

**МИНОБРНАУКИ РОССИИ**

**федеральное государственное бюджетное**

**образовательное учреждение высшего образования**

**«Казанский национальный исследовательский**

**технологический университет»**

**(ФГБОУ ВО «КНИТУ»)**

Кафедра Интеллектуальных систем и управления информационными ресурсами\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Направление (специальность) 02.03.03 Математическое обеспечение и администрирование информационных систем

Направленность (профиль, специализации) Информационные системы и базы данных

Группа 4321-23

**КУРСОВОЙ ПРОЕКТ**

по дисциплине «Базы данных и СУБД»

на тему «Разработка базы данных для системы учета посетителей музея»

|  |  |  |  |
| --- | --- | --- | --- |
| Исполнитель |  |  | МВЕЛВА К. Ч. |
|  | (дата, подпись) | (Ф.И.О) |
| Руководитель |  |  | Доц. каф. ИСУИР МАНГУШЕВА А.Р. |
|  | (дата, подпись) | (Должность, Ф.И.О) |
| Нормоконтролер |  |  | Доц. каф. ИСУИР МАНГУШЕВА А.Р. |
|  | (дата, подпись) | (Должность, Ф.И.О) |

|  |  |
| --- | --- |
|  |  |
| Работа(проект) защищена с оценкой |  |
| Руководитель |  |
|  | (дата, подпись) |

Казань, 2024 г.

ОГЛАВЛЕНИЕ

[ВВЕДЕНИЕ 3](#_Toc185543810)

[ЗАДАНИЕ 5](#_Toc185543811)

[ЛИСТ НОРМОКОНТРОЛЕРА 6](#_Toc185543812)

[1. Определение сущностей и атрибутов 10](#_Toc185543813)

[ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ 13](#_Toc185543814)

[1. Построение ER модель 14](#_Toc185543815)

[2. Нормализация и построения ER модель 17](#_Toc185543816)

[3. Определение отношений в базе данных 19](#_Toc185543817)

[ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ 21](#_Toc185543818)

[1. Определение типы данных 21](#_Toc185543819)

[2. Ограничения целостности 22](#_Toc185543820)

[3. Определение родительской и дочерней сущностей 23](#_Toc185543821)

[4. Добавление ограничений целостности 23](#_Toc185543822)

[5. Создание базы данных и ввод данных 26](#_Toc185543823)

[ЗАПРОСЫ К БД, ПРЕДСТАВЛЕНИЯ, ФУНКЦИИ И ТРИГГЕРЫ 29](#_Toc185543824)

[1. Запросы 29](#_Toc185543825)

[2. Создание представления 35](#_Toc185543826)

[3. Создание функции 39](#_Toc185543827)

[4. Создание триггеров 43](#_Toc185543828)

[ЗАКЛЮЧЕНИЕ 49](#_Toc185543829)

[СПИСОК ЛИТЕРАТУРЫ 51](#_Toc185543830)

# ВВЕДЕНИЕ

За последние несколько лет мир значительно продвинулся вперед в областях, связанных с информацией и анализом данных. На протяжении многих лет развитие крупных информационных систем, которым требовалась такая же крупная и подходящая структура управления данными, привело к созданию баз данных. Базы данных и их интеграция в информационные системы способствовали дальнейшему росту множества отраслей бизнеса, анализа информации и маркетинга. Благодаря их практическому применению во всех мыслимых областях и проявлениях общества, базы данных стали неотъемлемой частью компьютерных систем. Хорошо управляемая и поддерживаемая база данных не только улучшает общий поток и контроль информации, необходимой для определенных системных процессов, но и повышает масштабируемость и ремонтопригодность всей системы, обеспечивая целостность данных.

Базы данных также взаимодействуют с высокоуровневыми удобными для пользователя интерфейсами или клиент-ориентированными приложениями, представляя исходные данные в виде простых для чтения и четко определенных структур. Благодаря использованию систем управления базами данных базы данных могут взаимодействовать с системами и улучшать их функциональность.

Нигде это не проявляется так очевидно, как в системах, которые внедряются в государственных учреждениях, таких как больницы, музеи и галереи. Чтобы музеи могли оптимально и в достаточном объеме хранить и использовать информацию о посетителях, необходима надежная и масштабируемая база данных. Цель следующего исследования - продемонстрировать этапы разработки простой базы данных для поддержки системы управления посетителями в музее. База данных должна соответствовать основным требованиям, предъявляемым к системе управления посетителями, в том смысле, что она должна хранить информацию о посетителях, легко связывать выставочные данные с данными о посетителях и отзывами и оставаться центральной частью раздела хранения информации системы управления посетителями.

Целью следующего исследования было создание базы данных для учета посетителей музея. Этапами исследования были:

1. Исследование предметной области.
2. Концептуальное проектирование базы данных.
3. Логическое проектирование базы данных.
4. Физическое проектирование.
5. Создание запросов к базе данных и написание представлений, функций и триггеров.

# ЗАДАНИЕ

на курсовую работу (проект) студенту кафедры Мвелве Калебу Чанде

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Тема работы (проекта): <<Разработка базы данных для системы учета посетителей музея>> \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Исходные данные к проекту: Предметная область

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Содержание расчетно-пояснительной записки (включая перечень подлежащих разработке вопросов, включая вопросы стандартизации и контроля качества)

Концептуальное проектирование, логическое проектирование, физическое проектирование запросы, функции, и триггеры

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Перечень графического материала (схемной документации) презентация\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Консультанты по проекту (с указанием относящихся к ним разделов) \_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата выдачи задания «\_\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_20\_\_\_г.

Руководитель проекта \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (Мангушева А. Р.)

# ЛИСТ НОРМОКОНТРОЛЕРА

1. Лист является обязательным приложением к пояснительной записке дипломного (курсового) проекта.

2. Нормоконтролер имеет право возвращать документацию без рассмотрения в случаях:

- нарушения установленной комплектности,

- отсутствия обязательных подписей,

- нечеткого выполнения текстового и графического материала.

3. Устранение ошибок, указанных нормоконтролером, обязательно.

П е р е ч е н ь

замечаний и предложений нормоконтролера по дипломному (курсовому) проекту, студента

4321-23 К.Ч. Мвелвы

(группа, инициалы, фамилия)

|  |  |  |
| --- | --- | --- |
| Л(Лист (страница) | Условное обозначение (код ошибок) | Содержание замечаний и предложений со ссылкой на нормативный документ, стандарт или типовую документацию |
|  |  |  |

Дата «\_\_\_\_» \_\_\_\_\_\_\_\_\_20\_\_\_г.

Нормоконтролер \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А.Р. Мангушева

(подпись) (инициалы, фамилия)

ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Чтобы создать базу данных, соответствующую требованиям системы, необходимо провести исследование предметной области. Необходимо рассмотреть, как функционирует система управления посетителями и какую роль база данных будет играть в оптимизации функциональности системы

Музей — это учреждение, занимающееся экспонированием и сохранением объектов, имеющих культурное или научное значение. Многие музеи выставляют эти объекты на всеобщее обозрение, а некоторые располагают частными коллекциями, которые используются исследователями и специалистами.

Система управления посетителями (VMS) — это технология, основанная на программном обеспечении, которая позволяет организациям управлять и отслеживать перемещение посетителей по своим помещениям [1]. Такие системы позволяют организациям отслеживать такую информацию, как контактные данные, цель визита и даты посещения. Это позволяет организации полностью контролировать, кто и зачем посещает их учреждение.

Для музеев это не просто инструмент для отслеживания посещаемости, а комплексное решение, призванное улучшить впечатления посетителей при сохранении сохранности и целостности экспонатов. Эти системы обеспечивают простую и эффективную продажу билетов, навигацию и ведение журнала записей. Это не только повышает качество обслуживания, но и обеспечивает эффективность.

Каждая система управления пользователями обрабатывает и взаимодействует с большими объемами данных, начиная от информации о посетителях и заканчивая данными о выставках и покупках посетителей. В связи с этим рекомендуется, чтобы система управления посетителями имела хорошо продуманную базу данных. База данных — это организованный набор данных, хранящихся в компьютерной системе и обычно контролируемых системой управления базами данных (СУБД) [2].

База данных служит центральным источником для управления музейными записями. Она не только улучшает предметную область, но и обеспечивает простой анализ поведения посетителей, например, предпочтений в отношении покупок и расходов, безопасности и синхронизации по всему музею.

Базовая и эффективная система управления посетителями состоит из множества элементов. Они немного отличаются по структуре, но все они способствуют достижению главной цели. Это:

1. Системы самообслуживания — Киоски самообслуживания являются неотъемлемым компонентом современных систем управления посетителями (VMS) в музеях, упрощая процесс регистрации посетителей и предлагая более широкий спектр услуг для пользователей. Эти киоски позволяют посетителям самостоятельно управлять входом, регистрируясь, приобретая билеты или получая информацию без необходимости прямого взаимодействия с персоналом.
2. Мобильные приложения — представляют собой ключевую эволюцию в технологии управления посетителями, предлагая динамичную и гибкую платформу для улучшения работы музея. Эти приложения позволяют посетителям взаимодействовать с музейными экспонатами и услугами непосредственно со своих мобильных устройств, обеспечивая ранее недостижимый уровень удобства и персонализации.
3. Облачные приложения — представляют собой перспективный подход в области систем управления посетителями (VMS), предлагая музеям беспрецедентную гибкость и масштабируемость. Эти системы используют возможности облака для безопасного хранения данных и управления ими, обеспечивая легкий доступ из любого места и в любое время.

В условиях высокой конкуренции и стремительного развития музейного пространства важность эффективного и удобного для пользователей управления посещениями трудно переоценить.

Они отвечают за организацию работы музея и поддержание профессионального имиджа организации.

Основное требование, предъявляемое к базе данных системы управления посетителями музея, заключается в том, что она должна связывать как минимум три основные части организации: данные о посетителях и систему сбора обратной связи, выставки и предметы, включенные в нее, а также информацию о покупке билетов и товарах, продаваемых в музее.

КОНЦЕПТУАЛЬНОЕ ПРОЕКТИРОВАНИЕ

Первым шагом в создании любой базы данных является концептуальное проектирование базы данных. Чтобы соответствовать требованиям концептуального моделирования, необходимо определить основные объекты базы данных.

Проектирование базы данных — это процесс создания подробной модели данных базы данных. Он включает в себя определение структуры, и организации данных, хранящихся в системе баз данных [3]. Согласно большинству источников, концептуальный дизайн — это всего лишь высокоуровневый дизайн - какие таблицы должны существовать и какие связи существуют между ними. Другие источники, такие как Питер Чен, считают, что он не слишком отличается от логического дизайн. Однако все согласились с тем, что на данном этапе рассматриваются сущности и их атрибуты. Чтобы получить общее представление о том, как будет выглядеть структура базы данных, необходимо определить, какие сущности будут в ней присутствовать. Это первый этап концептуального проектирования.

## Определение сущностей и атрибутов

Сущности — это объекты или понятия, которые существуют в реальном мире и которые можно отличить друг от друга [4]. Примером объекта для базы данных системы управления посетителями музея является объект "посетители". Этот объект будет содержать всю информацию, относящуюся к посетителям.

Атрибуты — это свойства или характеристики объекта, которые определяют и отличают один объект от другого [5]. Атрибут — это не что иное, как фрагмент данных, который предоставляет дополнительную информацию об объекте.

Рассмотрим приведенную ниже концептуальную структуру базы данных системы управления посетителями музея.

* Сущность посетителей предназначен для хранения соответствующей информации о посетителях, которая может быть использована для идентификации посетителя.
* Сущность данных посещений содержит информацию о посещении музея. Это может быть информация о дате посещения, о том, кто из посетителей совершил это посещение и что было сделано во время посещения. Он также может содержать информацию о билетах, которые купил посетитель. Это должно быть разбито на две таблицы.
* Сущность данных экспонатах хранит соответствующую информацию о выставке, такую как характер выставки и продолжительность ее проведения. Также включена дополнительная информация, например, о представленных на выставке экспонатах.
* Сущность данных отзывов содержит информацию об отзывах посетителей о выставках. Его можно использовать для записи оценок, комментариев и предложений посетителей.
* Сущность данных о билетах рекомендуется использовать для хранения информации о билетах. Сюда входит информация о типе билета, записи о продаже билетов и о том, какие выставки включены в билет.
* Сущность данных выставок хранит основную информацию, относящуюся к музейным выставкам. Сюда входит название выставки, представленные на ней экспонаты, продолжительность выставки и идентификационный номер выставки.
* Сущность данных о покупках хранит информацию о покупках, совершенных посетителями во время их посещений, суммах, уплаченных посетителями, и купленных товарах. Этот объект также содержит идентификатор покупки

Создание сущностей приводит только к определению структур данных, которые будут содержаться в базе данных. Чтобы их можно было использовать, они должны быть связаны и структурно доработаны. Это делается на этапе логического проектирования.

# ЛОГИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

Логическое проектирование базы данных — это процесс определения логических структур данных, необходимых для поддержки информационных ресурсов в организации [6]. Процесс логического проектирования помогает создать модель базы данных, готовую к физической разработке и внедрению. Это полное и обоснованное логическое проектирование обеспечивает бесперебойную работу и целостность базы данных.

Для реляционных баз данных проектирование логической базы данных предполагает преобразование концептуальной модели в логическую схему с использованием модели данных, такой как модель сущности-отношения. Для этого необходимо определить атрибуты каждой сущности. После этого логическая модель нормализуется.

