|  |
| --- |
| МИНОБРНАУКИ РОССИИ |
| Федеральное государственное бюджетное образовательное учреждение высшего образования **«МИРЭА − Российский технологический университет»**  **РТУ МИРЭА** |

**Институт информационных технологий (ИИТ)**

**Кафедра инструментального и прикладного программного обеспечения (ИиИППО)**

**ОТЧЕТ ПО ПРАКТИЧЕСКИМ РАБОТАМ №1 - 4**

по дисциплине «Разработка баз данных»

|  |  |  |  |
| --- | --- | --- | --- |
| Студент группы ИНБО-01-17 | *ИКБО 33-22, Шило Юрий Сергеевич* | (подпись) | |
| Преподаватель | *Баев Игорь Борисович* | (подпись) | |
| Отчет представлен | «\_\_\_»\_\_\_\_\_\_\_\_202\_\_г. | |  | |

Москва 2024 г.

**СОЗДАНИЕ БАЗЫ ДАННЫХ И ТАБЛИЦ В НЕЙ**

Построим физическую модель нашей базы данных. Для этого мы будем использовать инструмент dbForge Studio.

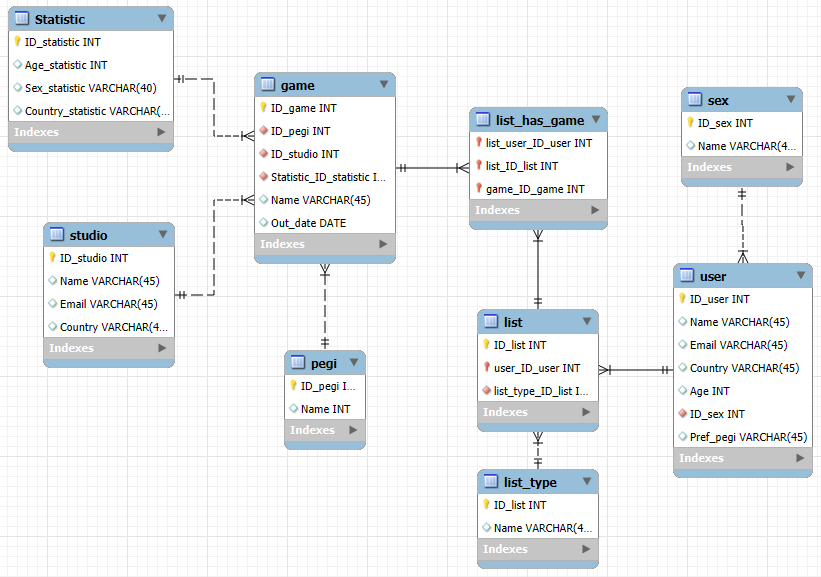


Рисунок 1 – Физическая модель реализуемой базы данных

Нам надо создать базу данных, которую назовем GameCalendar. Для этого в SQL существует оператор create database. Перед началом работы с базой данных надо указать серверу в какуй именно БД мы создаем таблицы, т.е. надо выбрать БД для работы. Для этого используется оператор use.



Рисунок 2 – Процесс создания и выбора базы данных

Для создания таблиц в SQL существует оператор create table. Создадим таблицу Statistic.



Рисунок 3 – Процесс создания таблицы «Statistic»

Создадим таблицу Studio.

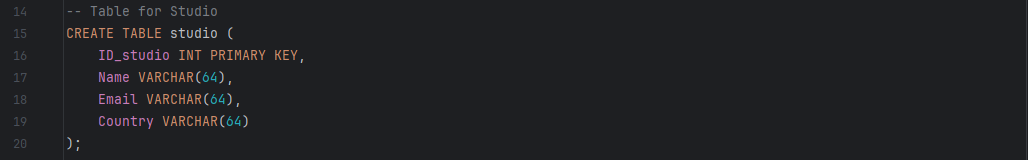
****

Рисунок 4 – Процесс создания таблицы «Studio»

Создадим таблицу PEGI.

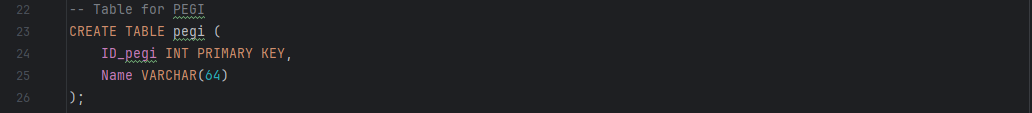
****

Рисунок 5 – Процесс создания таблицы «PEGI»

Создадим таблицу Sex.

****

Рисунок 6 – Процесс создания таблицы «Sex»

Создадим таблицу Game.

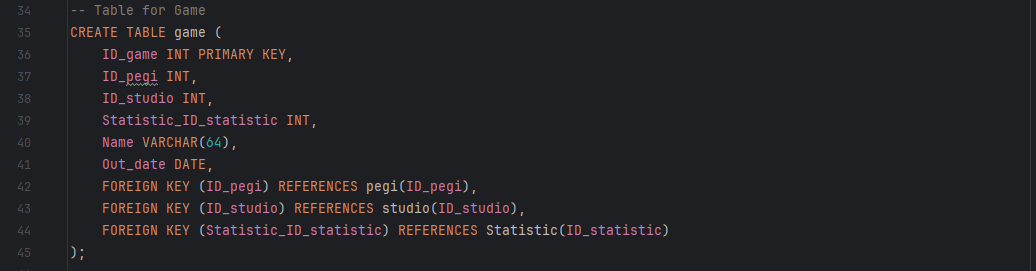
****

Рисунок 7 – Процесс создания таблицы «Game»

Создадим таблицу List Type.

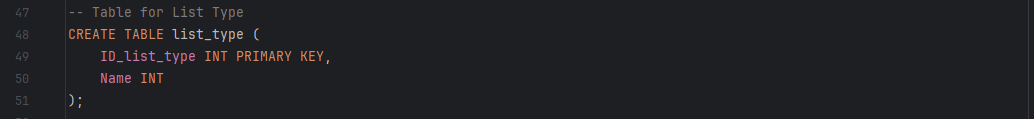
****

Рисунок 8 – Процесс создания таблицы «List Type»

Создадим таблицу User.

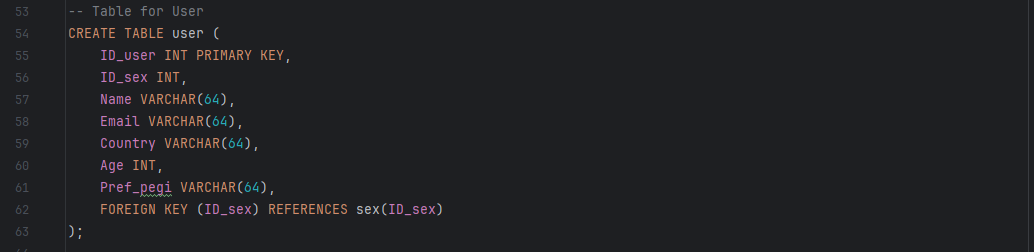
****

Рисунок 9 – Процесс создания таблицы «User»

Создадим таблицу List.

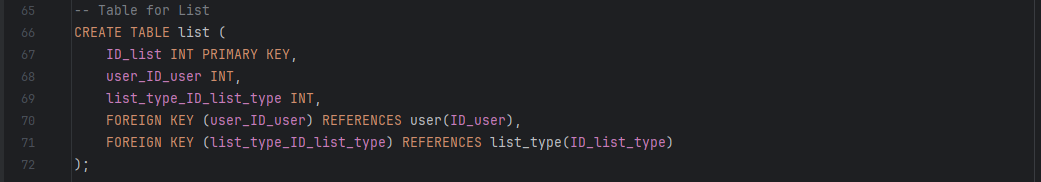
****

Рисунок 10 – Процесс создания таблицы «List»

Создадим таблицу List Has Game.

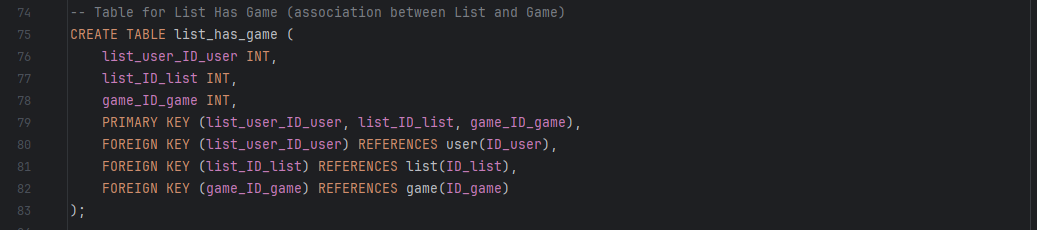
****

Рисунок 11 – Процесс создания таблицы «List Has Game»

В SQL существует возможность посмотреть какие БД у нас существуют, какие таблицы в них присутствуют, и какие столбцы эти таблицы содержат.

show databases — показать все имеющиеся БД.

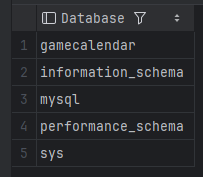
****

Рисунок 12 – Описание всех имеющиеся БД

show tables — показать список таблиц текущей БД (предварительно ее надо выбрать с помощью оператора use).

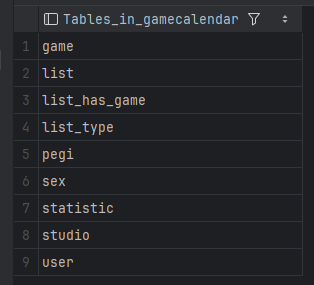
****

Рисунок 13 – Cписок таблиц БД «GameCalendar»

describe имя\_таблицы — показать описание столбцов указанной таблицы. Просмотрим таблицу statistic.

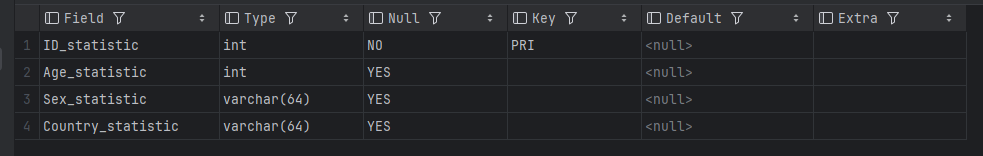
****

Рисунок 14 – Описание столбцов в таблице «statistic»

Просмотрим таблицу studio.

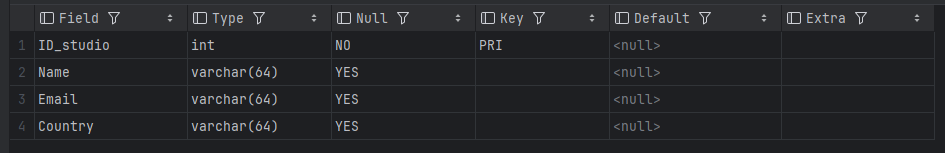
****

Рисунок 15 – Описание столбцов в таблице «studio»

Просмотрим таблицу pegi.

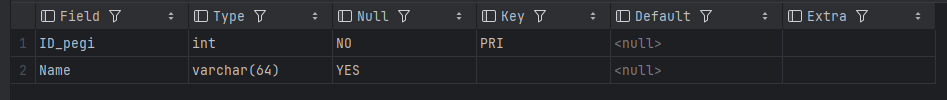
****

Рисунок 16 – Описание столбцов в таблице «pegi»

Просмотрим таблицу sex.

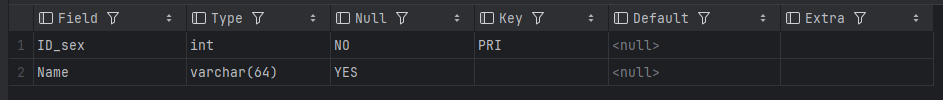
****

Рисунок 17 – Описание столбцов в таблице «sex»

Просмотрим таблицу game.

