

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«Магнитогорский государственный технический университет
им. Г. И. Носова»**
(ФГБОУ ВО «МГТУ им. Г.И. Носова»)

Кафедра бизнес-информатики и информационных технологий

Отчет
по учебной – технологической (проектно-технологической) практике

Исполнитель: Климов А.С. студент 2 курса, группы АПИб-22-2

Руководитель практики: _____ Масленникова Ольга Евгеньевна, к.п.н., доцент кафедры БИиИТ

Руководитель практики
от Профильной организации: _____ Ошурков Вячеслав Александрович, руководитель направления
консалтинга службы бизнес-анализа и консалтинга ЗАО «КонсОМ СКС»

Отчет защищен «16» июля 2024 г. с оценкой _____
(оценка) (подпись)

Магнитогорск, 2024

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Магнитогорский государственный технический университет
им. Г. И. Носова
(ФГБОУ ВО «МГТУ им. Г.И. Носова»)

Кафедра бизнес-информатики и информационных технологий

РАБОЧИЙ ПЛАН-ГРАФИК

по учебной – технологической (проектно-технологической) практике

в период с 03.07.2024 по 16.07.2024

Обучающемуся Климову Артему Сергеевичу

группы

АПИ6-22-2

№	Этапы практики по выполнению программы практики и индивидуального задания	Срок исполнения
1	Установочное собрание по организации практики	03.07.2024
2	Выполнение индивидуального задания	04.07.2024 – 09.07.2024
3	Оформление отчёта по учебно-эксплуатационной практике	10.07.2024 – 14.07.2024
4	Защита отчёта по учебно-эксплуатационной практике	16.07.2024

Руководитель практики от МГТУ им. Г.И. Носова

Доцент кафедры БИиИТ, к.п.н

(подпись)

/ Масленникова О.Е./

Руководитель практики от Профильной организации

Руководитель направления
консалтинга службы бизнес-
анализа и консалтинга ЗАО
«КонсОМ СКС»

(подпись)

/ Ошурков В.А./

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
**«Магнитогорский государственный технический университет
им. Г. И. Носова
(ФГБОУ ВО «МГТУ им. Г.И. Носова)»**

Кафедра бизнес-информатики и информационных технологий

ИНДИВИДУАЛЬНОЕ ЗАДАНИЕ

по учебной – технологической (проектно-технологической) практике

Обучающемуся Климову Артему Сергеевичу

группы

АПИБ-22-2

09.03.03 Прикладная информатика (разработка компьютерных игр и AR/VR-приложений
(виртуальной/дополненной реальности))

1. Период практики: с 03.07.2024 по 16.07.2024
2. Место прохождения практики ЗАО «КонсОМ СКС»

№ п/п	Содержание индивидуального задания (перечень задач, подлежащих выполнению)
1	Организационно-подготовительный этап
1.1	Участие в установочном собрании по организации практики. Получение индивидуального задания.
1.2	Вводный инструктаж представителя закрытого акционерного общества «КонсОМ СКС» обучающимся по правилам ТБ, производственной и противопожарной безопасности
2	Основной этап
2.1	Установка PostgreSQL сервера и программного продукта DBeaver
2.2	Описание SQL-запросов
3	Отчетный этап
3.1	Подготовка и защита отчета по практике

Руководитель практики
от МГТУ им. Г.И. Носова

(подпись)

/ Масленникова О.Е. /

Обучающийся

(подпись)

/ Климов А.С. /

СОГЛАСОВАНО:

Руководитель практики
от Профильной организации

(подпись)

/ Ошурков В.А. /

Дата выдачи 03.07.2024

Дневник прохождения практики

Студента Климов А.С.

Группы АПИб-22-2

курса 2

Направления 09.03.03 Прикладная информатика (разработка компьютерных игр и AR/VR-приложений (виртуальной/дополненной реальности))

Сроки практики: с 03.07.2024 по 16.07.2024 г.

Дата	Краткое содержание выполненной работы	Отметка о выполнении
1. Организационно-подготовительный этап		
03.07.2024	1.1 Участие в установочном собрании по организации практики. Получение индивидуального задания.	
10.07.2024	1.2 Вводный инструктаж представителя закрытого акционерного общества «КонсОМ СКС» обучающимся по правилам ТБ, производственной и противопожарной безопасности	
2. Основной этап		
04.07.2024	Установка PostgreSQL сервера и программного продукта DBeaver	
05.07.2024 – 09.07.2024	Описание SQL-запросов	
3. Отчетный этап		
11.07.2024 – 16.07.2024	Подготовка и сдача отчета по практике	

Руководитель практики
от МГТУ им. Г.И. Носова

(подпись)

Масленникова О.Е.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	6
1 ОРГАНИЗАЦИОННО-ПОДГОТОВИТЕЛЬНЫЙ ЭТАП	7
1.1 Участие в установочном собрании по организации практики. Получение индивидуального задание	7
1.2 Вводный инструктаж представителя закрытого акционерного общества «КонсОМ СКС» обучающимся	7
2 ОСНОВНОЙ ЭТАП	8
2.1 Установка PostgreSQL	8
2.2 Установка и подготовка программного продукта DBeaver	9
3 ОПИСАНИЕ SQL-ЗАПРОСОВ	14
3.1 Выполнение «Задания 1»	14
3.2 Выполнение «Задания 2»	15
3.3 Выполнение «Задания 3»	15
3.4 Выполнение «Задания 4»	16
4 ОТЧЕТНЫЙ ЭТАП	17
ЗАКЛЮЧЕНИЕ	18
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	19
ПРИЛОЖЕНИЯ	20
Приложение А Результат выполнения «Задания 1»	20
Приложение Б Результат выполнения «Задания 2»	24
Приложение В Результат выполнения «Задания 3»	28
Приложение Г Результат выполнения «Задания 4»	31
Приложение Д Справка о проверке в системе «Антиплагиат.ру»	35

ВВЕДЕНИЕ

В соответствии с учебным планом учебная - технологическая (проектно-технологическая) практика была пройдена в ЗАО «КонсОМ СКС» с 3 июля 2024 г. по 16 июля 2024 г. (14 дней). Учебная - технологическая (проектно-технологическая) практика проводится с целью закрепления полученных студентом теоретических знаний и приобретения практических навыков, необходимых для самостоятельной работы в IT-компании. Актуальность прохождения практики обуславливается тем, что написание дипломной работы основывается на материалах прохождения учебной практики, в связи с этим прохождение практики облегчает написание выпускной квалификационной работы.

Целями учебной - технологическая (проектно-технологическая) практики являются:

1. закрепление и углубление теоретических знаний, полученных при изучении дисциплин учебного плана;
2. приобретение и развитие необходимых практических умений и навыков в соответствии с требованиями к уровню подготовки выпускника;
3. приобретение опыта самостоятельной профессиональной деятельности.

Задачами учебной - технологическая(проектно-технологическая) практики являются:

1. изучение нотации SQL;
2. разработка SQL-запросов для выполнения индивидуального задания.

1 ОРГАНИЗАЦИОННО-ПОДГОТОВИТЕЛЬНЫЙ ЭТАП

1.1 Участие в установочном собрании по организации практики. Получение индивидуального задания

В первый день практики состоялось установочное собрание, на котором были обсуждены ключевые аспекты организации и проведения практики. На собрании присутствовали куратор практики от предприятия и студенты.

Основные темы, освещённые на собрании:

1. обзор структуры компании и её основных подразделений;
2. цели и задачи технологической практики;
3. правила внутреннего распорядка и требования к безопасности;
4. основные этапы практики и ожидаемые результаты.

После установочного собрания каждому студенту было выдано индивидуальное задание, включающее установку PostgreSQL сервера и программного продукта DBeaver, описание шагов и практических заданий.

1.2 Вводный инструктаж представителя закрытого акционерного общества «КонсОМ СКС» обучающимся

В первый день практики состоялся вводный инструктаж, проведённый представителем «КонсОМ СКС» для обучающихся. На инструктаже были рассмотрены следующие ключевые аспекты:

1. Общие сведения о компании. Представитель компании дал обзор деятельности «КонсОМ СКС», её основных направлений работы и достижений.
2. Организационная структура. были представлены основные подразделения компании и их функции, а также краткое описание ролей и обязанностей сотрудников.
3. Корпоративная культура. Обсуждены ценности компании, стандарты поведения, ожидания от сотрудников и принципы взаимодействия внутри коллектива.
4. Правила внутреннего распорядка. Описаны основные правила и нормы поведения, график работы, требования к посещаемости и оформлению документов.
5. Меры безопасности: Ознакомление с основными требованиями по охране труда и технике безопасности, а также противопожарной безопасности, актуальными для работы в компании.

2 ОСНОВНОЙ ЭТАП

2.1 Установка PostgreSQL

После получения индивидуального задания, необходимо было установить на свой персональный компьютер PostgreSQL сервер, представляющее из себя систему управления базами данных. С помощью PostgreSQL можно создавать, хранить базы данных и работать с данными с помощью запросов на языке SQL.

Для установки PostgreSQL необходимо заранее скачать с сайта <https://www.enterprisedb.com/downloads/postgres-postgresql-downloads> подходящую для персонального компьютера версию. После завершения скачивания, запускаем и проходим установку программного продукта, выбираем путь, где будет установлено ПО. Также для дальнейшей работы необходимо придумать пароль и порт, который установлен по умолчанию как 5432. Также необходимо выбрать необходимые для работы версию и драйвера. Эти данные представлены на рисунке 1 и 2 соответственно.

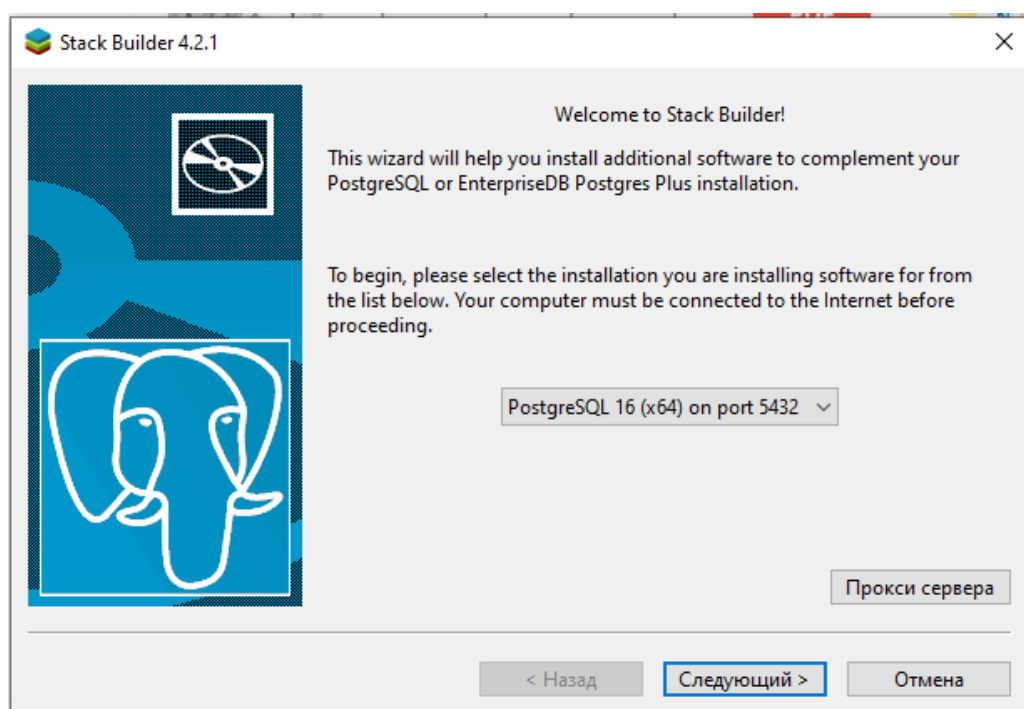


Рисунок 1 – Выбор версии для установки PostgreSQL

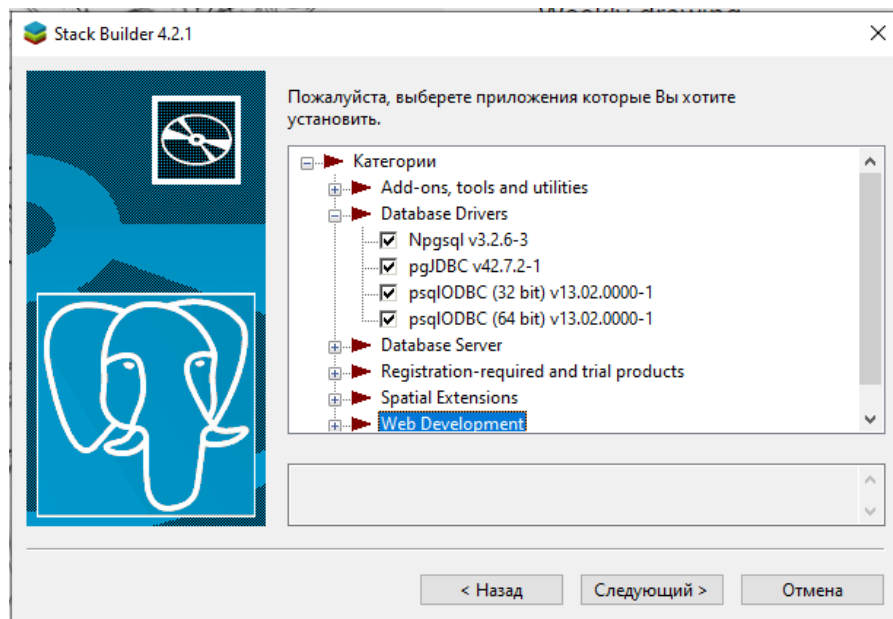


Рисунок 2 – Выбор драйверов для PostgreSQL

После необходимо выбрать директорию для загрузки выбранных пакетов и после их установки перезагрузить персональный компьютер. Проверить работу сервера можно в диспетчере задач, как представлено на рисунке 3.

