поля из таблицы `Production.WorkOrderRouting`.
2. Показать поля `ProductDescriptionID`, `[Description]` из таблицы `Production.ProductModelProductDescription`.
3. Показать поля `BusinessEntityID`, `RateChangeDate`, `Rate` из таблицы `HumanResources.EmployeePayHistory`. Показать только те записи, для которых почасовая ставка заработной платы (`Rate`) больше \$15 и меньше \$20. Использовать оператор `BETWEEN`.
4. Из таблицы `Sales.SalesOrderDetail` показать ID продажи (`SalesOrderID`) и ID товара (`ProductID`). Показать только те товары, у которых есть `CarrierTrackingNumber`.
5. Показать поля `ProductCategoryID` и `Name` из таблицы `Production.ProductCategory`. Показать только те товары, название которых начинается на букву 'C'.
6. Из таблицы `Sales.Store` показать ID сотрудника (`BusinessEntityID`), название магазина (`Name`) и ID продавца (`SalesPersonID`). Показать только значения поля `Name`, принадлежащие списку ('Next-Door Bike Store', 'Area Bike Accessories', 'Top of the Line Bikes', 'Valley Toy Store', 'Global Plaza'). Использовать оператор `IN`.
7. Показать поля `BusinessEntityID`, `AccountNumber`, `Name`, `PurchasingWebServiceURL` из таблицы `Purchasing.Vendor`. Известные адреса сайтов магазинов (`PurchasingWebServiceURL`) заменить значением 'not available'. Названия полей оставить без изменений.

8. Показать поля `ProductID`, `Name` из таблицы `Production.Product` с применением функции `COALESCE()`. Показать поле `Measurements`, так, чтобы, если значение в поле `Style` известно, то показать его, а иначе, показать значение в поле `Color`. Если и в поле `Color` значение неизвестно, то вывести значение `'UNKNOWN'`.
9. Из таблицы `Production.Culture` показать значения поля `Name` в алфавитном порядке.
10. Показать название (`Name`) и размер (`Size`) товара из таблицы `Production.Product`, если имеются товары только размера `'M'`. Заменить размер товара на `NULL`, если он имеет значение `'M'`. Названия поля `Size` заменить на `Class`.

ВАРИАНТ 6

1. Показать все поля из таблицы `Production.ScrapReason`.
2. Показать поля (`ProductPhotoID`, `ThumbnailPhotoFileName`, `LargePhotoFileName`) из таблицы `Production.ProductPhoto`.
3. Показать поля `TerritoryID`, `Name`, `Group`, `SalesYTD` из таблицы `Sales.SalesTerritory`. Показать только те записи, для которых продажи на территории с начала года (`SalesYTD`) больше \$3000000 и меньше \$5000000. Использовать оператор `BETWEEN`.
4. Из таблицы `Sales.Store` показать ID сотрудника (`BusinessEntityID`), название магазина (`Name`) и ID продавца (`SalesPersonID`). Показать только существующие ID продавцов.
5. Показать место документа в памяти (`DocumentNode`), заголовок документа (`Title`) и расширение файла (`FileExtension`) из таблицы `Production.Document`. Показать только файлы с расширением `'.doc'`.
6. Из таблицы `Production.ProductSubcategory` показать поля `ProductSubcategoryID`, `Name`. Показать только значения поля `Name` из списка «переключатели», «гарнитуры», «колеса» (`'Derailleurs'`, `'Headsets'`, `'Wheels'`). Использовать оператор `IN`.
7. Показать поля `BillOfMaterialsID`, `ProductAssemblyID`, `ComponentID`, `UnitMeasureCode` из таблицы `Production.BillOfMaterials`. Если значение поля `ProductAssemblyID` – `NULL`, заменить его на 0 и выделить в отдельное поле с названием `ProductAssemblyID_0`.

8. Показать поля `ProductID`, `Name` из таблицы `Production.Product` с применением функции `COALESCE()`. Показать поле `Measurement`, так, чтобы, если значение в поле `Color` известно, то показать его, а иначе, показать значение в поле `Style`. Если и в поле `Style` значение неизвестно, то вывести значение `'UNKNOWN'`.
9. Из таблицы `Production.Document` показать поля `DocumentNode`, `Title` и `FileExtension`. Отсортировать строки в алфавитном порядке в соответствии с полем `Title`.
10. Показать поля `BusinessEntityID`, `NationalIDNumber`, `LoginID`, `OrganizationNode` из таблицы `HumanResources.Employee`. Показать только существующие значения поля `OrganizationNode`. Если поле `OrganizationNode` имеет значение `0x5AE358`, то нужно обратить это значение в `NULL`. Названия полей оставить без изменений.

ВАРИАНТ 7

1. Показать все поля из таблицы `Production.ProductSubcategory`.
2. Показать поля `ProductReviewID`, `ProductID`, `ReviewerName` из таблицы `Production.ProductReview`.
3. + 4. Показать поля `ProductID`, `StartDate`, `EndDate`, `StandardCost` из таблицы `Production.ProductCostHistory`. Показать только те записи, для которых стандартная стоимость (`StandardCost`) больше \$150 и меньше \$200 и известна дата окончания стоимости продукта (`EndDate`). Использовать оператор `BETWEEN`.
5. Из таблицы `Production.ProductPhoto` показать ID фотографии товара (`ProductPhotoID`) и название файла с полноразмерной фотографией (`LargePhotoFileName`). Показать только те товары, миниатюрные фото (`ThumbnailPhotoFileName`) которых содержат в названии «зеленый» ('green').
6. Показать поля `BusinessEntityID`, `AccountNumber`, `Name` из таблицы `Purchasing.Vendor`. Показать только значения поля `Name` из списка ('Australia Bike Retailer', 'Chicago Rent-All', 'Light Speed', 'First National Sport Co.'). Использовать оператор `IN`.
7. Из таблицы `Production.Document` показать место документа в памяти (`DocumentNode`), заголовок документа (`Title`), расширение файла (`FileExtension`). Показать только строки со значением `NULL` в поле `DocumentSummary` и заменить все значения `NULL` в данном поле на 'No ' («нет»). Названия полей оставить без изменений.

8. Показать поля `ProductID`, `Name` из таблицы `Production.Product` с применением функции `COALESCE()`. Показать поле `Measurement`, так, чтобы, если значение в поле `Weight` известно, то показать его, а иначе, показать значение в поле `Style`. Если и в поле `Style` значение неизвестно, то вывести значение `'UNKNOWN'`.
9. Из таблицы `Production.WorkOrderRouting` показать уникальные значения поля `ProductID` и поле `LocationID`. Отсортировать все строки в алфавитном порядке относительно поля `LocationID`.
10. Из таблицы `Purchasing.Vendor` показать поля `BusinessEntityID`, `AccountNumber`, `Name` и `ActiveFlag`, если значение последнего поля равно 0. В поле `ActiveFlag` заменить все значения 0 на `NULL`, а название поля заменить на `'Vendor is actively used'`.

ВАРИАНТ 8

1. Показать все поля из таблицы `Production.BillOfMaterials`.
2. Показать поля `TransactionID`, `ProductID`, `ReferenceOrderID`, `TransactionType`, `Quantity`, `ActualCost` из таблицы `Production.TransactionHistory`.
3. + 4. Показать поля `ProductID`, `StartDate`, `EndDate`, `ListPrice` из таблицы `Production.ProductListPriceHistory`. Показать только те записи, для которых стоимость (`ListPrice`) больше \$50 и меньше \$60 и известна дата окончания стоимости продукта (`EndDate`). Использовать оператор `BETWEEN`.
5. Из таблицы `Production.ProductModel` показать ID модели товара (`ProductModelID`), наименование товара (`Name`) и дата модификации продукта `ModifiedDate`. Показать только те товары, у которых в названии содержится слово «перчатки» (`'Gloves'`).
6. Показать ID и наименование причины поломки товара (`ScrapReasonID` и `Name`), из таблицы `Production.ScrapReason`, где наименование причины поломки принадлежит списку «Неправильный цвет», «Отверстие слишком большое», «Отверстие слишком маленькое», «Процесс покраски не удался» (`'Color incorrect'`, `'Drill size too large'`, `'Drill size too small'`, `'Paint process failed'`). Использовать оператор `IN`.
7. + 10. Из таблицы `Production.Document` показать поля `DocumentNode`, `Title` и `FileExtension`. Все пустые значения поля

FileExtension заменить на NULL, а затем все NULL этого же поля заменить на '.txt'. Названия полей оставить без изменений.

8. Показать поля ProductID, Name из таблицы Production.Product с применением функции COALESCE(). Показать поле Measurement, так, чтобы, если значение в поле Color известно, то показать его, а иначе, показать значение в поле Weight. Если и в поле Weight значение неизвестно, то вывести значение 'UNKNOWN'.
9. Показать поля ProductID, BusinessEntityID, AverageLeadTime, StandardPrice из таблицы Purchasing.ProductVendor, используя вместо названия таблицы псевдоним 'p'. Отсортировать все строки в алфавитном порядке в соответствии с полем StandardPrice.

ВАРИАНТ 9

1. Показать все поля из таблицы `Production.ProductDescription`.
2. Показать поля `TransactionID`, `ProductID`, `TransactionType`, `Quantity`, `ActualCost` из таблицы `Production.TransactionHistoryArchive`.
3. Показать поля `ProductId`, `UnitPrice`, `LineTotal`, `ReceivedQty`, `RejectedQty` и `OrderQty` из таблицы `Purchasing.PurchaseOrderDetail`, где количество, фактически полученное от продавца (`ReceivedQty`) больше 60, а количество, отклоненное, во время проверки (`RejectedQty`), находится в диапазоне от 50 до 80.
4. Из таблицы `Production.Document` показать место документа в памяти (`DocumentNode`), уровень документа (`DocumentLevel`), заголовок документа (`Title`) и итоговый документ (`DocumentSummary`). Показать только те документы, которые не имеют итогового документа.
5. Показать место документа в памяти (`DocumentNode`), заголовок документа (`Title`) и расширение файла (`FileExtension`) из таблицы `Production.Document`, если значения поля `Title` содержат слово 'Assembly' («сборка»).
6. Показать поля `TransactionID`, `ProductID`, `ReferenceOrderID`, `TransactionType`, `Quantity`, `ActualCost` из таблицы `Production.TransactionHistory`. Показать только значения поля `TransactionType` из списка ('W', 'P'). Использовать оператор `IN`.

7. Из таблицы `HumanResources.Department` показать поля `DepartmentID`, `Name`, `GroupName`. Все значения поля `Name` равные `'Finance'` («финансы») заменить на `NULL`, а затем все `NULL` этого же поля заменить на `'Other'` («другое»). Получившиеся поле вынести отдельно и назвать `OtherName`.
8. Показать поля `ProductID`, `Name` из таблицы `Production.Product` с применением функции `COALESCE()`. Показать поле `Measurement`, так, чтобы, если значение в поле `Style` известно, то показать его, а иначе, показать значение в поле `Weight`. Если и в поле `Weight` значение неизвестно, то вывести значение `'UNKNOWN'`.
9. Показать поля `TransactionID`, `ProductID`, `ReferenceOrderID`, `TransactionType`, `Quantity`, `ActualCost` из таблицы `Production.TransactionHistory`, где значения поля `Quantity` больше 4. Отсортировать строки в алфавитном порядке относительно поля `ActualCost`.

ВАРИАНТ 10

1. Показать все поля из таблицы `Production.Product`.
2. Показать поля `FirstName`, `LastName` и `MiddleName` из `Person.Person`.
3. Показать поля `ProductId`, `UnitPrice` и `OrderQty` из таблицы `Purchasing.PurchaseOrderDetail`, где количество заказанного товара (`OrderQty`) больше 60, а цена (`UnitPrice`) находится в диапазоне от \$50 до \$100.
4. Показать идентификационный номер сотрудника (`BusinessEntityID`), номер аккаунта (`AccountNumber`), название поставщика (`Name`) и URL поставщика (`PurchasingWebServiceURL`) из таблицы `Purchasing.Vendor`, где у поставщика имеется URL.
5. Показать все названия товаров (`Name`), содержащие 'Frame' («рама») и ID моделей товаров (`ProductModelID`) из таблицы `Production.ProductModel`.
6. Показать сотовый и рабочий телефонный номер (`PhoneNumber`), идентификационный номер сотрудника (`BusinessEntityID`) и вид телефонного номера (`PhoneNumberTypeID`) из таблицы `Person.PersonPhone`, где вид телефонного номера или 1 (сотовый), или 3 (рабочий). Использовать оператор `IN`. Отсортировать строки в алфавитном порядке относительно поля `BusinessEntityID`.