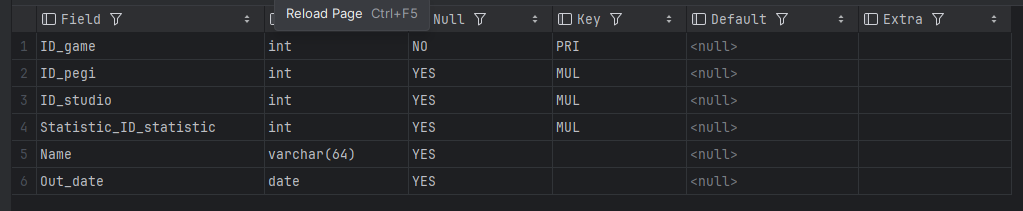
****

Рисунок 18 – Описание столбцов в таблице «game»

Просмотрим таблицу list\_type.

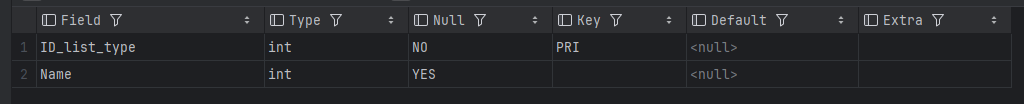
****

Рисунок 19 – Описание столбцов в таблице «list\_type»

Просмотрим таблицу user.

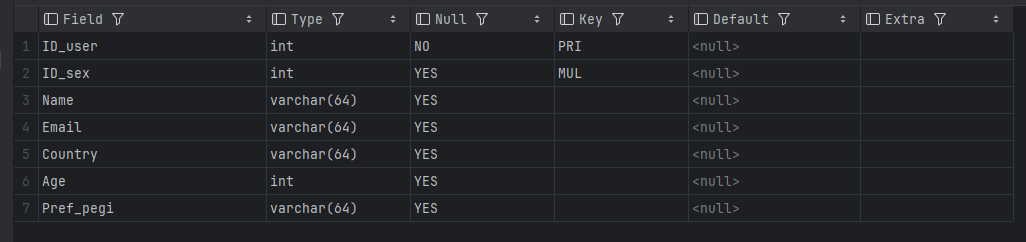
****

Рисунок 20 – Описание столбцов в таблице «user»

Просмотрим таблицу list.

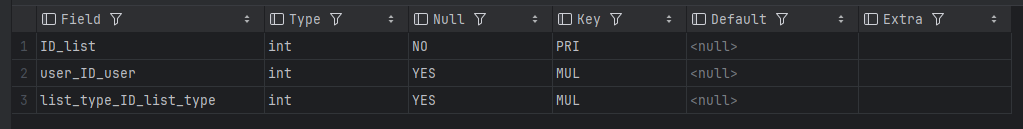
****

Рисунок 21 – Описание столбцов в таблице «list»

Просмотрим таблицу list\_has\_game.

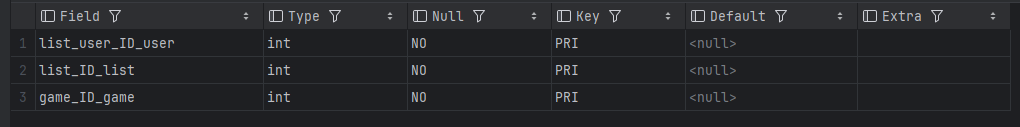
****

Рисунок 22 – Описание столбцов в таблице «list\_has\_game»

**ДОБАВЛЕНИЕ ДАННЫХ В ТАБЛИЦЫ БАЗЫ ДАННЫХ**

Теперь нам необходимо внести данные в наши таблицы. На сайтах, вы обычно вводите информацию в какие-нибудь html-формы, затем сценарий на каком-либо языке (php, java…) извлекает эти данные из формы и заносит их в БД. Делает он это посредством SQL-запроса на внесение данных в базу.

Для этого используется оператор INSERT. Добавим данные в нашу таблицу statistic.

****

Рисунок 1 – Добавление данных в таблицу «statistic»

Добавим данные в нашу таблицу studio.

****

Рисунок 2 – Добавление данных в таблицу «studio»

Добавим данные в нашу таблицу pegi.

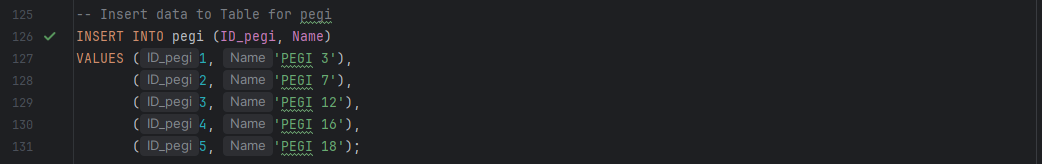
****

Рисунок 3 – Добавление данных в таблицу «pegi»

Добавим данные в нашу таблицу sex.

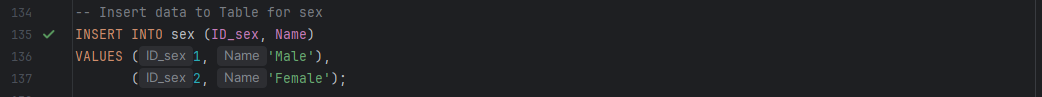
****

Рисунок 4 – Добавление данных в таблицу «sex»

Добавим данные в нашу таблицу game.

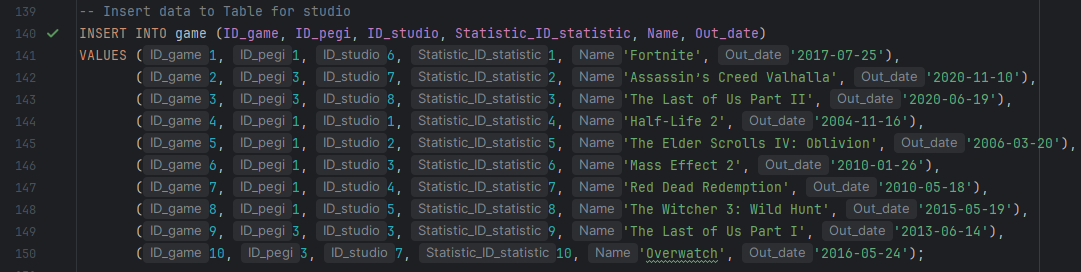
****

Рисунок 5 – Добавление данных в таблицу «game»

Добавим данные в нашу таблицу list\_type.

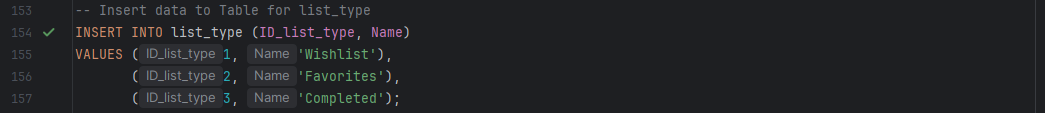
****

Рисунок 6 – Добавление данных в таблицу «list\_type»

Добавим данные в нашу таблицу user.

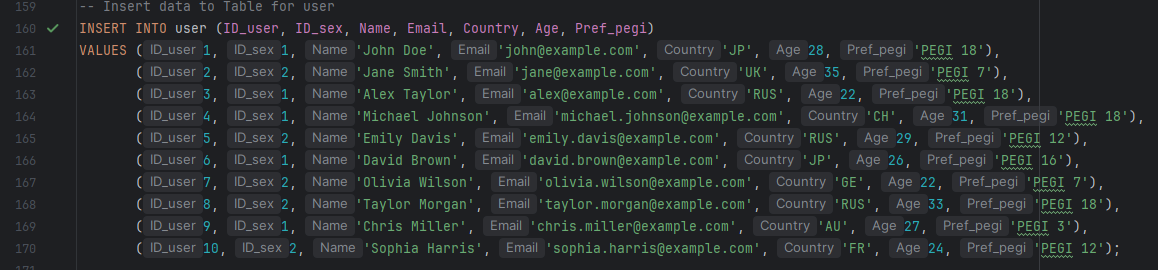
****

Рисунок 7 – Добавление данных в таблицу «user»

Добавим данные в нашу таблицу list.

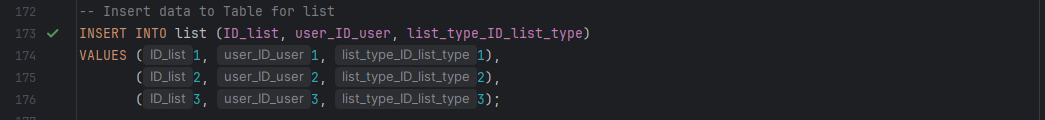
****

Рисунок 8 – Добавление данных в таблицу «list»

Добавим данные в нашу таблицу list\_has\_game.

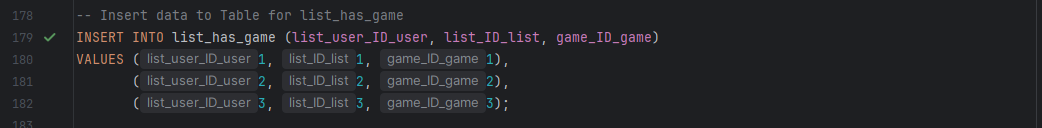
****

Рисунок 9 – Добавление данных в таблицу «list\_has\_game»

Чтобы посмотреть, какие данные у нас содержатся в таблицах. В SQL существует оператор SELECT. Просмотрим данные, которые мы добавили в таблицу statistic.

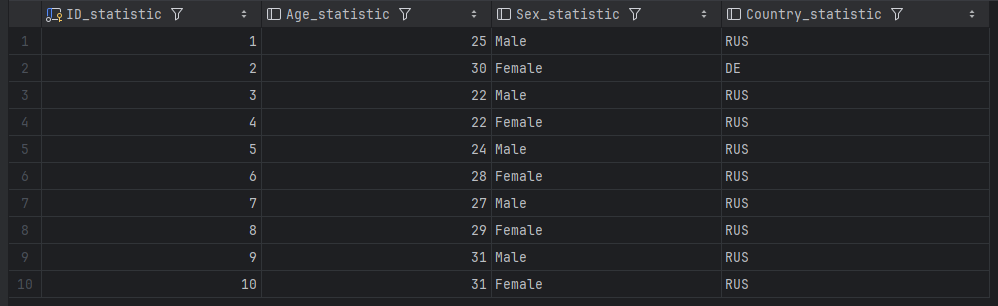
****

Рисунок 10 – Добавленные данные в таблицу «statistic»

Просмотрим данные, которые мы добавили в таблицу studio.

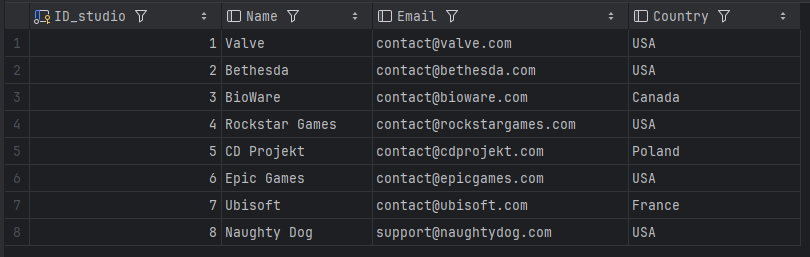
****

Рисунок 11 – Добавленные данные в таблицу «studio»

Просмотрим данные, которые мы добавили в таблицу pegi.

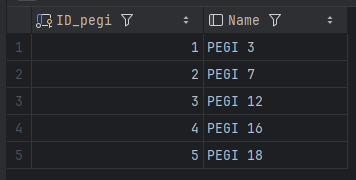
****

Рисунок 12 – Добавленные данные в таблицу «pegi»

Просмотрим данные, которые мы добавили в таблицу sex.

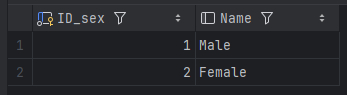
****

Рисунок 13 – Добавленные данные в таблицу «sex»

Просмотрим данные, которые мы добавили в таблицу game.