> pg_ctl - starts/stops/restarts the...	0%	0,9 МБ	0 МБ/с	0 Мбит/с
PostgreSQL Server	0%	2,1 МБ	0 МБ/с	0 Мбит/с
PostgreSQL Server	0%	1,8 МБ	0 МБ/с	0 Мбит/с
PostgreSQL Server	0%	1,7 МБ	0 МБ/с	0 Мбит/с
PostgreSQL Server	0%	2,8 МБ	0 МБ/с	0 Мбит/с
PostgreSQL Server	0%	2,0 МБ	0 МБ/с	0 Мбит/с
PostgreSQL Server	0%	1,8 МБ	0 МБ/с	0 Мбит/с
PostgreSQL Server	0%	2,9 МБ	0 МБ/с	0 Мбит/с

Рисунок 3 – Проверка работы PostgreSQL

После установки и проверки работы сервера можно начинать установку программного продукта DBeaver.

2.2 Установка и подготовка программного продукта DBeaver

Для предоставления результатов исследования нужно установить нужное программное обеспечение, в нашем случае это DBeaver.

DBeaver – это универсальный и многофункциональный клиентом для работы с базами данных, который поддерживает все популярные реляционные и нереляционные базы данных, такие

как MySQL, PostgreSQL, SQL Server, DB2 и многие другие. Он предоставляет удобный графический интерфейс, мощные возможности для написания SQL-запросов, инструменты для визуализации данных и доступен как в платной версии (DBeaver PRO), так и в бесплатной версии (DBeaver Community). Это отличный выбор для специалистов, работающих с базами данных, из-за широкого функционала, удобства использования и доступности.

Для начала заходим на сайт dbeaver.io и нажимаем кнопку «Download» как выделенно на рисунке 4.

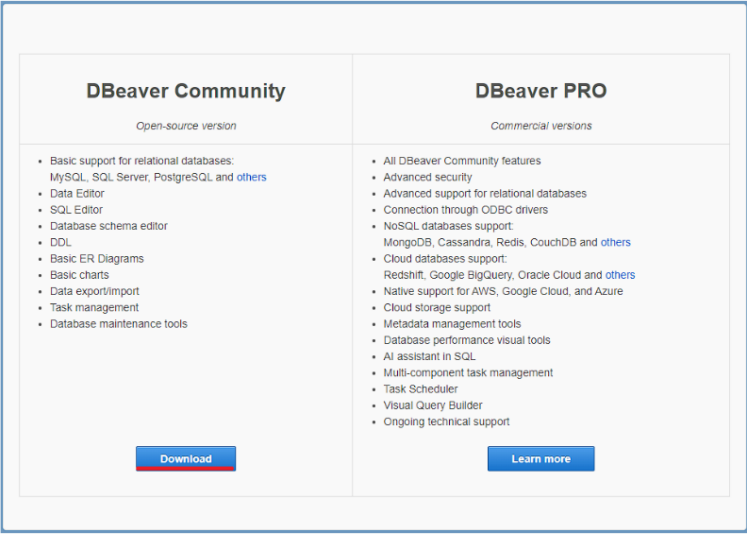


Рисунок 4 – Скачивание ПО (шаг 1)

Мы попадаем на сайт следующую вкладку, где нам надо нажать «Windows Installer» и скачать пакет ПО, как выделенно на рисунке 5.

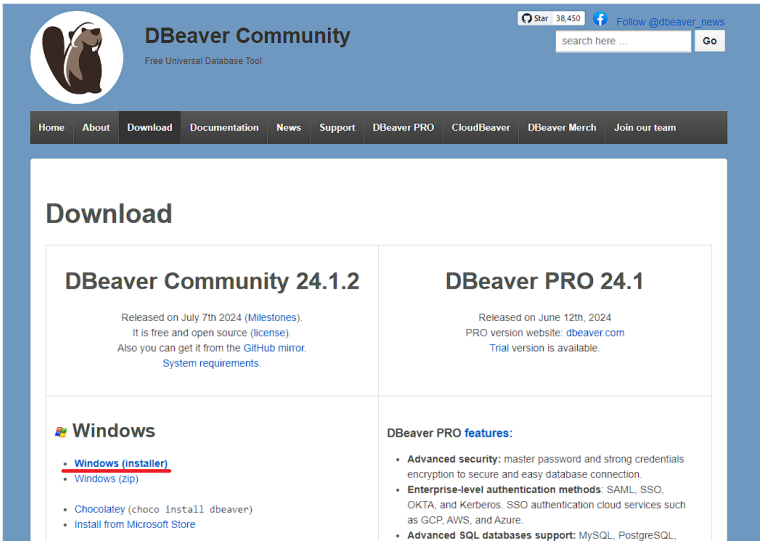


Рисунок 5- Скачивание ПО (шаг 2)

После завершения скачивания, запускаем и проходим установку программного продукта, выбираем путь, где будет установлено ПО, как показано на рисунке 6.

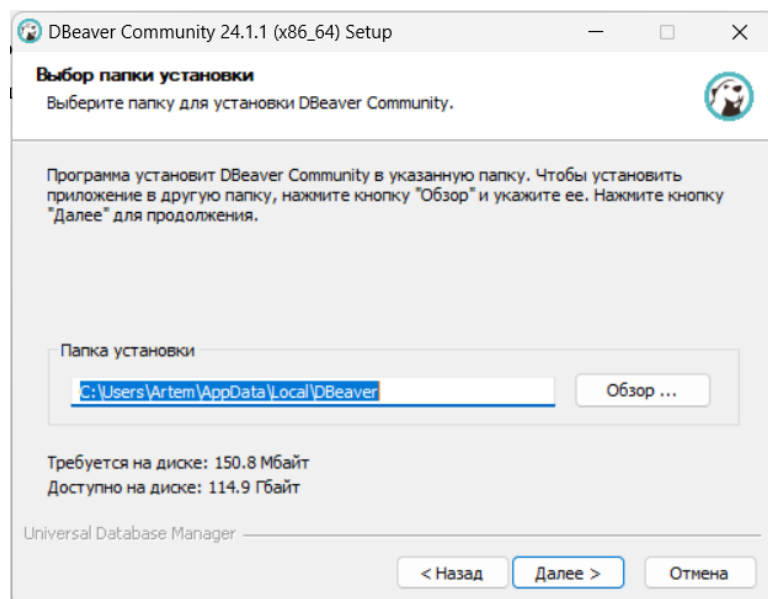


Рисунок 6 – Выбор репозитория ПО

После установки программного продукта необходимо его подключить к серверу SQL. Для этого необходимо в меню «База данных» выбрать «Новое соединение». После нажатия открывается меню, содержащее типы соединения для базы данных. После выполнения ручного поиска для добавления PostgreSQL, как показано на рисунке 7, отображенного цифрой «1». Когда тип соединения выбран, нажимается кнопка «Далее». Эти данные обозначены цифрой «2» на рисунке 7.

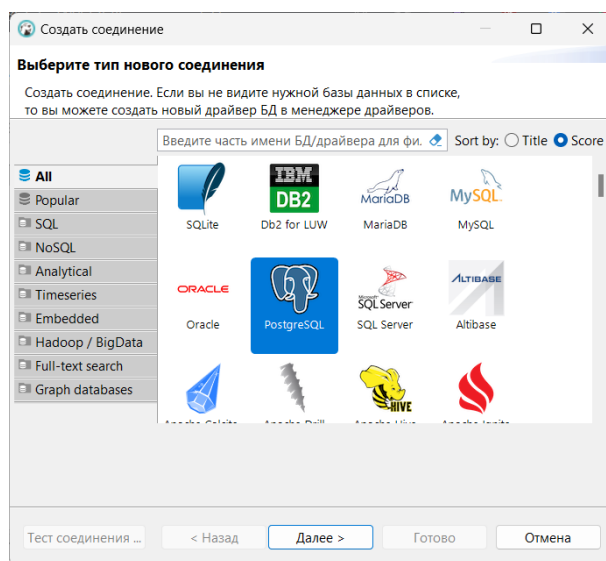


Рисунок 7 – Добавление PostgreSQL

После необходимо ввести хост, порт, логин и пароль пользователя, которые были введены при установке PostgreSQL. Данные с обозначены цифрами «1», «2» и «3» обведены на рисунке 8. А также при в разделе «Свойства драйвера» при необходимости скачать предлагаемые драйвера.

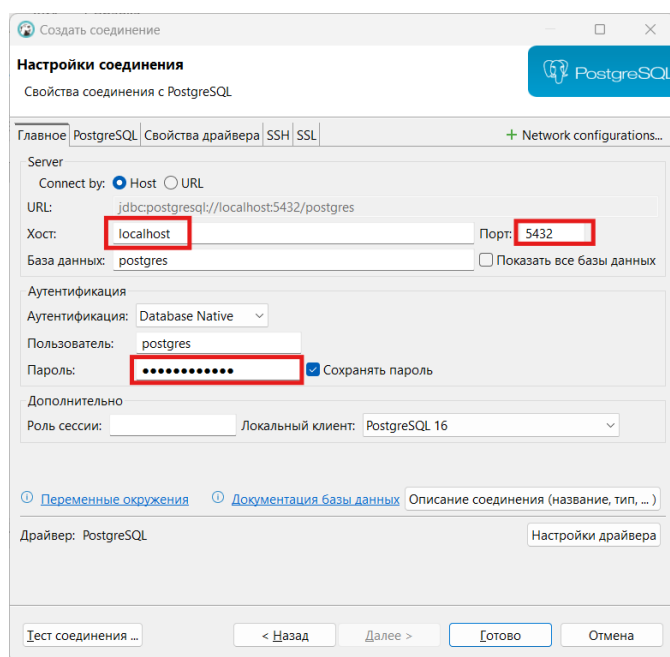


Рисунок 8 – настройка PostgreSQL

После подключения сервера необходимо восстановить базу данных из файла «dvd-rental.backup». Чтобы выбрать необходимую базу данных, нужно раскрыть сервер PostgreSQL в области «Проекты», затем раскрыть папку «General» и выбрать нужную базу данных «postgres» правой кнопкой мыши. В контекстном меню выбираем необходимое действие «Восстановить» для подключения к базе данных, как показано на рисунке 9.