7. Показать поля AddressID, AddressLine1, AddressLine2, City из таблицы Person.[Address]. Заменить неизвестное значение в поле AddressLine2 на 'NO'. Названия полей оставить без изменений.
8. Показать поля ProductID, Name из таблицы Production.Product с применением функции COALESCE(). Показать поле Meauserement, так, чтобы, если значение в поле Size известно, то показать его, а иначе, показать значение в поле Color. Если и в поле Color значение неизвестно, то вывести значение 'UNKNOWN'.
9. Отсортировать строки таблицы Production.ScrapReason в обратном алфавитном порядке значения брака Name. Показать поля ScrapReasonID, Name, ModifiedDate.
10. Заменить на NULL значение в поле ListPrice, если цена равна нулю. Показать поля ProductID, Name, Size, Color из таблицы Production.Product.

ВАРИАНТ 11

1. Показать все поля из таблицы `Production.Illustration`.
2. Показать поля `UnitMeasureCode`, `Name` из таблицы `Production.UnitMeasure`.
3. Показать поля `TransactionID`, `ProductID`, `ReferenceOrderID`, `TransactionType`, `Quantity`, `ActualCost` из таблицы `Production.TransactionHistory`, где значения поля `Quantity` больше 4, а значения поля `ActualCost` находятся в диапазоне от 100 до 150.
4. Показать поля `ProductID`, `BusinessEntityID`, `AverageLeadTime`, `StandardPrice`, `LastReceiptDate` из таблицы `Purchasing.ProductVendor`, для товаров, имеющих дату последнего получения продукта продавцом (`LastReceiptDate`).
5. Показать идентификационный номер и причину производственного отказа (`ScrapReasonID` и `Name`) из таблицы `Production.ScrapReason`, где причина производственного отказа связана со сверлением ('Drill').
6. Показать поля `ProductID`, `Name`, `ProductSubcategoryID` из таблицы `Production.Product`, где идентификационный номер подкатегории (`ProductSubcategoryID`) равен или 1, или 2, или 3, или 4, или 5. Использовать оператор `IN`.
7. Показать поля `PurchaseOrderID`, `RevisionNumber`, `Status`, `ShipDate` из таблицы `Purchasing.PurchaseOrderHeader`. Если текущее состояние заказа (`Status`) "Ожидание" (= 1) и ориентировочная дата отгрузки от поставщика (`ShipDate`) неизвестна,

то заменить значение поля ShipDate на текущую дату. Названия полей оставить без изменений.

8. Показать поля ProductID, Name из таблицы Production.Product с применением функции COALESCE(). Показать поле Measurement, так, чтобы, если значение в поле Style известно, то показать его, а иначе, показать значение в поле Weight. Если и в поле Weight значение неизвестно, то вывести значение 'UNKNOWN'.
9. Из таблицы Purchasing.PurchaseOrderDetail показать идентификационный номер товара (ProductID), цену за 1 единицу товара (UnitPrice) и количество заказов данного товара (OrderQty). Отсортировать строки по возрастанию цены в поле UnitPrice.
10. Показать ID заказов, находящихся в разработке (WorkOrderID), ID товаров (ProductID), количество заказов (OrderQty), количество имеющихся в наличии товаров (StockedQty) и ID производственного отказа (ScrapReasonID) из таблицы Production.WorkOrder. Показать только существующие ID производственного отказа, а все значения данного поля, равные 1, заменить на NULL. Полученные значения вынести в отдельное поле с названием OtherScrapReasonID.

ВАРИАНТ 12

1. Показать все поля из таблицы `Production.ProductDocument`.
2. Показать поля `ProductModelID`, `IllustrationID` из таблицы `Production.ProductModelIllustration`.
3. Показать поля `ProductDescriptionID`, `Description` из таблицы `Production.ProductDescription`, для которых идентификатор (`ProductDescriptionID`) больше 3 и меньше 122. Использовать оператор `BETWEEN`.
4. Из таблицы `HumanResources.JobCandidate` показать ID кандидатов на должность (`JobCandidateID`), ID сотрудников (`BusinessEntityID`) и резюме (`Resume`). Показать только существующие ID сотрудников.
5. Показать ID товаров (`ProductID`), имена и адреса электронных почт покупателей (`ReviewerName` и `EmailAddress`) и комментарии (`Comments`) из таблицы `Production.ProductReview`, где имя покупателя начинается на 'J'.
6. Из таблицы `HumanResources.Department` показать поля `DepartmentID`, `Name` и `GroupName`. Показать только значения поля `GroupName` из списка ('Manufacturing', 'Quality Assurance', 'Research and Development'). Использовать оператор `IN`.
7. Показать ID сотрудников (`BusinessEntityID`), доли продаж (`SalesQuota`), продажи в этом году (`SalesYTD`) и продажи в прошлом году (`SalesLastYear`) из таблицы `Sales.SalesPerson`. Неизвестные доли продаж заменить на 0 и вынести в отдельное поле с названием `OtherSalesQuota`.

8. Показать поля `ProductID`, `Name` из таблицы `Production.Product` с применением функции `COALESCE()`. Показать поле `Measurement`, так, чтобы, если значение в поле `Size` известно, то показать его, а иначе, показать значение в поле `Class`. Если и в поле `Class` значение неизвестно, то вывести значение `'UNKNOWN'`.
9. Отсортировать в обратном алфавитном порядке строки таблицы `Person.AddressType` в соответствии с полем `Name`. Показать поля `AddressTypeID` и `Name`.
10. Из таблицы `Person.PersonPhone` показать ID сотрудников (`BusinessEntityID`), номера телефонов (`PhoneNumber`) и типы номеров телефонов (`PhoneNumberTypeID`). Отсортировать все строки в порядке возрастания значений в поле `BusinessEntityID` и все значения поля `PhoneNumberTypeID` равные 1 заменить на `NULL`. Названия полей оставить без изменений.

ВАРИАНТ 13

1. Показать все поля из таблицы `Production.ProductCostHistory`.
2. Показать поля `ProductModelID`, `Name` из таблицы `Production.ProductModel`.
3. Показать поля `CurrencyRateID`, `CurrencyRateDate`, `FromCurrencyCode`, `ToCurrencyCode`, `AverageRate`, `EndOfDayRate` из таблицы `Sales.CurrencyRate`, для которых средний курс обмена за день (`AverageRate`) больше \$1 и меньше \$10. Использовать оператор `BETWEEN`.
4. Из таблицы `Purchasing.ProductVendor` показать ID товара (`ProductID`), минимальное и максимальное количество заказов (`MinOrderQty` и `MaxOrderQty`) и существующее количество заказов на данный момент (`OnOrderQty`).
5. Показать поля `LocationID`, `ScheduledStartDate` и `ScheduledEndDate` из таблицы `Production.WorkOrderRouting`, где поле `LocationID` содержит 45.
6. Из таблицы `Sales.CreditCard` показать тип кредитной карты (`CardType`), номер кредитной карты (`CardNumber`), месяц и год, до которого действует кредитная карта (`ExpMonth` и `ExpYear`). Показать только те кредитные карты, крайний месяц эксплуатации которых или апрель, или июнь, или август, или октябрь (месяцы указаны цифрами). Использовать оператор `IN`.
7. Показать тип сотрудника (`PersonType`), его ФИО (`FirstName`, `MiddleName` и `LastName`) из таблицы `Person.Person`. Все неизвестные значения поля `MiddleName` заменить на '---'. Названия полей оставить без изменений.

8. Из таблицы `Production.Product` показать поля `ProductID`, `Name` с применением функции `COALESCE()`. Показать поле `Measurement`, так, чтобы, если значение в поле `Weight` известно, то показать его, а иначе, показать значение в поле `ProductLine`. Если и в поле `ProductLine` значение неизвестно, то вывести значение `'UNKNOWN'`.
9. Показать ID электронной почты (`EmailAddressID`) и саму электронную почту (`EmailAddress`) из таблицы `Person.EmailAddress`. Все электронные почты расположить в обратном алфавитном порядке.
10. Из таблицы `HumanResources.EmployeePayHistory` показать поля `BusinessEntityID`, `Rate` и `PayFrequency`. Все строки поля `PayFrequency` со значением 2 заменить на `NULL` и вынести в отдельное поле с названием `OtherPayFrequency`.

ВАРИАНТ 14

1. Показать все поля из таблицы `Production.ProductListPriceHistory`.
2. Показать поля `DocumentNode`, `DocumentLevel`, `Title`, `[Owner]`, `[FileName]`, `FileExtension` из таблицы `Production.Document`.
3. Показать поля `SalesOrderID`, `SalesOrderDetailID`, `ProductID`, `OrderQty` из таблицы `Sales.SalesOrderDetail`, для которых количество заказываемого продукта (`OrderQty`) больше 1 и меньше 3. Использовать оператор `BETWEEN`.
4. Из таблицы `HumanResources.EmployeeDepartmentHistory` показать ID сотрудника (`BusinessEntityID`), дату начала работы (`StartDate`) и дату увольнения (`EndDate`). Показать только тех сотрудников, у которых неизвестна дата увольнения.
5. Показать все поля из таблицы `Person.PhoneNumberType`, где строки поля `Name` содержат 'e'.
6. Из таблицы `Sales.SalesTerritory` показать поля `TerritoryID`, `Name` и `CountryRegionCode`, где значения поля `CountryRegionCode` принадлежат списку ('CA', 'US'). Использовать оператор `IN`.
7. Показать поля `JobCandidateID`, `BusinessEntityID` и `Resume` из таблицы `HumanResources.JobCandidate`. Неизвестные значения поля `BusinessEntityID` заменить на 0. Названия полей оставить без изменений.

8. Показать поля `ProductID`, `Name` из таблицы `Production.Product` с применением функции `COALESCE()`. Показать поле `Measurement`, так, чтобы, если значение в поле `Style` известно, то показать его, а иначе, показать значение в поле `ProductLine`. Если и в поле `ProductLine` значение неизвестно, то вывести значение `'UNKNOWN'`.
9. Из таблицы `Person.ContactType` показать поля `ContactTypeID` и `Name`, отсортированные в порядке убывания относительно поля `ContactTypeID`.
10. Показать поля `SalesReasonID`, `Name` и `ReasonType` из таблицы `Sales.SalesReason`. Заменить все значения поля `ReasonType` на `NULL`, если они принимают значение `'Other'`. Названия полей оставить без изменений.

ВАРИАНТ 15

1. Показать все поля из таблицы `Production.Illustration`.
2. Показать поля `ProductModelID`, `ProductDescriptionID`, `CultureID` из таблицы `Production.ProductModelProductDescriptionCulture`.
3. Показать поля `SalesTaxRateID`, `StateProvinceID`, `TaxType`, `TaxRate`, `Name` из таблицы `Sales.SalesTaxRate`, для которых сумма налоговой ставки (`TaxRate`) больше \$7 и меньше \$14. Использовать оператор `BETWEEN`.
4. Из таблицы `Sales.SalesTerritoryHistory` показать ID территории (`TerritoryID`), дату начала (`StartDate`) и существующую (`NOT NULL`) дату окончания работы фирмы на данной территории (`EndDate`).
5. Из таблицы `Purchasing.Vendor` показать номер и название аккаунта (`AccountNumber` и `Name`), где номер аккаунта заканчивается на '1'.
6. Из таблицы `Person.AddressType` показать поля `AddressTypeID` и `Name`. Значения поля `Name` взять либо `Billing`, либо `Primary`, либо `Archive`. Использовать оператор `IN`.
7. Показать ID товара (`ProductID`), его цену (`ListPrice`), дату начала (`StartDate`) и окончания (`EndDate`) действия цены из таблицы `Production.ProductListPriceHistory`. Если цена действующая (т. е. `EndDate = NULL`), то заменить её на текущую дату. Названия полей оставить без изменений.
8. Показать поля `ProductID`, `Name` из таблицы `Production.Product` с применением функции `COALESCE()`.

Показать поле Measurement, так, чтобы, если значение в поле Size известно, то показать его, а иначе, показать значение в поле ProductLine. Если и в поле ProductLine значение неизвестно, то вывести значение 'UNKNOWN'.

9. Показать поля LocationID и Name из таблицы Production.Location. Отсортировать все строки в алфавитном порядке в соответствии с полем Name.
10. Из таблицы Production.Location показать наименования товаров (Name), цену на товар (CostRate) и количество товаров в наличии (Availability). Если значения полей CostRate и Availability равны 0, то нужно заменить их на NULL и вынести в отдельные поля с названиями NullCostRate и NullAvailability соответственно.