Отношения — это связи между сущностей. Примером отношений может служить "посетитель, имеющий членство". Проектируемая база данных является реляционной. Это означает, что все объекты в ней должны быть связаны каким-либо отношением. Рассмотрим типы отношений ниже

Существует несколько типов связей, которые могут существовать между объектами в реляционной базе данных. Это:

* + 1. "Один к одному" (1:1) — отношения "один к одному" являются двунаправленными, что означает, что они однозначны в обоих направлениях [7].
    2. "Один ко многим" (1:M) — отношение "один ко многим" возникает, когда один объект имеет многозначное отношение с другим объектом.
    3. "Многие ко многим" (М:М) — Отношения "многие ко многим" — это отношения, которые многозначны в обоих направлениях[8].

Нормализация базы данных — это процесс упорядочения атрибутов базы данных с целью уменьшения или устранения избыточности данных [9].

Ниже приведены шаги, предпринятые для создания логической модели проектирования базы данных управления посетителями музея.

## Построение ER модель

Это очень важный шаг для устранения ошибок в структуре базы данных и уменьшения избыточности во всей модели. Чтобы нормализовать базы данных, мы должны сначала построить ER-модель базы данных, сведя в таблицу объекты и их атрибуты, как это определено в концептуальном проекте.

Таблица 1 — visitors

|  |  |
| --- | --- |
| Атрибут | Описание атрибута |
| Visitor\_id | Столбец с уникальными числовыми идентификаторами для каждой строки таблицы |
| Surname | Это фамилия посетителя |
| Name | Это имя посетителя |
| Middle Name | Это отчество или второе имя посетителя |
| Date of birth | Здесь хранится дата рождения посетителя |
| Gender | Это показывает пол посетителя |
| Contact Information | Здесь отображается контактная информация посетителя по электронной почте |
| Membership\_id | Индикатор, показывающий членский номер посетителя |

Таблица 2 — visit\_details

|  |  |
| --- | --- |
| Атрибут | Описание атрибута |
| Visit\_id | Уникальный индикатор для отображения идентификационного номера посещения |
| Visit\_date | Атрибут, показывающий дату посещения |
| Purchase\_id | Атрибут, отображающий уникальный идентификатор покупки |
| Ticket\_id | Атрибут, отображающий уникальный идентификатор билета |

Таблица 3 — exhibitions

|  |  |
| --- | --- |
| Атрибут | Описание атрибута |
| Exhibition\_id | атрибут, отображающий уникальный идентификатор выставки |
| Exhibition\_name | показывает название каждой выставки |
| Start\_date | показывает дату начала каждой выставки |
| End\_date | показывает дату окончания каждой выставки |

Таблица 4 — items

|  |  |
| --- | --- |
| Атрибут | Описание атрибута |
| Item\_id | Это уникальный идентификатор каждого элемента в таблице |
| Item\_name | Название каждого элемента в таблице |
| Entry\_date | Здесь указана дата поступления каждого предмета в музейное хранилище |
| Exhbition\_id | Это ссылка на таблицу выставок, которая показывает, к какой выставке относится каждый предмет |

Таблица 5 — feedback

|  |  |
| --- | --- |
| Атрибут | Описание атрибута |
| Feedback\_id | Это уникальный идентификатор для каждой строки в таблице |
| Visitor\_id | ссылается на таблицу посетителей и показывает идентификатор посетителя для каждого отзыва |
| Comments | Здесь отображаются комментарии, оставленные в обзоре, в виде текста. |
| Ratings | Показывает числовой рейтинг музея |
| Suggestions | Учет предложений по улучшению предоставления услуг в музее. |

Таблица 6 — tickets

|  |  |
| --- | --- |
| Атрибут | Описание атрибута |
| Ticket\_id | Уникальный идентификатор для каждого билета |
| Ticket type | Указывает тип билета. Индивидуальный, студенческий или для старшеклассников |
| Date of purchase | Этот атрибут показывает дату, на которую был приобретен билет |
| Exhibition\_id | Этот атрибут ссылается на таблицу выставок и показывает идентификатор выставки |
| Ticket price | Этот атрибут показывает стоимость билета |

Таблица 7 — purchases

|  |  |
| --- | --- |
| Атрибут | Описание атрибута |
| Purchase\_id | Здесь отображается уникальный цифровой идентификатор каждой совершенной покупки |
| Visitor\_id | Этот атрибут ссылается на таблицу посетителей и показывает уникальный идентификатор посетителя |
| Item\_name | В нем отображается название приобретенного товара |
| Total\_amount | В таблице покупок отображается общая стоимость товаров |

Таблица 8— memberships

|  |  |
| --- | --- |
| Атрибут | Описание атрибута |
| Memership\_id | Уникальный индикатор, показывающий номер участника |
| Membership type | Показывает тип членства, которое имеет пользователь. Оно может быть премиум, стандартным или базовым |
| Start\_date | Показывает дату начала действия членства |
| Expriy\_date | Показывает дату окончания срока действия членства |

Теперь, когда объекты базы данных сведены в таблицу, можно удалить избыточные объекты и атрибуты. Это называется нормализацией и является следующим шагом в разработке базы данных

## Нормализация и построения ER модель

Следующим этапом в логическом проектировании базы данных является нормализация данных. Как было заявлено ранее нормализация — это процесс упорядочивания данных в базе данных. Он включает в себя установление связей между таблицами в соответствии с правилами, разработанными как для защиты данных, так и для повышения гибкости базы данных за счет устранения избыточности и несогласованной зависимости.

На каждом этапе нормализации мы сравниваем данные, чтобы определить, соответствуют ли они нормальным формам. Существует три основных нормальных формы. Рассмотрим следующие характеристики каждой нормальной формы:

Первая нормальная форма (1н.ф) — Необходимо исключать повторяющиеся группы в отдельных таблицах, создать отдельную таблицу для каждого набора связанных данных, и определить каждый набор связанных данных с помощью первичного ключа. В качестве примера первой нормализации рассмотрим таблицу 2, <<visit\_details>>, которая является избыточной, поскольку ссылается на несколько таблиц, и ее можно упростить, полностью избавившись от атрибутов <<Ticket\_id>>, <<Purchase\_id>>, <<Visit\_date>> и <<Visit\_id>>.

Первичный ключ — это столбец в таблице реляционной базы данных, который отличается для каждой записи. Это обеспечивает целостность и идентифицируемость каждой записи. Для каждой из приведенных выше таблиц базы данных столбец \_id, содержащий уникальный числовой идентификатор, будет являться первичным ключом.

Вторая нормальная форма (2н.ф) — Эта форма характеризуется созданием отдельных таблиц для наборов значений, которые применяются к нескольким записям, и связью этих таблиц с внешним ключом. Внешний ключ — это поле (или набор полей) в одной таблице, которое ссылается на первичный ключ в другой таблице [10]. Каждый экземпляр первичного ключа одной таблицы в другой таблице указывается как внешний ключ. Например, поле <<membership\_id>> в таблице <<visitors>> будет внешним ключом, который ссылается на таблицу 8 — memberships.

Третья нормальная форма (3н.ф). Исключаются поля, которые не зависят от ключа. Результатом нормализации является функциональная и логически связанная модель базы данных. Ниже приведены таблицы после нормализации и их отношения.

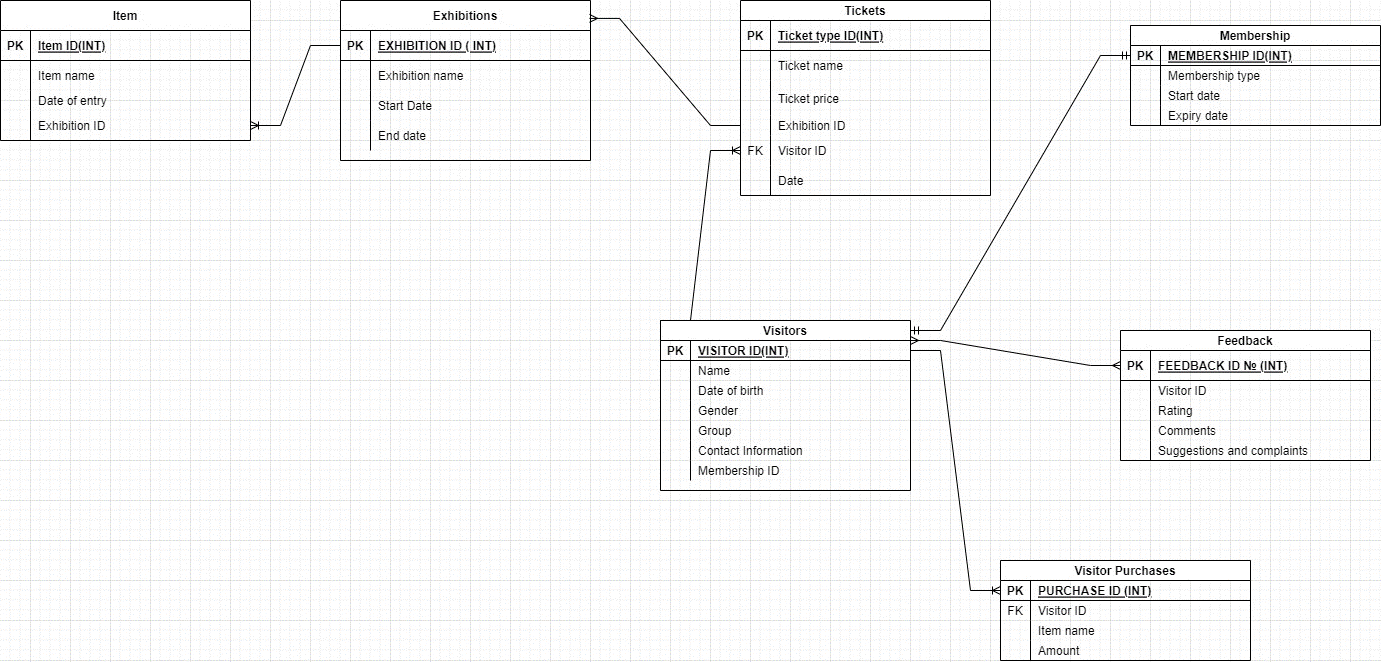


Рисунок 1 — ER модель базы данных

На основе модели ER, представленной на рисунке 1, можно создать нормализованные связи между объектами для каждого объекта, который связан с другим объектом. Это последний этап процесса логического проектирования

## Определение отношений в базе данных

* Items — exhibitions (M:1) На одной выставке может быть представлено много предметов/артефактов, но одновременно только один предмет может принадлежать одной выставке.
* Exhibitions—tickets (1:M) В билете одновременно указана только одна выставка.
* Visitors – memberships (1:1) Посетитель может иметь только один абонемент в музей.
* Visitors – feedback (1:M) Посетитель может оставить несколько отзывов под одним и тем же идентификатором посетителя.
* Visitors – tickets(1:M) Один посетитель может купить несколько билетов на выставку.
* Visitors – visitor\_Purchases (1:M) Один посетитель может завершить несколько покупки

Результатом логического проектирования базы данных является адаптируемая и оптимизированная ER-модель, которая является достаточно масштабируемой и гибкой для удовлетворения потребностей системы управления посетителями музея.

# ФИЗИЧЕСКОЕ ПРОЕКТИРОВАНИЕ

Целью физического проектирования базы данных является определение того, как будет реализован логический дизайн базы данных. Целью этого этапа проектирования является определение набора структур таблиц, типов данных для полей и ограничений для этих таблиц, таких как первичный ключ, внешний ключ, уникальный ключ и значение “not\_null”. В определенных случаях это также включает в себя разработку функций безопасности для системы баз данных, включая назначение уровня безопасности.

Для начала, чтобы определить наилучшие типы данных для каждого столбца, необходимо рассмотреть различные типы данных.

## Определение типы данных

Числовые типы данных

* INT (размер): Диапазон значений со знаком - от -2147483648 до 2147483647. Диапазон значений без знака - от 0 до 4294967295.
* DECIMAL (размер, d): Точное число с фиксированной запятой. Общее количество цифр указывается в параметре размер. Количество цифр после запятой указывается в параметре d.
* SERIAL: хранит последовательное целое число типа INT, которое автоматически присваивается сервером базы данных при вставке новой строки. По умолчанию начальный порядковый номер равен 1.
* DOUBLE PRECISION (размер, d): это неточный числовой тип с переменной точностью.

Строковые типы данных

* CHAR (размер): Строка фиксированной длины, которая может содержать буквы, цифры и специальные символы. Параметр size определяет длину столбца в символах - может быть от 0 до 255.
* VARCHAR (размер): Строка переменной длины, которая может содержать буквы, цифры и специальные символы. Параметр размер определяет максимальную длину столбца в символах - может быть от 0 до 65535.
* TEXT (размер) Содержит строку максимальной длиной 65 535 байт.

Датовские типы данных

* DATE: дата. Формат: ГГГГ-ММ-ДД. Поддерживаемый диапазон значений - от "1000-01-01" до "9999-12-31".
* TIMESTAMP: Временная метка. Значения временных меток хранятся в виде количества секунд, прошедших с начала эпохи Unix ('1970-01-01 00:00:00' UTC). Формат: ГГГГ-ММ-ДД, чч:мм:сс.

## Ограничения целостности

Ограничения целостности — это набор предопределенных правил, которые используются для поддержания качества информации. Ограничения целостности гарантируют, что ввод данных, обновление данных, удаление данных и другие процессы должны выполняться таким образом, чтобы не нарушалась целостность данных.