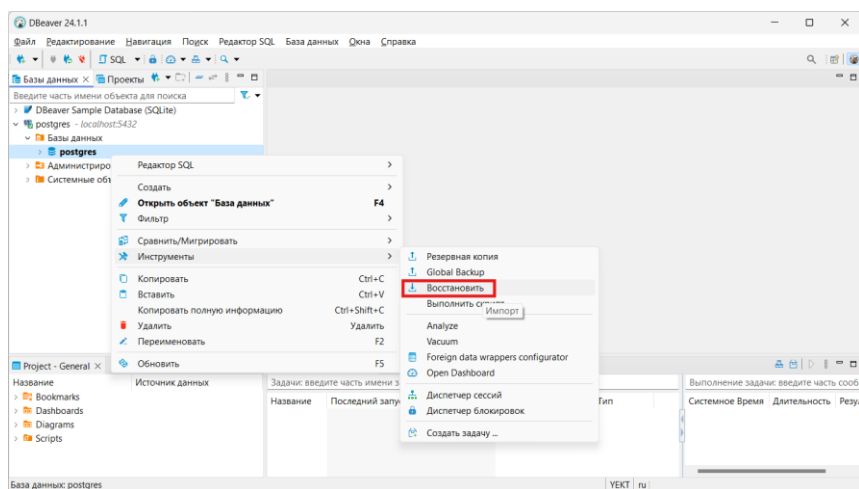


Рисунок 9 – Восстановление базы данных

После необходимо ввести выбрать нужный файл и нажать кнопку «Старт», как показано на рисунке 10. В итоге после восстановления базы данных, она открывается в виде диаграммы, где видны таблицы и их связи, как представлено на рисунке 11.

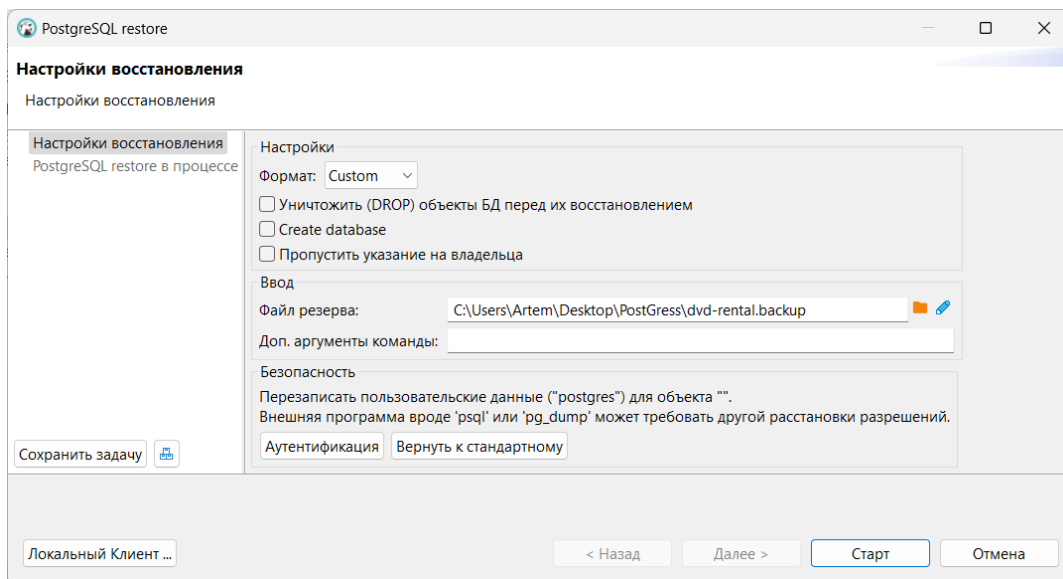


Рисунок 10 – Выбор файла «dvd-rental.backup»

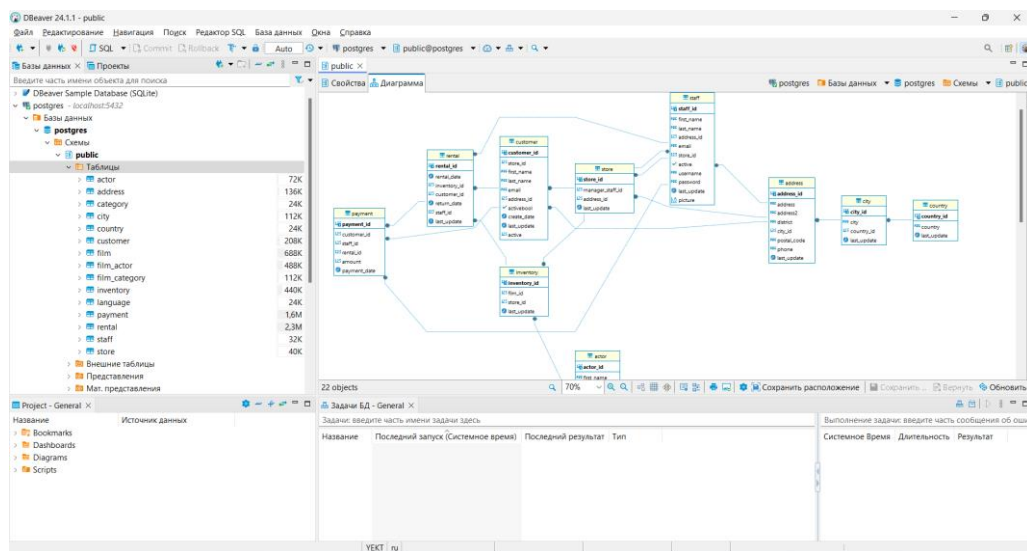


Рисунок 11 – Восстановленная диаграмма

После установки программного обеспечения можно приступить к выполнению.

3 ОПИСАНИЕ SQL-ЗАПРОСОВ

3.1 Выполнение «Задания 1»

В рамках данного задания необходимо было продемонстрировать умение писать простые запросы типа SELECT, задавать названия колонкам, навыки фильтрации и сортировки строк в таблицах с использованием основных операторов языка SQL, а также навыки выполнения преобразования текстовых, числовых значений и дат с помощью функций языка SQL по работе со строками, датами и числами.

Для выполнения заданий в этом блоке необходимо было использовать следующие команды и операторы:

- select – позволяет взять данные из таблицы;
- distinct – указывает на то, что должны быть выведены только уникальные значения;
- as – используется для переименования столбца или таблицы с псевдонимами и создания вычисляемых колонок;
- from – указывает на то из какой таблицы берутся данные;
- order by – сортирует числа по возрастанию и строки по алфавиту;
- where – позволяет задать условия для отбора данных;
- like – позволяет задать условие для нахождения «похожих» данных;
- not – позволяет инвертировать условие;
- between – используется для выбора данных в заданном диапазоне;
- and и or – логический оператор, позволяющий совмещать условия;
- order by desc – используется для сортировки по убыванию;
- limit – задает максимальное количество строк, которые нужно вывести;
- concat_ws – соединят поля столбцов с разделителем;
- char_length – позволяет подсчитать количества символов в строке;
- cast – используется для приведения типов данных;
- lower – приводит символы в строке к нижнему регистру;
- split_part – позволяет разделить строку на части с помощью разделителя и вернуть нужную часть.

Все SQL-запросы, а также результаты из выполнения представлены в приложении А.

3.2 Выполнение «Задания 2»

В рамках выполнения данной группы заданий была поставлена цель продемонстрировать знания по функциям агрегации и группировке строк, умение фильтровать сгруппированные строки, а также навыки использования методов соединения таблиц с помощью разных вариаций JOIN.

Для выполнения заданий в этом блоке необходимо было использовать следующие команды и операторы:

- select, distinct, as, from, order by, order by desc, limit, where – изученные в предыдущем задании;
- left join – предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию и дополняются записями из левой таблицы, даже если они не соответствуют условию;
- right join – предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию и дополняются записями из правой таблицы, даже если они не соответствуют условию;
- inner join – предназначен для соединения таблиц и вывода результирующей таблицы, в которой данные полностью пересекаются по условию;
- cross join – реализует операцию декартова произведения в реляционной алгебре;
- round – округляет значения до определенного знака;
- sum() – возвращает общую сумму числового столбца;
- min() – возвращает минимальное значение в столбце;
- max() – возвращает максимальное значение в столбце;
- group by – группирует строки с одинаковым значением в сводные строки;
- having – используется для фильтрации результата с оператором group by;
- count() – возвращает количество строк, соответствующее заданному критерию;
- case when условие then *результат1* else *результат* end – возвращает условие, которое выполняется.

Все SQL-запросы, а также результаты из выполнения представлены в приложении Б.

3.3 Выполнение «Задания 3»

В рамках выполнения данной группы заданий была поставлена цель продемонстрировать знания в работе с оконными функциями.

Оконные функции – это особый тип функций, которые вычисляют значение для каждой строки в результирующем наборе, основанном на группе связанных строк.

Для выполнения заданий в этом блоке необходимо было использовать следующие функции:

- select, distinct, as, from, order by, order by desc, limit, where, inner join и др. – изученные в предыдущем задании;
- row_number() – присваивает уникальный номер каждой строке в определенном наборе;
- dense_rank() – функция возвращает ранг каждой строки. Но в отличие от функции rank(), она для одинаковых значений возвращает ранг, не пропуская следующий;
- over – определяет окно или определяемый пользователем набор строк внутри результирующего набора запроса;
- partition by – разделяет результирующий набор запроса на секции;
- unbounded preceding – указывает, что окно начинается с первой строки группы;
- lag() – обращается к данным из предыдущей строки окна;
- last_value() – выводит последнее значение в определенном наборе;
- unbounded following – указать, что окно заканчивается на последней строке группы.
- date() – возвращает только дату без времени
- with – временный результирующий набор данных, к которому можно обращаться в последующих запросах.

Все SQL-запросы, а также результаты из выполнения представлены в приложении В.

3.4 Выполнение «Задания 4»

Целью данного задания было продемонстрировать навыки работы с различными инструментами SQL для эффективного анализа данных. В частности, задание включало в себя работу с массивами, подзапросами, табличными выражениями, материализованными представлениями и анализом производительности запросов.

Для выполнения заданий в этом блоке необходимо было использовать следующие команды и операторы:

- select, distinct, as, from, order by, order by desc, limit, where, inner join и др. – изученные в предыдущем задании;
- any – возвращает значение true, если любое из значений подчиненного запроса удовлетворяет условию;
- all – возвращает значение true, если все значения подчиненного запроса удовлетворяют условию.

Все SQL-запросы, а также результаты из выполнения представлены в приложении Г.

4 ОТЧЕТНЫЙ ЭТАП

На отчетном этапе мы заключили нашу практику, представив отчет о проделанной работе. Этот этап включал подготовку структурированного отчета, в котором мы анализировали полученные результаты. После этого мы подготовились к защите отчета, где представили его перед научным руководителем. В результате обсуждения мы получили обратную связь и дальнейшие рекомендации для улучшения нашей работы.

ЗАКЛЮЧЕНИЕ

В ходе практики были изучены нотации SQL и разработаны SQL-запросы для выполнения индивидуального задания. Были установлены PostgreSQL сервер и программное обеспечение DBeaver, настроено подключение к серверу SQL и восстановлена база данных из файла «dvd-rental.backup». В результате прохождения практики были получены практические навыки работы с SQL, PostgreSQL и DBeaver.

Также стоит отметить, что мы улучшили наши навыки сбора информации и коммуникации, впервые начали проходить практику в предприятии. Трудности проявились в начале, после получения задания, так как до практики с базами данных мы ещё не работали.