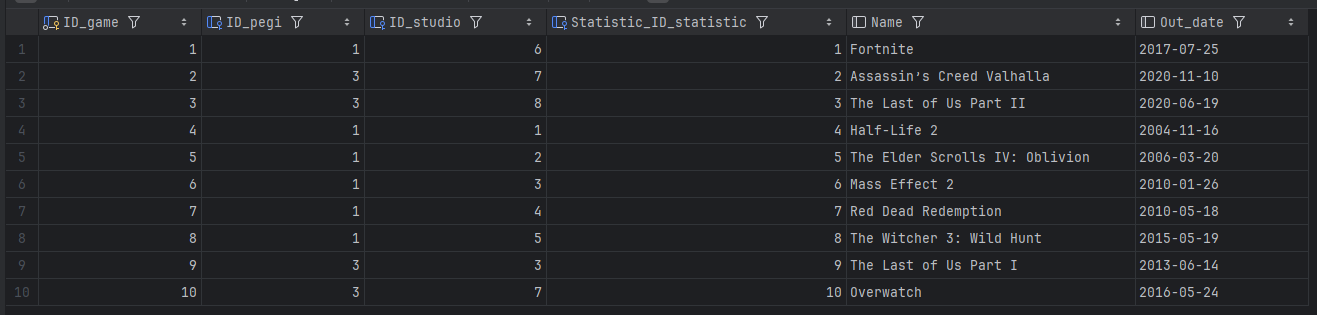
****

Рисунок 14 – Добавленные данные в таблицу «game»

Просмотрим данные, которые мы добавили в таблицу list\_type.

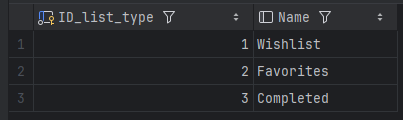
****

Рисунок 15 – Добавленные данные в таблицу «list\_type»

Просмотрим данные, которые мы добавили в таблицу user.

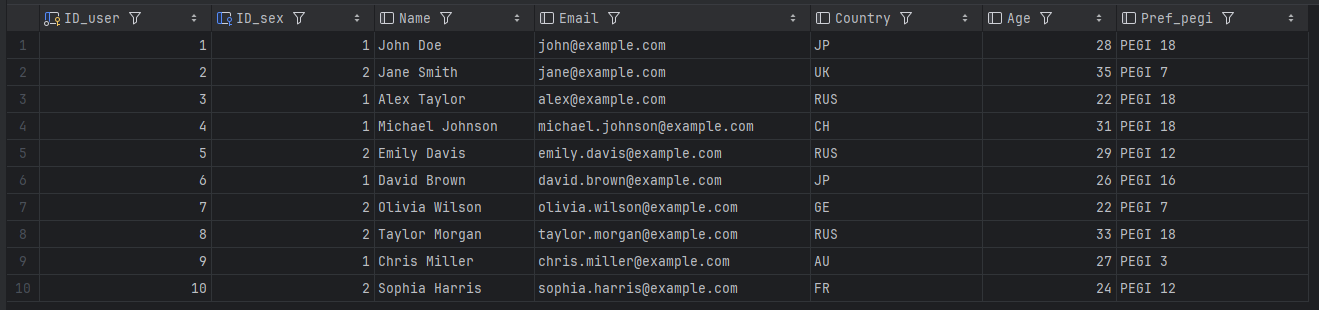
****

Рисунок 16 – Добавленные данные в таблицу «user»

Просмотрим данные, которые мы добавили в таблицу list.

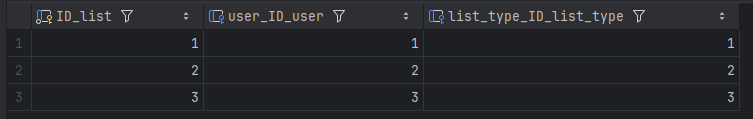
****

Рисунок 17 – Добавленные данные в таблицу «list»

Просмотрим данные, которые мы добавили в таблицу list\_has\_game.

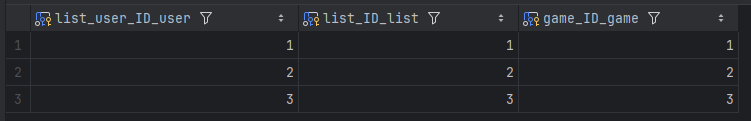
****

Рисунок 18 – Добавленные данные в таблицу «list\_has\_game»

**ВЫБОРКА И СОРТИРОВКА ДАННЫХ**

Очень часто бывает, что все информация из таблицы не нужна. Например, необходимо узнать, какие темы были созданы определеным пользователем. Для этого в SQL есть ключевое слово WHERE и специальные операторы.

Применим оператор равно. При его использовании отбираются значения равные указанному.



Рисунок 1 – Просмотр таблицы user с применением оператора ID\_sex = 1

Применим оператор больше. При его использовании отбираются значения больше указанного.

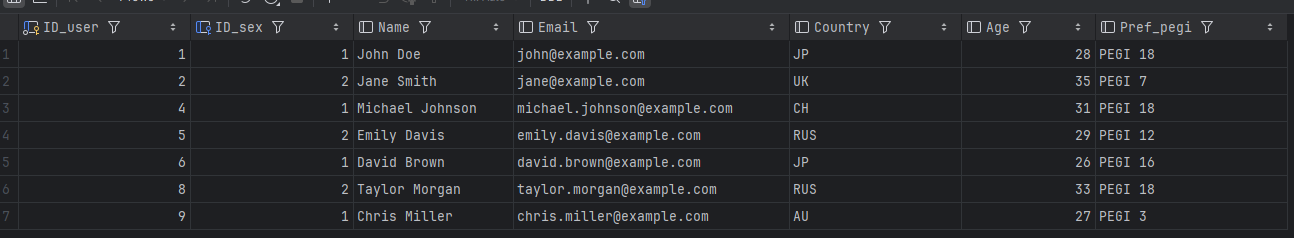


Рисунок 2 – Просмотр таблицы user с применением оператора Age > 25

Применим оператор меньше. При его использовании отбираются значения меньше указанного.



Рисунок 3 – Просмотр таблицы user с применением оператора Age < 25;

Применим оператор больше или равно. При его использовании отбираются значения большие и равные указанному.

****

Рисунок 4 – Просмотр таблицы user с применением оператора Age >= 25;

Применим оператор меньше или равно. При его использовании отбираются значения меньшие и равные указанному.

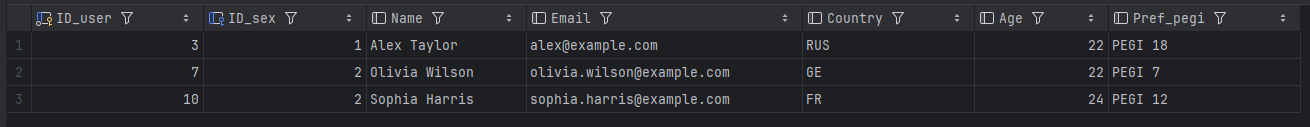
****

Рисунок 5 – Просмотр таблицы user с применением оператора Age <= 25;

Применим оператор IS NOT NULL. При его использовании отбираются строки, имеющие значения в указанном поле.

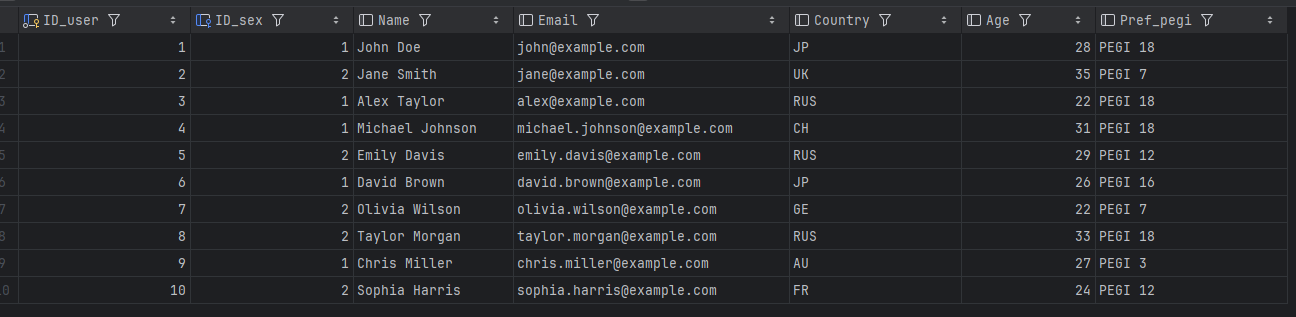
****

Рисунок 6 – Просмотр таблицы user с применением оператора ID\_user IS NOT NULL

Применим оператор IS NULL. При его использовании отбираются строки, не имеющие значения в указанном поле.

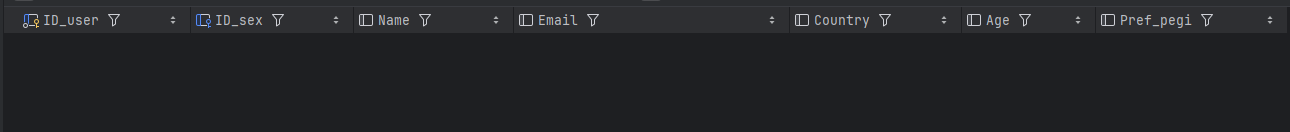
****

Рисунок 7 – Просмотр таблицы user с применением оператора ID\_user IS NULL;

Применим оператор BETWEEN (между). При его использовании отбираются значения, находящиеся между указанными.

****

Рисунок 8 – Просмотр таблицы user с применением оператора ID\_user BETWEEN 3 AND 6

Применим оператор IN (значение содержится). При его использовании отбираются значения, соответствующие указанным.

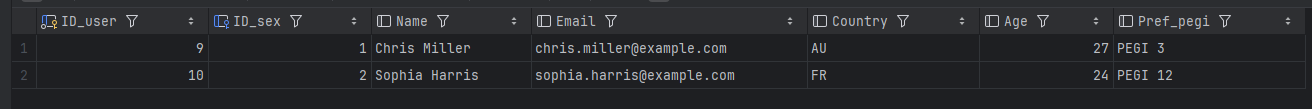
****

Рисунок 9 – Просмотр таблицы user с применением оператора Age IN (24, 27)

Применим оператор NOT IN (значение не содержится). При его использовании отбираются значения, кроме указанных.

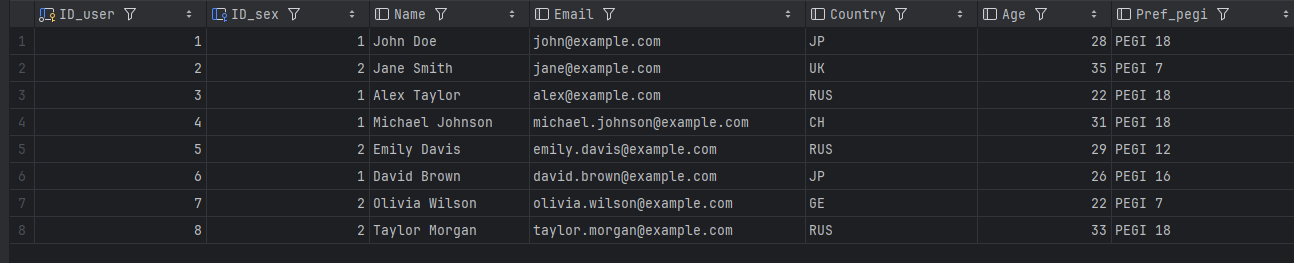
****

Рисунок 10 – Просмотр таблицы user с применением оператора Age NOT IN (24, 27);

Применим оператор LIKE (соответствие). При его использовании отбираются значения, соответствующие образцу.

****

Рисунок 11 – Просмотр таблицы user с применением оператора Name LIKE '%il%';

Применим оператор NOT LIKE (не соответствие). При его использовании отбираются значения, не соответствующие образцу.