* PRIMARY KEY — указывает, что атрибуты первичного ключа должны быть уникальными, а не нулевыми.
* UNIQUE — каждое значение в столбце должно быть уникальным
* DEFAULT — устанавливает значение по умолчанию для столбца, если оно не задано
* CHECK (condition) — проверяет, удовлетворяет ли значение, содержащееся в столбце, указанному условию
* REFERENCES имя\_таблицы(имя\_столбца) — может быть указан между двумя таблицами. В случае ограничений ссылочной целостности, если внешний ключ в таблице ссылается на первичный ключ другой таблицы, то каждое значение внешнего ключа в таблице должно быть равно null или доступно в таблице, на которую оно ссылается.
* NOT NULL — указывает, что внутри кортежа атрибуты, для которых задано ограничение not null, не должны содержать никакого нулевого значения.

## Определение родительской и дочерней сущностей

Это важный этап, предшествующий созданию базы данных или сущностей. Чтобы знать, какие базы данных необходимо определить в первую очередь, необходимо определить, какие сущности являются родительскими, а какие - дочерними. Разница заключается в том, что в отношениях между родительской и дочерней сущностями родительская таблица содержит первичный ключ, на который ссылается другая таблица, например, <<membership\_id>> из таблицы <<memberships>>, на которую ссылается таблица <<visitors>>. Дочерняя таблица поддерживается родительской таблицей и на самом деле не является полной без данных из родительского поля [11]. Для структуры базы данных на рисунке 1 родительскими объектами являются членства и выставки. Остальные объекты являются дочерними объектами.

## Добавление ограничений целостности

Таблица 4.1 — Типы данных и ограничения сущности items

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип | Ограничение |
| Item\_id | serial | Primary key |
| item\_name | text | Not null; default ‘no\_name’ |
| Date\_of\_entry | date | Not null |
| Exhibition\_id | Int | Not null References Exhibitions(Exhibition\_id) |

Таблица 4.2— Типы данных и ограничения сущности exhibitions

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип | Ограничение |
| Exhibition\_id | serial | Primary key |
| Exhibition\_name | text | Not null; default ‘no\_name’ |
| Start\_date | date | Not null; (check start\_date<end\_date |
| End\_date | Int | Not null; (check end\_date>start\_date |

Таблица 4.3 — Типы данных и ограничения сущностиmemberships

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип | Ограничение |
| membership\_id | serial | Primary key |
| Membership\_type | text | Not null; |
| Start\_date | date | Not null; check (expiry\_date > start\_date) |
| Expiry\_date | date | Not null; check (expiry\_date > start\_date) |

Таблица 4.4— Типы данных и ограничения сущности visitors

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип | Ограничение |
| Visitor\_id | serial | Primary key |
| surname | text | Not null; |
| name | text | Not\_null; |
| Date\_of\_birth | date | Not null; (check start\_date<end\_date |
| gender | Varchar(10) | Not null; (check end\_date>start\_date |
| Contact\_information | Varchar(50) | Not null; |
| Membership\_id | int | Not null, unique, References Memberships (Membership\_id) |

(Продолжение таблицы 4.4)

Таблица 4.5 — Типы данных и ограничения сущности tickets

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип | Ограничение |
| Ticket\_id | serial | Primary key |
| Ticket\_name | text | Not null; |
| Ticket\_price | Decimal (8,2) | Not\_null; |
| Exhibition\_id | Int | Not null; References Exhibitions (Exhibition\_id) |
| Visitor\_id | Int | Not null; References Visitors (Visitor\_id) |
| Date | Date | Not null; |

Таблица 4.6 — Типы данных и ограничения сущности feedback

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип | Ограничение |
| Feedback\_id | serial | Primary key |
| Visitor\_id | Int | Not null; |
| rating | Int | Not\_null; check (rating<5) check(rating>0) |
| comments | text | Not null; |

Таблица 4.7 — Типы данных и ограничения сущности visitor\_purchases

|  |  |  |
| --- | --- | --- |
| Атрибут | Тип | Ограничение |
| Purchase\_id | serial | Primary key |
| Visitor\_id | int | Not null; Foreign Key(visitors) |
| Item\_name | text | Not\_null; |
| amount | Decimal (8,2) | Not null; |

(Продолжение таблицы 4.7)

## Создание базы данных и ввод данных

Прежде чем вводить таблицы и информацию о них, необходимо сначала создать базу данных. Это делается с помощью команды CREATE DATABASE, синтаксис которой показан ниже. Эта команда создаст базу данных с параметрами по умолчанию.

CREATE DATABASE имя\_базы данных (museum\_db, на пример)

Для реализации дизайна базы данных используется система управления базами данных, PostgreSQL, таблицы создаются с помощью SQL-команд CREATE TABLE, а данные вставляются в них с помощью команды INSERT INTO. Команда CREATE TABLE создаст новую, изначально пустую таблицу в текущей базе данных. Синтаксис команды, следующий:

CREATE TABLE имя\_таблицы

(Тип данных столбца 1 (размер), тип данных столбца 2 (размер),

тип данных столбца(размер)

);

Рассмотрим скрипт для создания таблицы "Exhibitions".

CREATE TABLE EXHIBITIONS(

exhibition\_id serial primary key,

exhibition\_name text not null default 'no\_name',

start\_date date not null,

end\_date date not null,

check (end\_date > start\_date)

);

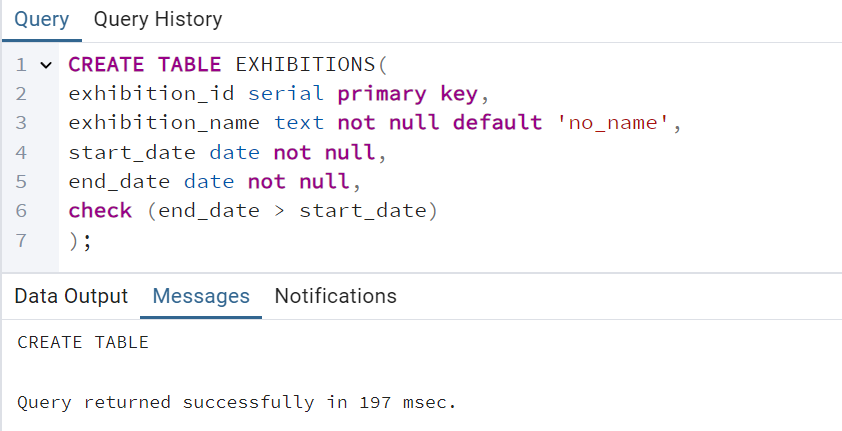


Рисунок 5.1 — Создание таблицы Exhbitions

Поле exhibition\_id является последовательным первичным ключом. Для каждой строки в базе данных будет автоматически сгенерировано новое целочисленное значение. Кроме того, для exhibition\_name были установлены ограничения “not null” и “default”. Для проверки достоверности дат установлено дополнительное ограничение.

Команда INSERT INTO используется для создания новых строк в таблице. Эти строки содержат данные и относятся к объектам реального мира. Синтаксис команды, следующий:

INSERT INTO имя\_таблицы

VALUES (значение1, значение2, значение3, ...);

Рассмотрим скрипт для внесения данных в таблице "Exhibitions".

INSERT INTO EXHIBITIONS(exhibition\_name,start\_date,end\_date)

VALUES ('Ancient Egypt','2024-01-04','2025-01-04'),

('Ancient Rome','2022-04-12','2025-01-04'),('English Kings','2024-01-04','2025-05-04'),

('Mayan Culture','2024-03-25','2024-04-25'),('Emperors of Rome','2024-08-01','2024-09-01'),

('Napeleon Bonaparte','2024-01-01','2025-03-01'),('John F. Kennedy','2024-11-01','2025-12-01'),

('Shaka Zulu','2024-02-18','2025-03-18'),('American Presidents','2023-01-04','2024-01-04'),

('Dictators of History','2024-06-10','2025-06-30'),('The Dark Ages','2024-11-01','2025-01-04'),

('World War 1','2024-05-09','2025-06-22'), ('World War 2','2024-05-09','2025-06-22'),

('Muhammad Ali','2024-07-03','2024-09-03'), ('Black Heroes','2024-02-01','2025-03-01');

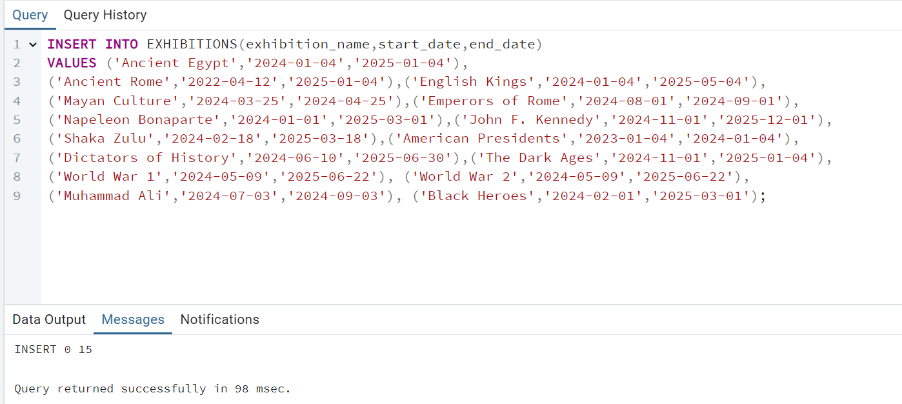


Рисунок 5.2 — Внесение данных в таблице Exhbitions

Создание таблиц базы данных и вставка данных в таблицы — это последний этап физического проектирования. В нем мы используем команды CREATE TABLE и INSERT INTO для создания базы данных с требуемыми типами данных и размерами. Другие скрипты, написанные для создания таблиц и вставки в них данных, доступны в электронной копии этого отчета по курсовому проекту. Однако для демонстрации эффективности и возможностей базы данных необходимо использовать запросы, которые обращаются к определенным наборам данных. Для дальнейшей демонстрации функциональности базы данных мы также используем представления, функции и триггеры.

# ЗАПРОСЫ К БД, ПРЕДСТАВЛЕНИЯ, ФУНКЦИИ И ТРИГГЕРЫ

## Запросы

SQL-запрос — это инструмент, который используется для работы с данными в базе данных [12]. Он содержит ряд ключевых слов, которые указывают системе управления базами данных, что делать. Запросы для отображения информации записываются с использованием команда SELECT. Рассмотрим примеры использования запросов к базе данных.

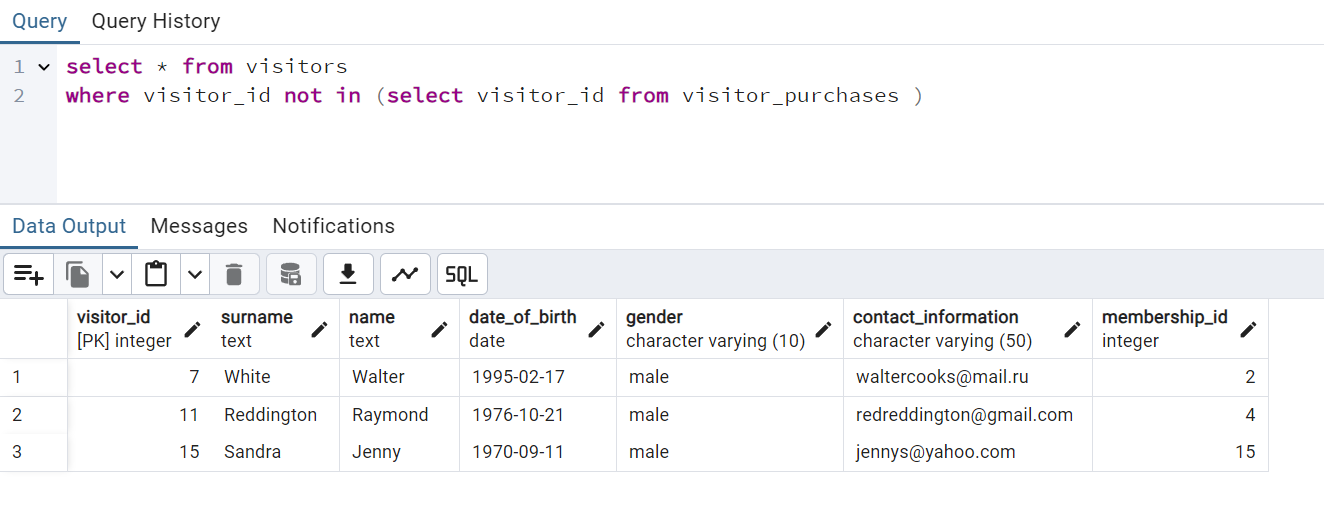
1. Выведите данные обо всех посетителях, которые не совершали никаких покупок.

Рисунок 1.1— Результат выполнения 1 запрос

1. Покажите количество экспонатов на выставке 7.