Представленные запросы были успешно защищены на итоговой презентации. Проведение публичной защиты наших выводов дало нам возможность убедиться в глубоком понимании изученного материала и уверенно выступить перед аудиторией.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Используем все возможности индексов в PostgreSQL // Хабр. [Электронный ресурс]. Режим доступа: <https://habr.com/ru/companies/vk/articles/453046/> (Дата обращения: 12.07.2024).
2. Массивы и Списки в SQL Server // Interface. [Электронный ресурс]. Режим доступа: <https://www.interface.ru/home.asp?artId=22765> (Дата обращения: 12.07.2024).
3. Обобщённое табличное выражение, оператор WITH // SQL Academy. [Электронный ресурс]. Режим доступа: <https://sql-academy.org/ru/guide/operator-with> (Дата обращения: 07.07.2024).
4. Оконные функции SQL // SQL Academy. [Электронный ресурс]. Режим доступа: <https://sql-academy.org/ru/guide/windows-functions> (Дата обращения: 12.07.2024).
5. Оконные функции SQL простым языком с примерами // Хабр. [Электронный ресурс]. Режим доступа: <https://habr.com/ru/articles/664000/> (Дата обращения: 12.07.2024).
6. Подробная Шпаргалка SQL на 2023 год // Uproger.com [Электронный ресурс]. Режим доступа: <https://uproger.com/shpargalki-sql-2023/> (Дата обращения: 12.07.2024).
7. Справочник SQL // Code.mu. [Электронный ресурс]. Режим доступа: <https://code.mu/ru/sql/manual/> (Дата обращения: 12.07.2024).
8. Учимся применять оконные функции // This is data. [Электронный ресурс]. Режим доступа: <https://thisisdata.ru/blog/uchimsya-primenyat-okonnyye-funktsii/> (Дата обращения: 12.07.2024).
9. SQL JOIN // SchoolsW3. [Электронный ресурс]. Режим доступа: https://www.schoolsw3.com/sql/sql_join.php (Дата обращения: 12.07.2024).
10. SQL JOIN: типы и примеры // TProger. [Электронный ресурс]. Режим доступа: <https://tproger.ru/articles/sql-join> (Дата обращения: 12.07.2024).

ПРИЛОЖЕНИЯ

Приложение А Результат выполнения «Задания 1»

--Задание 1. Выведите уникальные названия городов из таблицы городов

```
select distinct city from city c order by city;
```

city 1 ×

select distinct city from city c order by city Введите SQL выражение чтобы отфильтровать

Таблица	ABC city
1	A Corua (La Corua)
2	Abha
3	Abu Dhabi
4	Acua
5	Adana
6	Addis Abeba

Рисунок А. 1 – Результат выполнения задания 1

/*Задание 2. Доработайте запрос из предыдущего задания,
* чтобы запрос выводил только те города, названия которых начинаются на "L" и заканчиваются на "a",
* и названия не содержат пробелов.*/

```
select distinct city from city c where city like 'L%a' and city not like '% %' order by city;
```

city 1 ×

select distinct city from city c where city like 'L%a' and city not like '% %' Введите SQL выражение чтобы отфильтровать результаты

Таблица	ABC city
1	Liepaja
2	Lima
3	Loja
4	Luzinia

Рисунок А. 2 – Результат выполнения задания 2

/*Задание 3. Получите из таблицы платежей за прокат фильмов информацию по платежам,
* которые выполнялись в промежуток с 17 июня 2005 года по 19 июня 2005 года
* включительно и стоимость которых превышает 1.00. Платежи нужно отсортировать по дате платежа*/

```
select * from payment p where payment_date between '2005-06-17' and '2005-06-20' and amount > 1.00  
order by payment_date ;  
select * from payment p;
```

payment 1 ×

select * from payment p where payment_date between '2005-06-17' and '2005-06-20' Введите SQL выражение чтобы отфильтровать результаты

Таблица	123 payment_id	123 customer_id	123 staff_id	123 rental_id	123 amount	payment_date
1	9 809	363	1	1 847	4,99	2005-06-17 00:05:22.000
2	9 218	341	1	1 849	7,99	2005-06-17 00:13:19.000
3	13 917	517	1	1 850	4,99	2005-06-17 00:31:35.000
4	7 330	271	2	1 852	2,99	2005-06-17 00:38:20.000
5	1 015	37	2	1 854	3,99	2005-06-17 00:43:57.000
6	13 262	492	2	1 855	4,99	2005-06-17 00:54:58.000
7	4 378	161	1	1 856	2,99	2005-06-17 01:02:00.000

Рисунок А. 3 – Результат выполнения задания 3

--Задание 4. Выведите информацию о 10-ти последних платежах за прокат фильмов

```
select * from payment p order by payment_date desc limit 10;
```

payment 1 x

select * from payment p order by payment_date desc limit 10

	123 payment_id	123 customer_id	123 staff_id	123 rental_id	123 amount	payment_date
1	630	23	2	15 532	2,99	2006-02-14 15:16:03.000
2	302	11	1	11 646	0,99	2006-02-14 15:16:03.000
3	600	22	1	12 222	4,99	2006-02-14 15:16:03.000
4	417	15	2	13 968	0	2006-02-14 15:16:03.000
5	253	9	1	15 813	4,99	2006-02-14 15:16:03.000
6	145	5	2	13 209	0,99	2006-02-14 15:16:03.000
7	416	15	1	13 798	3,98	2006-02-14 15:16:03.000
8	578	21	1	14 933	2,99	2006-02-14 15:16:03.000
9	385	14	1	13 780	4,99	2006-02-14 15:16:03.000
10	781	28	2	12 938	2,99	2006-02-14 15:16:03.000

Рисунок А. 4 – Результат выполнения задания 4

/* Задание 5. Выведите следующую информацию по покупателям:

- Фамилия и имя (в одной колонке через пробел)
- Электронная почта
- Длину значения поля email
- Дату последнего обновления записи о покупателе (без времени)

Каждой колонке задайте наименование на русском языке.

```
select
concat_ws(' ',first_name,last_name) as Имя_покупателя,
email as Электронная_почта,
char_length(email) as Длина_почты,
cast(last_update as date) as Дата_последнего_обновления_записи
from customer c ;
```

customer 1 x

select concat_ws(' ',first_name,last_name) as Имя_покупателя, email as Электронная_почта, char_length(email) as Длина_почты, cast(last_update as date) as Дата_последнего_обновления_записи

	Имя_покупателя	Электронная_почта	123 Длина_почты	Дата_последнего_обновления_записи
1	MARY SMITH	MARY.SMITH@sakilacustomer.org	29	2006-02-15
2	PATRICIA JOHNSON	PATRICIA.JOHNSON@sakilacustomer.org	35	2006-02-15
3	LINDA WILLIAMS	LINDA.WILLIAMS@sakilacustomer.org	33	2006-02-15
4	BARBARA JONES	BARBARA.JONES@sakilacustomer.org	32	2006-02-15
5	ELIZABETH BROWN	ELIZABETH.BROWN@sakilacustomer.org	34	2006-02-15
6	JENNIFER DAVIS	JENNIFER.DAVIS@sakilacustomer.org	33	2006-02-15
7	MARIA MILLER	MARIA.MILLER@sakilacustomer.org	31	2006-02-15

Рисунок А. 5 – Результат выполнения задания 5

/*Задание 6. Выведите одним запросом только активных покупателей, имена которых KELLY или WILLIE.

* Все буквы в фамилии и имени из верхнего регистра должны быть переведены в нижний регистр*/

```
select
lower(concat_ws(' ',first_name,last_name))
from customer c
where (first_name = 'KELLY' or first_name = 'WILLIE') and activebool = true;
```

Результат 1 x

select lower(concat_ws(' ',first_name,last_name)) from customer c

	lower
1	kelly torres
2	willie howell
3	willie markham
4	kelly knott

Рисунок А. 6 – Результат выполнения задания 6

/*Задание 7. Выведите одним запросом информацию о фильмах, у которых рейтинг "R" и стоимость аренды указана от 0.00 до 3.00 включительно, а также фильмы с рейтингом "PG-13" и стоимостью аренды больше или равной 4.00*/

```
select * from film f where (rating = 'R' and rental_rate between 0.00 and 3.00) or (rating = 'PG-13' and rental_rate >= 4.00);
```

film_id	title	description	release_year	language_id	original_language_id	rental_duration	rental_rate	length	replacement_cost	rating	last_update
7	AIRPLANE SIERRA	A Touching Saga of a Hunter An	2 006	1 07	[NULL]	6	4.99	62	28.99	PG-13	2006-02-15 05:03:42.000
17	ALONE TRIP	A Fast-Paced Character Study of	2 006	1 07	[NULL]	3	0.99	82	14.99	R	2006-02-15 05:03:42.000
23	ANACONDA CONFESS	A Lackluster Display of a Dentis	2 006	1 07	[NULL]	3	0.99	92	9.99	R	2006-02-15 05:03:42.000
24	ANALYZE HOOSIERS	A Thoughtful Display of a Explo	2 006	1 07	[NULL]	6	2.99	181	19.99	R	2006-02-15 05:03:42.000
28	ANTHEM LUKE	A Touching Panorama of a Wait	2 006	1 07	[NULL]	5	4.99	91	16.99	PG-13	2006-02-15 05:03:42.000
30	ANYTHING SAVANNAH	A Epic Story of a Pastry Chef An	2 006	1 07	[NULL]	4	2.99	82	27.99	R	2006-02-15 05:03:42.000
40	ARMY FLINTSTONES	A Boring Saga of a Database Ad	2 006	1 07	[NULL]	4	0.99	148	22.99	R	2006-02-15 05:03:42.000
44	ATTACKS HATE	A Fast-Paced Panorama of a Tec	2 006	1 07	[NULL]	5	4.99	113	21.99	PG-13	2006-02-15 05:03:42.000
45	ATTRACTION NEWTON	A Astounding Panorama of a Co	2 006	1 07	[NULL]	5	4.99	83	14.99	PG-13	2006-02-15 05:03:42.000
48	BACKLASH UNDEFEAT	A Stunning Character Study of a	2 006	1 07	[NULL]	3	4.99	118	24.99	PG-13	2006-02-15 05:03:42.000
49	BADMAN DAWN	A Emotional Panorama of a Pior	2 006	1 07	[NULL]	6	2.99	162	22.99	R	2006-02-15 05:03:42.000

Рисунок А. 7 – Результат выполнения задания 7

/*Задание 8. Получите информацию о трёх фильмах с самым длинным описанием фильма

```
select * from film f order by char_length(description) desc limit 3 ;
```

film_id	title	description	release_year	language_id	original_language_id	rental_duration	rental_rate	length	replacement_cost	rating	last_update
217	DAZED PUNK	A Action-Packed Story of a Pion	2 006	1 07	[NULL]	6	4.99	120	20.99	G	2006-02-15 05:03:42.000
116	CANDIDATE PERDITION	A Brilliant Epistle of a Composer	2 006	1 07	[NULL]	4	2.99	70	10.99	R	2006-02-15 05:03:42.000
274	EGG IGBY	A Beautiful Documentary of a B	2 006	1 07	[NULL]	4	2.99	67	20.99	PG	2006-02-15 05:03:42.000

Рисунок А. 8 – Результат выполнения задания 8

/*Задание 9. Выведите Email каждого покупателя, разделив значение Email на 2 отдельных колонки:

- в первой колонке должно быть значение, указанное до @,
- во второй колонке должно быть значение, указанное после @

```
select
split_part(email, '@', 1) as Пользователь,
split_part(email, '@', 2) as Домен
from customer c ;
```

Пользователь	Домен
MARY.SMITH	sakilacustomer.org
PATRICIA.JOHNSON	sakilacustomer.org
LINDA.WILLIAMS	sakilacustomer.org
BARBARA.JONES	sakilacustomer.org
ELIZABETH.BROWN	sakilacustomer.org
JENNIFER.DAVIS	sakilacustomer.org
MARIA.MILLER	sakilacustomer.org
SUSAN.WILSON	sakilacustomer.org
MARGARET.MOORE	sakilacustomer.org
DOROTHY.TAYLOR	sakilacustomer.org
LISA.ANDERSON	sakilacustomer.org

Рисунок А. 9 – Результат выполнения задания 9

/*Задание 10. Доработайте запрос из предыдущего задания, скорректируйте значения в новых колонках:
 * первая буква должна быть заглавной, остальные строчными.*/

```

select initcap(split_part(email, '@', 1)) as Пользователь,
initcap(split_part(email, '@', 2)) as Домен
from customer c ; --Тут слова после точек тоже с большой пишутся
select
upper(substring(split_part(email, '@', 1) from 1 for 1)) ||
lower(substring(split_part(email, '@', 1) from 2)) as Пользователь,
upper(substring(split_part(email, '@', 2) from 1 for 1)) ||
lower(substring(split_part(email, '@', 2) from 2)) as Домен
from customer c ; --Тут правильно но громоздко

```

Результат 1 ×

select initcap(split_part(email, '@', 1)) as Пользователь, initcap(split_part(email, '@', 2)) as Домен