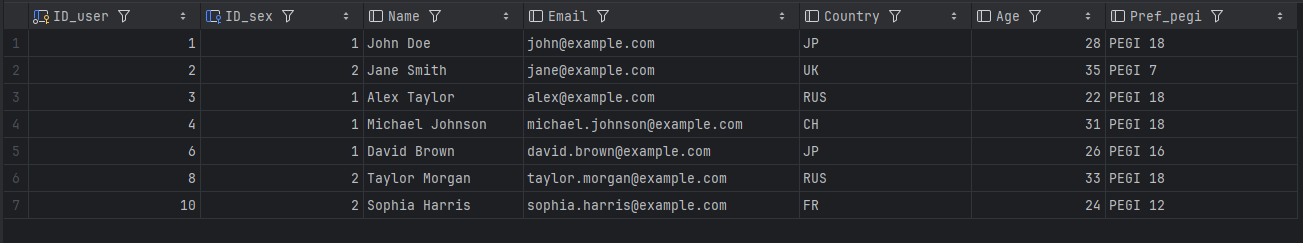
****

Рисунок 12 – Просмотр таблицы user с применением оператора Name NOT LIKE '%il%';

Для сортировки в SQL существует ключевое слово ORDER BY после которого указывается имя столбца, по которому будет происходить сортировка.

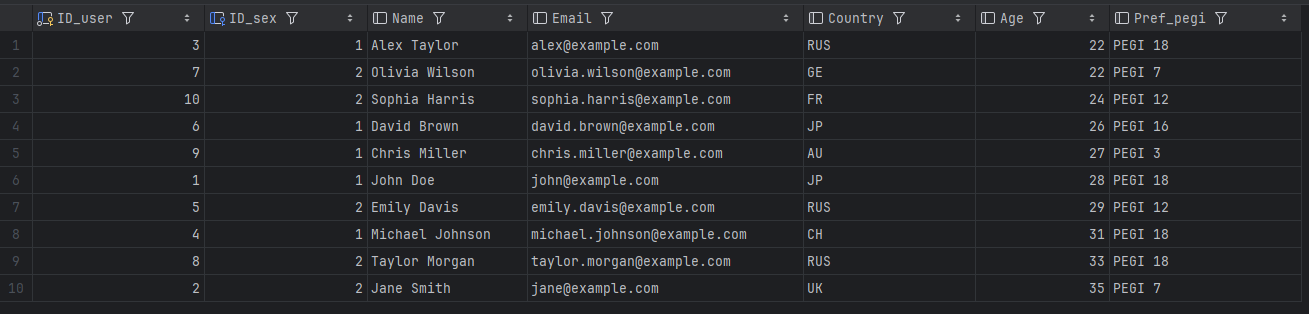
****

Рисунок 13 – Отсортированные данные по возрастанию по столбцу Age

По умолчанию сортировка идет по возрастанию, но это можно изменить, добавив ключевое слово DESC.

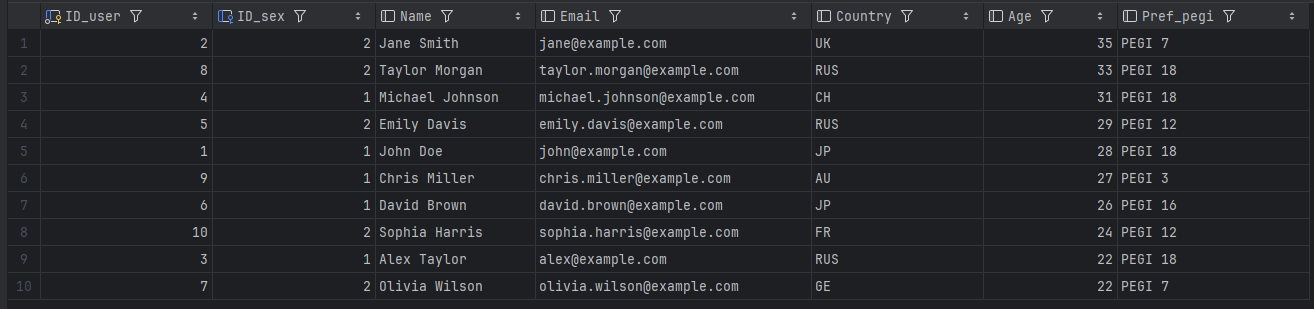
****

Рисунок 14 – Отсортированные данные по убыванию по столбцу Age

**ИЗМЕНЕНИЕ ДАННЫХ В ТАБЛИЦЕ**

Для добавления столбцов в таблицу используется оператор ALTER TABLE — ADD COLUMN.

Для того, чтобы указать местоположение столбца используются ключевые слова: FIRST — новый столбец будет первым, и AFTER — указывает после какого столбца поместить новый.

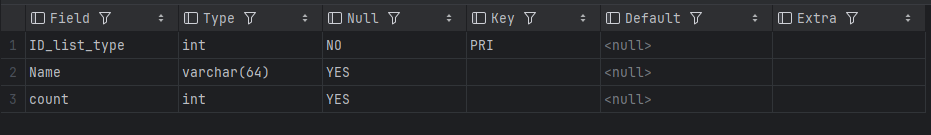
****

Рисунок 1 – Добавление столбца count в таблицу list\_type

Для изменения имени существующего столбца используется оператор CHANGE.

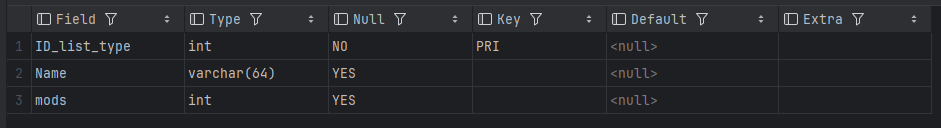
****

Рисунок 2 – Изменение столбца count на mods в таблице list\_type

Рассмотрим — оператор DELETE, который позволяет удалять строки из таблицы.

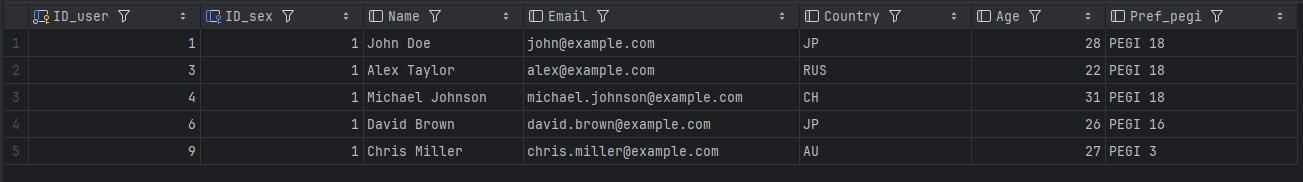
****

Рисунок 3 – Удаление всех данных из таблицы user, где ID\_sex = 2

**РЕАЛИЦИОННАЯ АЛГЕБРА**

**Операция проекции**. Осуществляется выбор только части по лей таблицы, т.е. производится вертикальная выборка данных.

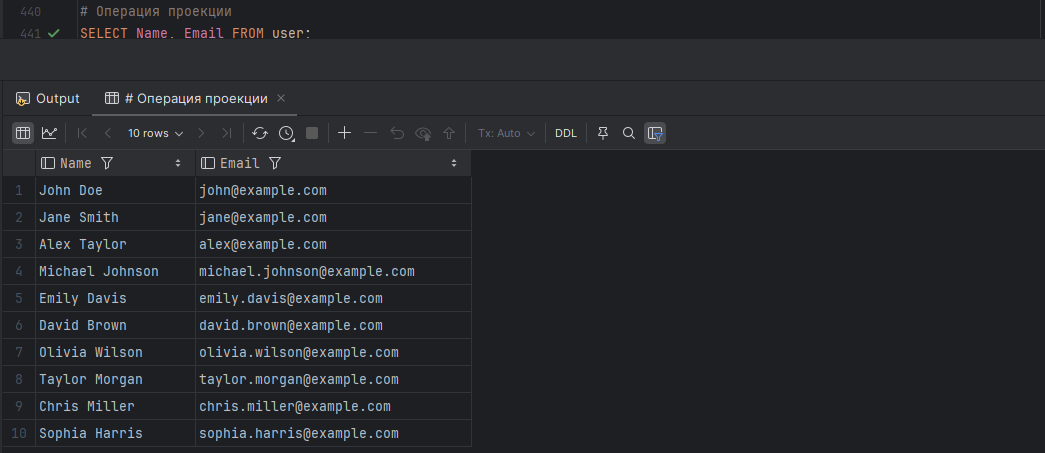


Рисунок 1 – Применение операции проекции

**Операция селекции.** Осуществляется горизонтальная выборка – в результат попадают только записи, удовлетворяющие условию.

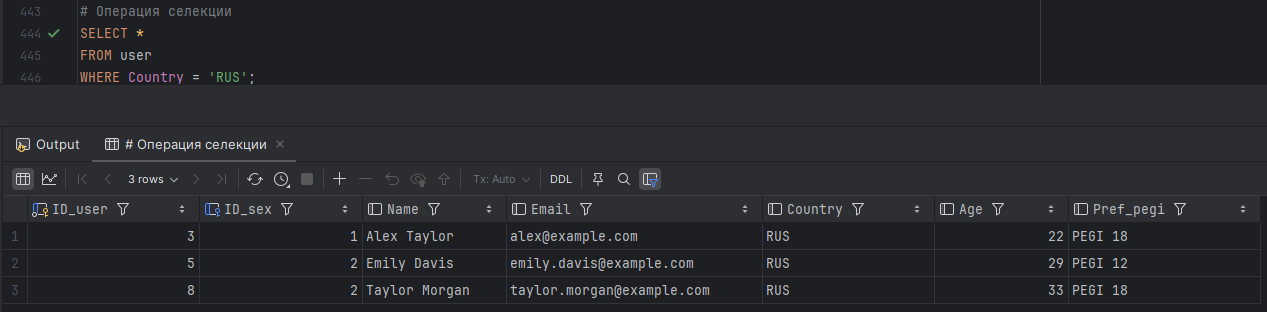


Рисунок 2 – Применение операции селекции

**Операции соединения.** Здесь следует выделить декартово произведение и на его основе соединение по условию, а также естественное соединение (по одноименным полям или равенству полей с одинаковым смыслом).

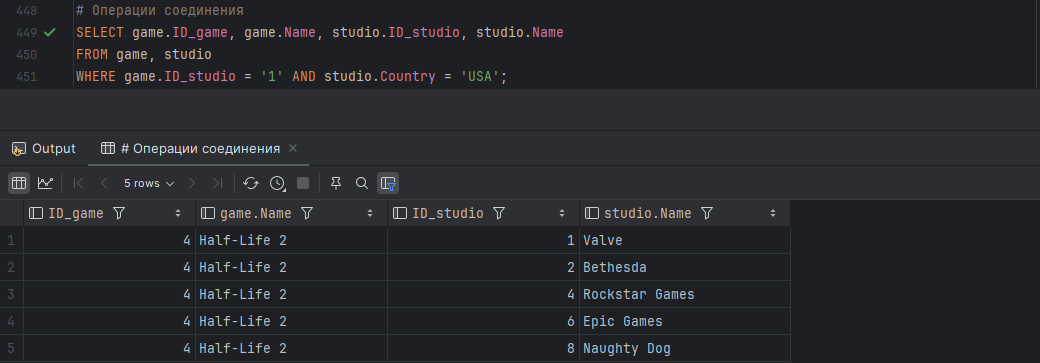


Рисунок 2 – Применение операции соединения

**Операция объединения.** Теоретико-множественные операции часто можно записать с помощью логических операций, примененных в конструкции WHERE запроса. Например, нужно получить список зачетов и экзаменов, которые сдают студенты 901 или 902 групп в 1 семестре. Таким образом, нужно объединить два множества, соответствующие двум разным группам. Объединение можно задать с помощью логического ИЛИ.