ВАРИАНТ 16

1. Показать все поля из таблицы `Production.ProductModelProductDescriptionCulture`.
2. Показать поля `WorkOrderID`, `ProductID`, `OrderQty`, `StockedQty`, `ScrappedQty`, `ScrapReasonID` из таблицы `Production.WorkOrder`.
3. Показать поля `BusinessEntityID`, `TerritoryID`, `StartDate`, `EndDate` из таблицы `Sales.SalesTerritoryHistory`, для которых дата окончания работы торгового представителя на территории (`EndDate`) находится между 2012-05-29 и 2012-11-29. Использовать оператор `BETWEEN`.
4. Показать поле `BillOfMaterialsID` и даты прекращения использования компонента в сборочном элементе (`EndDate`). Пустые значения поля `EndDate` показывать не надо. Таблица `Production.BillOfMaterials`.
5. Из таблицы `Person.EmailAddress` показать все идентификационные номера e-mail адресов (`EmailAddressID`) и сами e-mail адреса (`EmailAddress`), содержащие в названии 'la'.
6. Показать все новые товары ('New Product') и все снятые с производства товары ('Discontinued Product') из поля (`Type`) таблицы `Sales.SpecialOffer`. Также вывести поля `SpecialOfferID`, `Description`, `DiscountPct`. Использовать оператор `IN`.
7. Из таблицы `HumanResources.JobCandidate` показать поля `JobCandidateID`, `BusinessEntityID` и `Resume`. Все неизвестные

значения поля `BusinessEntityID` заменить на 0, а само поле переименовать на `BusinessEntityID_0`.

8. Показать поля `ProductID`, `Name` из таблицы `Production.Product` с применением функции `COALESCE()`. Показать поле `Measurement`, так, чтобы, если значение в поле `Color` известно, то показать его, а иначе, показать значение в поле `Size`. Если и в поле `Size` значение неизвестно, то вывести значение `'UNKNOWN'`.
9. Отсортировать все строки таблицы `Sales.CreditCard` в обратном алфавитном порядке относительно типа кредитной карты (`CardType`). Показать ID кредитных карт (`CreditCardID`), их тип (`CardType`), а также номера карт (`CardNumber`).
10. Из таблицы `Production.ProductModelProductDescriptionCulture` показать поля `ProductModelID` и `CultureID`. Все значения поля `CultureID`, равные `'zh-cht'` заменить на `NULL` и вынести в отдельное поле с названием `NullCultureID`.

ВАРИАНТ 17

1. Показать все поля из таблицы `Production.ProductPhoto`.
2. Показать поля `WorkOrderID`, `ProductID`, `OperationSequence`, `LocationID`, `ActualStartDate`, `ActualEndDate`, `ActualCost` из таблицы `Production.WorkOrderRouting`.
3. В таблице `Production.Product` найти товары, дата продажи (`SellEndDate`) которых находится в диапазоне от 2012-05-29 до 2013-05-29. Показать поля `ProductID`, `Name`, `Color`, `Size`, `SellEndDate`. Использовать оператор `BETWEEN`.
4. Показать названия городов (`City`) и их первые адресные строки (`AddressLine1`) и существующую (`NOT NULL`) вторые адресные строки (`AddressLine2`) из таблицы `Person.Address`.
5. Из таблицы `Person.Person` показать `ID`, имя, отчество и фамилию сотрудников (`BusinessEntityID`, `FirstName`, `MiddleName` и `LastName`). Показать только тех сотрудников, имя которых начинается на D.
6. Показать поля `WorkOrderID`, `ProductID`, `OperationSequence` и `LocationID` из таблицы `Production.WorkOrderRouting`. Показать только значения поля `OperationSequence` равные или 3, или 5. Использовать оператор `IN`.
7. Показать из таблицы `Production.Product` `ID` (`ProductID`), название (`Name`) и цвет (`Color`) товаров, которые имеют цвет.
8. Показать поля `ProductID`, `Name` из таблицы `Production.Product` с применением функции `COALESCE()`. Показать поле `Measurement`, так, чтобы, если значение в поле `Weight` известно, то показать его, а

иначе, показать значение в поле Color. Если и в поле Color значение неизвестно, то вывести значение 'UNKNOWN'.

9. Из таблицы Sales.SalesReason показать поля SalesReasonID, Name и ReasonType. Отсортировать все строки таблицы в алфавитном порядке относительно поля Name.
10. Показать идентификационные номера заказов и причин продажи (SalesOrderID и SalesReasonID) из таблицы Sales.SalesOrderHeaderSalesReason. Все идентификационные номера причин продажи, равные 5, заменить на NULL. Названия полей оставить без изменений.

ВАРИАНТ 18

1. Показать все поля из таблицы `Production.ProductProductPhoto`.
2. Показать поля `LocationID`, `Name`, `CostRate`, `Availability` из таблицы `Production.Location`.
3. Из таблицы `Purchasing.PurchaseOrderHeader` показать товары, налог на вывоз (`TaxAmt`) которых не менее 300 и не более 3000. Показать поля `PurchaseOrderID`, `EmployeeID`, `OrderDate`, `ShipDate` и `TaxAmt`. Использовать оператор `BETWEEN`.
4. Показать все небракованные товары из таблицы `Production.WorkOrder`. Показать поля `WorkOrderID`, `ScrappedQty` (количество забракованных товаров) и `ScrapReasonID` (номер причины брака).
5. Показать все описания неисправностей (`Name`), связанных с температурой (`temperature`) из таблицы `Production.ScrapReason`.
6. Из таблицы `Sales.Customer` показать идентификационные номера магазинов (`StoreID`) и территории (`TerritoryID`), на которой они располагаются. Показать только ID территорий равные или 4, или 6, или 8, используя оператор `IN`.
7. Показать поля `CustomerID`, `PersonID` и `AccountNumber` из таблицы `Sales.Customer`. Все неизвестные значения поля `PersonID` заменить на -1. Названия полей оставить без изменений.
8. Показать поля `ProductID`, `Name` из таблицы `Production.Product` с применением функции `COALESCE()`. Показать поле `Measurement`,

так, чтобы, если значение в поле `Color` известно, то показать его, а иначе, показать значение в поле `Class`. Если и в поле `Class` значение неизвестно, то вывести значение `'UNKNOWN'`.

9. Показать все поля и строки таблицы `Person.BusinessEntityAddress` в порядке убывания в соответствии с полем `AddressID`.
10. Из таблицы `Person.StateProvince` показать названия территориальных субъектов (`Name`) и их флаг уникальности (`IsOnlyStateProvinceFlag`) (если территориальный субъект единственный на территории страны, флаг = 1, если не единственный, то флаг = 0). все флаги уникальности, равные 0, заменить на `NULL`. Названия полей оставить без изменений.

ВАРИАНТ 19

1. Показать все поля из таблицы `Production.ProductReview`.
2. Показать поля `ScrapReasonID`, `Name` из таблицы `Production.ScrapReason`.
3. Из таблицы `Sales.CurrencyRate` показать ID курса валюты (`CurrencyRateID`), даты, в которые проверяется курс валюты (`CurrencyRateDate`) и конечный курс валюты на данную дату (`EndOfDayRate`), взятый от \$1 до \$10. Использовать оператор `BETWEEN`.
4. Показать цену на товар (`ListPrice`), дату начала действия цены (`StartDate`) и дату окончания действия цены (`EndDate`) из таблицы `Production.ProductListPriceHistory`. Показать только те цены, которые актуальны (`EndDate = NULL`).
5. Показать все описания (`Name`) неисправностей (`damage`). Таблица `Production.ScrapReason`.
6. Из таблицы `Production.ProductInventory` показать поля `ProductID`, `Shelf` и `Quantity`. Вывести только те значения поля `Shelf`, которые принадлежат списку ('E', 'H', 'R', 'T'). Использовать оператор `IN`.
7. Показать ID организации (`BusinessEntityID`) и ID её территориального расположения (`TerritoryID`) из таблицы `Sales.SalesPerson`. Все неизвестные значения поля `TerritoryID` заменить на 0. Названия полей оставить без изменений.
8. Показать поля `ProductID`, `Name` из таблицы `Production.Product` с применением функции `COALESCE()`. Показать поле `Measurement`,

так, чтобы, если значение в поле Color известно, то показать его, а иначе, показать значение в поле Weight. Если и в поле Weight значение неизвестно, то вывести значение 'UNKNOWN'.

9. Отсортировать все поля (BusinessEntityID, RateChangeDate, Rate) таблицы HumanResources.EmployeePayHistory в порядке возрастания относительно поля Rate.
10. Из таблицы Person.BusinessEntityContact показать поля PersonID и ContactTypeID. Все значения поля ContactTypeID, равные 11, заменить на NULL и вынести в отдельное поле с названием NullContactTypeID.

ВАРИАНТ 20

1. Показать все поля из таблицы `Production.TransactionHistory`.
2. Показать поля `ProductSubcategoryID`, `ProductCategoryID`, `Name` из таблицы `Production.ProductSubcategory`.
3. Показать поля `WorkOrderID`, `ProductID`, `ScheduledStartDate`, `ScheduledEndDate` из таблицы `Production.WorkOrderRouting` при условии, что запланированная дата начала производства (`ScheduledStartDate`) начинается не ранее 03.06.2011 (включительно), а запланированная дата окончания производства (`ScheduledEndDate`) заканчивается не позднее 14.06.2011 (включительно).
4. Из таблицы `Production.Product` показать идентификационный номер и название товара (`ProductID` и `Name`), а также СУЩЕСТВУЮЩИЕ единицы измерения размера и веса (`SizeUnitMeasureCode` и `WeightUnitMeasureCode`).
5. Показать из таблицы `Production.ProductSubcategory` все названия товаров (`Name`), содержащих в названии 'Bike'.
6. Из таблицы `Purchasing.ProductVendor` показать поля `ProductID` и `UnitMeasureCode`, где значения поля `UnitMeasureCode` принадлежат списку ('GAL', 'CS', 'EA'). Использовать оператор `IN`.
7. Показать ID специального предложения (`SpecialOfferID`), тип и категорию товаров (`[Type]` и `Category`), а также минимальное и максимальное количество товаров, попавших в специальное предложение (`MinQty` и `MaxQty`) из таблицы `Sales.SpecialOffer`.

Неизвестное максимальное количество товаров заменить на 0. Названия полей оставить без изменений.

8. Показать поля `ProductID`, `Name` из таблицы `Production.Product` с применением функции `COALESCE()`. Показать поле `Measurement`, так, чтобы, если значение в поле `Color` известно, то показать его, а иначе, показать значение в поле `Size`. Если и в поле `Size` значение неизвестно, то вывести значение `'UNKNOWN'`.
9. Отсортировать в порядке убывания строки таблицы `Production.ProductCostHistory` в соответствии с полем `StartDate`. Показать поля `ProductID`, `StartDate`, `EndDate` и `StandardCost`.
10. Из таблицы `Person.StateProvince` показать поля `StateProvinceID`, `StateProvinceCode`, `CountryRegionCode` и `IsOnlyStateProvinceFlag`. Все значения поля `IsOnlyStateProvinceFlag`, равные 0, заменить на `NULL`. Названия полей оставить без изменений.

ВАРИАНТ 21

1. Показать все поля из таблицы `Production.TransactionHistoryArchive`.
2. Показать поля `ProductDescriptionID`, `Description` из таблицы `Production.ProductDescription`.
3. Показать поля (`WorkOrderID`, `ProductID`, `ActualCost`) из таблицы `Production.WorkOrderRouting`, при условии, что актуальная цена (`ActualCost`) находится в диапазоне между \$50 и \$100. Использовать оператор `BETWEEN`.
4. Показать все наименования модели «рама» (`Frame`) поля `Name` и `ProductModelID`, из таблицы `Production.ProductModel`.
5. Из таблицы `Person.Address` показать адрес (`AddressLine1`), город (`City`) и ID территориального субъекта (`StateProvinceID`), где адрес содержит 'Dr. '.
6. Показать поля `TransactionID` и `TransactionType` из таблицы `Production.TransactionHistoryArchive`, где значения поля `TransactionType` принадлежат списку ('P', 'S'). Использовать оператор `IN`.
7. Из таблицы `Production.Product` показать ID товара (`ProductID`), его название (`Name`) и тип линейки товаров (`ProductLine`). Известные значения линейки товаров заменить на 'no'. Названия полей оставить без изменений.
8. Показать поля `ProductID`, `Name` из таблицы `Production.Product` с применением функции `COALESCE()`. Показать поле `Measurement`, так, чтобы, если значение в поле `Class` известно, то показать его, а

иначе, показать значение в поле `Color`. Если и в поле `Color` значение неизвестно, то вывести значение `'UNKNOWN'`.

9. Из таблицы `Purchasing.ShipMethod` показать идентификационный номер метода доставки товаров (`ShipMethodID`) и название транспортной компании (`Name`). Отсортировать все строки в алфавитном порядке относительно поля `Name`.
10. Из таблицы
`Production.ProductModelProductDescriptionCulture`
показать поля `ProductModelID` и `CultureID`. Все значения поля `CultureID`, равные `'fr'`, заменить на `NULL` и вынести в отдельное поле с названием `NullCultureID`.