	Пользователь	Домен
1	Mary.Smith	Sakilacustomer.Org
2	Patricia.Johnson	Sakilacustomer.Org
3	Linda.Williams	Sakilacustomer.Org
4	Barbara.Jones	Sakilacustomer.Org
5	Elizabeth.Brown	Sakilacustomer.Org
6	Jennifer.Davis	Sakilacustomer.Org
7	Maria.Miller	Sakilacustomer.Org
8	Susan.Wilson	Sakilacustomer.Org
9	Margaret.Moore	Sakilacustomer.Org
10	Dorothy.Taylor	Sakilacustomer.Org
11	Lisa.Anderson	Sakilacustomer.Org

Рисунок А. 10 – Результат выполнения задания 10

Приложение Б

Результат выполнения «Задания 2»

--Задание 1. Выведите для каждого покупателя его адрес, город и страну проживания.
 select c3.country as Страна, c2.city as Город, a.address as Адрес, c.first_name || ' ' || c.last_name as Имя_покупателя from customer c
 inner join address a on a.address_id = c.address_id
 inner join city c2 on c2.city_id = a.city_id
 inner join country c3 on c3.country_id = c2.country_id ;

country(+)	Страна	Город	Адрес	Имя_покупателя
1	Japan	Sasebo	1913 Hanoi Way	MARY SMITH
2	United States	San Bernardino	1121 Loja Avenue	PATRICIA JOHNSON
3	Greece	Athenai	692 Joliet Street	LINDA WILLIAMS
4	Myanmar	Myingyan	1566 Inegl Manor	BARBARA JONES
5	Taiwan	Nantou	53 Idfu Parkway	ELIZABETH BROWN
6	United States	Laredo	1795 Santiago de Compostela Way	JENNIFER DAVIS
7	Yugoslavia	Kragujevac	900 Santiago de Compostela Parkway	MARIA MILLER
8	New Zealand	Hamilton	478 Joliet Way	SUSAN WILSON
9	Oman	Masqat	613 Korolev Drive	MARGARET MOORE
10	Iran	Esfahan	1531 Sal Drive	DOROTHY TAYLOR
11	Japan	Sagamihara	1542 Tarlac Parkway	LISA ANDERSON
12	India	Yamuna Nagar	808 Bhopal Manor	NANCY THOMAS
13	Turkey	Osmaniye	270 Amroha Parkway	KAREN JACKSON

Рисунок Б. 1 – Результат выполнения задания 1

/*Задание 2. С помощью SQL-запроса посчитайте для каждого магазина количество его покупателей.
 • Доработайте запрос и выведите только те магазины, у которых количество покупателей больше 300. Для решения используйте фильтрацию по сгруппированным строкам с функцией агрегации.
 • Доработайте запрос, добавив в него информацию о городе магазина, фамилии и имени продавца, который работает в нём.

```

select store_id as Магазин, count(*) as Количество_покупателей from customer c
group by c.store_id;

select store_id as Магазин, count(*) as Количество_покупателей from customer c
group by c.store_id
having count(*) > 300;

select c2.city as Город_магазина, s2.first_name as Имя_продавца, s2.last_name as Фамилия_продавца, s.store_id as Магазин, count(*) as Количество_покупателей from customer c
inner join store s on s.store_id = c.store_id
inner join address a on a.address_id = s.address_id
inner join city c2 on c2.city_id = a.city_id
inner join staff s2 on s2.store_id = s.store_id
group by c2.city, s2.first_name, s2.last_name, s.store_id
having count(*) > 300;
  
```

Магазин	Количество_покупателей
1	326
2	273

Рисунок Б. 2 – Результат выполнения задания 2

/*Задание 2. С помощью SQL-запроса посчитайте для каждого магазина количество его покупателей.
 • Доработайте запрос и выведите только те магазины, у которых количество покупателей больше 300. Для решения используйте фильтрацию по сгруппированным строкам с функцией агрегации.
 • Доработайте запрос, добавив в него информацию о городе магазина, фамилии и имени продавца, который работает в нём.

```

select store_id as Магазин, count(*) as Количество_покупателей from customer c
group by c.store_id;

select store_id as Магазин, count(*) as Количество_покупателей from customer c
group by c.store_id
having count(*) > 300;

select c2.city as Город_магазина, s2.first_name as Имя_продавца, s2.last_name as Фамилия_продавца, s.store_id as Магазин, count(*) as Количество_покупателей from customer c
inner join store s on s.store_id = c.store_id
inner join address a on a.address_id = s.address_id
inner join city c2 on c2.city_id = a.city_id
inner join staff s2 on s2.store_id = s.store_id
group by c2.city, s2.first_name, s2.last_name, s.store_id
having count(*) > 300;
  
```

Магазин	Количество_покупателей
1	326

Рисунок Б. 3 – Результат выполнения задания 2

/*Задание 2. С помощью SQL-запроса посчитайте для каждого магазина количество его покупателей.

- Доработайте запрос и выведите только те магазины, у которых количество покупателей больше 300. Для решения используйте фильтрацию по сгруппированным строкам с функцией агрегации.
- Доработайте запрос, добавив в него информацию о городе магазина, фамилии и имени продавца, который работает в нём.

```

select store_id as Магазин, count(*) as Количество_покупателей from customer c
group by c.store_id;

select store_id as Магазин, count(*) as Количество_покупателей from customer c
group by c.store_id
having count(*) > 300;

select c2.city as Город_магазина, s2.first_name as Имя_продавца, s2.last_name as Фамилия_продавца, s.store_id as Магазин, count(*) as Количество_покупателей from customer c
inner join store s on s.store_id = c.store_id
inner join address a on a.address_id = s.address_id
inner join city c2 on c2.city_id = a.city_id
inner join staff s2 on s2.store_id = s.store_id
group by c2.city, s2.first_name, s2.last_name, s.store_id
having count(*) > 300;

```

city(+) 1 x

select c2.city as Город_магазина, s2.first_name as Имя_продавца, s2.last_name as Фамилия_продавца, s.store_id as Магазин, count(*) as Количество_покупателей

Город_магазина	Имя_продавца	Фамилия_продавца	Магазин	Количество_покупателей
Lethbridge	Mike	Hillyer	1	326

Рисунок Б. 4 – Результат выполнения задания 2

--Задание 3. Выведите топ-5 покупателей, которые взяли в аренду за всё время наибольшее количество фильмов

```

select customer_id as Покупатель, count(payment_id) as Количество_аренд from payment p
group by customer_id
order by count(payment_id) desc limit 5;

```

payment 1 x

select customer_id as Покупатель, count(payment_id) as Количество_аренд

Покупатель	Количество_аренд
148	46
526	45
236	42
144	42
75	41

Рисунок Б. 5 – Результат выполнения задания 3

/*Задание 4. Посчитайте для каждого покупателя 4 аналитических показателя:

- количество взятых в аренду фильмов;
- общую стоимость платежей за аренду всех фильмов (значение округлите до целого числа);
- минимальное значение платежа за аренду фильма;
- максимальное значение платежа за аренду фильма.

```

select
customer_id as Покупатель,
count(rental_id) as Количество_взятых_фильмов,
round(sum(amount)) as Общая_стоимость_платежей,
min(amount) as Минимальная_стоимость_платежа,
max(amount) as Максимальная_стоимость_платежа
from payment p
group by customer_id
order by customer_id ;

```

payment 1 x

select customer_id as Покупатель, count(rental_id) as Количество_взятых_фильмов, round(sum(amount)) as Общая_стоимость_платежей, min(amount) as Минимальная_стоимость_платежа, max(amount) as Максимальная_стоимость_платежа

Покупатель	Количество_взятых_фильмов	Общая_стоимость_платежей	Минимальная_стоимость_платежа	Максимальная_стоимость_платежа
1	32	119	0,99	9,99
2	27	129	0,99	10,99
3	26	136	0,99	10,99
4	22	82	0,99	8,99
5	38	145	0,99	9,99
6	28	94	0,99	7,99
7	33	152	0,99	8,99
8	24	93	0,99	9,99
9	23	90	0,99	7,99
10	25	100	0,99	8,99
11	24	107	0,99	9,99

Рисунок Б. 6 – Результат выполнения задания 4

*/Задание 5. Используя данные из таблицы городов, составьте одним запросом всевозможные пары городов так, чтобы в результате не было пар с одинаковыми названиями городов.
* Для решения необходимо использовать декартово произведение*/

```
select c.city as Первый_город, c2.city as Второй_город from city c
cross join city c2
where c.city != c2.city; --Убираем повторяющиеся названия, но пары А Б и Б А также останутся

select c.city, c2.city from city c, city c2 where c.city < c2.city; --Вроде как надо
```

city 1 ×

select c.city, c2.city from city c, city c2 where c.city < c2.city

	city	city
1	A Corua (La Corua)	Abha
2	A Corua (La Corua)	Abu Dhabi
3	A Corua (La Corua)	Acua
4	A Corua (La Corua)	Adana
5	A Corua (La Corua)	Addis Abeba
6	A Corua (La Corua)	Aden
7	A Corua (La Corua)	Adoni
8	A Corua (La Corua)	Ahmadnagar
9	A Corua (La Corua)	Akishima
10	A Corua (La Corua)	Akron

Рисунок Б. 7 – Результат выполнения задания 5

/Задание 6. Используя данные из таблицы rental о дате выдачи фильма в аренду (поле rental_date) и дате возврата (поле return_date), вычислите для каждого покупателя среднее количество дней, за которые он возвращает фильмы/

```
select customer_id as Покупатель, round(avg(date(return_date)-date(rental_date)),2) as Средняя_разница_в_днях from rental r
group by customer_id
order by customer_id ;
```

rental 1 ×

select customer_id as Покупатель, round(avg(date(return_date)-date(rental_date)),2) as Средняя_разница_в_днях

	Покупатель	Средняя_разница_в_днях
1	1	4,47
2	2	5,52
3	3	5,88
4	4	3,86
5	5	5,03
6	6	5,43
7	7	5,61
8	8	4,54
9	9	4,55
10	10	6
11	11	5,7
12	12	5,29
13	13	5,04
14	14	5,11

Рисунок Б. 8 – Результат выполнения задания 6

/Задание 7. Посчитайте для каждого фильма, сколько раз его брали в аренду, а также общую стоимость аренды фильма за всё время/

```
select f.title as Название_фильма, count(r.rental_id) as Сколько_раз_брали_в_аренду, sum(coalesce(p.amount,0)) as Общая_стоимость_аренды from rental r
left join inventory i on i.inventory_id = r.inventory_id
right join film f on f.film_id = i.film_id --right потому что может оказаться что фильм никто не арендовал
left join payment p on p.rental_id = r.rental_id
group by f.film_id
order by f.film_id ; --Группируем по ID потому что могут быть разные фильмы с одним названием и их группирует.
```

film 1 ×

select title as Название_фильма, count(r.rental_id) as Сколько_раз_брали_в_аренду, sum(coalesce(p.amount,0)) as Общая_стоимость_аренды

	Название_фильма	Сколько_раз_брали_в_аренду	Общая_стоимость_аренды
1	ACADEMY DINOSAUR	23	36,77
2	ACE GOLDFINGER	7	52,93
3	ADAPTATION HOLES	12	37,88
4	AFFAIR PREJUDICE	23	91,77
5	AFRICAN EGG	12	51,88
6	AGENT TRUMAN	21	126,79
7	AIRPLANE SIERRA	15	82,85
8	AIRPORT POLLOCK	18	102,82
9	ALABAMA DEVIL	12	71,88
10	ALADDIN CALENDAR	23	131,77
11	ALAMO VIDEOTAPE	24	35,76
12	ALASKA PHANTOM	26	44,74
13	ALICE IN WONDERLAND	26	44,74

Рисунок Б. 9 – Результат выполнения задания 7

--Задание 8. Доработайте запрос из предыдущего задания и выведите с помощью него фильмы, которые ни разу не брали в аренду.

```

select f.title as Название_фильма, count(r.rental_id) as Сколько_раз_брали_в_аренду from rental r
left join inventory i on i.inventory_id = r.inventory_id
right join film f on f.film_id = i.film_id --right потому что нужны фильмы, которые никто не арендовал
group by f.film_id
having count(r.rental_id) = 0
order by f.film_id ;
  