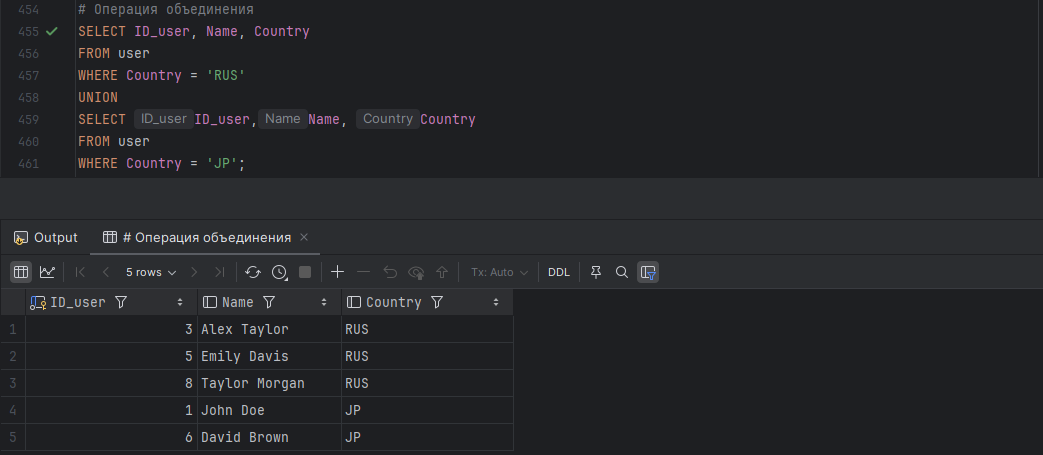


Рисунок 2 – Применение операции объединения

**Операция пересечения.** В простых случаях эту операцию можно описать с помощью логической операции AND. В более сложных случаях эта операция определяется чаще всего с помощью подзапроса и ключевого слова EXISTS, которое показывает наличие похожего элемента во множестве, которое задается подзапросом.

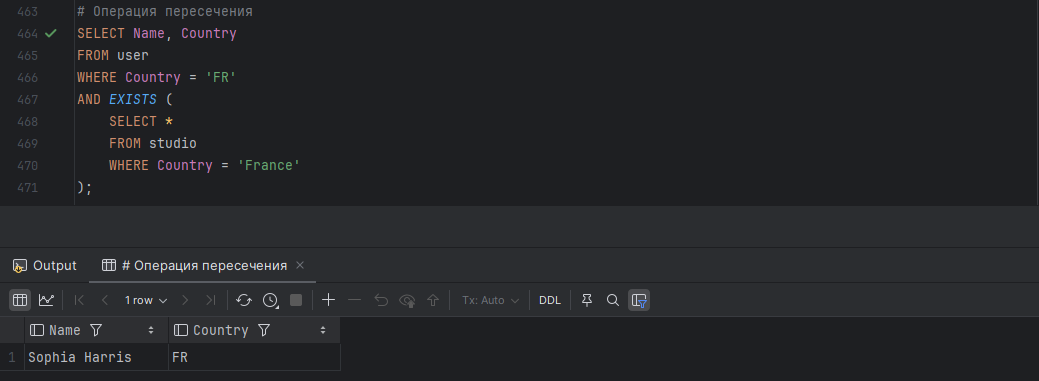


Рисунок 2 – Применение операции пересечения

**Операция разности.** Эта операция также определяется часто с помощью подзапроса с ключевым словом NOT EXISTS, которое показывает отсутствие элемента во множестве, задаваемом подзапросом.

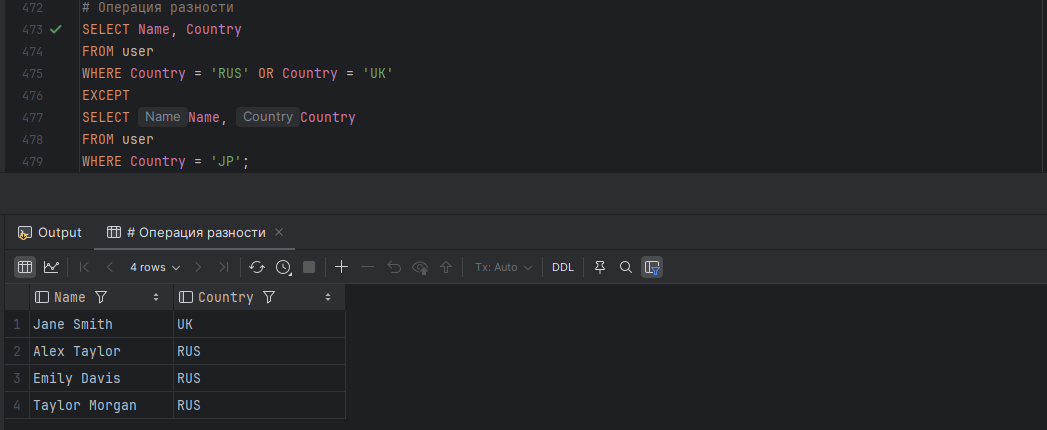


Рисунок 2 – Применение операции разности

**Операция группировки.** Эта операция связана со своеобразной сверткой таблицы по полям группировки. Помимо полей группировки результат запроса может содержать итоговые агрегирующие функции по группам (COUNT, SUM, AVG, MAX, MIN).

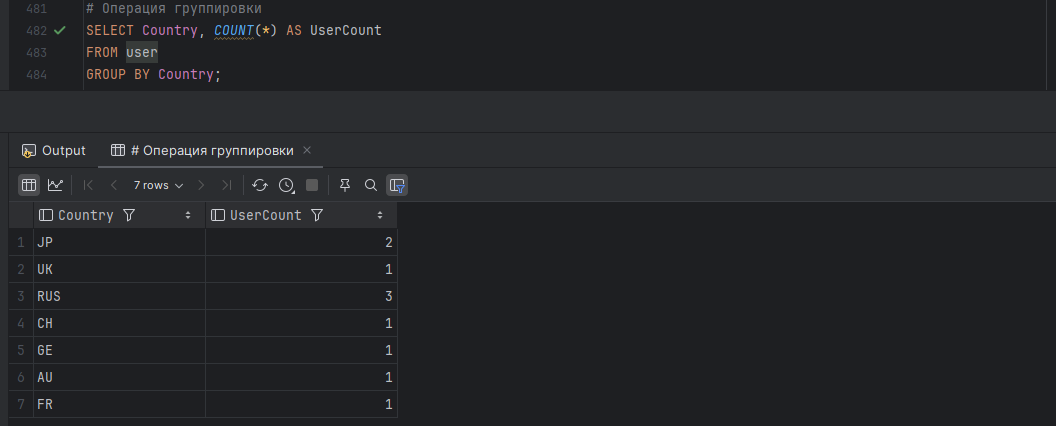


Рисунок 2 – Применение операции группировки

**Операция сортировки.** Вывести всех преподавателей, которым сдают студенты зачеты-экзамены в первом семестре, в порядке убывания количества зачетов-экзаменов. Для этого следует сначала выбрать нужные элементы таблицы Sessions, затем осуществить естественное соединение полученной таблицы с таблицей Teachers, после чего производится группировка записей в результате запроса и последующая сортировка.

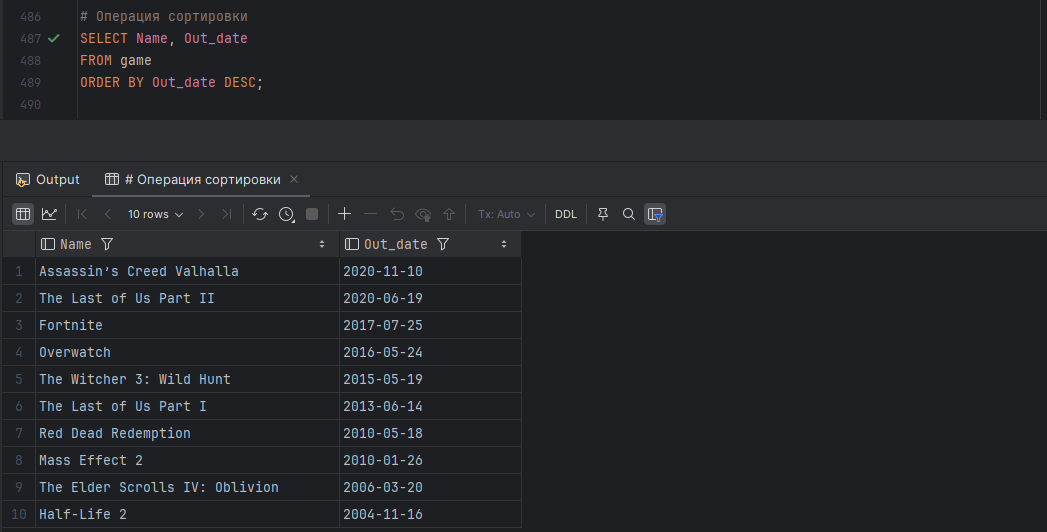


Рисунок 2 – Применение операции сортировки

**Операция деления.** Это самая нетривиальная операция реляционной алгебры, которая обычно применяется тогда, когда требуется найти все записи первой таблицы, которые соединяются естественным образом со всеми записями второй таблицы. Например, нам требуется найти тех преподавателей, которым должны сдать в первом семестре зачеты-экзамены студенты всех групп факультета. Запрос получается достаточно сложный и он связан с выполнением двух операций разности (первая разность - из всевозможных комбинаций групп и преподавателей вычитаются реальные комбинации этих полей, т.е. 46 результатом становятся всевозможные нереальные пары, вторая разность – выбираются преподаватели, которые в нереальных парах не присутствуют).

**ХРАНИМЫЕ ПРОЦЕДУРЫ, ФУНКЦИИ И ТРИГГЕРЫ**

Хранимые процедуры, функции и триггеры вводятся в базу данных для обеспечения бизнес-логики приложения на уровне серверной его компоненты. Обычно хранимые процедуры и функции представляют собой утилиты, которые определенным образом обрабатывают данные или реализуют достаточно сложный алгоритм вычисления некоторых показателей.

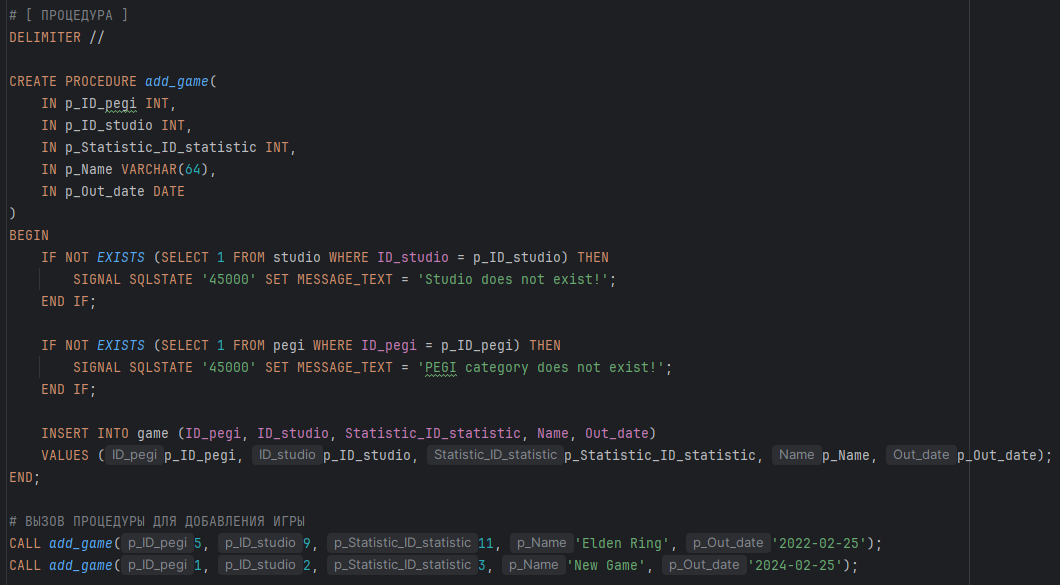
****

Рисунок 1 – Код процедуры для добавления игры и ее последующий вызов

Вызовем написанную нами процедуру используя оператор CALL.