ВАРИАНТ 22

1. Показать все поля из таблицы `Production.UnitMeasure`.
2. Показать поля `BillOfMaterialsID`, `ProductAssemblyID`, `ComponentID`, `UnitMeasureCode` из таблицы `Production.BillOfMaterials`.
3. Из таблицы `Production.ProductInventory` показать идентификационные номера и количество товаров (`ProductID` и `Quantity`). Показать только те товары, количество которых больше 400, но меньше 600. Использовать оператор `BETWEEN`.
4. Показать поля `CustomerID` и все существующую (`NOT NULL`) значения поля `StoreID` из таблицы `Sales.Customer`.
5. Из таблицы `Person.CountryRegion` показать код (`CountryRegionCode`) и название (`Name`) стран, которые заканчиваются на 'n'.
6. Показать код и название валюты (`CurrencyCode` и `Name`) из таблицы `Sales.Currency`. Показать только валюты из списка ('Kroon', 'Armenian Dram', 'US Dollar').
7. Из таблицы `Production.ProductCostHistory` показать среднюю цену на товар (`StandardCost`), дату начала (`StartDate`) и окончания (`EndDate`) действия цены. Если дата окончания действия цены неизвестна, заменить её на текущую дату. Название полей оставить без изменений.
8. Показать поля `ProductID`, `Name` из таблицы `Production.Product` с применением функции `COALESCE()`. Показать поле `Measurement`, так, чтобы, если значение в поле `Class` известно, то показать его, а

иначе, показать значение в поле `Style`. Если и в поле `Style` значение неизвестно, то вывести значение `'UNKNOWN'`.

9. Отсортировать строки таблицы `Sales.SalesTaxRate` в соответствии с полем `Name`. Показать поля `TaxType`, `TaxRate` и `Name`.
10. Показать поля `CountryRegionCode` и `Name` из таблицы `Person.StateProvince`. Все значение поля `CountryRegionCode`, равные `US`, заменить на `NULL` и вынести в отдельное поле с названием `NullCountryRegionCode`.

ВАРИАНТ 23

1. Показать все поля из таблицы `Production.ProductModelIllustration`.
2. Показать поля `ProductID`, `Name`, `ProductNumber`, `Color`, `ListPrice` из таблицы `Production.Product`.
3. Из таблицы `Production.ProductModelProductDescriptionCulture` показать идентификационные номера товаров (`ProductDescriptionID`) в диапазоне от 1000 до 1300. Показать поля `ProductDescriptionID` и `CultureID`. Использовать оператор `BETWEEN`.
4. Показать из таблицы `Sales.SpecialOffer` значения полей `[Type]`, `MinQty` и СУЩЕСТВУЮЩИЕ значения поля `MaxQty`.
5. Из таблицы `HumanResources.Employee` показать логин и должность сотрудника (`LoginID` и `JobTitle`), его дату рождения (`BirthDate`) и дату принятия сотрудника на должность (`HireDate`). Показать должности, которые связаны с дизайном (`Design`).
6. Показать поля `PurchaseOrderDetailID`, `OrderQty`, `ProductID` и `UnitPrice` из таблицы `Purchasing.PurchaseOrderDetail`. Показать значения поля `OrderQty`, равные или 4, или 60, или 1250. Использовать оператор `IN`.
7. Из таблицы `Sales.SalesOrderHeader` показать поля `AccountNumber`, `CustomerID` и `SalesPersonID`. Все неизвестные значения поля `SalesPersonID` заменить на 0. Название полей оставить без изменений.

8. Показать поля `ProductID`, `Name` из таблицы `Production.Product` с применением функции `COALESCE()`. Показать поле `Measurement`, так, чтобы, если значение в поле `Class` известно, то показать его, а иначе, показать значение в поле `ProductLine`. Если и в поле `ProductLine` значение неизвестно, то вывести значение `'UNKNOWN'`.
9. Отсортировать в обратном алфавитном порядке уникальные значения типов сотрудников (`PersonType`) из таблицы `Person.Person`.
10. Из таблицы `Production.ProductModelIllustration` показать идентификационные номера моделей и фотографий товаров (`ProductModelID` и `IllustrationID`). Все ID фотографий, равные 4, заменить на `NULL` и переименовать поле в `NullIllustrationID`.

ВАРИАНТ 24

1. Показать все поля из таблицы `Production.ProductModel`.
2. Показать поле `IllustrationID` из таблицы `Production.Illustration`.
3. Показать поля `ProductID`, `LocationID`, `Shelf`, `Quantity` из таблицы `Production.ProductInventory`, при условии, что количество товара (`Quantity`) больше 400.
4. Показать идентификационный номер адреса (`AddressID`), первый адрес (`AddressLine1`), названия городов (`City`) из таблицы `Person.Address`, у которых имеется вторая строка адреса (`AddressLine2`).
5. Из таблицы `Sales.CreditCard` показать ID кредитной карты (`CreditCardID`), тип и номер кредитной карты (`CardType` и `CardNumber`). Показать только те карты, в названии которых (`CardType`) содержится `Card`.
6. Показать все поля таблицы `Production.ProductReview`, у которых ID товара (`ProductID`) или 798, или 937. Использовать оператор `IN`.
7. Из таблицы `Person.Person` показать имя, отчество, фамилию и суффикс (`FirstName`, `MiddleName`, `LastName` и `Suffix`). Все неизвестные отчества заменить на `'---'`, а все неизвестные суффиксы – на `'NO'`. Названия полей оставить без изменений.
8. Показать поля `ProductID`, `Name` из таблицы `Production.Product` с применением функции `COALESCE()`. Показать поле `Measurement`, так, чтобы, если значение в поле `Class` известно, то показать его, а

иначе, показать значение в поле Weight. Если и в поле Weight значение неизвестно, то вывести значение 'UNKNOWN'.

9. Из таблицы Sales.SalesTerritory показать поля TerritoryID, Name и CountryRegionCode. Отсортировать строки в алфавитном порядке в соответствии с полем Name.
10. Показать поля TaxType, TaxRate и Name из таблицы Sales.SalesTaxRate. Все значения поля TaxType, равные 2, заменить на NULL. Названия полей оставить без изменений.

ВАРИАНТ 25

1. Показать все поля из таблицы `Production.ProductModelProductDescriptionCulture`.
2. Показать поля `ProductID`, `DocumentNode` из таблицы `Production.ProductDocument`.
3. Из таблицы `Sales.PersonCreditCard` показать все поля, если значения поля `CreditCardID` больше 19000.
4. Показать ID сотрудников (`BusinessEntityID`), существующую (`NOT NULL`) квоты продаж (`SalesQuota`) и дополнительные продажи (`Bonus`) из таблицы `Sales.SalesPerson`.
5. Показать поля `ProductAssemblyID` и `UnitMeasureCode` из таблицы `Production.BillOfMaterials`, где значения поля `UnitMeasureCode` содержат 'OZ'.
6. Из таблицы `Production.Location` показать ID местоположения и название магазина (`LocationID` и `Name`). Название магазина должно быть или `Metal Storage`, или `Paint Storage`, или `Finished Goods Storage`. Использовать оператор `IN`.
7. Показать поля `BusinessEntityID`, `PersonType`, `Title` и `LastName` из таблицы `Person.Person`. Все неизвестные значения поля `Title` заменить на 'Dear.'. Названия полей оставить без изменения.
8. Показать поля `ProductID`, `Name` из таблицы `Production.Product` с применением функции `COALESCE()`. Показать поле `LastDate`, так, чтобы, если значение в поле `SellEndDate` известно, то показать его, а

иначе, показать значение в поле `DiscontinuedDate`. Если и в поле `DiscontinuedDate` значение неизвестно, то вывести текущую дату.

9. Показать и отсортировать в обратном алфавитном порядке все уникальные значения поля `CultureID` из таблицы `Production.ProductModelProductDescriptionCulture`.
10. Из таблицы `Sales.SalesTerritory` показать поля `TerritoryID`, `Name` и `CountryRegionCode`. Все значения поля `CountryRegionCode`, равные `US`, заменить на `NULL`. Названия полей оставить без изменений.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. PostgresPro/education [Электронный ресурс] // PostgresPro: [сайт]. [2023]. URL: <https://postgrespro.ru/> (дата обращения: 10.03.2023).
2. АО «Кодекс». МЕЖГОСУДАРСТВЕННЫЙ СТАНДАРТ ГОСТ 7.32-2017 2021. URL: <https://docs.cntd.ru/document/1200157208> (дата обращения: 16.11.2021).
3. Портал федеральных государственных образовательных стандартов высшего образования. ИНФОРМАЦИОННАЯ БЕЗОПАСНОСТЬ // fgosvo. 2023. URL: https://fgosvo.ru/uploadfiles//FGOS%20VO%203%2B%2B/Spec/10.05.01_C_3_08112022.pdf (дата обращения: 16.03.2023).
4. Портал федеральных государственных образовательных стандартов высшего образования. УПРАВЛЕНИЕ В ТЕХНИЧЕСКИХ СИСТЕМАХ // fgosvo. 2023. URL: https://fgosvo.ru/uploadfiles/FGOS%20VO%203++/Bak/270304_B_3_31082020.pdf (дата обращения: 16.3.2023).
5. Paradox (database) [Электронный ресурс] // Wikipedia: [сайт]. [2022]. URL: [https://en.wikipedia.org/wiki/Paradox_\(database\)](https://en.wikipedia.org/wiki/Paradox_(database)) (дата обращения: 17.03.2023).
6. Васильева М.А. Навигационный способ доступа к базе данных. Методические указания к лабораторной работе по дисциплине «Информационное обеспечение систем управления». Москва: МИИТ, 2007. 16 с.
7. Васильева М.А. Работа со связанными таблицами. Методические указания к лабораторной работе по дисциплине "Информационное обеспечение систем управления". Москва: МИИТ, 2011. 22 с.

8. Васильева М.А. Создание таблиц баз данных. Методические указания к лабораторной работе по дисциплине «Информационное обеспечение систем управления». Москва: МИИТ, 2007. 26 с.
9. Васильева М.А., Балакина Е.П. Введение в базы данных. Учебное пособие по дисциплине «Информационное обеспечение систем управления». Москва: МИИТ, 2007. 80 с.
10. Васильева М.А., Балакина Е.П. Реляционные способы доступа к базам данных. Методические указания к лабораторной работе по дисциплине «Информационное обеспечение систем управления». Москва: МИИТ, 2008. 24 с.
11. Васильева М.А., Балакина Е.П. Информационное обеспечение систем управления. Методические указания к курсовому проектированию. 1-е изд. Москва: Гуманитарный институт федерального государственного бюджетного образовательного учреждения высшего образования "Российский университет транспорта (МИИТ)" (Москва), 2007. 32 с.
12. Delphi (язык программирования) [Электронный ресурс] // Википедия: [сайт]. [2023]. URL: [https://ru.wikipedia.org/wiki/Delphi_\(язык_программирования\)](https://ru.wikipedia.org/wiki/Delphi_(язык_программирования)) (дата обращения: 17.03.2023).
13. Знакомство с SQL Server 2022 [Электронный ресурс] // Microsoft: [сайт]. [2023]. URL: <https://www.microsoft.com/ru-ru/sql-server/> (дата обращения: 17.03.2023).
14. Васильева М.А., Тимофеева О.А., Филиппченко К.М. Фильтрация набора данных: Учебно-методическое пособие. 1-е изд. Москва: РУТ (МИИТ), 2020. 31 с.
15. Васильева М.А., Филиппченко К.М. Система контроля версия. Основы командной разработки: учебное пособие для ВУЗов. Санкт-Петербург: Лань, 2022. 144 с.

16. Васильева М.А. Комплексное взаимодействие лингвистических и выпускающих кафедр в техническом вузе. Международная научно-практическая конференция посвященная 125-летию РУТ (МИИТ). // Цифровое образование как ответ вызовам сегодняшнего дня. Москва. 2021. С. 98–102.
17. Васильева М.А. Современные технологии в науке и образовании - СТНО-2021 // Применение системы контроля версий git для организации учебного процесса в вузе. Рязань. 2021. Т. 10. С. 15–18.
18. Васильева М.А., Меркулов Д.А. Группировка и обобщение данных. Рекомендации по выполнению работы и перечень типовых заданий: Учебно-методическое пособие. Москва: РУТ (МИИТ), 2023. 46 с.
19. Васильева М.А., Ракинцев Н.А. Соединение данных из множества таблиц. Рекомендации по выполнению работы и перечень типовых заданий: Учебно-методическое пособие. Москва: РУТ (МИИТ), 2023. 60 с.
20. Васильева М.А., Хобта Д.О. Фильтрация набора данных. Рекомендации по выполнению работы и перечень типовых заданий. 2-е изд. Москва: РУТ (МИИТ), 2023. 150 с.
21. Васильева М.А., Балакина Е.П., Филипченко К.М. Информационное обеспечение систем управления. Методические указания к курсовому проектированию. Учебно-методическое пособие. 2-е изд. Москва: РУТ (МИИТ), 2023. 102 с.
22. Васильева М.А., Балакина Е.П., Филипченко К.М. Информационное обеспечение систем управления. Проектирование базы данных с заданиями: Учебник для ВУЗов. Санкт-Петербург: Лань, 2023. 210 с.
23. Васильева М.А., Быкова Е.А., Викторов К.А. Информационное обеспечение автоматизированной системы выбора энергооптимальных режимов управления поездом метрополитена. // Цифровые

инфокоммуникационные технологии: сборник научных трудов ФГБОУ ВО РГУПС., 2022. С. 35 – 40.