```

film 1 ×

select f.title as Название_фильма, count(r.rental_id) as Сколько_раз_брали_в_аренду

	Название_фильма	Сколько_раз_брали_в_аренду
1	ALICE FANTASIA	0
2	APOLLO TEEN	0
3	ARGONAUTS TOWN	0
4	ARK RIDGEMONT	0
5	ARSENIC INDEPENDENCE	0
6	BOONDOCK BALLROOM	0
7	BUTCH PANTHER	0
8	CATCH AMISTAD	0
9	CHINATOWN GLADIATOR	0
10	CHOCOLATE PUSS	0

Рисунок Б. 10– Результат выполнения задания 8

/*Задание 9. Посчитайте количество продаж, выполненных каждым продавцом.
 * Добавьте вычисляемую колонку «Премия». Если количество продаж превышает 7 300, то значение в колонке будет «Да»,
 * иначе должно быть значение «Нет».*/

```

select
s.staff_id as Продавец,
count(p.payment_id) as Количество_продаж,
(case when count(p.payment_id)>7300 then 'Да' else 'Нет' end) as Премия
from payment p
inner join staff s on s.staff_id = p.staff_id
group by s.staff_id
order by s.staff_id ;
  
```

staff 1 ×

select s.staff_id as Продавец, count(p.payment_id) as Количество_продаж

	Продавец	Количество_продаж	Премия
1	1	8 057	Да
2	2	7 992	Да

Рисунок Б. 11 – Результат выполнения задания 9

Приложение В

Результат выполнения «Задания 3»

*/Задание 1. Сделайте запрос к таблице payment и с помощью оконных функций добавьте вычисляемые колонки согласно условиям:

- Пронумеруйте все платежи от 1 до N по дате
 - Пронумеруйте платежи для каждого покупателя, сортировка платежей должна быть по дате
 - Посчитайте нарастающим итогом сумму всех платежей для каждого покупателя, сортировка должна быть сперва по дате платежа, а затем по сумме платежа от наименьшей к большей
 - Пронумеруйте платежи для каждого покупателя по стоимости платежа от наибольших к меньшим так, чтобы платежи с одинаковым значением имели одинаковое значение номера.
- Можно составить на каждый пункт отдельный SQL-запрос, а можно объединить все колонки в одном запросе.

```
select |
payment_date as Дата_платежа, --Выводим сначала дату
row_number() over (order by payment_date) as Номер_платежа, --Далее нумеруем платежи по датам от первой даты и до последней
customer_id as Айди_покупателя, --Теперь выводим айди покупателей чтобы отслеживать кто сколько потратил
row_number() over(partition by customer_id) as Номер_платежа_покупателя, --Нумеруем платежи для каждого покупателя, у них отдельные пронумерованные списки платежей отсортированные
amount as Сумма_платежа, --Теперь для удобства выводим сумму отдельного платежа
dense_rank() over (partition by customer_id order by amount desc) as Номер_платежа_по_стоимости, --Здесь с помощью функции dense_rank проставляем номера платежам по отдельности
sum(amount) over(partition by customer_id order by payment_date, amount rows unbounded preceding) as Нарастающий_итог --Тут с помощью агрегирующей функции sum считаем суммы
from payment p
order by payment_date, amount ;
```

1	Дата_платежа	123	Номер_платежа	123	Айди_покупателя	123	Номер_платежа_покупателя	123	Сумма_платежа	123	Номер_платежа_по_стоимости	123	Нарастающий_итог
1	2005-05-24 22:53:30.000		1		130		1		2,99		5		2,99
2	2005-05-24 22:54:33.000		2		459		1		2,99		8		2,99
3	2005-05-24 23:03:39.000		3		408		1		3,99		5		3,99
4	2005-05-24 23:04:41.000		4		333		1		4,99		3		4,99
5	2005-05-24 23:05:21.000		5		222		1		6,99		2		6,99
6	2005-05-24 23:08:07.000		6		549		1		0,99		6		0,99
7	2005-05-24 23:11:53.000		7		269		1		1,99		8		1,99
8	2005-05-24 23:31:46.000		8		239		1		4,99		5		4,99
9	2005-05-25 00:00:40.000		9		126		1		4,99		5		4,99

Рисунок В. 1– Результат выполнения задания 1

*/Задание 2. С помощью оконной функции выведите для каждого покупателя стоимость платежа и стоимость платежа из предыдущей строки со значением по умолчанию 0.0 с сортировкой по дате.

```
select
customer_id as Айди_покупателя,
amount as Платеж,
lag(amount,1,0.0) over (partition by customer_id order by payment_date) as Предыдущий_платеж
from payment p
order by customer_id, payment_date ;
```

123	Айди_покупателя	123	Платеж	123	Предыдущий_платеж
1	1		2,99		0
2	1		0,99		2,99
3	1		5,99		0,99
4	1		0,99		5,99
5	1		9,99		0,99
6	1		4,99		9,99
7	1		4,99		4,99
8	1		0,99		4,99
9	1		3,99		0,99
10	1		5,99		3,99
11	1		5,99		5,99
12	1		4,99		5,99

Рисунок В. 2 – Результат выполнения задания 2

*/Задание 3. С помощью оконной функции определите, на сколько каждый следующий платеж покупателя больше или меньше текущего.

```
select |
customer_id as Айди_покупателя,
amount as Размер_платежа_покупателя,
amount - lead (amount,1,0.0) over (partition by customer_id) as Разница_разм_след_и_текущ_платежа
from payment p ;
```

123	Айди_покупателя	123	Размер_платежа_покупателя	123	Разница_разм_след_и_текущ_платежа
1	1		2,99		2
2	1		0,99		-5
3	1		5,99		5
4	1		0,99		-9
5	1		9,99		5
6	1		4,99		0
7	1		4,99		4
8	1		0,99		-3
9	1		3,99		-2
10	1		5,99		0

Рисунок В. 3 – Результат выполнения задания 3

/*Задание 4. С помощью оконной функции для каждого покупателя выведите данные о его последней оплате аренды.

```

select
*
from payment p
where payment_date in ((select last_value(payment_date) over (partition by customer_id from payment p2)) ; --Здесь используется in с перечнем одинаковых значений отображающих дату последней оплаты

```

payment 1 X

select * from payment p where payment_date in (select last_value(123 Введите SQL выражение чтобы отфильтровать результаты

123 payment_id	123 customer_id	123 staff_id	123 rental_id	123 amount	123 payment_date
1	59	1	15 315	5,99	2005-08-22 20:03:46.000
2	85	2	15 907	4,99	2005-08-23 17:39:35.000
3	107	3	15 619	2,99	2005-08-23 07:10:14.000
4	145	4	15 635	1,99	2005-08-23 07:43:00.000
5	173	5	13 209	0,99	2006-02-14 15:16:03.000
6	206	6	15 603	0,99	2005-08-23 06:41:32.000
7	230	7	14 222	5,99	2005-08-21 04:49:48.000
8	253	8	15 805	4,99	2005-08-23 14:31:19.000
9	278	9	15 813	4,99	2006-02-14 15:16:03.000
10	302	10	15 370	5,99	2005-08-22 21:59:29.000
11		11	11 646	0,99	2006-02-14 15:16:03.000

Results edit key: payment_id |

Рисунок В. 4 – Результат выполнения задания 4

/*Задание 5. С помощью оконной функции выведите для каждого сотрудника сумму продаж за август 2005 года с нарастающим итогом по каждому сотруднику и по каждой дате продажи (без учёта времени) с сортировкой по дате*/

```

select
staff_id as Айди_работника,
date(payment_date) as Дата,
sum(amount) over (partition by staff_id order by payment_date, amount rows unbounded preceding) as Нарастающий_итог_по_сотруднику,
sum(amount) over (partition by staff_id order by payment_date, amount rows between unbounded preceding and unbounded following) as Сумма_продаж_за_август_2005,
sum(amount) over (partition by staff_id, date(payment_date) order by date(payment_date), amount rows unbounded preceding) as Нарастающий_итог_по_дате,
sum(amount) over (partition by staff_id, date(payment_date)) as Сумма_продаж_по_дням
from payment p
where payment_date between '2005-08-01' and '2005-09-01'
order by staff_id, date(payment_date);

```

payment 1 X

select staff_id as Айди_работника, date(payment_date) as Дата, su(123 Введите SQL выражение чтобы отфильтровать результаты

123 Айди_работника	123 Дата	123 Нарастающий_итог_по_сотруднику	123 Сумма_продаж_за_август_2005	123 Нарастающий_итог_по_дате	123 Сумма_продаж_по_дням
1	2005-08-01	1 135,2	11 853,65	0,99	1 442,55
2	2005-08-01	767,12	11 853,65	1,98	1 442,55
3	2005-08-01	66,84	11 853,65	2,97	1 442,55
4	2005-08-01	786,08	11 853,65	3,96	1 442,55
5	2005-08-01	224,48	11 853,65	4,95	1 442,55
6	2005-08-01	77,81	11 853,65	5,94	1 442,55
7	2005-08-01	1 039,46	11 853,65	6,93	1 442,55
8	2005-08-01	161,63	11 853,65	7,92	1 442,55
9	2005-08-01	854,92	11 853,65	8,91	1 442,55
10	2005-08-01	860,9	11 853,65	9,9	1 442,55
11	2005-08-01	1 037,40	11 853,65	10,90	1 442,55

Рисунок В. 5 – Результат выполнения задания 5

/*Задание 6. 20 августа 2005 года в магазинах проходила акция: покупатель каждого сотого платежа получал дополнительную скидку на следующую аренду. С помощью оконной функции выведите всех покупателей, которые в день проведения акции получили скидку*/

```

with NumberedCustomers as (
select
*,
row_number () over (order by date(payment_date)) as numbered --Использую with для удобства вз
from payment p
where date(payment_date) = '2005-08-20'
)
select
date(payment_date) as Дата,
numbered as Номер_покупателя,
customer_id as Айди_покупателя
from NumberedCustomers
where numbered % 100 = 0;

```

ayment 1 X

with NumberedCustomers as (select *, row_number () over (order by date(payment_date)) as numbered --Использую with для удобства вз

123 Дата	123 Номер_покупателя	123 Айди_покупателя
2005-08-20	100	87
2005-08-20	200	189
2005-08-20	300	273
2005-08-20	400	372
2005-08-20	500	458
2005-08-20	600	569

Рисунок В. 6 – Результат выполнения задания 6

```

with RankedCustomers as (
  select
    c.customer_id, c.first_name, c.last_name, c3.country, count(p.payment_id) over (partition by c.customer_id as num_rentals, sum(p.amount) over (partition by c.customer_id as total_amount, row
  from customer c
  inner join payment p on p.customer_id = c.customer_id
  inner join rental r on r.rental_id = p.rental_id
  inner join address a on a.address_id = c.address_id
  inner join city c2 on c2.city_id = a.city_id
  inner join country c3 on c3.country_id = c2.country_id
),
MaxRentals as (
  select
    country, MAX(num_rentals) as max_num_rentals, MAX(total_amount) as max_total_amount, max(payment_date) as max_payment_date
  from RankedCustomers
  group by country
)
select
  rc.country, rc.customer_id, rc.first_name, rc.last_name, rc.num_rentals, rc.total_amount,
  case
    when rc.num_rentals = mr.max_num_rentals then rc.num_rentals
    else null
  end AS customer_with_most_rentals,
  case
    when rc.payment_date = mr.max_payment_date then rc.payment_date
    else null
  end as customer_last_rental,
  case
    when rc.total_amount = mr.max_total_amount THEN rc.total_amount
    else null
  end as customer_with_highest_amount
from RankedCustomers rc
join MaxRentals mr on rc.country = mr.country
where rc.rental_order = 1 and not(rc.num_rentals != mr.max_num_rentals and rc.payment_date != mr.max_payment_date and rc.total_amount != mr.max_total_amount)
order by rc.country, rc.rental_order desc;