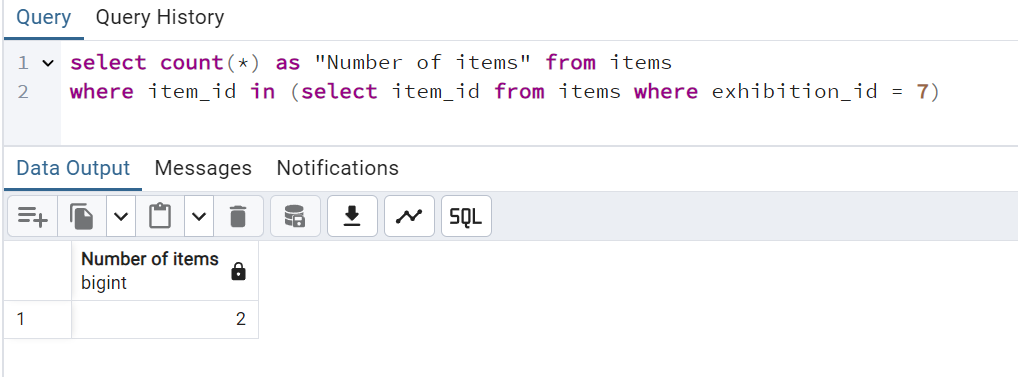


Рисунок 1.2— Результат выполнения 2 запрос

1. Выведите идентификаторы посетителей, которые оставили отзыв более двух раз.

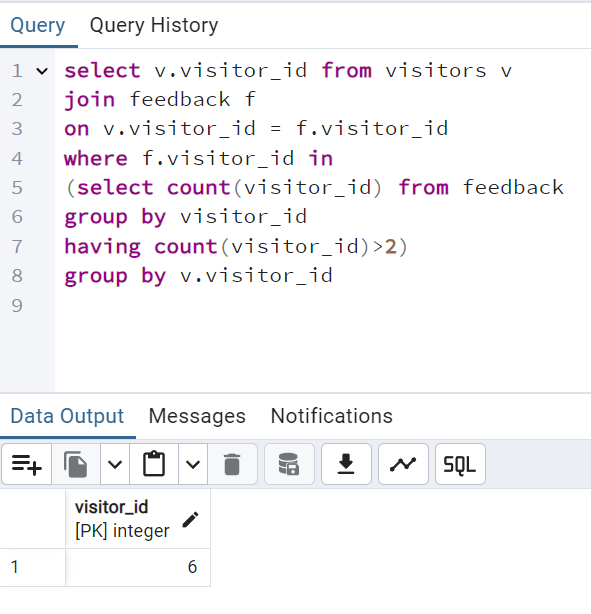


Рисунок 1.3— Результат выполнения 3 запрос

1. Выведите каждый тип билета и количество раз, когда он был куплен.

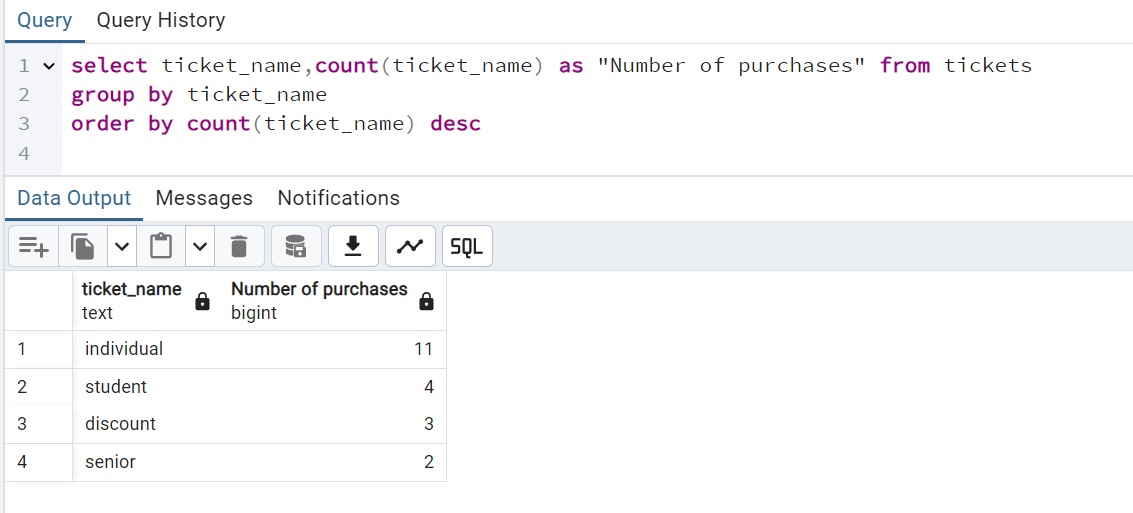


Рисунок 1.4— Результат выполнения 4 запрос

1. Отображайте информацию о каждой сделанной покупке на каком-нибудь пакете.

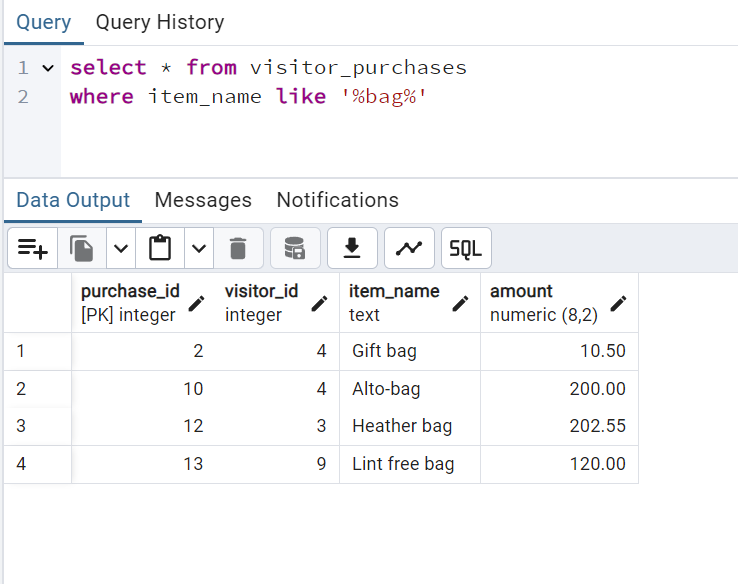


Рисунок 1.5— Результат выполнения 5 запрос

1. Покажите список названий выставок и количество экспонатов в каждой из них.

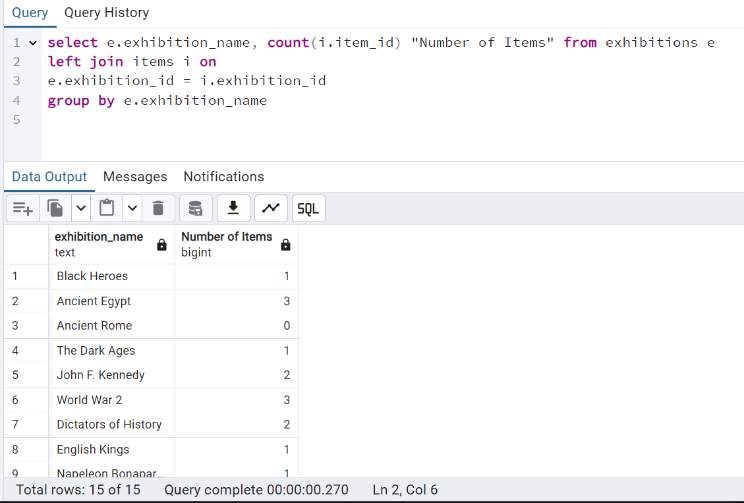


Рисунок 1.6— Результат выполнения 6 запрос

1. Выведите данные о посетителе и все комментарии к отзывам, сделанные посетителем с именем "John Mayer".

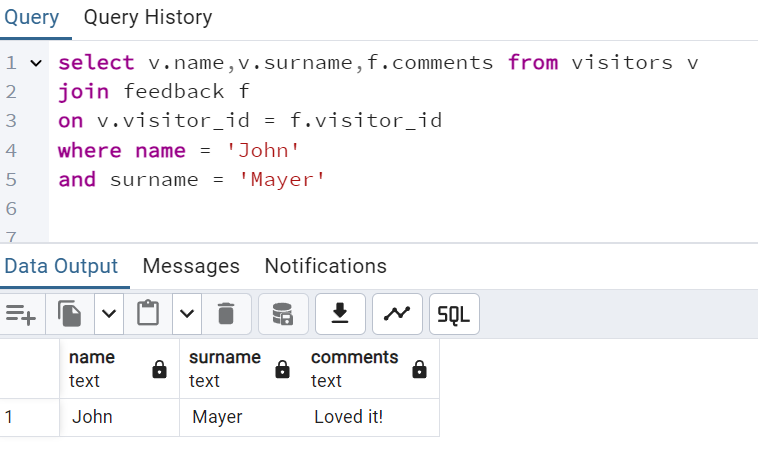


Рисунок 1.7— Результат выполнения 7 запрос

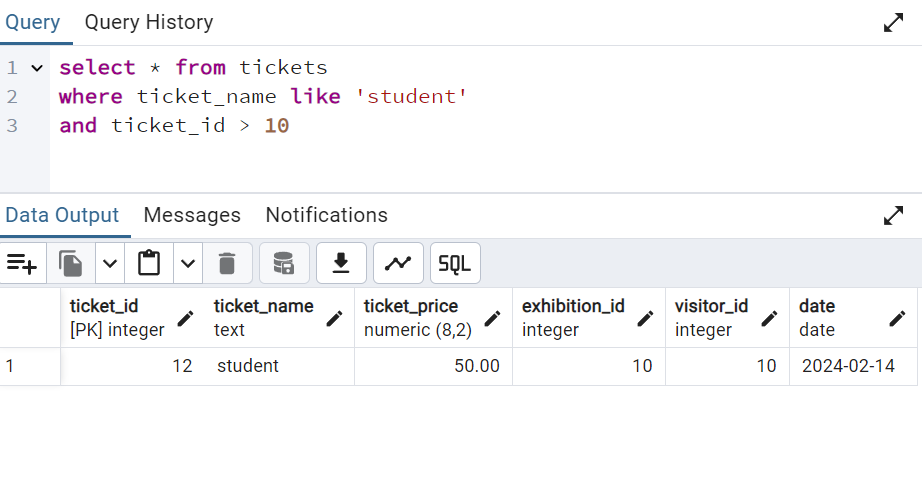
1.  Отображать информацию обо всех студенческих билетах с идентификатором 10 и выше.

Рисунок 1.8— Результат выполнения 8 запрос

1. Составьте топ-5 самых продолжительных выставок.

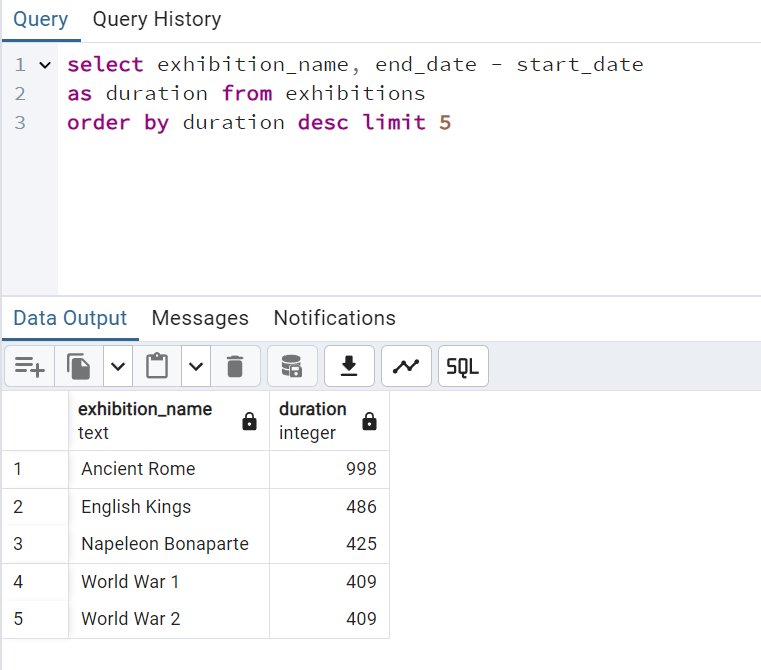


Рисунок 1.9— Результат выполнения 9 запрос

1. Вывести сведений обо всех посетителях, которые зарегистрировались с помощью адресов электронной почты yahoo.