24. Васильева М.А. Цифровая трансформация транспорта: проблемы и перспективы: материалы Международной научно-практической конференции «Цифровые технологии транспорта и логистики», (28 сентября 2022 г.). // Автоматизированная система выбора энергооптимальных режимов управления. Москва. 2022. С. 218-221.
25. Блог SkillFactory [Электронный ресурс] // DBeaver: [сайт]. [2023]. URL: <https://blog.skillfactory.ru/glossary/dbeaver/> (дата обращения: 14.03.2023).
26. Поиск по шаблону [Электронный ресурс] // Postgres Pro: [сайт]. [2023]. URL: <https://postgrespro.ru/docs/postgresql/9.6/functions-matching> (дата обращения: 05.11.2023).
27. Моргунов Е.П., Рогова Е.В., Лузанова П.В. Основы языка SQL: учеб. пособие. СПб: БХВ-Петербург, 2018. 336 с.
28. support.microsoft.com [Электронный ресурс] // Special characters : [сайт]. [2021]. URL: <https://support.microsoft.com/en-us/office/insert-a-symbol-in-word-2a061ae9-5a6c-4407-b618-8dc3c9fd4f44> (дата обращения: 18.11.2021).

ПРИЛОЖЕНИЕ А

**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«РОССИЙСКИЙ УНИВЕРСИТЕТ ТРАНСПОРТА»
(РУТ (МИИТ))**

**Институт транспортной техники и систем управления
Кафедра «Управление и защита информации»**

**ОТЧЁТ
О ЛАБОРАТОРНОЙ РАБОТЕ №1
По дисциплине «Наименование дисциплины»
ВАРИАНТ Х**

**Выполнил: ст. гр. Название группы
Фамилия Имя Отчество
Проверил: к.т.н., доц. Васильева М. А.**

Москва ГОД

ПРИЛОЖЕНИЕ Б

УСТАНОВКА ПРИЛОЖЕНИЯ DBeaver

Скачайте с официального сайта (Рисунок 64) приложение DBeaver.

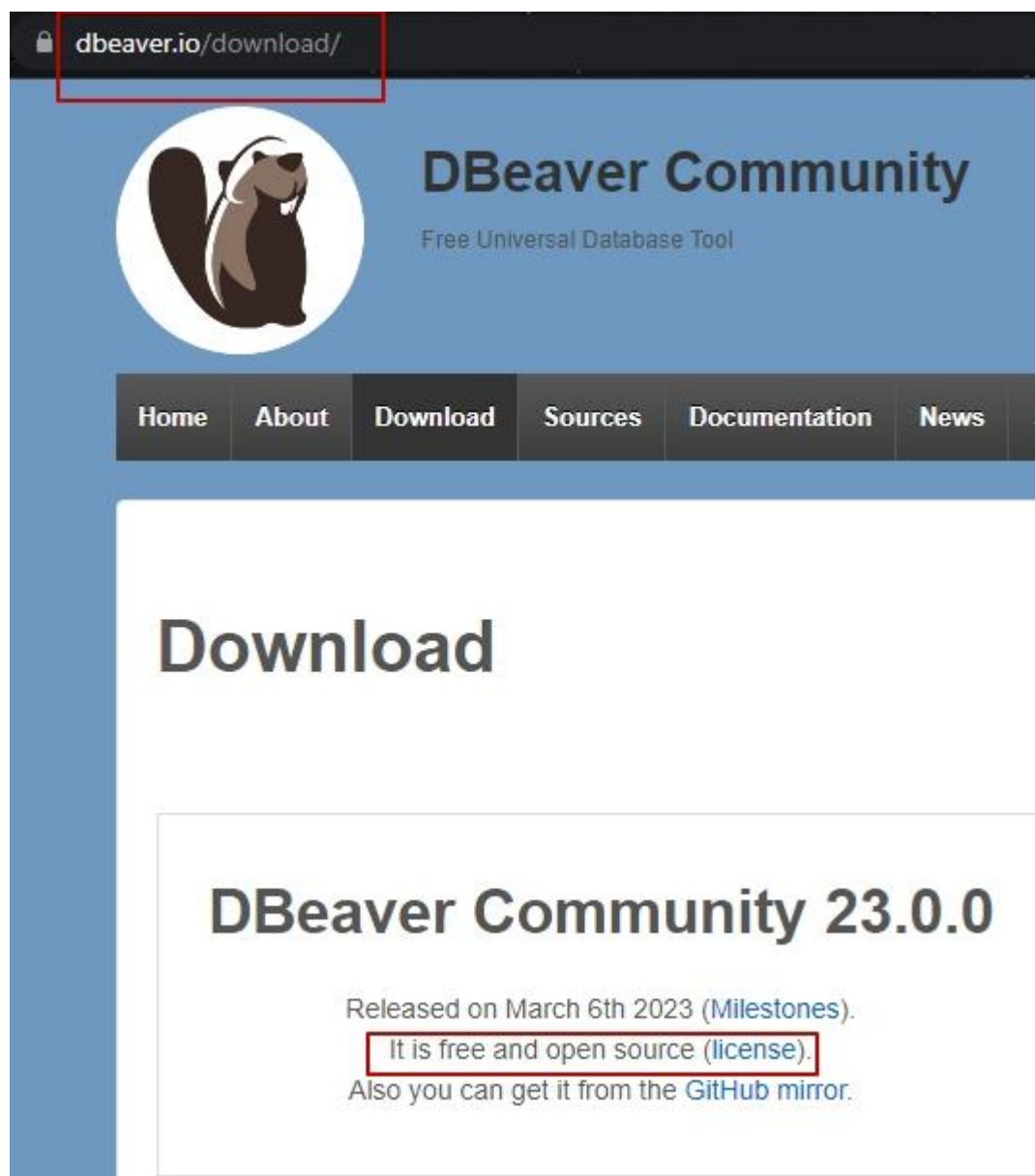


Рисунок 64 – Официальный сайт для скачивания приложения DBeaver

После загрузки вы можете открыть и запустить программу установки. Мастер установки все сделает за вас. Просто нажимайте «Next» (Рисунок 65).

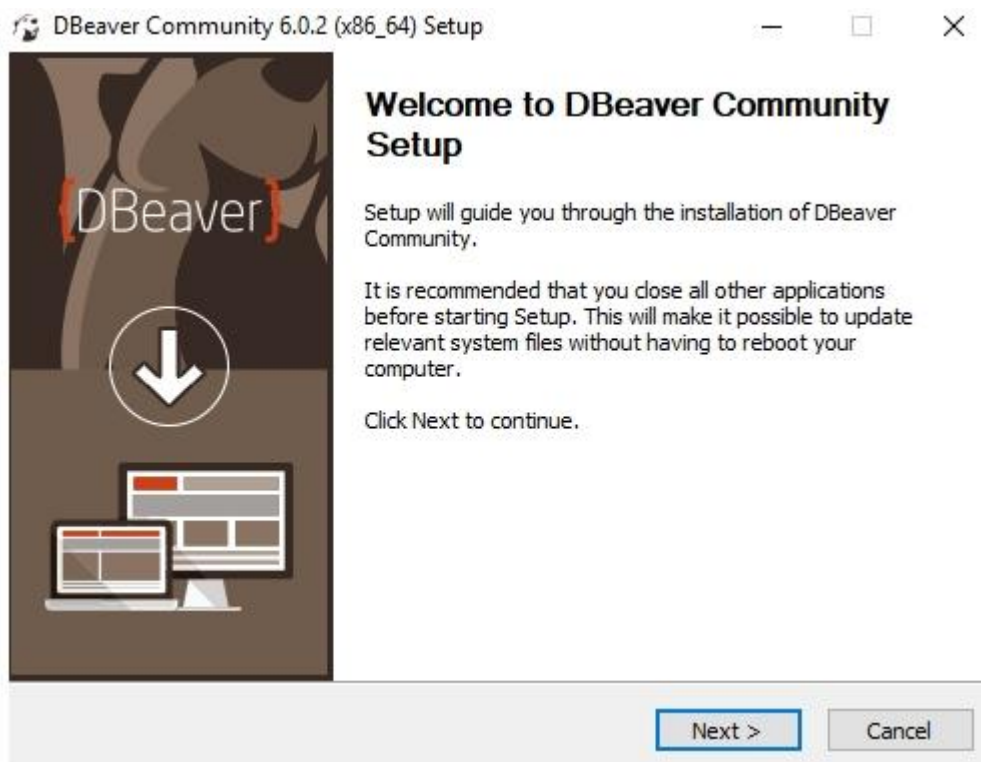


Рисунок 65 – Мастер установки приложения

Нажмите на кнопку согласия принятия лицензии «I Agree» (Рисунок 66).

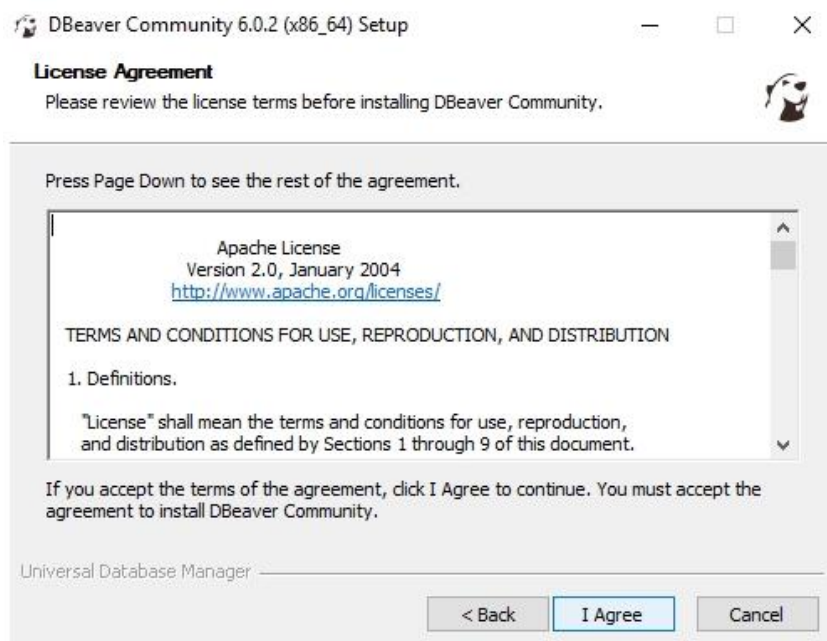


Рисунок 66 – Подтверждение лицензии



Рисунок 67 – Выбор языка

Выберите круг пользователей, для которых будет установлено приложение (Рисунок 68).

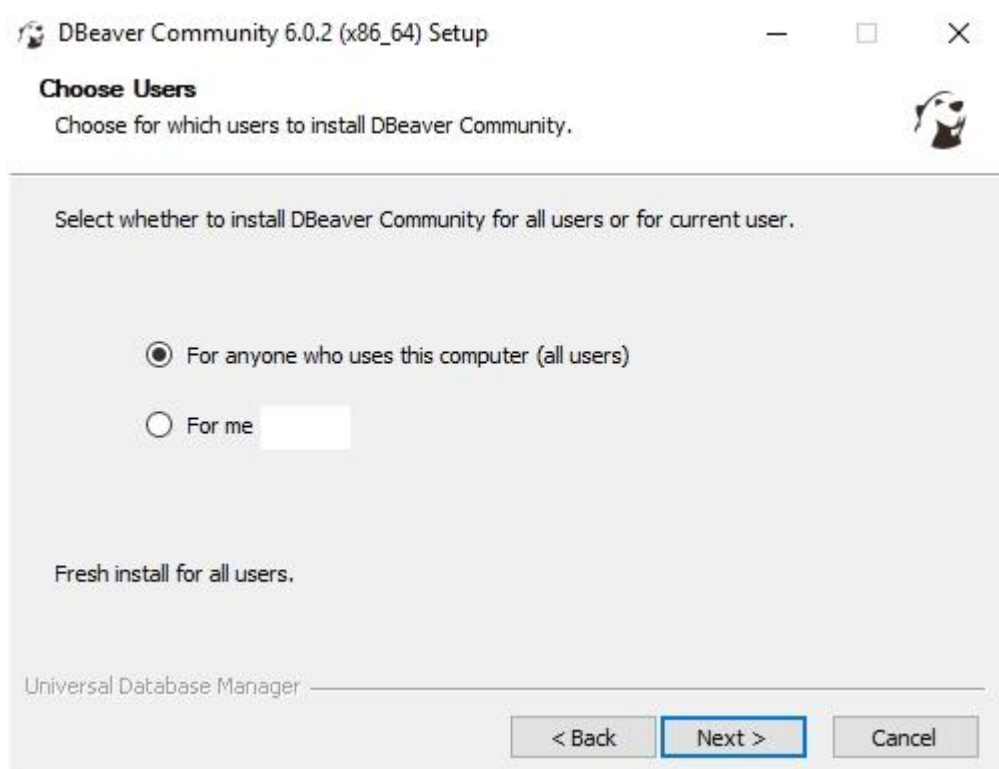


Рисунок 68 – Приложение будет использовано всеми пользователями

Если вы хотите, чтобы все файлы с расширением «.sql» ассоциировались с программой DBEaver (файл будет открыт программой DBEaver), отметьте последний пункт для установки соответствующего компонента (Рисунок 69).