```

country(+): 1 X

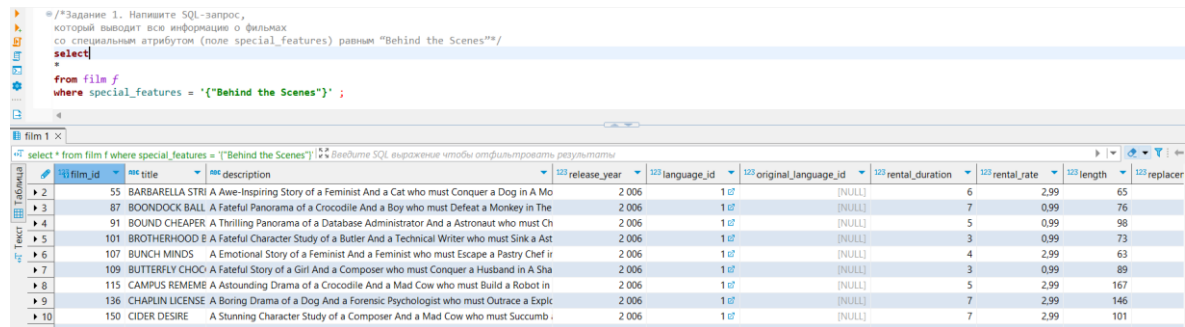
with RankedCustomers as (select c.customer_id, c.first_name, c.last_name, c3.country, count(p.payment_id) over (partition by c.customer_id as num_rentals, sum(p.amount) over (partition by c.customer_id as total_amount, row

country	customer_id	first_name	last_name	num_rentals	total_amount	customer_with_most_rentals	customer_last_rental	customer_with_highest_amount
USA	218	VERA	MCCOY	18	67.82	18	2005-08-23 05:30:19.000	67.82

Рисунок В. 7 – Результат выполнения задания 7

Приложение Г

Результат выполнения «Задания 4»



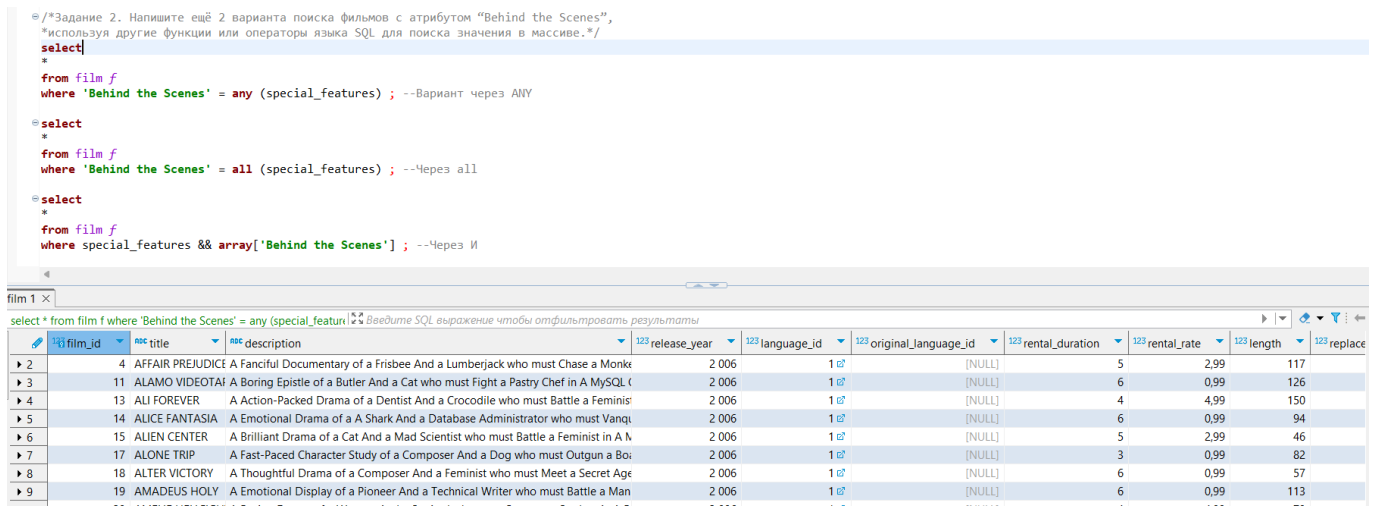
SQL query:

```
select
*
from film f
where special_features = ('Behind the Scenes');
```

Results table:

film_id	title	description	release_year	language_id	original_language_id	rental_duration	rental_rate	length	replacer
55	BARBARELLA STRI	A Awe-Inspiring Story of a Feminist And a Cat who must Conquer a Dog in A Mo	2 006	1 07	[NULL]	6	2,99	65	
87	BOONDOCK BALL	A Fateful Panorama of a Crocodile And a Boy who must Defeat a Monkey in The	2 006	1 07	[NULL]	7	0,99	76	
91	BOUND CHEAPER	A Thrilling Panorama of a Database Administrator And a Astronaut who must Ch	2 006	1 07	[NULL]	5	0,99	98	
101	BROTHERHOOD E	A Fateful Character Study of a Butler And a Technical Writer who must Sink a Ast	2 006	1 07	[NULL]	3	0,99	73	
107	BUNCH MINDS	A Emotional Story of a Feminist And a Feminist who must Escape a Pastry Chef ir	2 006	1 07	[NULL]	4	2,99	63	
109	BUTTERFLY CHOC	A Fateful Story of a Girl And a Composer who must Conquer a Husband in A Sha	2 006	1 07	[NULL]	3	0,99	89	
115	CAMPUS REMEME	A Astounding Drama of a Crocodile And a Mad Cow who must Build a Robot in	2 006	1 07	[NULL]	5	2,99	167	
136	CHAPLIN LICENSE	A Boring Drama of a Dog And a Forensic Psychologist who must Outrace a Explic	2 006	1 07	[NULL]	7	2,99	146	
150	OLDER DESIRE	A Stunning Character Study of a Composer And a Mad Cow who must Succumb	2 006	1 07	[NULL]	7	2,99	101	

Рисунок Г. 1 – Результат выполнения задания 1



SQL queries:

```
select
*
from film f
where 'Behind the Scenes' = any (special_features); --Вариант через ANY

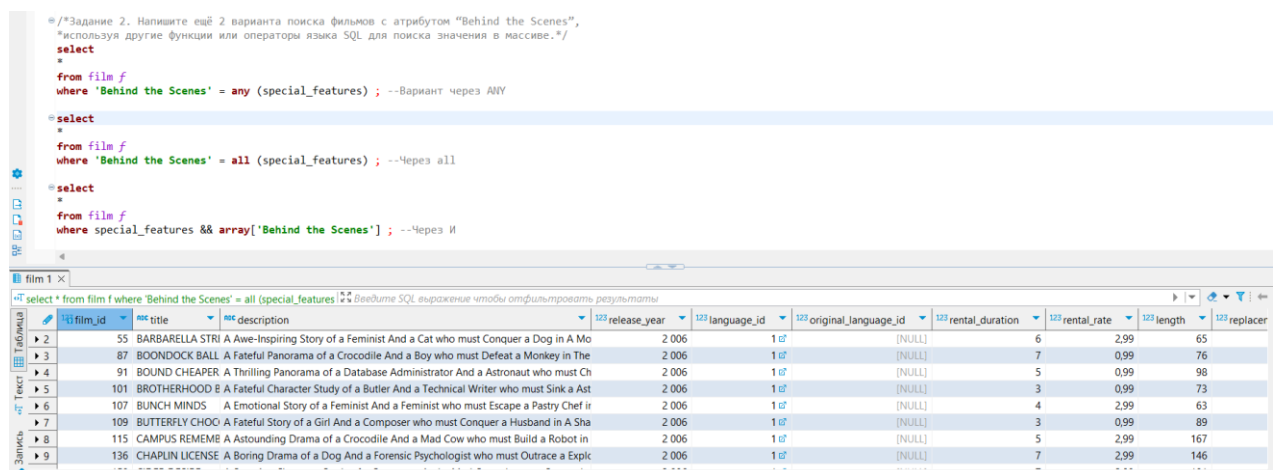
select
*
from film f
where 'Behind the Scenes' = all (special_features); --Через all

select
*
from film f
where special_features && array['Behind the Scenes']; --Через И
```

Results table:

film_id	title	description	release_year	language_id	original_language_id	rental_duration	rental_rate	length	replacer
4	AFFAIR PREJUDICE	A Fanciful Documentary of a Frisbee And a Lumberjack who must Chase a Monke	2 006	1 07	[NULL]	5	2,99	117	
11	ALAMO VIDEOTA	A Boring Epistle of a Butler And a Cat who must Fight a Pastry Chef in A MySQL C	2 006	1 07	[NULL]	6	0,99	126	
13	ALI FOREVER	A Action-Packed Drama of a Dentist And a Crocodile who must Battle a Feminis	2 006	1 07	[NULL]	4	4,99	150	
14	ALICE FANTASIA	A Emotional Drama of a A Shark And a Database Administrator who must Vanqu	2 006	1 07	[NULL]	6	0,99	94	
15	ALIEN CENTER	A Brilliant Drama of a Cat And a Mad Scientist who must Battle a Feminist in A N	2 006	1 07	[NULL]	5	2,99	46	
17	ALONE TRIP	A Fast-Paced Character Study of a Composer And a Dog who must Outgun a Boi	2 006	1 07	[NULL]	3	0,99	82	
18	ALTER VICTORY	A Thoughtful Drama of a Composer And a Feminist who must Meet a Secret Age	2 006	1 07	[NULL]	6	0,99	57	
19	AMADEUS HOLY	A Emotional Display of a Pioneer And a Technical Writer who must Battle a Man	2 006	1 07	[NULL]	6	0,99	113	

Рисунок Г. 2 – Результат выполнения задания 2



SQL queries:

```
select
*
from film f
where 'Behind the Scenes' = any (special_features); --Вариант через ANY

select
*
from film f
where 'Behind the Scenes' = all (special_features); --Через all

select
*
from film f
where special_features && array['Behind the Scenes']; --Через И
```

Results table:

film_id	title	description	release_year	language_id	original_language_id	rental_duration	rental_rate	length	replacer
55	BARBARELLA STRI	A Awe-Inspiring Story of a Feminist And a Cat who must Conquer a Dog in A Mo	2 006	1 07	[NULL]	6	2,99	65	
87	BOONDOCK BALL	A Fateful Panorama of a Crocodile And a Boy who must Defeat a Monkey in The	2 006	1 07	[NULL]	7	0,99	76	
91	BOUND CHEAPER	A Thrilling Panorama of a Database Administrator And a Astronaut who must Ch	2 006	1 07	[NULL]	5	0,99	98	
101	BROTHERHOOD E	A Fateful Character Study of a Butler And a Technical Writer who must Sink a Ast	2 006	1 07	[NULL]	3	0,99	73	
107	BUNCH MINDS	A Emotional Story of a Feminist And a Feminist who must Escape a Pastry Chef ir	2 006	1 07	[NULL]	4	2,99	63	
109	BUTTERFLY CHOC	A Fateful Story of a Girl And a Composer who must Conquer a Husband in A Sha	2 006	1 07	[NULL]	3	0,99	89	
115	CAMPUS REMEME	A Astounding Drama of a Crocodile And a Mad Cow who must Build a Robot in	2 006	1 07	[NULL]	5	2,99	167	
136	CHAPLIN LICENSE	A Boring Drama of a Dog And a Forensic Psychologist who must Outrace a Explic	2 006	1 07	[NULL]	7	2,99	146	

Рисунок Г. 3 – Результат выполнения задания 2

/*Задание 2. Напишите ещё 2 варианта поиска фильмов с атрибутом "Behind the Scenes",
 используя другие функции или операторы языка SQL для поиска значения в массиве.*/

```

select
*
from film f
where 'Behind the Scenes' = any (special_features) ; --Вариант через ANY

select
*
from film f
where 'Behind the Scenes' = all (special_features) ; --Через all

select
*
from film f
where special_features && array['Behind the Scenes'] ; --Через И
  