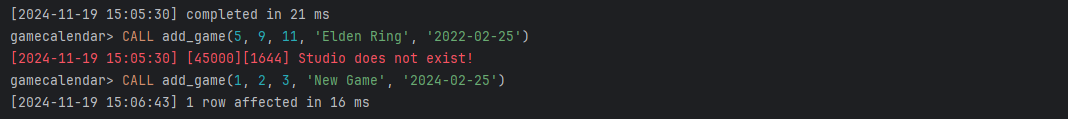


Рисунок 2 – Результат отработки процедуры

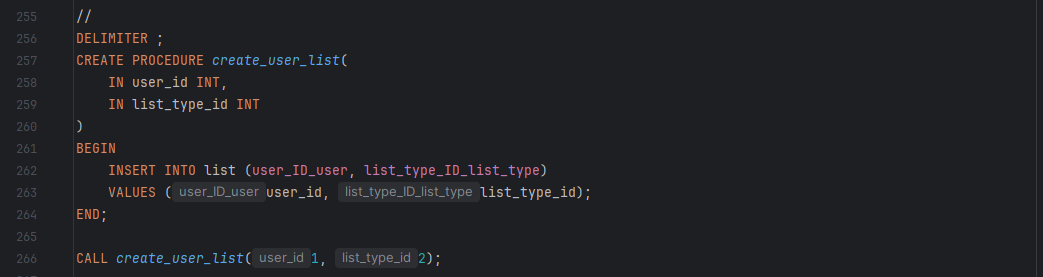


Рисунок 3 – Код процедуры для создания списка пользователя и ее последующий вызов

Вызовем написанную нами процедуру используя оператор CALL.



Рисунок 4 – Результат отработки процедуры

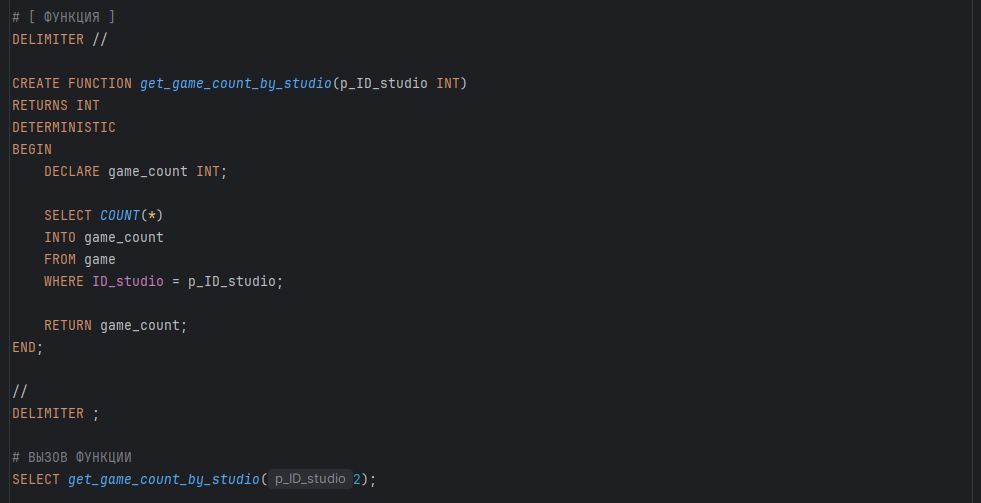


Рисунок 5 – Код функции для просмотра количества игр у студии и ее вызов

Вызовем написанную нами функцию.

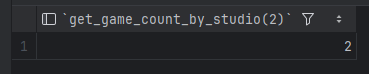


Рисунок 6 – Результат отработки функции

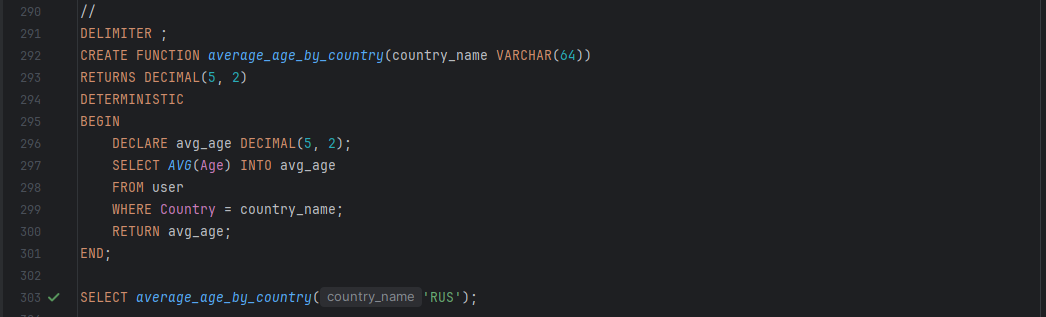


Рисунок 7 – Код функции для просмотра среднего возраста по стране и ее вызов

Вызовем написанную нами функцию.

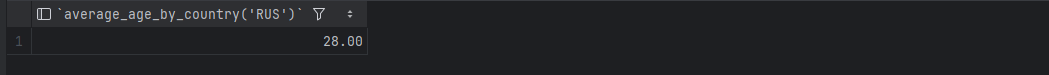


Рисунок 8 – Результат отработки функции

Триггеры – это частный случай хранимой процедуры, который выполняется автоматически при выполнении команд обновления данных (INSERT, DELETE, UPDATE). Триггеры привязываются к конкретным таблицам базы данных. Для каждой команды должны быть свои триггеры.

Создадим таблицу для логирования отработки наших триггеров.

****

Рисунок 9 – Создание таблицы для логирования изменений

Напишем триггер, который будет срабатывать при изменение уже существующих данных в таблице game.

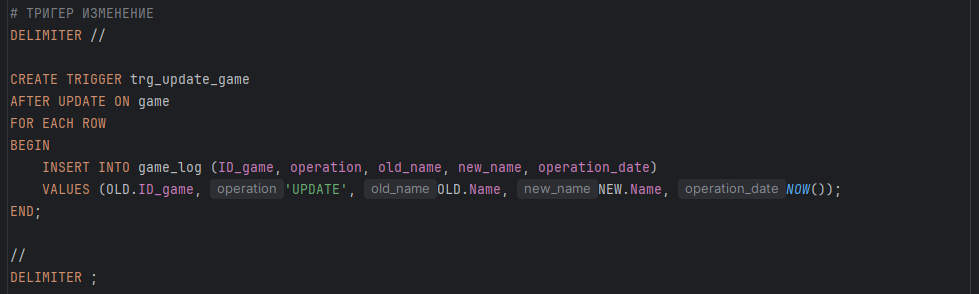
****

Рисунок 10 – Триггер на изменение игры

При изменении данных в нашей таблице у нас вызывается триггер, который записывает изменения в таблицу логирования, созданную ранее.

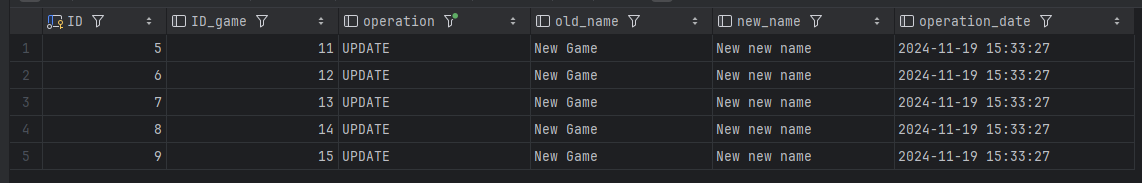
****

Рисунок 11 – Срабатывание триггера на изменение игры

Напишем триггер, который будет срабатывать при обновлении данных пользователя.

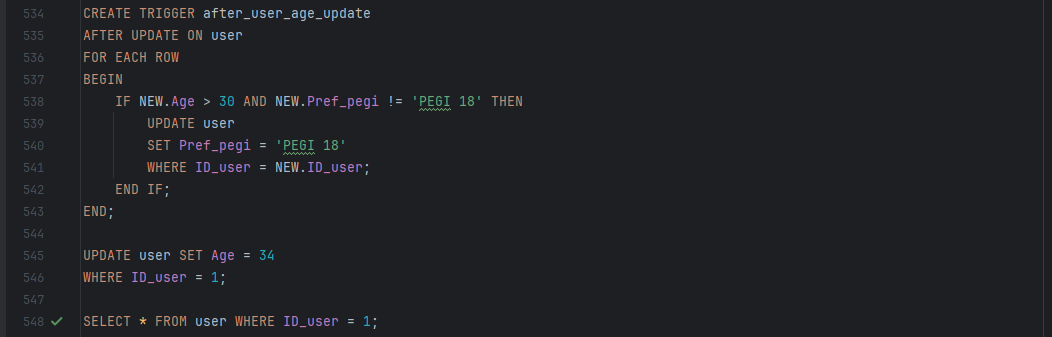
****

Рисунок 12 – Триггер на добавления пользователя

При изменении возраста пользователя у нас вызывается триггер, который изменяет предпочтительный PEGI на PEGI-18.

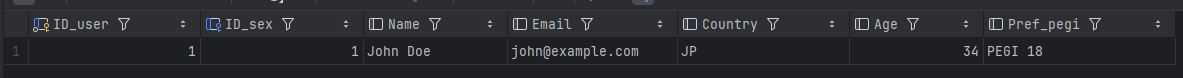
****

Рисунок 13 – Срабатывание триггера на попытку добавление не соответствующего пользователя

Напишем триггер, который будет срабатывать при добавлении пользователя и проверять правильно ли написан email.

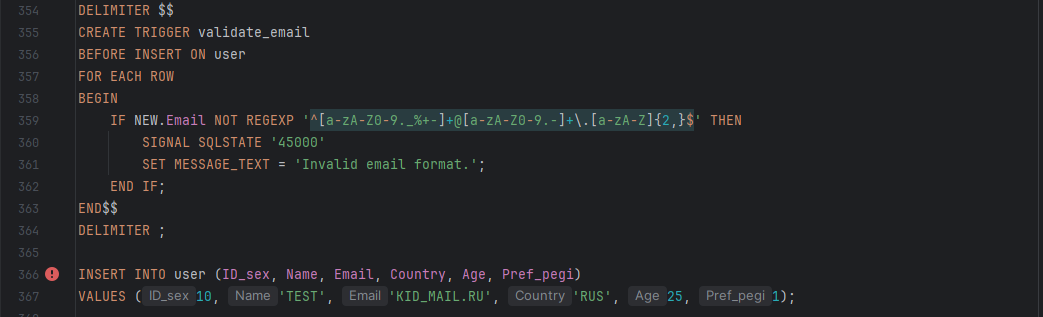
****

Рисунок 14 – Триггер на добавления пользователя

При добавлении пользователя у нас вызывается триггер, который записывает проверяет почту пользователя на соответствие.

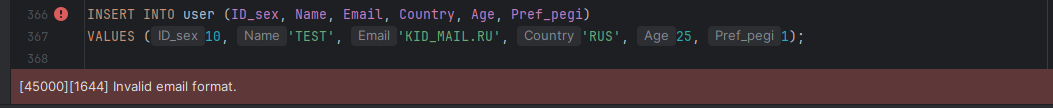


Рисунок 15 – Срабатывание триггера на попытку добавление не правильного email

Напишем триггер, который будет срабатывать при добавлении игры в список.