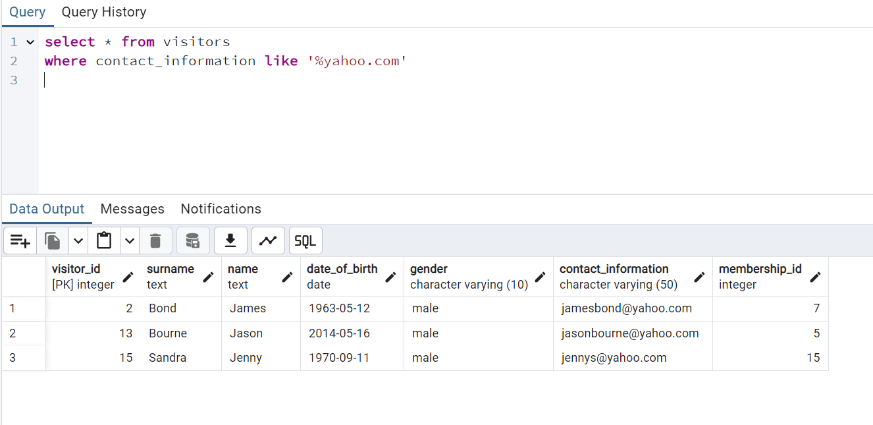


Рисунок 1.10— Результат выполнения 10 запрос

1. Выведите максимальную продолжительность (в днях) членства

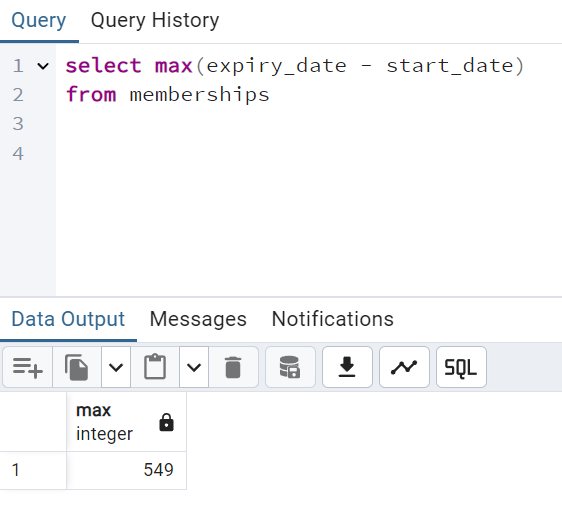


Рисунок 1.11— Результат выполнения 11 запрос

1. Выведите имена посетителей и их комментарии к отзывам

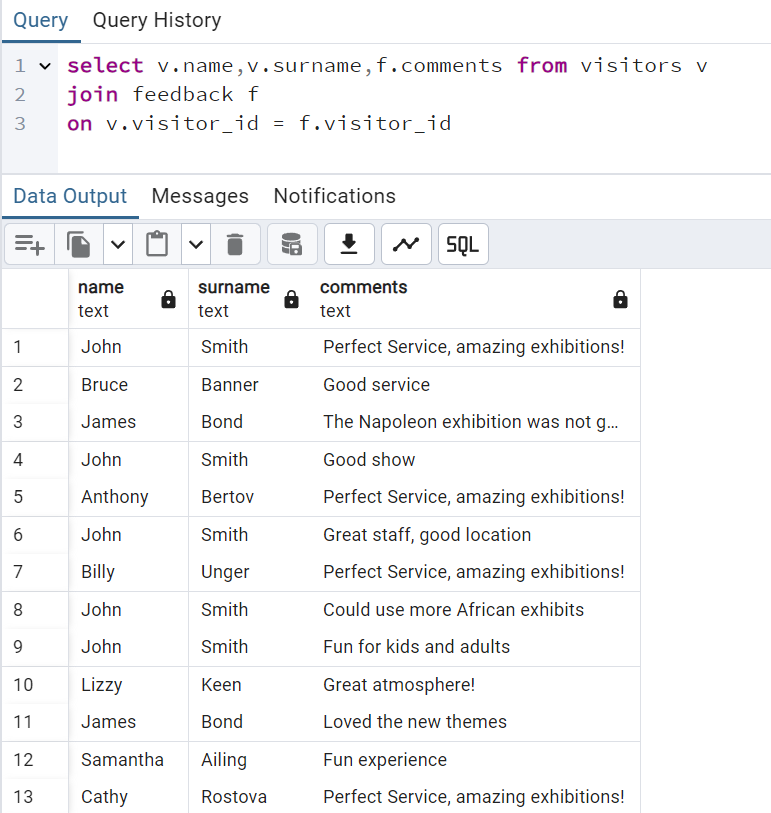


Рисунок 1.12— Результат выполнения 12 запрос

1. Выведите подробную информацию обо всех выставках, на которых нет никаких экспонатов

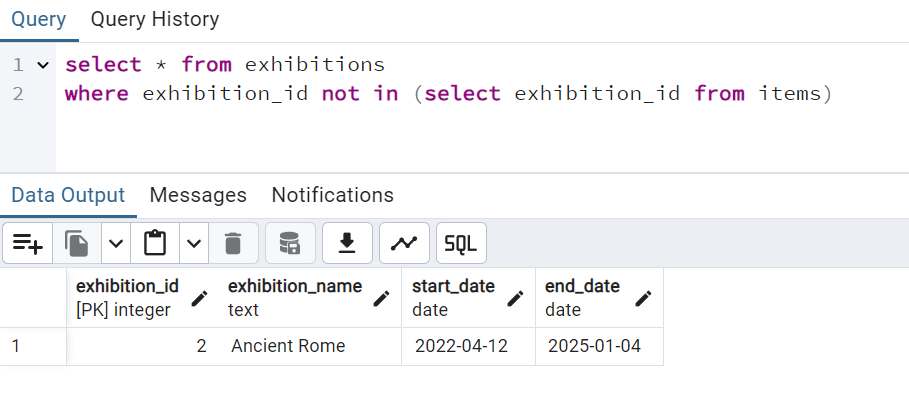


Рисунок 1.13— Результат выполнения 13 запрос

1. Выведите среднюю цену покупок, совершенных в музее.

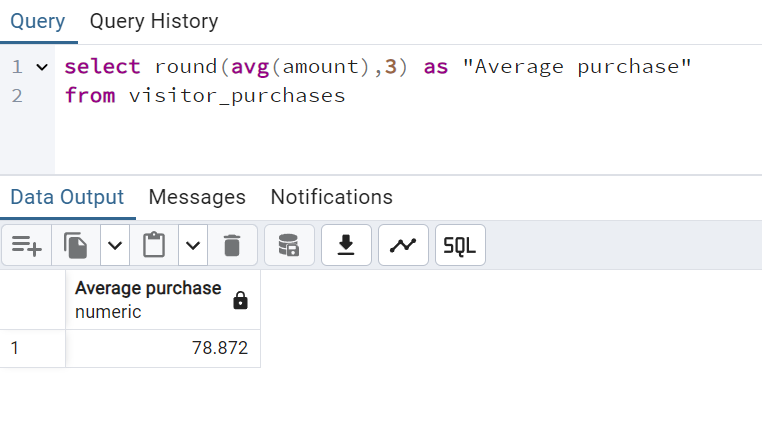


Рисунок 1.14— Результат выполнения 5 запрос

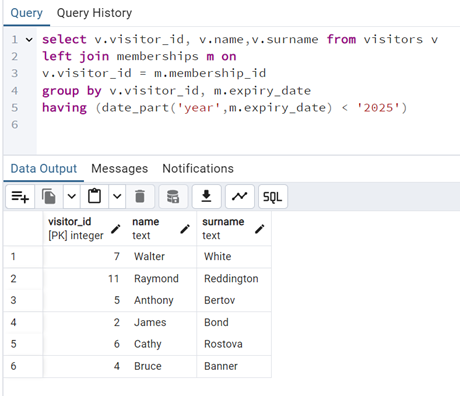
15 Выведите информацию о посетителях, срок действия членства которых истекает до 2025 года.

Рисунок 1.15— Результат выполнения 15 запрос

## Создание представления

Представление в терминологии SQL — это отдельная таблица, которая является производной от других таблиц [13]. Эти другие таблицы могут быть базовыми таблицами или другими представлениями. Оно считается виртуальной таблицей. Для создания представления используется команда CREATE VIEW. Рассмотрим синтаксис команды и пример ее использования.

CREATE VIEW схема.имя\_представления AS SELECT (тело запроса).

1. Представлениеvisitor\_memberships

Представление visitor\_memberships предназначено для отображения информации о посетителях и их членских данных для удобства использования. Чтобы создать его, мы выбираем <<surname>> и <<name>> посетителя и его <<membership\_id>> из таблицы посетителей и объединяем их с таблицей членства, из которой мы отображаем дату начала и окончания срока действия членства посетителя.

CREATE VIEW public.visitor\_memberships AS

select v.surname, v.name, v.membership\_id, m.start\_date, m.expiry\_date

from visitors v

join memberships m

on v.membership\_id = m.membership\_id;

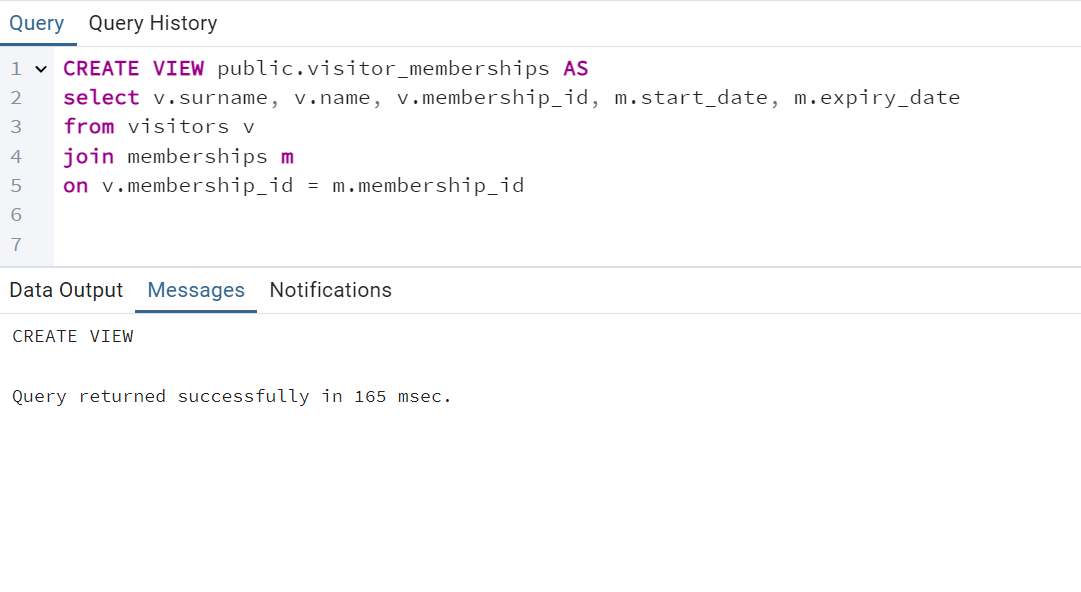
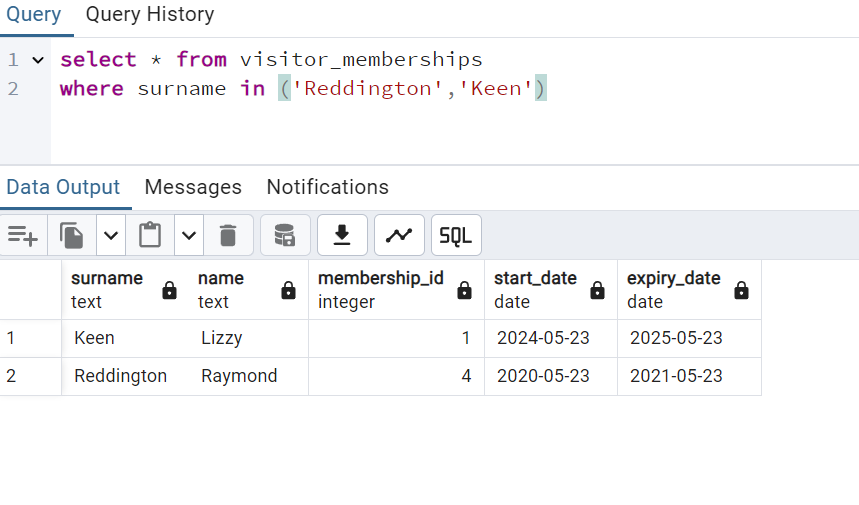


Рисунок 2.1.1 — Результат выполнения скрипта для создания представления

Чтобы протестировать созданное представление, пишется простой запрос. Рассмотрим результаты работы скрипта ниже

Рисунок 2.1.2 — Результат выполнения скрипта для тестирования.

1. Представление items\_exhibitions:

Представление items\_exhibitions предназначено для отображения информации о музейных экспонатах и о том, к какой выставке они в данный момент относится. Чтобы создать представление, в таблице items выбираются 'item\_name' и 'date\_of\_entry', а в таблице exhibitions— ‘exhibition\_name’. Эти таблицы объединяются по exhibition\_id.

CREATE VIEW items\_exhibitions AS

select i.item\_name, i.date\_of\_entry, e.exhibition\_name

from items i join exhibitions e

on i.exhibition\_id = e.exhibition\_id;