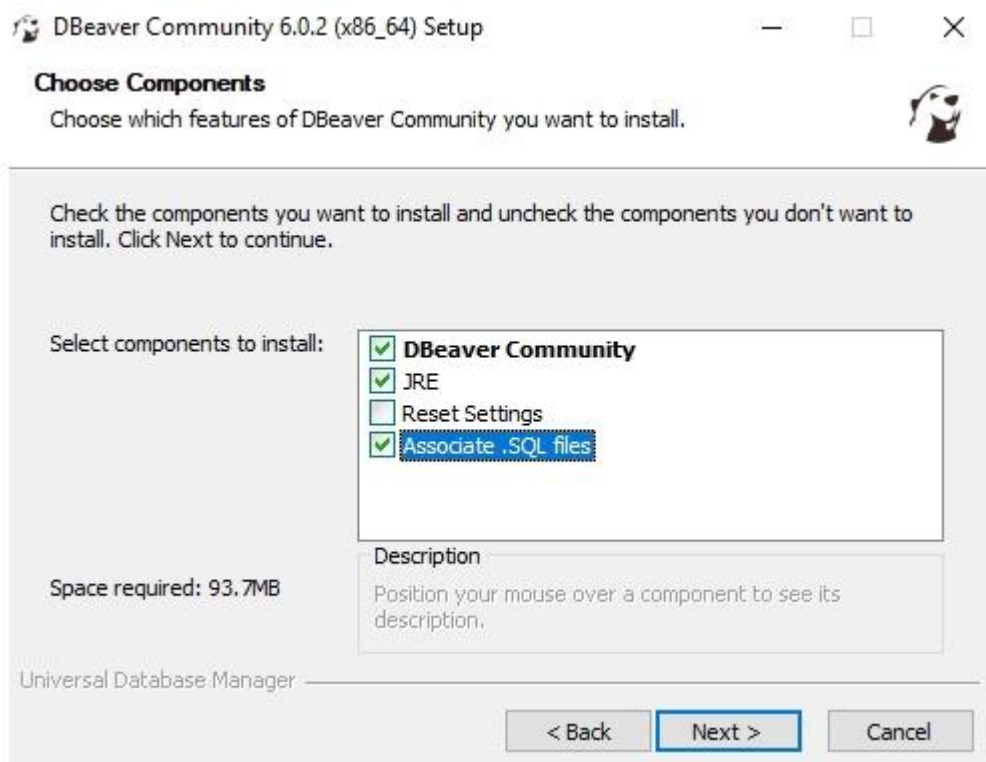


Рисунок 69 – Необходимые для установки компоненты

Выберите место для установки приложения (Рисунок 70).

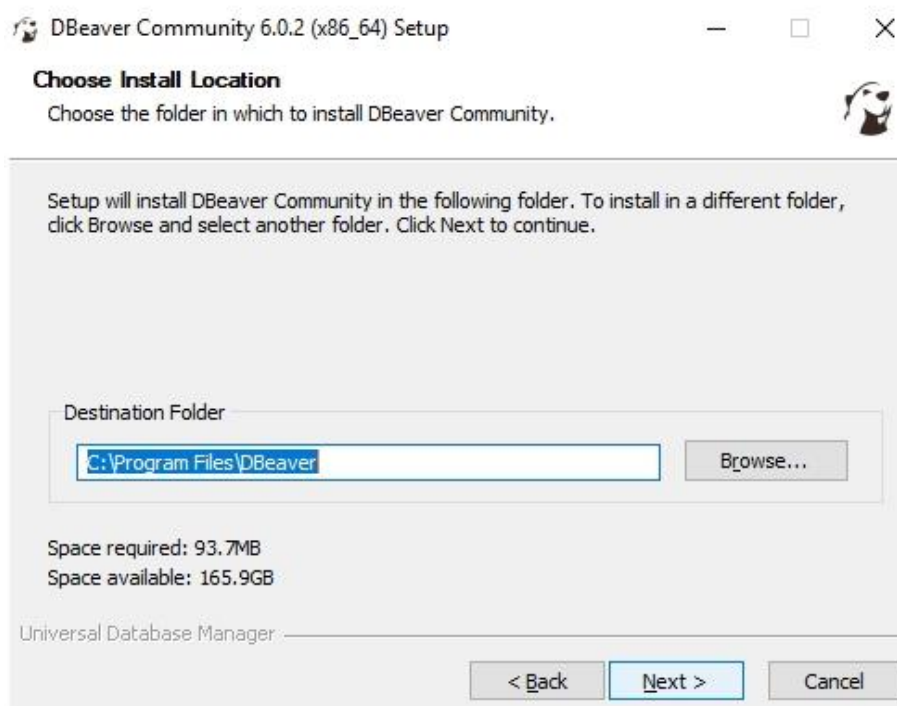


Рисунок 70 – Выбор места для установки программы

Для выбора группы значков для запуска приложения, выберите папку (Рисунок 71), чтобы запускать приложение через меню Пуск (Рисунок 72).

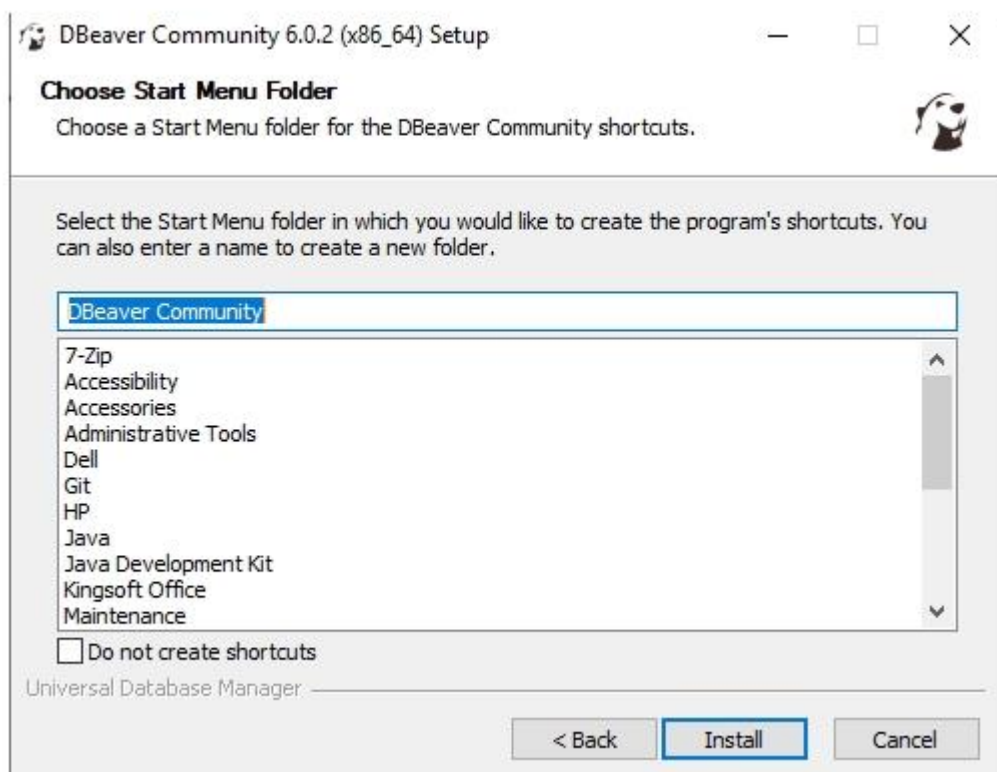


Рисунок 71 – Выбор группы значков для запуска приложения

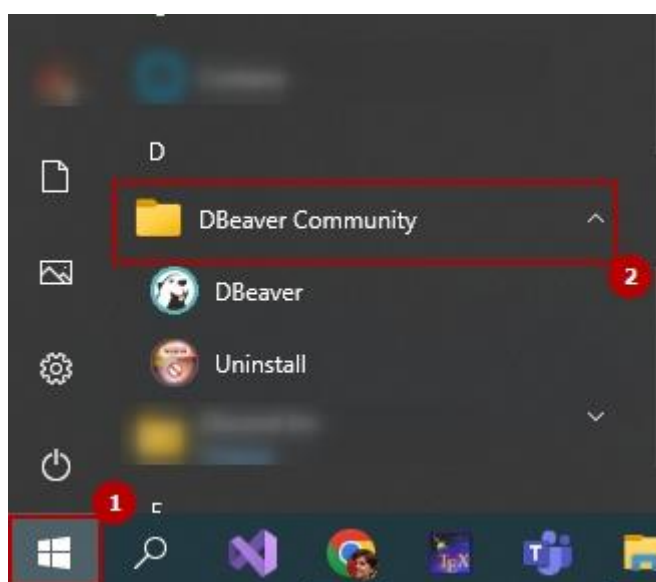


Рисунок 72 – Пример группы значков для запуска приложения

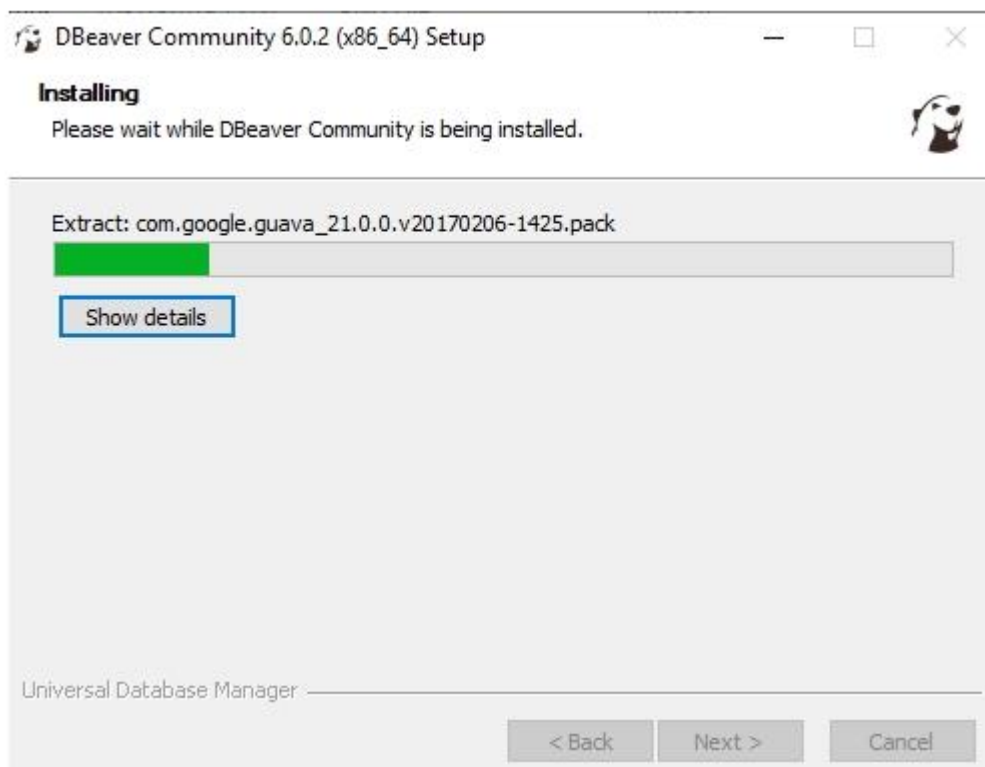


Рисунок 73 – Процесс установки приложения

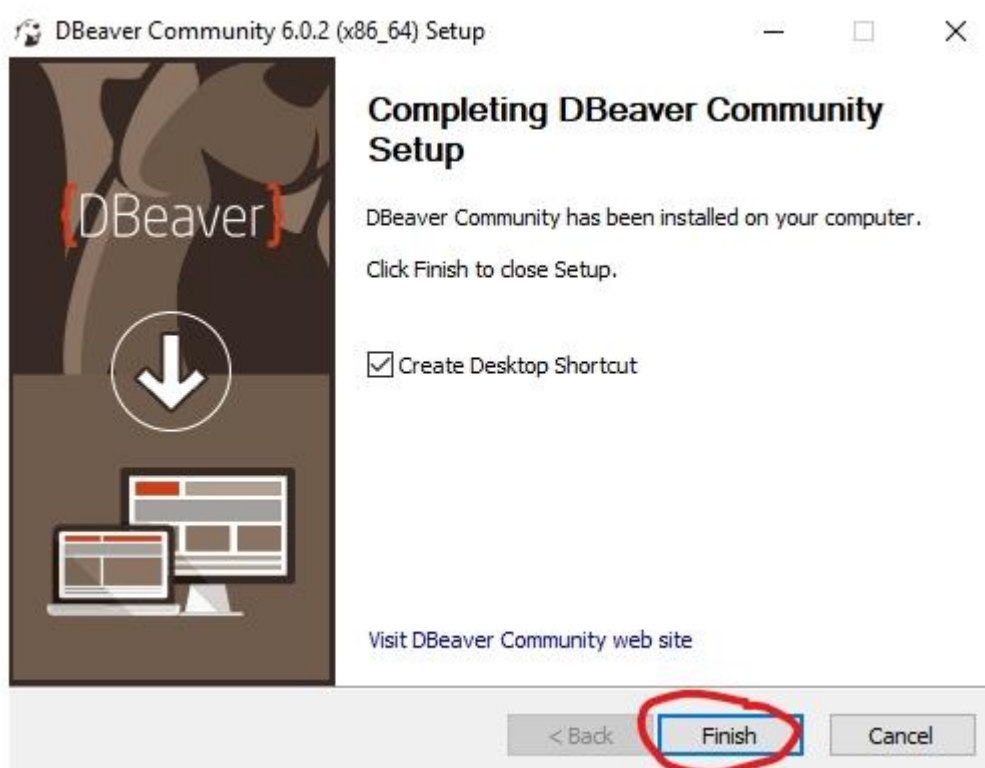


Рисунок 74 – Приложение установлено

Для работы с СУБД PostgreSQL необходимо настроить приложение DBeaver. Выберите пункт меню Database/New Database Connection (Рисунок 75).