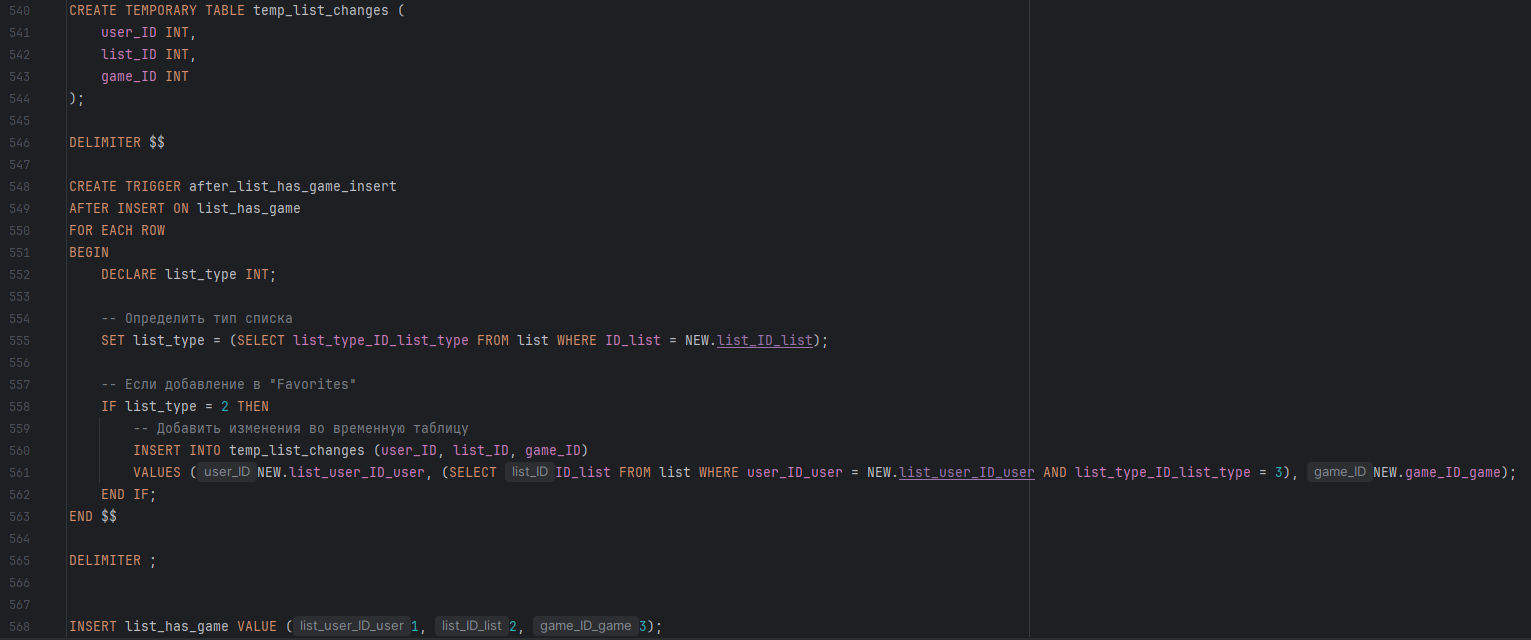
****

Рисунок 16 – Триггер на добавление игры в список

При добавлении игры в список Избранное у нас вызывается триггер, который добавляет игру в список Пройдено.

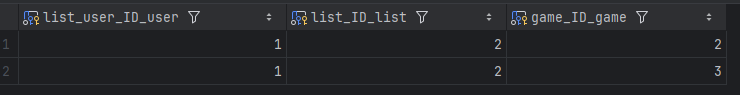
****

Рисунок 17 – Срабатывание триггера на попытку добавления игры

Напишем триггер, который будет срабатывать при добавлении игры в список пользователя.

****

Рисунок 18 – Триггер на добавление игры

При добавлении игры у в список пользователя у нас вызывается триггер, который увеличивает количество игр в списке.

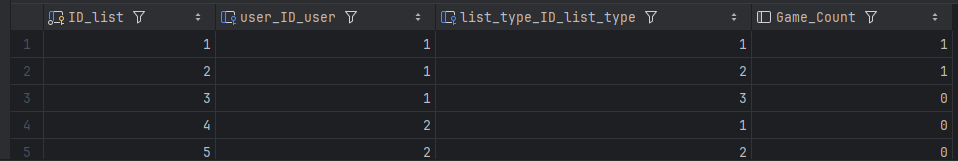
****

Рисунок 19 – Срабатывание триггера на попытку добавление не вышедшей игры

**ОКОННЫЕ ФУНКЦИИ**

Оконные функции в MySQL — это специальный тип функции, который позволяет выполнять агрегатные и аналитические операции над группами строк, которые определены внутри отдельного окна (или оконного фрейма). Оконные функции представляют собой удобный инструмент по работы с данными и предоставляют более гибкий способ обращения с ними, чем традиционные агрегатные функции за счёт того, что они могут учитывать порядок сортировки данных и разбивать их на группы без фактической группировки.

В рамках данной работы разделим оконные функции на следующие 3 группы:

1. **Агрегатные функции:**

Агрегатными функциями называются функции, которые выполняют арифметические вычисления на наборе данных и возвращают итоговое значение.

COUNT(\*) — вычисляет количество значений в столбце (не учитывает значения NULL)

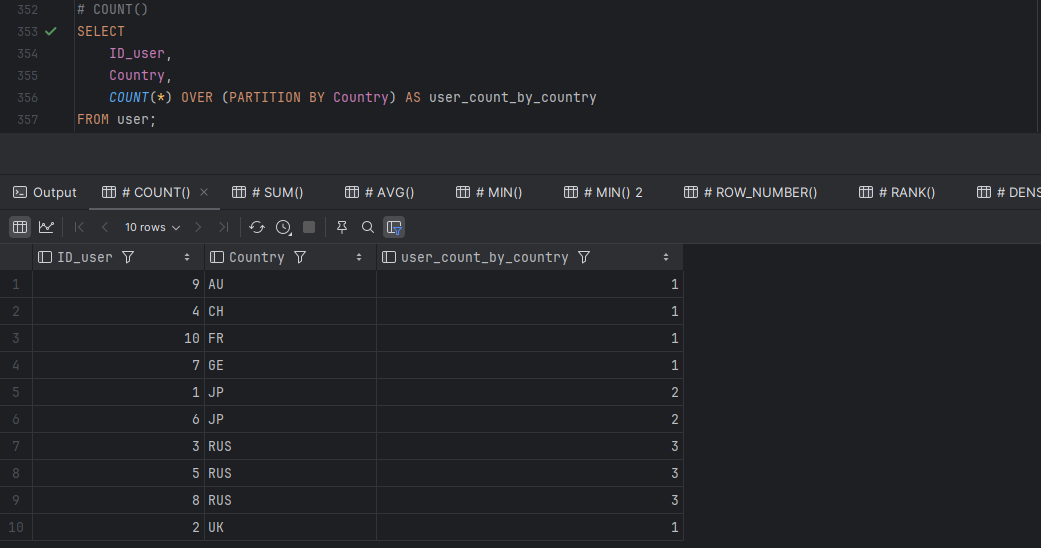


Рисунок 1 – Код и выполнение агрегатной функции COUNT

SUM(\*) — возвращает сумму значений в столбце;

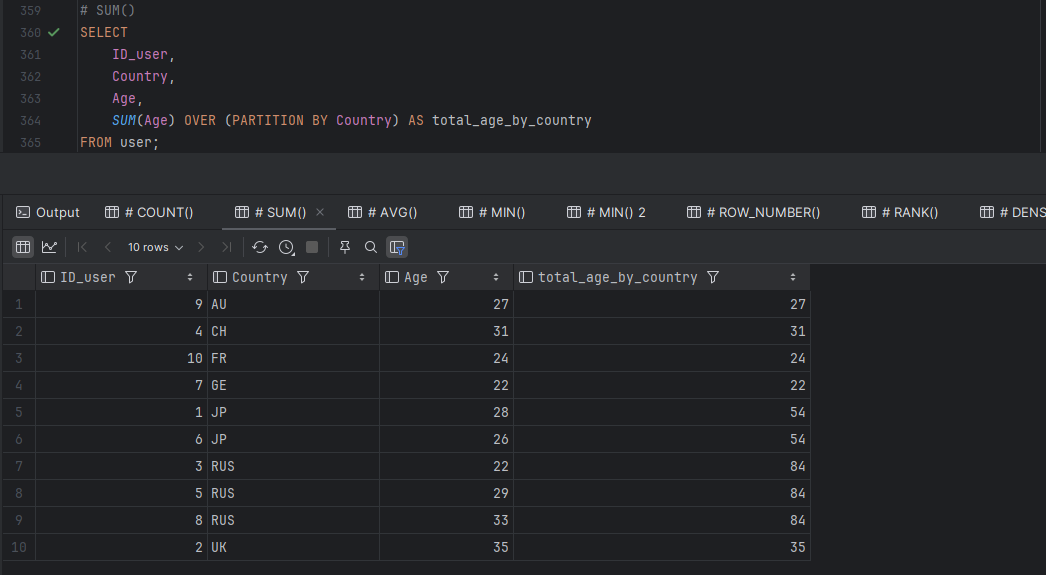


Рисунок 2 – Код и выполнение агрегатной функции SUM

AVG(\*) — определяет среднее значение в столбце

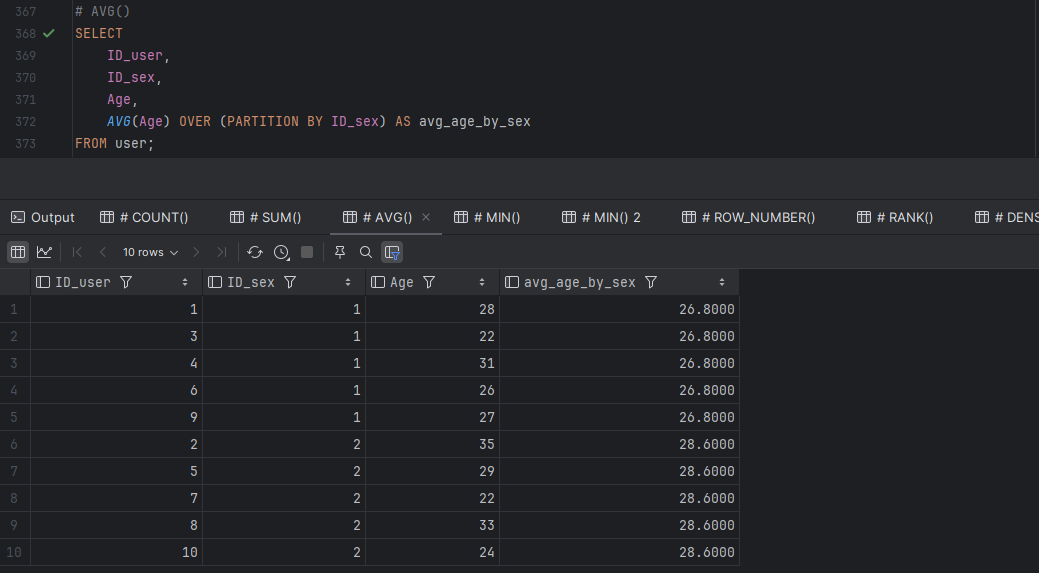


Рисунок 3 – Код и выполнение агрегатной функции AVG

MIN(\*) — определяет минимальное значение в столбце.