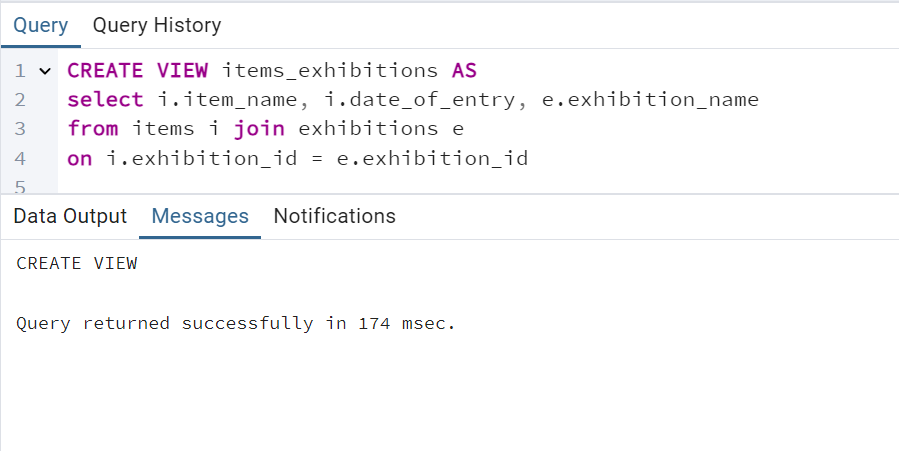
****

Рисунок 2.2.1 — Результат выполнения скрипта для создания представления

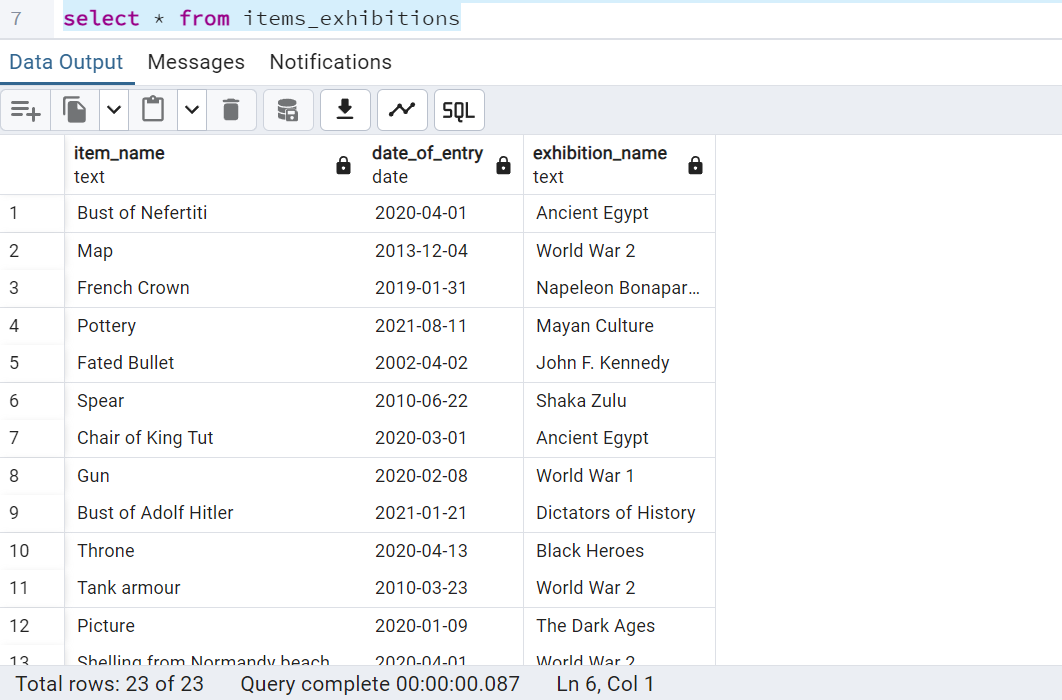
****

Рисунок 2.2.2 — Результат выполнения скрипта для тестирования представления

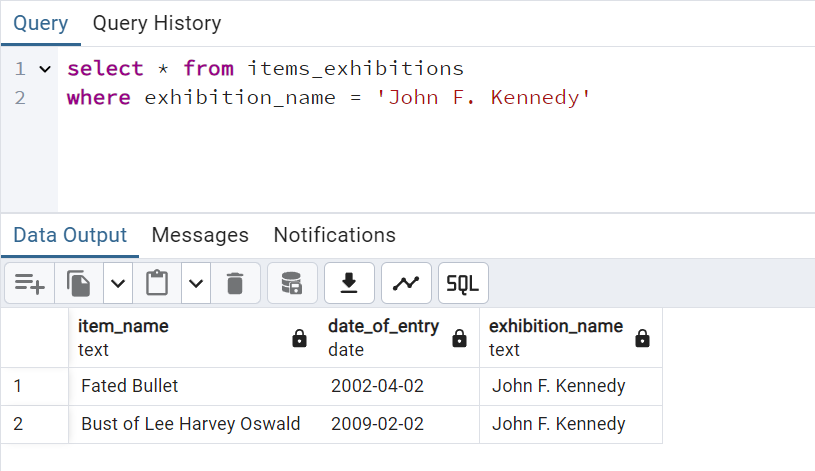
****

Рисунок 2.2.3 — Результат выполнения скрипта для тестирования

1. Представление visitors\_feedback:

Представление visitors\_feedback создано для эффективного представления информации о посетителях и их отзывах. Чтобы создать это представление, в таблице посетителей выбираются идентификатор посетителя — ‘visitor\_id’—, его имя и фамилия— ‘name’ и ‘surname’. После объединения таблиц посетителей и отзывов, рейтинг и комментарии добавляются в выходные данные.

CREATE VIEW visitors\_feedback AS

select v.visitor\_id, v.surname,v.name, f.rating,f.comments

from visitors v

join feedback f

on v.visitor\_id = f.visitor\_id;

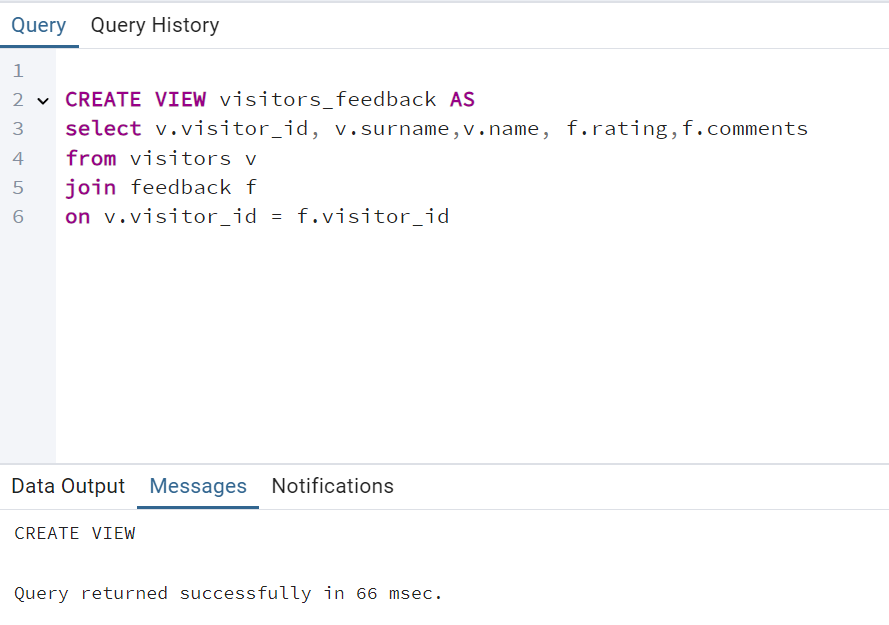
****

Рисунок 2.3.1 — Результат выполнения скрипта для создания представления

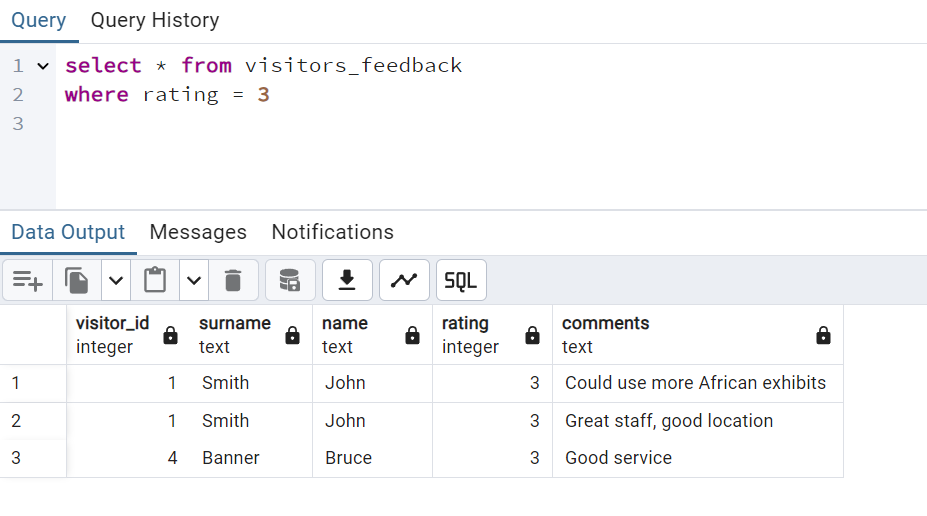
****

Рисунок 2.3.1 — Результат выполнения скрипта для создания представления

## Создание функции

Функции SQL — это программы, которые программное обеспечение для управления базами данных запускает в базе данных для выполнения операций с данными или объектами базы данных [14].

1. GetExhibitionDuration (name text). Функция отображает информацию о продолжительности конкретной выставки (возвращает значение в днях). В качестве входных параметров используется название выставки.

CREATE FUNCTION getExhibitionDuration(name text)

RETURNS INTEGER AS

$$

BEGIN

select (end\_date - start\_date) as Duration from exhibitions

where name = exhibition\_name;

RETURN duration;

END;

$$ LANGUAGE plpgsql;

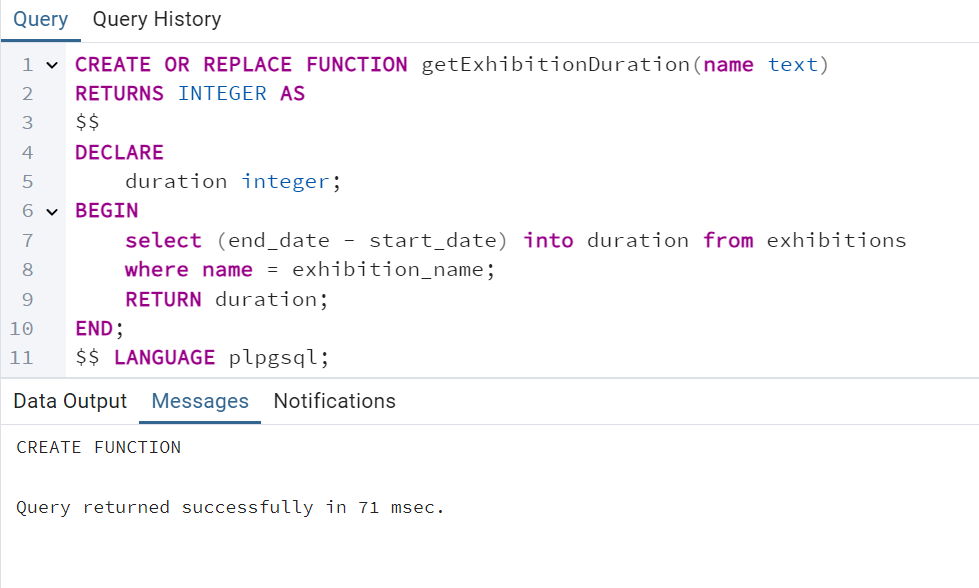
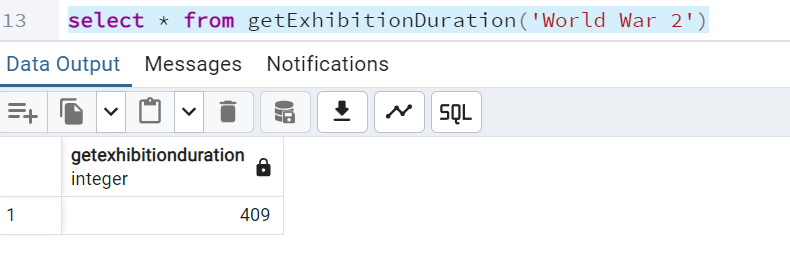


Рисунок 3.1.1 — Результат выполнения скрипта для создания функций



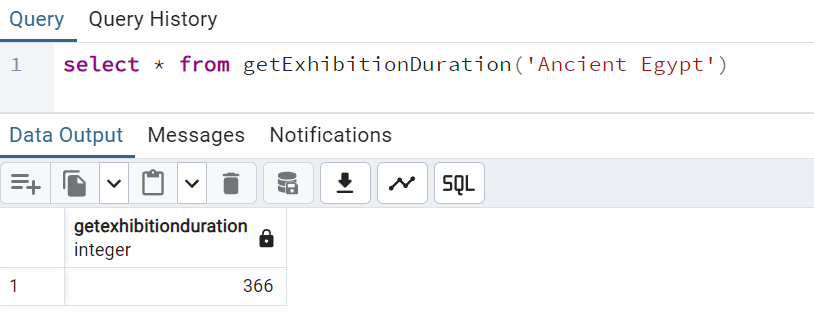


Рисунок 3.1.2 — Результат выполнения скрипта для предсталения функций

1. GetAverageRating (visitor\_name text). Функция отображает информацию о пользователе и его среднем рейтинге. Она принимает имя пользователя (имя и фамилию) в качестве входных параметров и возвращает таблицу, в которой указаны имя и фамилия пользователя и его средний рейтинг.