Рисунок 75 – Создание нового соединения

Выберите СУБД из имеющегося списка (Рисунок 76).

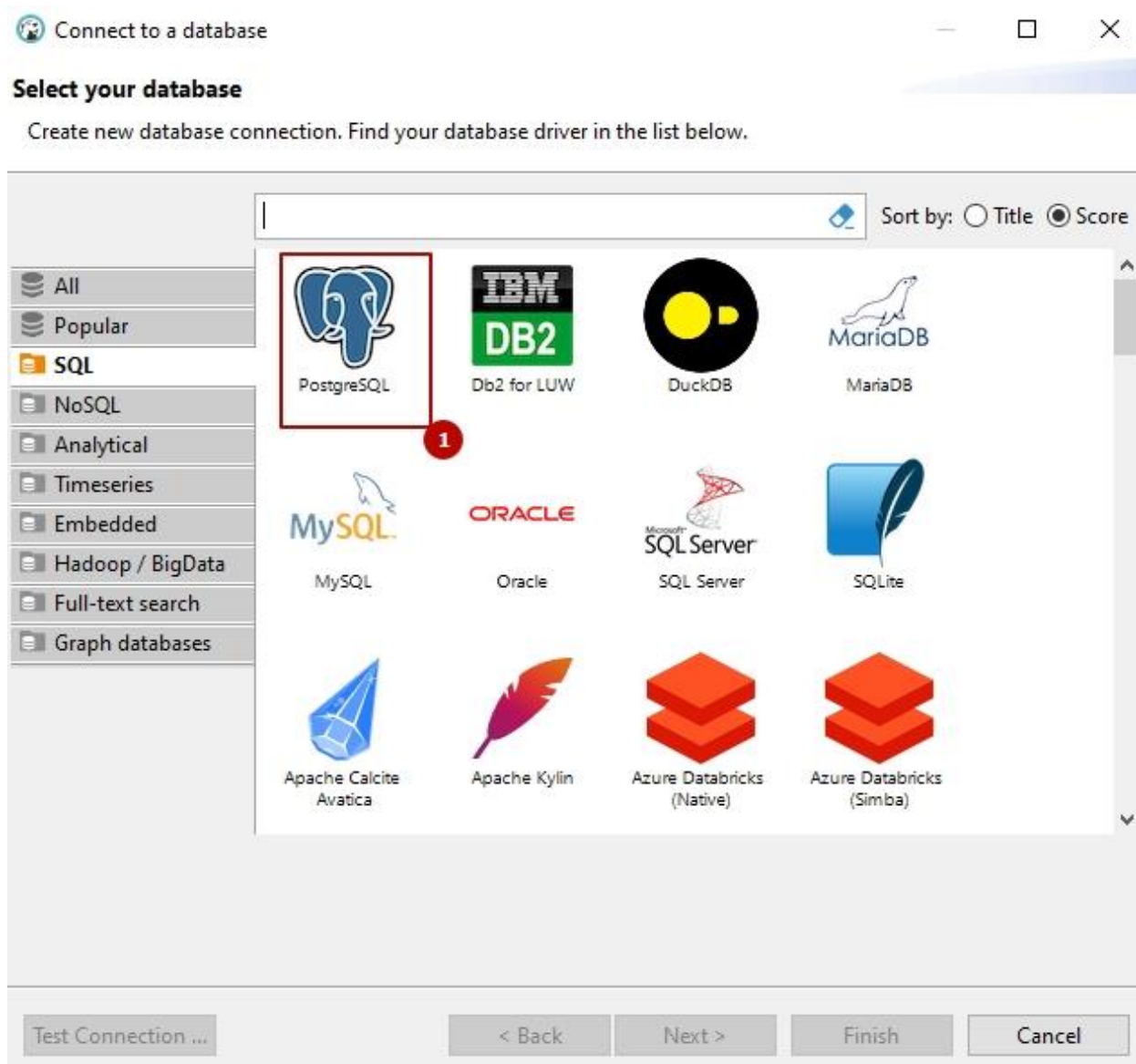


Рисунок 76 – Окно выбора СУБД для работы

Настройте соединение, установите пароль, который был использован при установке PostgreSQL (Рисунок 77).

The screenshot shows a window titled "Connect to a database" with a PostgreSQL logo in the top right corner. The window has several tabs: "Main", "PostgreSQL", "Driver properties", "SSH", "Proxy", and "SSL". The "Main" tab is selected, displaying the "PostgreSQL connection settings".

Server

Connect by: ☒ Host ☐ URL

URL: jdbc:postgresql://localhost:5432/postgres

Host: localhost Port: 5432

Database: postgres

Authentication

Authentication: Database Native

Username: postgres

Password: [masked] ☒ Save password locally

Advanced

Session role: Local Client: PostgreSQL 15

i You can use variables in connection parameters. [Connection details \(name, type, ...\)](#)

Driver name: PostgreSQL [Driver Settings](#) [Driver license](#)

Buttons at the bottom: Test Connection ..., < Back, Next >, Finish, Cancel.

Рисунок 77 – Окно настройки соединения

Для правильной работы приложения с СУБД PostgreSQL рекомендуется загрузить последнюю версию JDBC (Java DataBase Connectivity) – драйвер для соединения с различными СУБД с официального сайта (Рисунок 78).

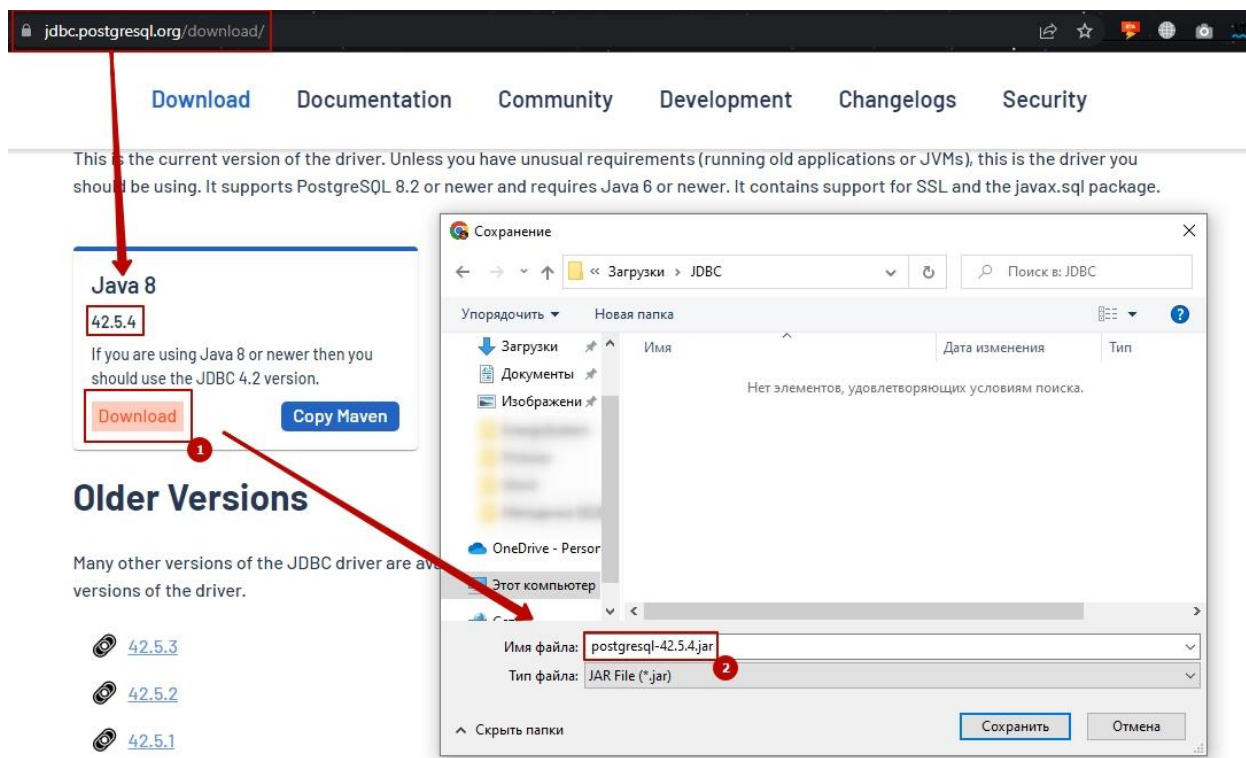


Рисунок 78 – Загрузка последней версии JDBC с официального сайта

Выберите кнопку «Driver Settings» (Рисунок 79) и в открывшемся окне на вкладке Libraries нажмите на кнопку «Add file». Выберите путь, куда был скачен драйвер JDBC (Рисунок 80) и выбираем его (Рисунок 81).