```

film 1 ×

select * from film f where special_features && array['Behind the Scenes'] Введите SQL выражение чтобы отфильтровать результаты

film_id	title	description	release_year	language_id	original_language_id	rental_duration	rental_rate	length	replacement_cost
2	4 AFFAIR PREJUDICE	A Fanciful Documentary of a Frisbee And a Lumberjack who must Chase a Monk	2006	1	[NULL]	5	2.99	117	
3	11 ALAMO VIDEOTA!	A Boring Epistle of a Butler And a Cat who must Fight a Pastry Chef in A MySQL C	2006	1	[NULL]	6	0.99	126	
4	13 ALI FOREVER	A Action-Packed Drama of a Dentist And a Crocodile who must Battle a Feminis	2006	1	[NULL]	4	4.99	150	
5	14 ALICE FANTASIA	A Emotional Drama of a A Shark And a Database Administrator who must Vanqu	2006	1	[NULL]	6	0.99	94	
6	15 ALIEN CENTER	A Brilliant Drama of a Cat And a Mad Scientist who must Battle a Feminist in A N	2006	1	[NULL]	5	2.99	46	
7	17 ALONE TRIP	A Fast-Paced Character Study of a Composer And a Dog who must Outgun a Boi	2006	1	[NULL]	3	0.99	82	
8	18 ALTER VICTORY	A Thoughtful Drama of a Composer And a Feminist who must Meet a Secret Age	2006	1	[NULL]	6	0.99	57	
9	19 AMADEUS HOLY	A Emotional Display of a Pioneer And a Technical Writer who must Battle a Man	2006	1	[NULL]	6	0.99	113	

Рисунок Г. 4 – Результат выполнения задания 2

/*Задание 3. Для каждого покупателя посчитайте, сколько он брал в аренду фильмов со специальным атрибутом "Behind the Scenes".
 Обязательное условие для выполнения задания: используйте запрос из задания 1, помещённый в CTE*/

```

with customers_films as(
select distinct
r.customer_id as Айди_покупателя,
count(f.film_id) over (partition by r.customer_id) as films_count
from film f
inner join inventory i on i.film_id = f.film_id
inner join rental r on r.inventory_id = i.inventory_id
where special_features = '{"Behind the Scenes"}'
order by r.customer_id
)
select
*
from customers_films;
  
```

rental 1 ×

with customers_films as(select distinct r.customer_id as Айди_покупателя, count(f.film_id) over (partition by r.customer_id) as films_count) Введите SQL выражение чтобы отфильтровать результаты

Айди_покупателя	films_count
1	2
2	4
3	5
4	6
5	8
6	9
7	10
8	11

Рисунок Г. 5 – Результат выполнения задания 3

/*Задание 4. Для каждого покупателя посчитайте, сколько он брал в аренду фильмов со специальным атрибутом "Behind the Scenes".
 Обязательное условие для выполнения задания: используйте запрос из задания 1, помещённый в подзапрос, который необходимо использовать для решения задания.*/

```

select distinct
r.customer_id as Айди_покупателя,
count(f.film_id) over (partition by r.customer_id) as films_count
from film f
inner join inventory i on i.film_id = f.film_id
inner join rental r on r.inventory_id = i.inventory_id
where f.film_id in (select film_id from film f2 where special_features = '{"Behind the Scenes"}')
order by r.customer_id ;
  
```

rental 1 ×

select distinct r.customer_id as Айди_покупателя, count(f.film_id) over (partition by r.customer_id) as films_count Введите SQL выражение чтобы отфильтровать результаты

Айди_покупателя	films_count
1	2
2	4
3	5
4	6
5	8
6	9
7	10
8	11
9	12

Рисунок Г. 6 – Результат выполнения задания 4

<pre> --Задание 5. Создайте материализованное представление с запросом из предыдущего задания и напишите запрос для обновления материализованного представления create materialized view SpecialView as select distinct r.customer_id as Айди_покупателя, count(f.film_id) over (partition by r.customer_id) as films_count from film f inner join inventory i on i.film_id = f.film_id inner join rental r on r.inventory_id = i.inventory_id where f.film_id in (select film_id from film f2 where special_features = '{"Behind the Scenes"}') order by r.customer_id ; refresh materialized view SpecialView; </pre>	
<pre> --explain analyze select * from SpecialView ; </pre>	
Статистика 1 X	
Name	Value
Updated Rows	0
Query	refresh materialized view SpecialView
Start time	Sun Jul 14 18:25:50 YEKT 2024
Finish time	Sun Jul 14 18:25:50 YEKT 2024

Рисунок Г. 7 – Результат выполнения задания 5

<pre> /*Задание 6. С помощью explain analyze проведите анализ скорости выполнения запросов из предыдущих заданий и ответьте на вопросы: с каким оператором или функцией языка SQL, используемыми при выполнении домашнего задания, поиск значения в массиве происходит быстрее; какой вариант вычислений работает быстрее: с использованием CTE или с использованием подзапроса */ explain analyze -- Plannig time ~ 0.260 ; Execution Time ~ 1.9 with customers_films as(select distinct r.customer_id as Айди_покупателя, count(f.film_id) over (partition by r.customer_id) as films_count from film f inner join inventory i on i.film_id = f.film_id inner join rental r on r.inventory_id = i.inventory_id where special_features = '{"Behind the Scenes"}' order by r.customer_id) select * from customers_films; </pre>	
Результат 1 X	
<pre> explain analyze with customers_films as(select distinct r.customer_id as Айди_покупателя, c </pre>	
QUERY PLAN	
11	-> Hash Join (cost=68.38.151.27 rows=321 width=8) (actual time=0.196.0.621 rows=319 loops=1)
12	Hash Cond: (f.film_id = f2.film_id)
13	-> Seq Scan on inventory i (cost=0.00.70.81 rows=4581 width=6) (actual time=0.006.0.158 rows=
14	-> Hash (cost=67.50.67.50 rows=70 width=4) (actual time=0.169.0.169 rows=70 loops=1)
15	Buckets: 1024 Batches: 1 Memory Usage: 11kB
16	-> Seq Scan on film f (cost=0.00.67.50 rows=70 width=4) (actual time=0.013.0.163 rows=70 l
17	Filter: (special_features = '{"Behind the Scenes"}':text[])
18	Rows Removed by Filter: 930
19	-> Index Scan using idx_fk_inventory_id on rental r (cost=0.29.0.51 rows=4 width=6) (actual time=0.00
20	Index Cond: (inventory_id = i.inventory_id)
21	Planning Time: 0.260 ms

Рисунок Г. 8 – Результат выполнения задания 6

<pre> --explain analyze -- Plannig time ~ 0.380 ; Execution Time ~ 2 select distinct r.customer_id as Айди_покупателя, count(f.film_id) over (partition by r.customer_id) as films_count from film f inner join inventory i on i.film_id = f.film_id inner join rental r on r.inventory_id = i.inventory_id where f.film_id in (select film_id from film f2 where special_features = '{"Behind the Scenes"}') order by r.customer_id ; --explain analyze -- Plannig time ~ 0.035 ; Execution Time ~ 0.05 Просто из любопытства select * from SpecialView ; </pre>	
<pre> /*ВЫВОД: С помощью explain analyze был проведен анализ скорости выполнения запросов с использованием CTE и подзапросов. * Поиск значений в массиве происходит быстрее с использованием CTE. </pre>	
Результат 1 X	
<pre> explain analyze select distinct r.customer_id as Айди_покупателя, c </pre>	
QUERY PLAN	
16	-> Hash Join (cost=68.38.136.01 rows=70 width=8) (actual time=0.182.0.268 rows=70 loops=
17	Hash Cond: (f.film_id = f2.film_id)
18	-> Seq Scan on film f (cost=0.00.65.00 rows=1000 width=4) (actual time=0.003.0.046 row
19	-> Hash (cost=67.50.67.50 rows=70 width=4) (actual time=0.169.0.169 rows=70 loops=1)
20	Buckets: 1024 Batches: 1 Memory Usage: 11kB
21	-> Seq Scan on film f2 (cost=0.00.67.50 rows=70 width=4) (actual time=0.012.0.163 rc
22	Filter: (special_features = '{"Behind the Scenes"}':text[])
23	Rows Removed by Filter: 930
24	-> Index Scan using idx_fk_inventory_id on rental r (cost=0.29.0.51 rows=4 width=6) (actual time=0.00
25	Index Cond: (inventory_id = i.inventory_id)
26	Planning Time: 0.380 ms
27	Execution Time: 1.987 ms

Рисунок Г. 9 – Результат выполнения задания 6

explain analyze -- Plannig time ~ 0.035 ; Execution Time ~ 0.05 Просто из любопытства

```

select
*
from SpecialView ; --Запрос к материализованному представлению

```

/*Вывод: С помощью explain analyze был проведен анализ скорости выполнения запросов с использованием CTE и подзапросов.
* Поиск значений в массиве происходит быстрее с использованием CTE.
* Ещё быстрее, конечно, происходит поиск в материализованном представлении, но его придется обновлять и хранить.
*/

Результат 1

explain analyze select * from SpecialView

1	Seq Scan on specialview (cost=0.00..9.14 rows=514 width=10) (actual time=0.005..0.025 rows=514 loops=1)
2	Planning Time: 0.101 ms
3	Execution Time: 0.041 ms

Рисунок Г. 10 – Результат выполнения задания 6

--Задание 7. Используя оконную функцию, выведите для каждого сотрудника сведения о первой его продаже.

```

with rental_info as(
select distinct
r.staff_id as Айди_продавца,
first_value(r.rental_date) over (partition by r.staff_id order by r.rental_date) as first_rental
from rental r
)
select
*
from rental
inner join rental_info on rental_info.first_rental= rental.rental_date

```

rental 1

rental_id	rental_date	inventory_id	customer_id	return_date	staff_id	last_update	Айди_продавца	first_rental
1	2005-05-24 22:53:30.000	367	130	2005-05-26 22:04:30.000	1	2006-02-15 21:30:53.000	1	2005-05-24 22:53:30.000
2	2005-05-24 23:04:41.000	2452	333	2005-06-03 01:43:41.000	2	2006-02-15 21:30:53.000	2	2005-05-24 23:04:41.000

Рисунок Г. 11 – Результат выполнения задания 7

/*Задание 8. Для каждого магазина определите и выведите одним SQL-запросом следующие аналитические показатели:

- день, в который арендовали больше всего фильмов (в формате год-месяц-день);
- количество фильмов, взятых в аренду в этот день;
- день, в который продали фильмов на наименьшую сумму (в формате год-месяц-день);
- сумму продаж в этот день.

```

with shops_ranking as (
select distinct
date(r.rental_date) as day_rental_date,
count(r.rental_id) over (partition by date(r.rental_date)) as day_film_count,
sum(p.amount) over (partition by date(r.rental_date)) as day_amount_sum
from store s
inner join inventory i on i.store_id = s.store_id
inner join rental r on r.inventory_id = i.inventory_id
inner join payment p on p.rental_id = r.rental_id
), min_max as (
select
max(day_film_count) as max_rent,
min(day_amount_sum) as min_amount
from shops_ranking
)
select
*
from shops_ranking sr
join min_max mm on mm.min_amount = sr.day_amount_sum or mm.max_rent = sr.day_film_count
order by day_rental_date

```

Результат 1

with shops_ranking as (select distinct date(r.rental_date) as day_rental_date,

day_rental_date	day_film_count	day_amount_sum	max_rent	min_amount
2005-05-24	12	38,88	679	38,88
2005-07-31	679	2 868,21	679	38,88

Рисунок Г. 12 – Результат выполнения задания 8

Приложение Д

Справка о проверке в системе «Антиплагиат.ру»

ТАРИФЫ

Demo ⚠

ИЗМЕНИТЬ

ПРОВЕРКИ

1 в 6 минут ?

ПРОВЕРИТЬ ДОКУМЕНТ

ПОЛЬЗОВАТЕЛЬ

artem_klimov_2002@bk.ru

ВОЙТИ В КАБИНЕТ

МЕНЮ

Поиск по названиям документов

УДАЛЕННЫЕ ДОКУМЕНТЫ | 1/1

ПЕРЕМЕСТИТЬ | УДАЛИТЬ | ИСТОРИЯ ОТЧЕТОВ | ПРОВЕРИТЬ ПРАВОПИСАНИЕ | ОФОРМИТЬ БИБЛИОГРАФИЮ

Название	Дата загрузки	Оригинальность
<input type="checkbox"/> PostgreSQL_Отчет_Климов_Артём	✓ 14 Июл 2024 18:30	99,11%

ПОСМОТРЕТЬ РЕЗУЛЬТАТЫ

Рисунок Д.1 – Результаты проверки