CREATE OR REPLACE FUNCTION getAverageRating(visitor\_name text, visitor\_surname text)

RETURNS TABLE (surname text, name text, average\_rating numeric)

AS

$$

BEGIN

RETURN QUERY

select v.surname, v.name, round(avg(f.rating),2) from visitors v

join feedback f on

v.visitor\_id = f.visitor\_id

where visitor\_surname = v.surname

and

visitor\_name = v.name

group by f.visitor\_id, v.surname, v.name;

END;

$$ LANGUAGE plpgsql;

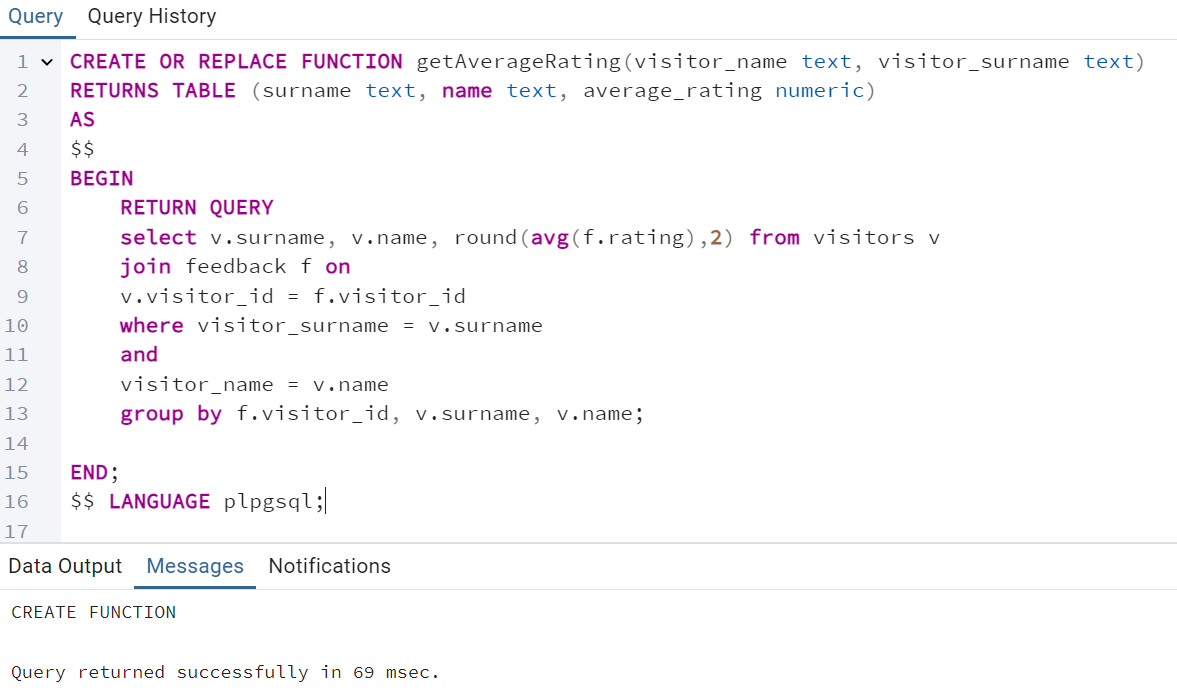


Рисунок 3.2.1 — Результат выполнения скрипта для создания функций

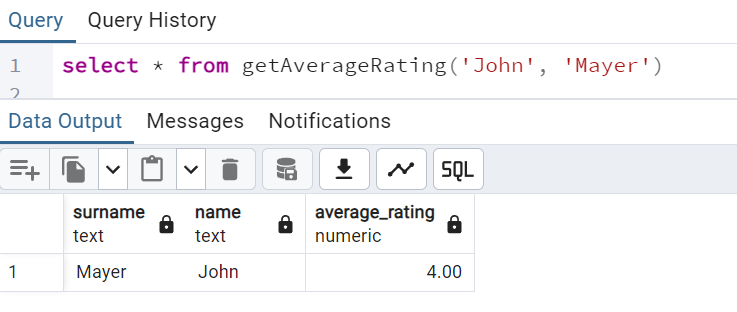


Рисунок 3.2.2 — Результат выполнения скрипта для тестирования функций

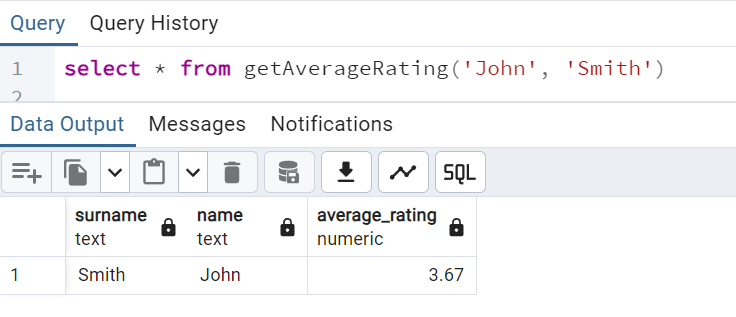


Рисунок 3.2.3 — Результат выполнения скрипта для создания функций

1. GetVisitorPurchases (Visitor\_name text). Функция предназначена для отображения информации о средней сумме покупок посетителя. То есть она принимает имя и фамилию пользователя в качестве входных параметров и затем возвращает таблицу средней суммы покупок посетителя.

CREATE OR REPLACE FUNCTION getVisitorPurchases(visitor\_name text, visitor\_surname text)

RETURNS TABLE (surname text, name text, average\_amount numeric)

AS

$$

BEGIN

RETURN QUERY

select v.surname, v.name, round(avg(vp.amount))

from visitors v

join visitor\_purchases vp

on v.visitor\_id = vp.visitor\_id

where visitor\_name = v.name

and

visitor\_surname = v.surname

group by v.visitor\_id, v.surname, v.name;

END;

$$ LANGUAGE plpgsql;

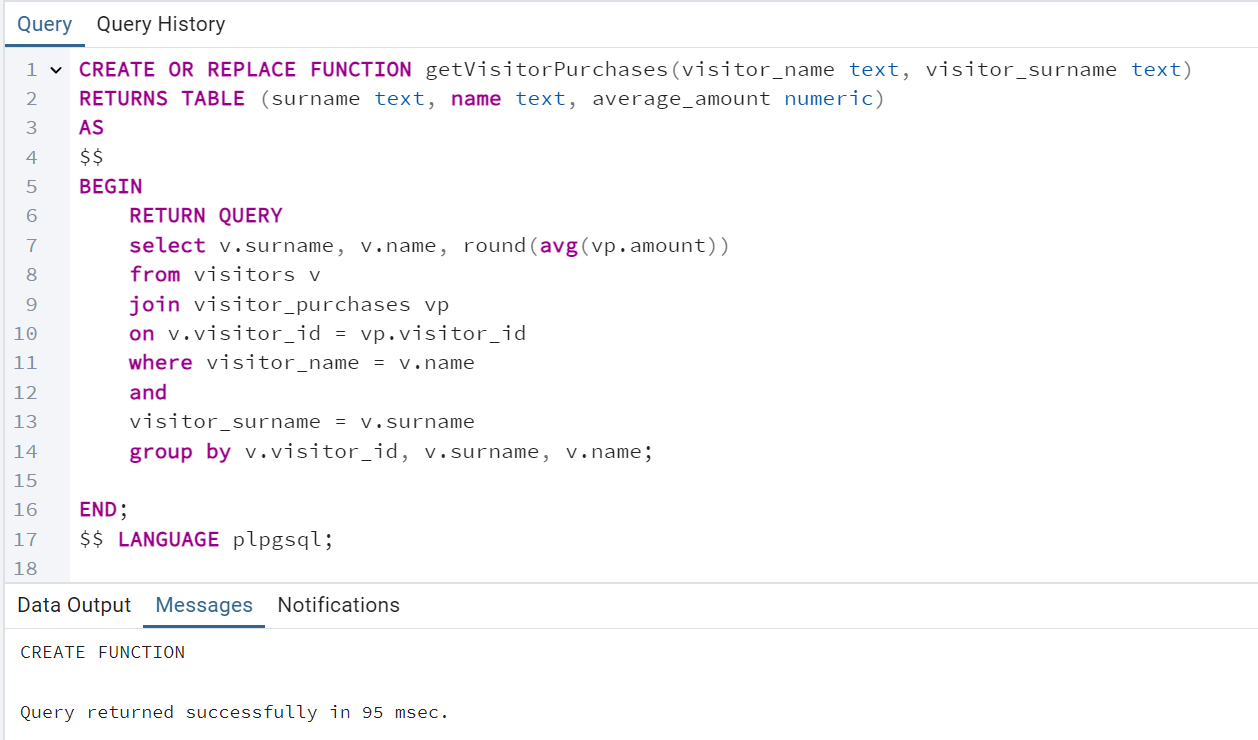
****

Рисунок 3.3.1 — Результат выполнения скрипта для создания функций

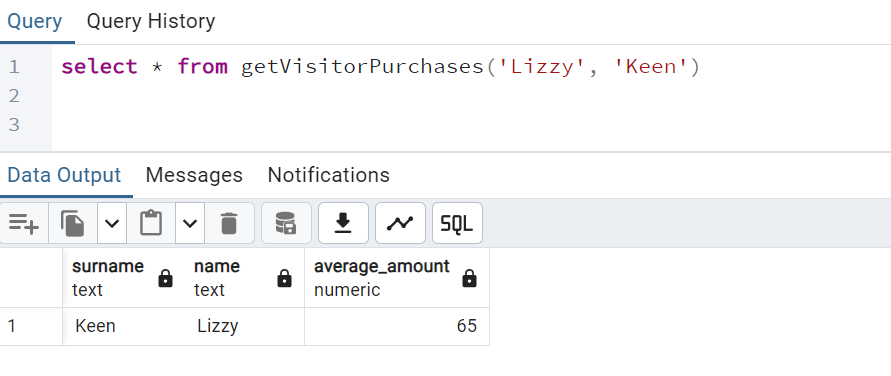
****

Рисунок 3.3.2 — Результат выполнения скрипта для представления функций

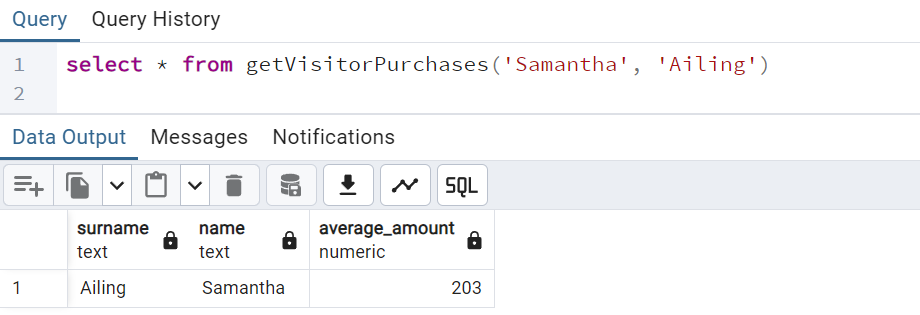
****

Рисунок 3.3.3 — Результат выполнения скрипта для создания функций

## Создание триггеров

Триггеры SQL — это важные инструменты в системах управления базами данных (СУБД), которые позволяют автоматически выполнять набор инструкций SQL при возникновении определенных событий в базе данных, таких как операции INSERT, UPDATE или DELETE [15].

1. User\_Update\_Trigger: Триггер user\_update\_trigger предназначен для отображения уведомления после успешной вставки новых данных в таблицу “visitors”. Это означает, что в нем есть инструкция "AFTER INSERT", которая запускает триггерную функцию user\_insert. Эта функция возвращает выходные данные триггера типа и отображает новую информацию для каждой строки новых данных.

CREATE OR REPLACE FUNCTION user\_insert()

RETURNS TRIGGER AS

$$

BEGIN

RAISE notice 'Update to visitors table:';

raise notice '%',new;

END

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER user\_update\_trigger

AFTER INSERT ON visitors

FOR EACH ROW

EXECUTE FUNCTION user\_insert();

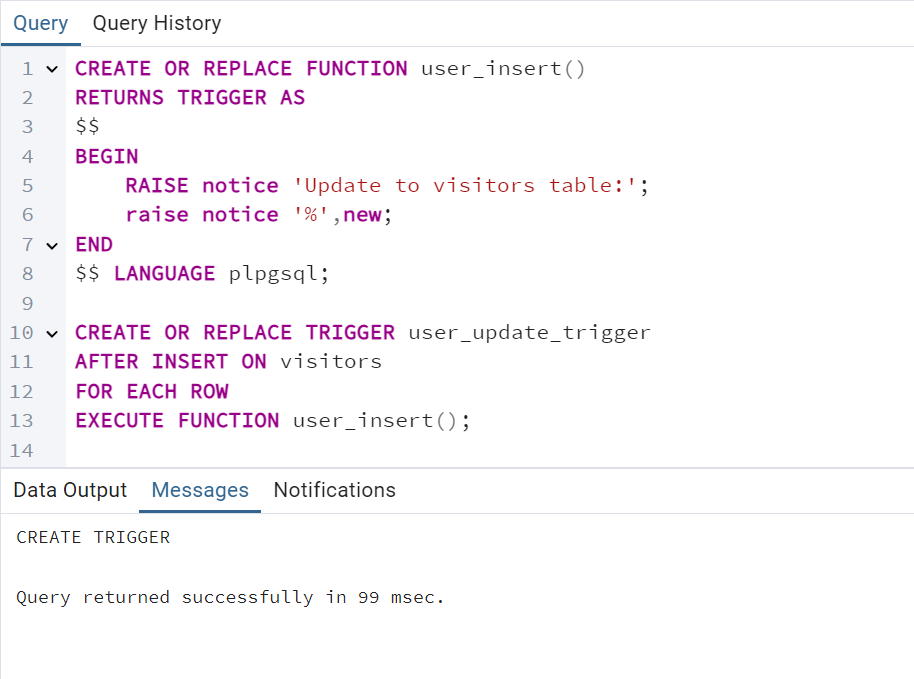


Рисунок 4.1.1 — Результат выполнения скрипта для создания триггера

Чтобы протестировать созданный триггер, мы должны вставить информацию в таблицу посетителей. Скрипт для тестирования кода показан ниже

****

Рисунок 4.1.2 — Результат выполнения скрипта для создания триггера

1. Item\_Update: Триггер Item\_update содержит инструкцию "AFTER UPDATE", которая отображает каждую строку старых и новых данных после успешного обновления данных в таблице Items. Соответствующая триггерная функция update\_details() возвращает выходные данные типа 'trigger' и выдает уведомление, в котором отображаются все обновленные данные.