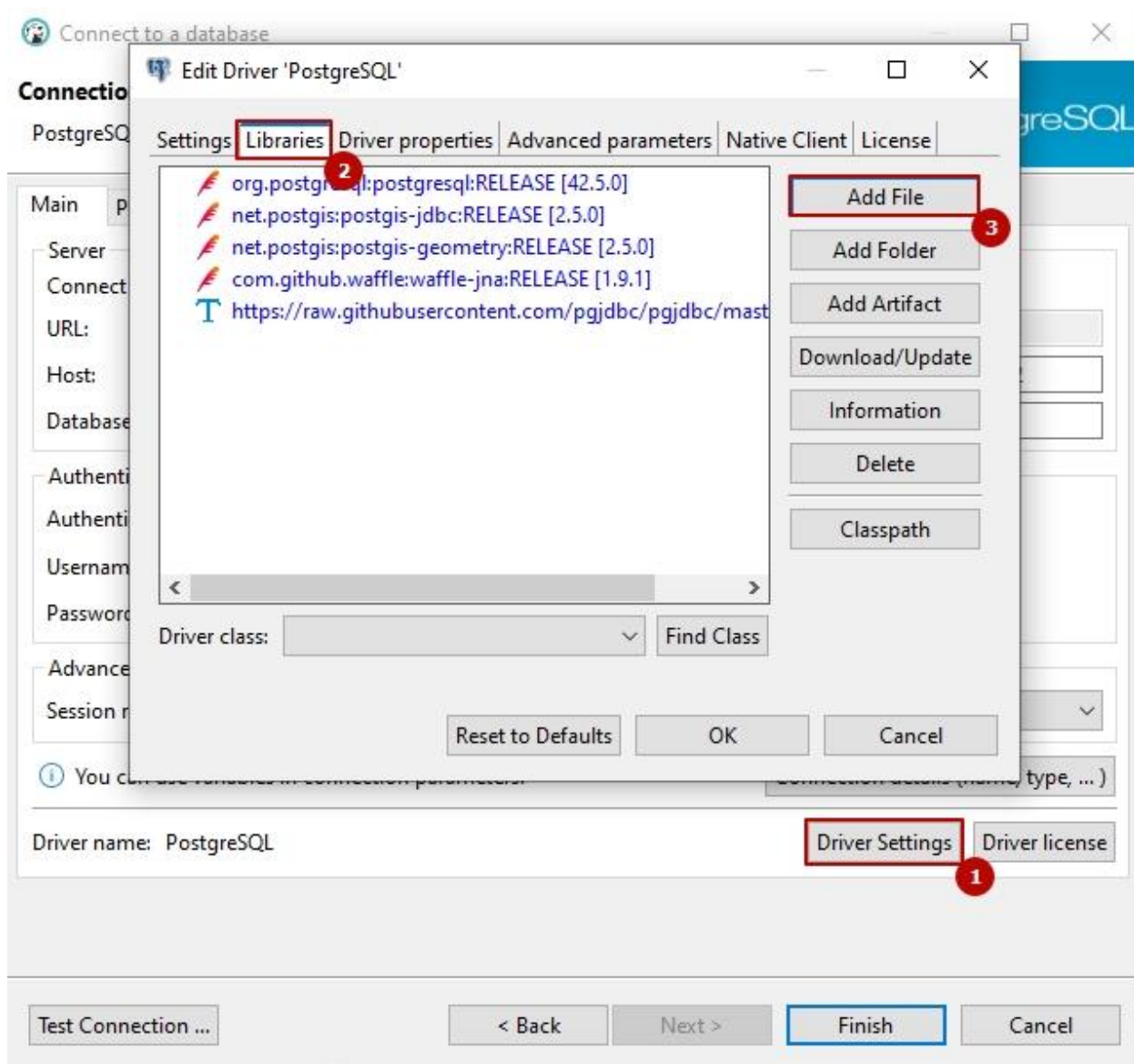


Рисунок 79 – Добавление необходимых драйверов

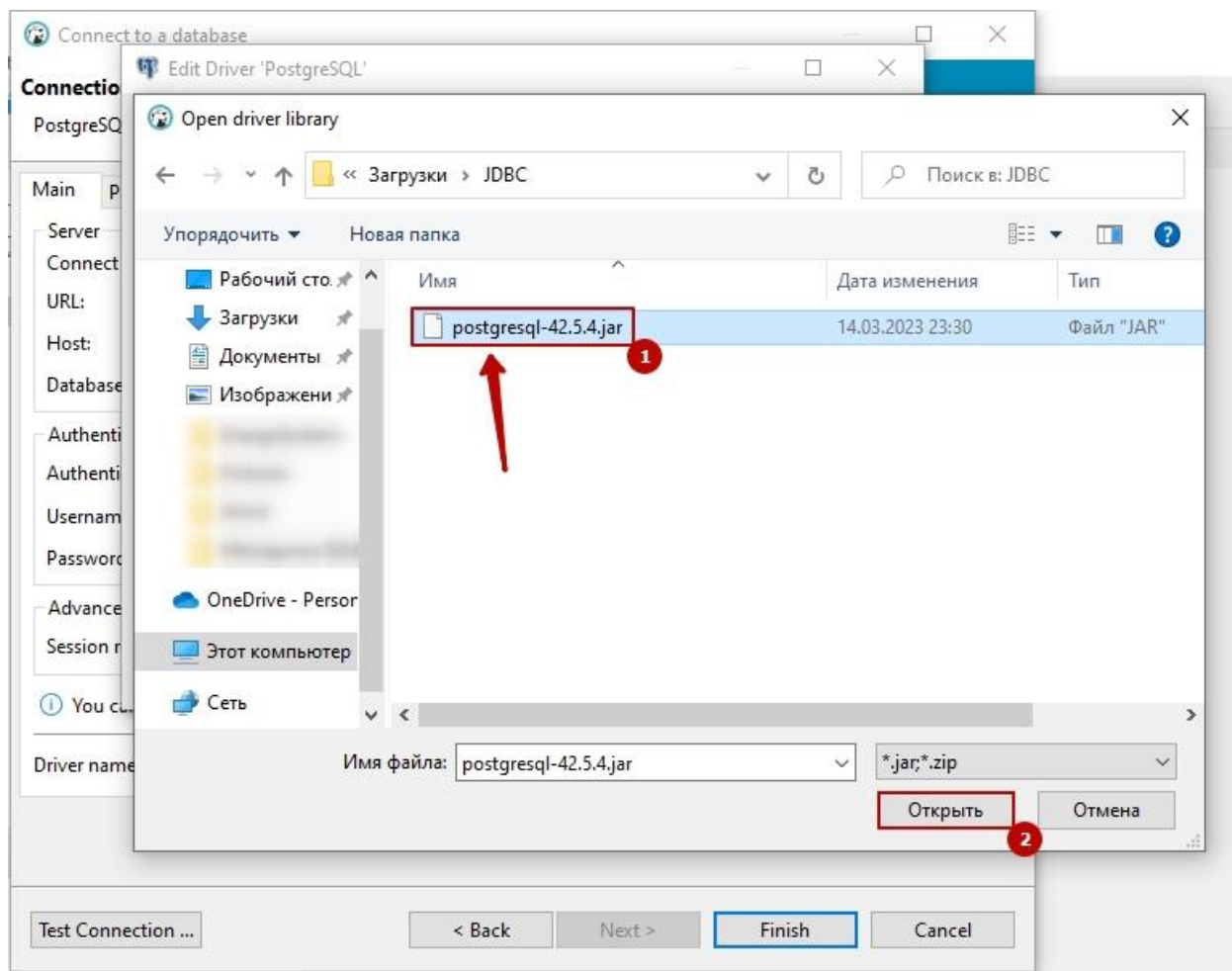


Рисунок 80 – Выбор драйвера JDBC

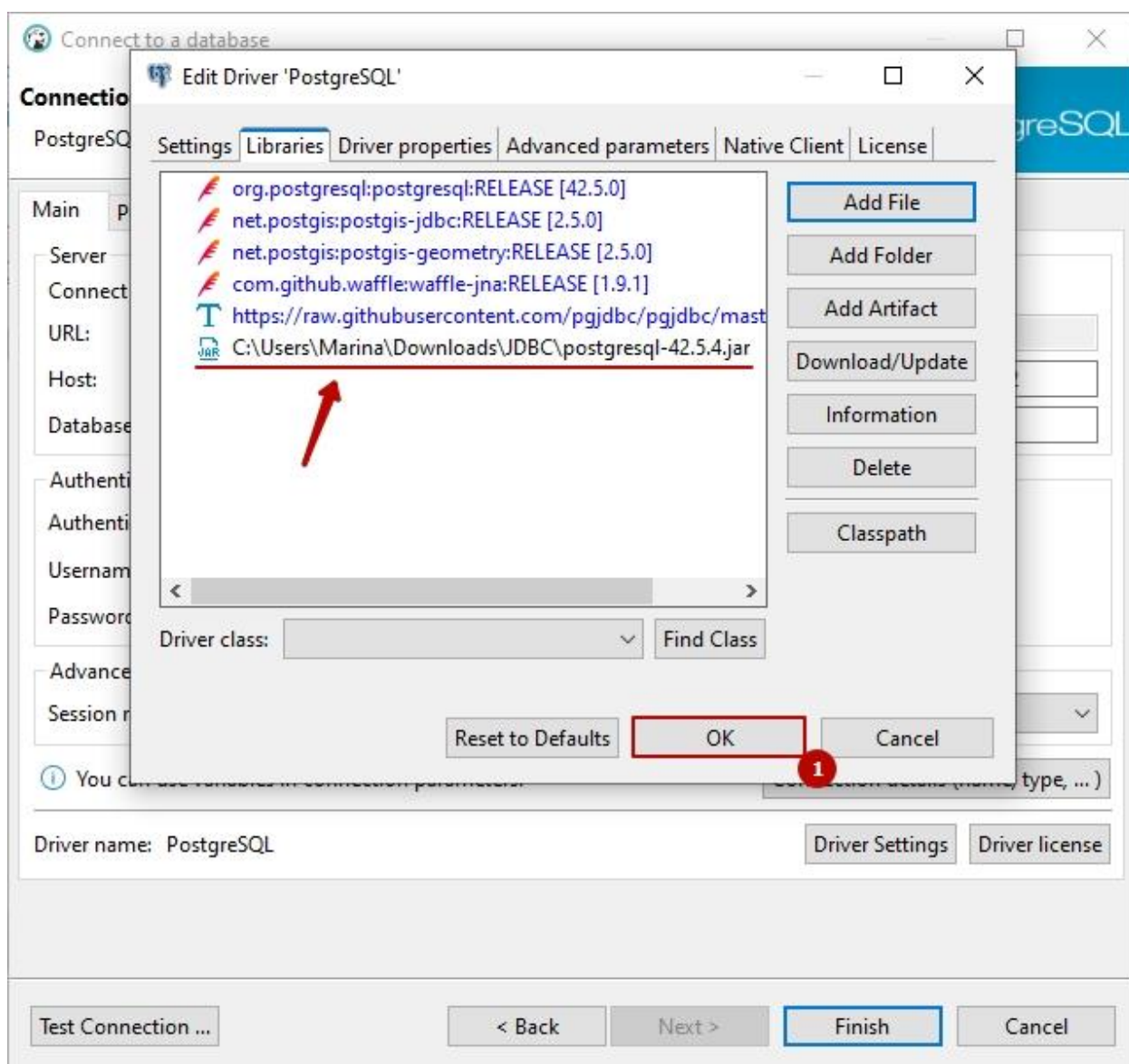


Рисунок 81 – Установка драйвера

Для проверки созданного соединения, выберите кнопку «Test Connection» (Рисунок 82). Ниже представлен пример удачного соединения (Рисунок 82).

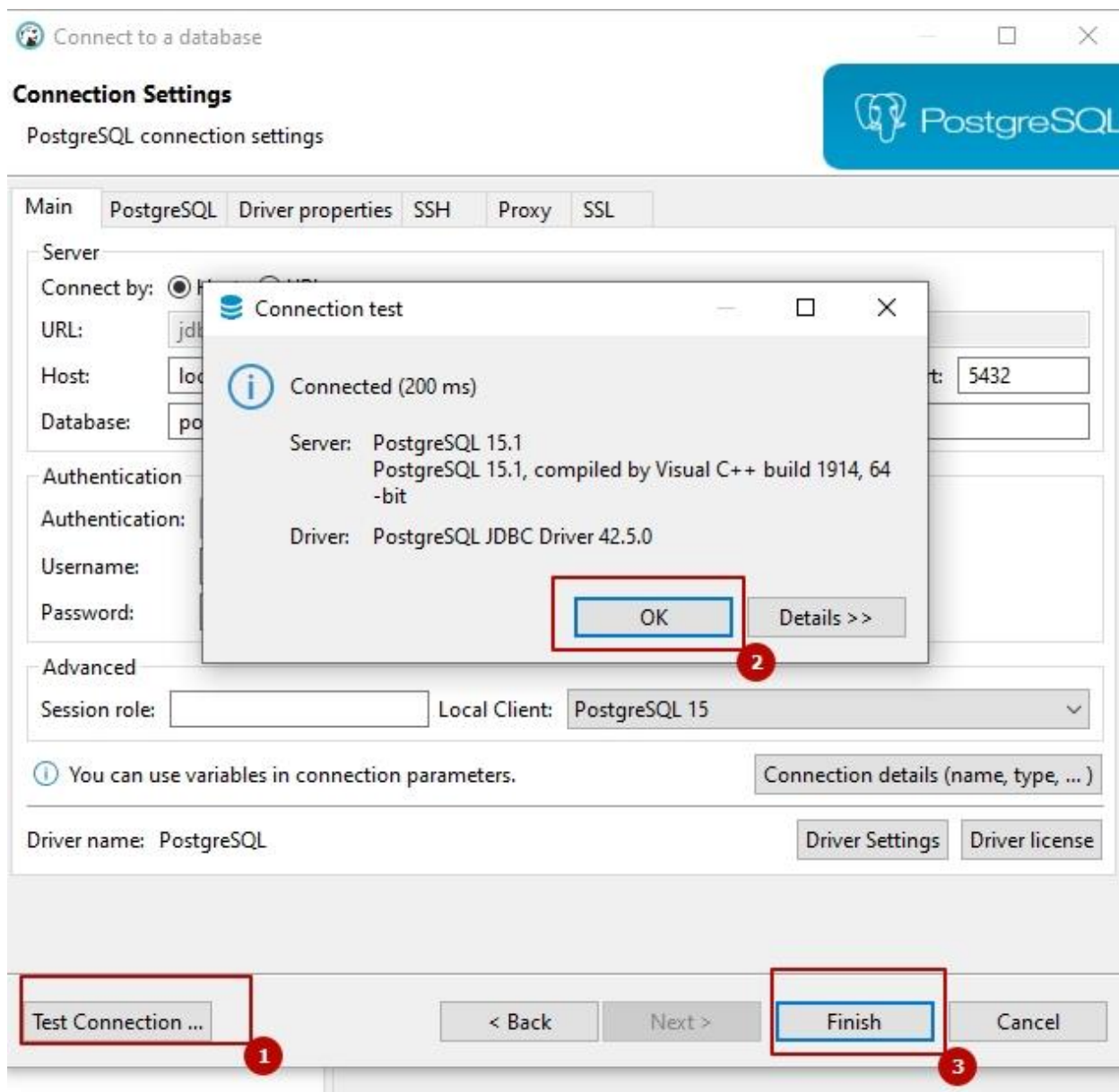


Рисунок 82 – Проверка соединения

УЧЕБНО-МЕТОДИЧЕСКОЕ ИЗДАНИЕ

Васильева Марина Алексеевна

Филипченко Константин Михайлович

Пугачёв Владислав Александрович

Фильтрация набора данных.
Рекомендации по выполнению работы
и перечень типовых заданий

Учебно-методическое пособие