CREATE OR REPLACE FUNCTION update\_details()

RETURNS TRIGGER AS

$$

BEGIN

RAISE notice 'There has been update to the items table from:';

RAISE notice '%', OLD;

RAISE notice 'Values updated to:';

RAISE notice '%', NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER item\_update

AFTER UPDATE ON items

FOR EACH ROW

EXECUTE FUNCTION update\_details();

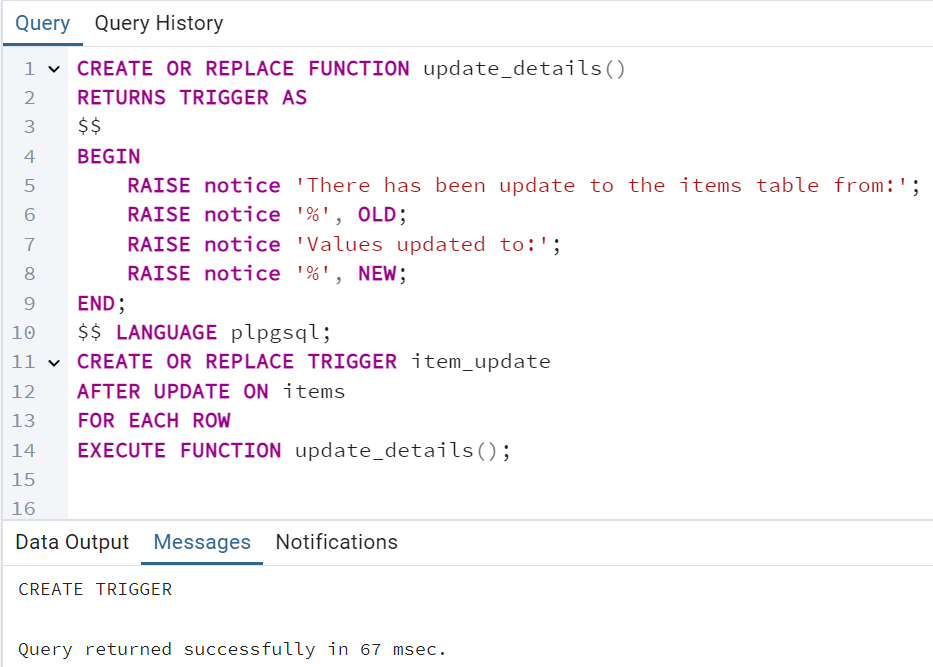
****

Рисунок 4.2.1 — Результат выполнения скрипта для создания триггера

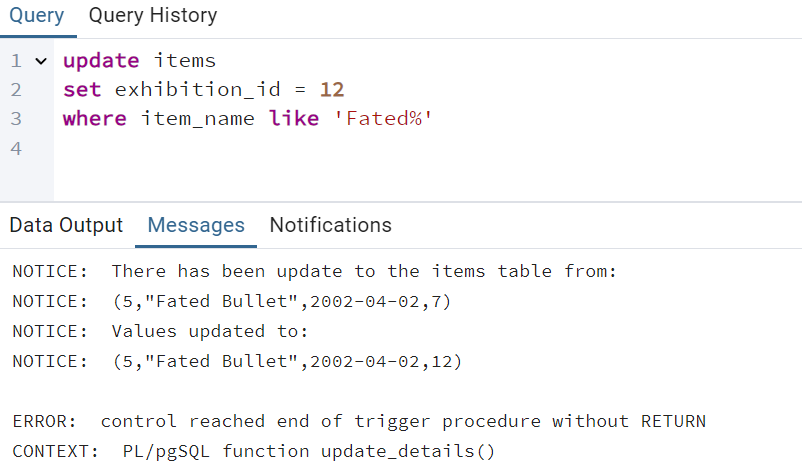
****

Рисунок 4.2.2 — Результат выполнения скрипта для тестирования триггера

1. On\_Feedback\_Update:

Триггер ‘on\_feedback\_update’ выполняет проверку перед вводом или обновлением рейтинга в таблице отзывов. Триггер вызывает функцию, которая выполняется для каждой строки обновленных данных. Соответствующая функция 'feedback\_validate()' содержит два предложения для проверки оценки, указанной в предлагаемом обновлении. Если оценка больше 5, выдается сообщение об ошибке, и пользователю предлагается изменить оценку. Аналогично, если рейтинг ниже 1, в качестве выходных данных отображаются сообщение об ошибке и запрос.

CREATE OR REPLACE FUNCTION feedback\_validation()

RETURNS TRIGGER AS

$$

BEGIN

IF NEW.rating > 5 THEN

RAISE EXCEPTION 'We know we are good but that rating is too high.

Please edit the rating: %', New.rating;

ELSE IF NEW.rating <1 THEN

RAISE EXCEPTION 'Are we that bad? Please edit the rating:%', New.rating;

END IF;

END IF;

RETURN NEW;

END;

$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER on\_feedback\_update

BEFORE UPDATE OR INSERT ON feedback

FOR EACH ROW

EXECUTE FUNCTION feedback\_validate();

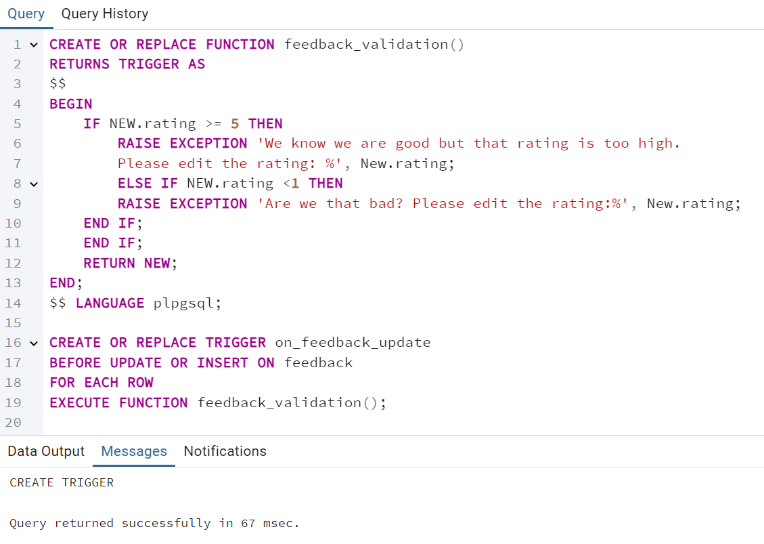


Рисунок 4.3.1 — Результат выполнения скрипта для создания триггера

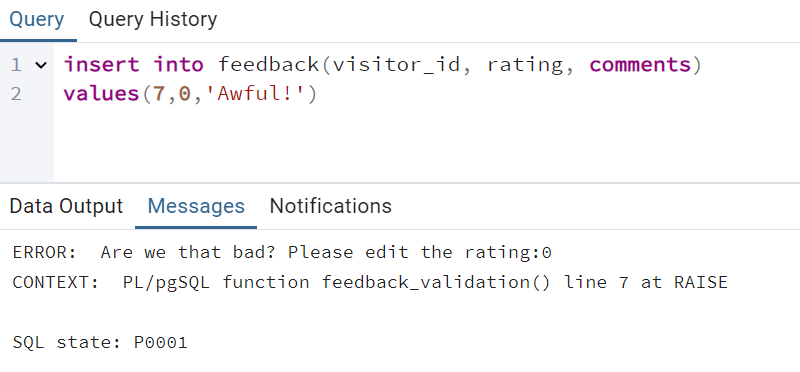
****

Рисунок 4.3.2 — Результат выполнения скрипта для тестирования триггера

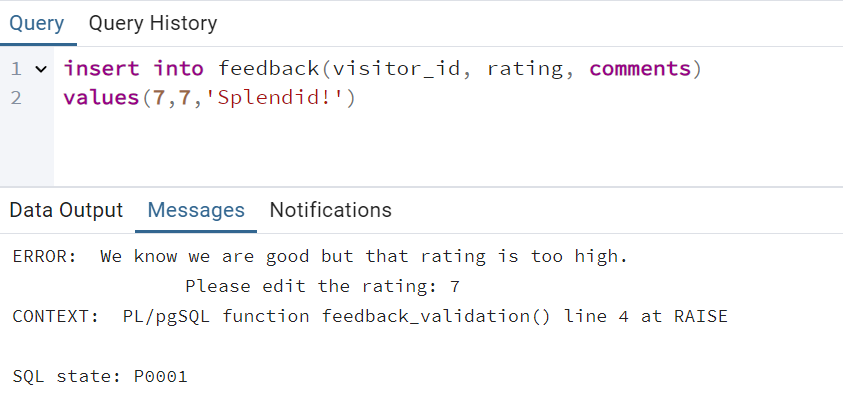
****

Рисунок 4.3.1 — Результат выполнения скрипта для тестирования триггера

# ЗАКЛЮЧЕНИЕ

Системы управления посетителями являются важной частью сектора безопасности и аналитики любой организации. Они упрощают управление и контроль за физическими помещениями организации и обеспечивают сохранность информации и активов в помещениях. Кроме того, они являются важным способом сбора информации о посетителях. Меньше мест используют такие системы в большей степени, чем музеи.

Безопасность музеев и управление экспозициями всегда были и будут предметом озабоченности. Внедрение систем управления посетителями в музеях — это быстрорастущая и хорошо документированная тенденция, и несколько компаний активно используют ее и предлагают эти услуги. Потребность в повышении безопасности и контроля требует, чтобы системы работали со все большим количеством данных, и это приводит к увеличению потребности в стабильном и масштабируемом подходе к обработке этих данных. По этой причине проект включал в себя не только исследование изучаемой области, но и создание и внедрение базы данных, которая удовлетворяла бы основные потребности стандартного музея.

Эти потребности включают, но не ограничиваются ими:

1. Эффективное хранение пользовательской информации для удобства использования
2. Эффективная увязка информации о билетах и экспонатах
3. Хранение и представление данных о поведении посетителей. Сюда входят данные о покупках посетителей и посещенных выставках.

Разработанная база данных позволяет системе управления посетителями автоматически сохранять данные о посетителях. Благодаря структурированному процессу проектирования, который включал концептуальный дизайн, логический дизайн и физический дизайн базы данных, полученная база данных отвечала основным требованиям. Он предназначен для хранения информации об экспонатах, базовой информации о билетах и привязки всей информации о посетителях к музейным системам

В дальнейшем можно было бы усовершенствовать структуру базы данных, включив в нее механизмы хранения большего количества информации для детального анализа привычек посетителей музея. Это помогло бы формировать музейную политику в соответствии с отзывами посетителей. Можно провести дальнейшие исследования для улучшения информационной безопасности, добавив ограничения доступа в рамках политики защиты данных. Таким образом, вся информация о музеях и посетителях, хранящаяся в базе данных, будет защищена от нежелательного доступа.

В этом случае, благодаря дополнительным усовершенствованиям базовой структуры базы данных, представленной в этом проекте, может быть внедрена масштабируемая, надежная и эффективная система управления данными для поддержки любой системы управления посетителями музея.

# СПИСОК ЛИТЕРАТУРЫ

1.Эйлин Х. Г. Музеи и их посетители глава 8, стр. 171

2. Автономная система управления базами данных (СУБД) R13 / глава 1 Департамента компьютерных наук и инженерии стр. 4

3. Питер П. Ч. Подход на основе сущностных отношений к информационному моделированию и анализу, Института ER (1981); Библиотека Массачусетского технологического института; Том II. 1 глава 2.1 страницы 10-12

4. Элмасри Н. Основы систем баз данных, Pearson Education, глава 7.3.1; Сущности и атрибуты, Том 1. 2 с 203 — 204

5. Элмасри Н. Основы систем баз данных, Pearson Education, глава 7.3.1; Сущности и атрибуты, Том 1. 2 с 204 — 206

6. Риккардо Т. Системы баз данных: Полная книга, Второе издание, стр. 141

7. Гарсия-Молина Г. Ульман Д. Д. Уидом Д. Системы реляционных баз данных: Глава 1.1.2; факультет компьютерных наук Стэнфордского университета.

с 54-57

8. Системы баз данных: концепции, языки и архитектура, Часть 1. Реляционные базы данных Глава 2.1 с 15-20.

9. Нормализация базы данных CSC 443 Университета Торонто с 1 по 4.

10. Шарма Н. Перниу Л. Чонг Р. Ф. Основы работы с базами данных, Глава 4, с. 89 – 103.

11. Введение в системы баз данных, глава 9.1, с. 263.

12. Герханы Н. Система баз данных CS 431/ Технологический институт Нью-Джерси, глава SQL, разделы 63-66.

13. Абитебул С. Халл А Основы баз данных.

14. SQL, Третье издание, Энди Оппель, Роберт Шелдон, стр. 126

15. Спилпейннинг Д. Галлус Д. П. До Моделирование баз данных и проектирование реляционных баз данных, ORACLE, глава 12, стр. 183

Интернет-источники

* + https://damedia.co.uk/visitor-management-systems-for-museum
  + <https://www.geeksforgeeks.org/what-is-database/>
  + <https://www.tutorialspoint.com/conceptual-database-design>
  + <https://stackoverflow.com/questions/4279089/what-is-the-difference-between-logical-data-model-and-conceptual-data-model>
  + https://github.com/sujeet-agrahari/awesome-database-design#awesome-database-design-
  + <https://www.ibm.com/docs/en/db2-for-zos/12?topic=model-entities-different-types-relationships>
  + <https://www.geeksforgeeks.org/database-design-fundamentals/>
  + <https://en.wikipedia.org/wiki/Database>
  + <https://www.geeksforgeeks.org/introduction-of-er-model/>
  + <https://www.geeksforgeeks.org/introduction-of-er-model/>
  + <https://www.ibm.com/docs/en/db2-for-zos/12?topic=model-entities-different-types-relationships>
  + https://study.com/learn/lesson/sql-functions-examples-uses