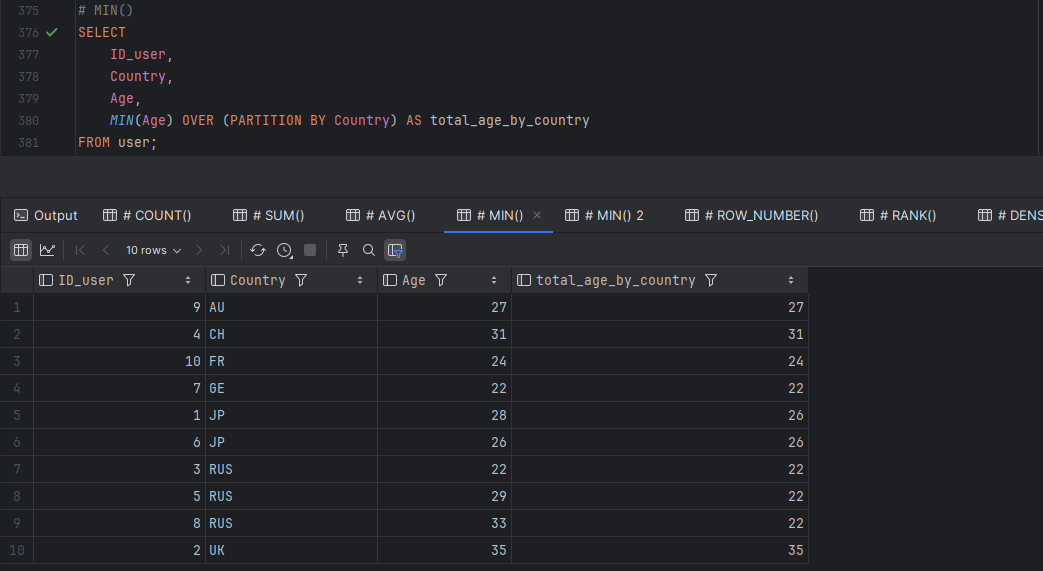


Рисунок 4 – Код и выполнение агрегатной функции MIN

MAX(\*) — определяет максимальное значение в столбце

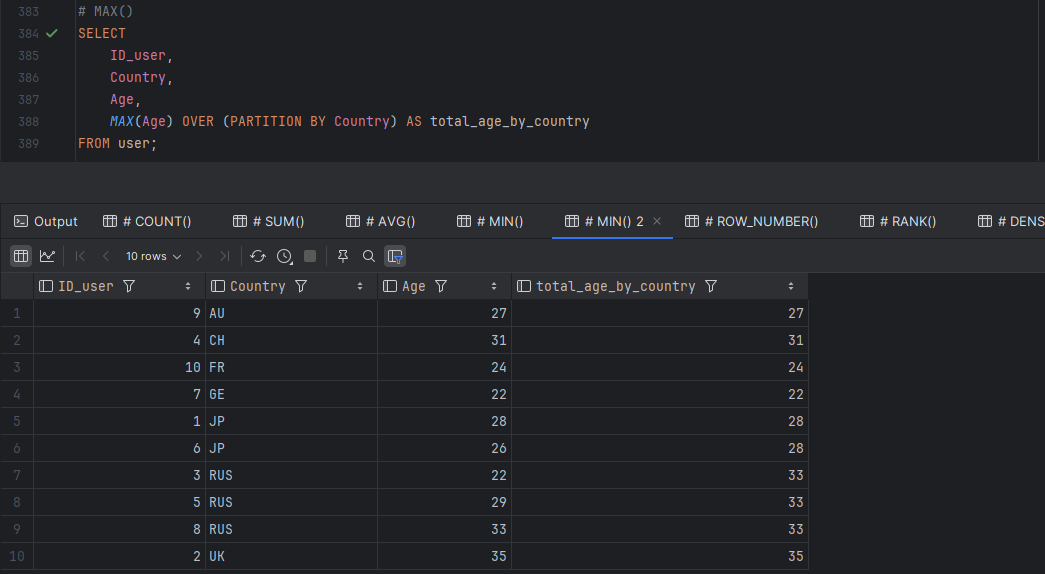


Рисунок 5 – Код и выполнение агрегатной функции MAX

1. **Ранжирующие функции:**

Ранжирующие функции — это функции, которые определяют ранг для каждой строки в окне. Например, их можно использовать для присвоения порядковых номеров или для составления рейтинга.

ROW\_NUMBER() — функция возвращает номер строки и используется для нумерации.

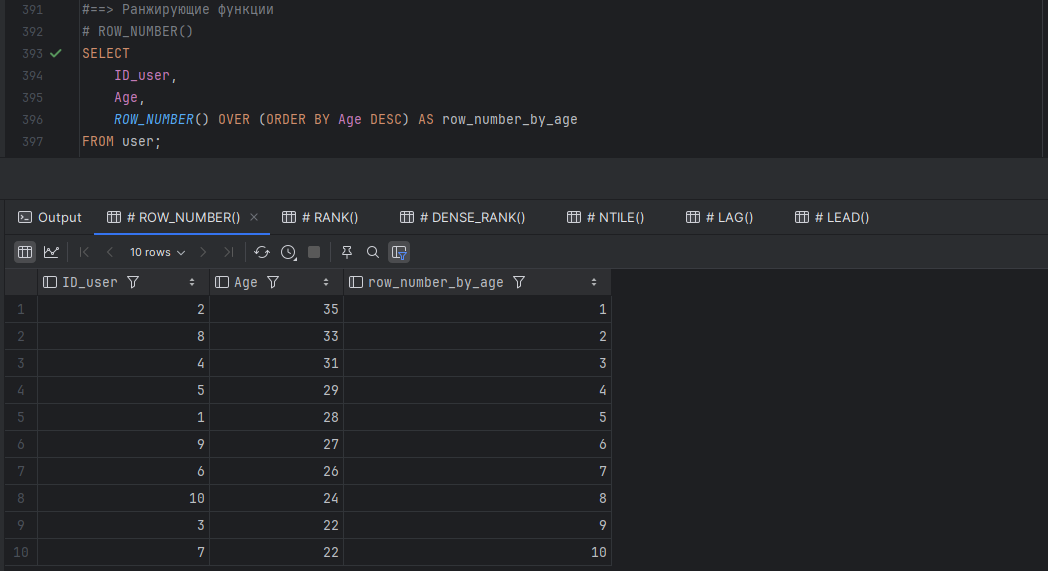


Рисунок 6 – Код и выполнение ранжирующей функции ROW\_NUMBER()

RANK() — функция возвращает ранг каждой строки. Данная функция в том числе анализирует данные и, в случае нахождения одинаковых – возвращает одинаковый ранг с пропуском следующего значения (например, два различных товара были проданы на одинаковую сумму по итогам месяца. При использовании данной функции для оценки ранга продаж за месяц обоим товарам будет выставлен ранг 1, а следующий за ними товар получит ранг 3).

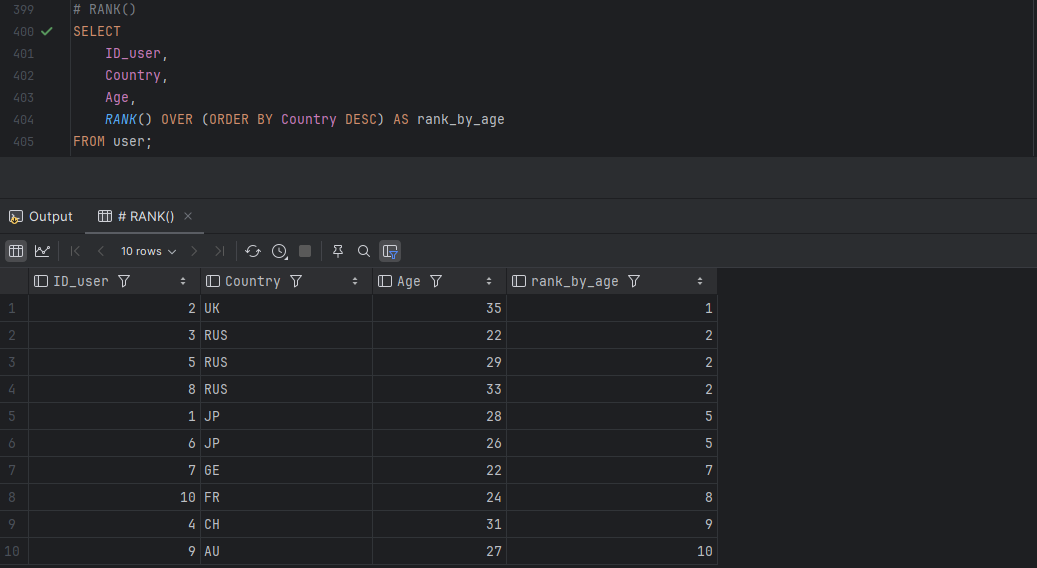


Рисунок 7 – Код и выполнение ранжирующей функции RANK()

DENSE\_RANK() — так же, как и прошлая функция, возвращает ранг каждой строчки, но в отличие от функции RANK, следующий ранг пропускаться не будет.

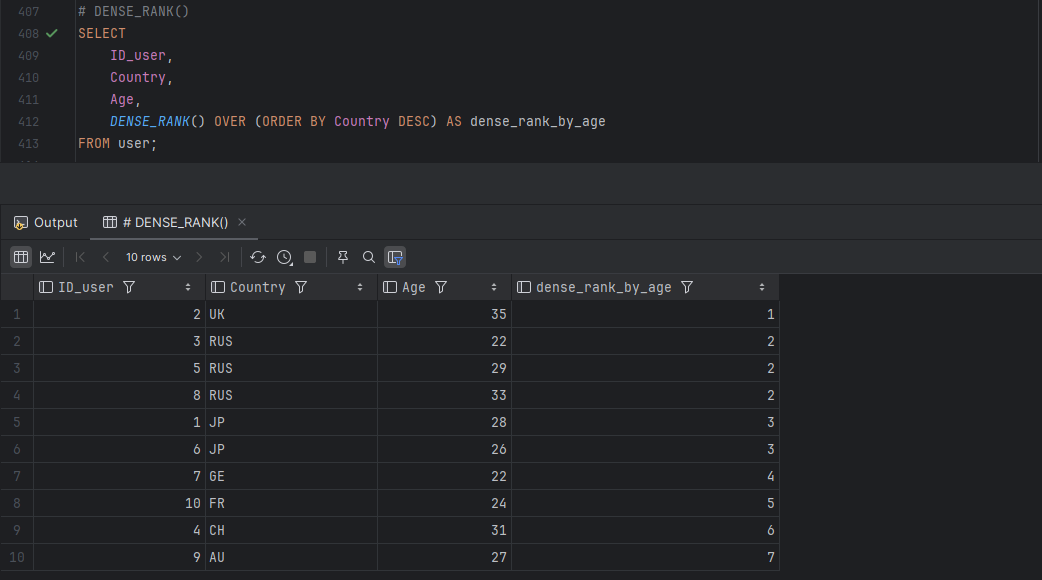


Рисунок 8 – Код и выполнение ранжирующей функции DENSE\_RANK()

NTILE(\*) — это функция, которая позволяет определить, к какой группе относится текущая строка. Количество групп задаётся в скобках.

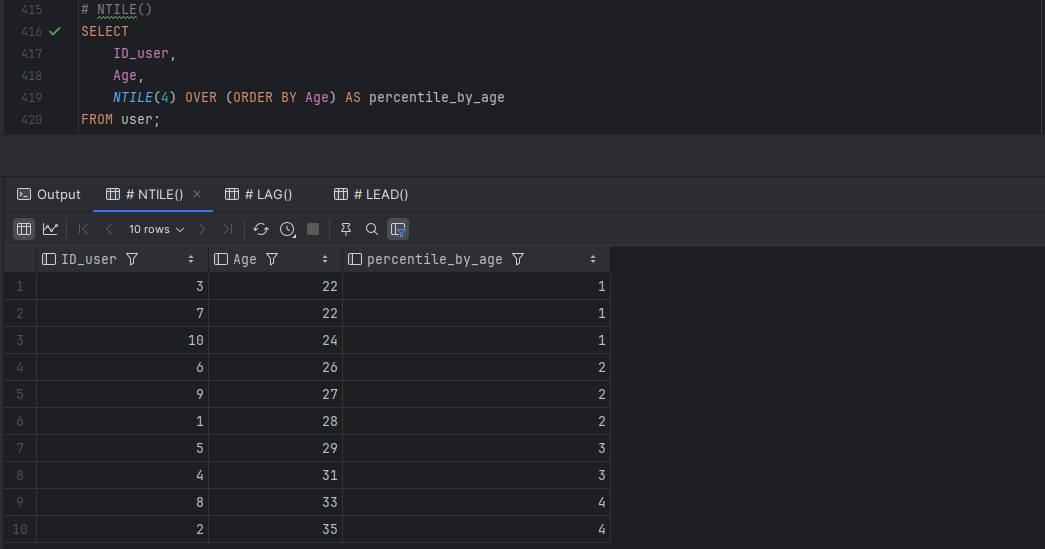


Рисунок 9 – Код и выполнение ранжирующей функции NTILE()

1. **Функции смещения:**

Функции смещения — это функции, которые позволяют перемещаться и обращаться к разным строкам в окне относительно текущей строки, а также обращаться к значениям в начале или в конце окна.

LAG(\*) и LEAD(\*) — функция LAG обращается к данным из предыдущей строки окна, а LEAD к данным из следующей строки. Функцию можно использовать для сравнения текущего значения строки с предыдущим или следующим. Имеет три параметра: столбец, значение которого необходимо вернуть, количество строк для смещения (по умолчанию это 1) и значение, которое необходимо вернуть, если после смещения возвращается значение NULL;

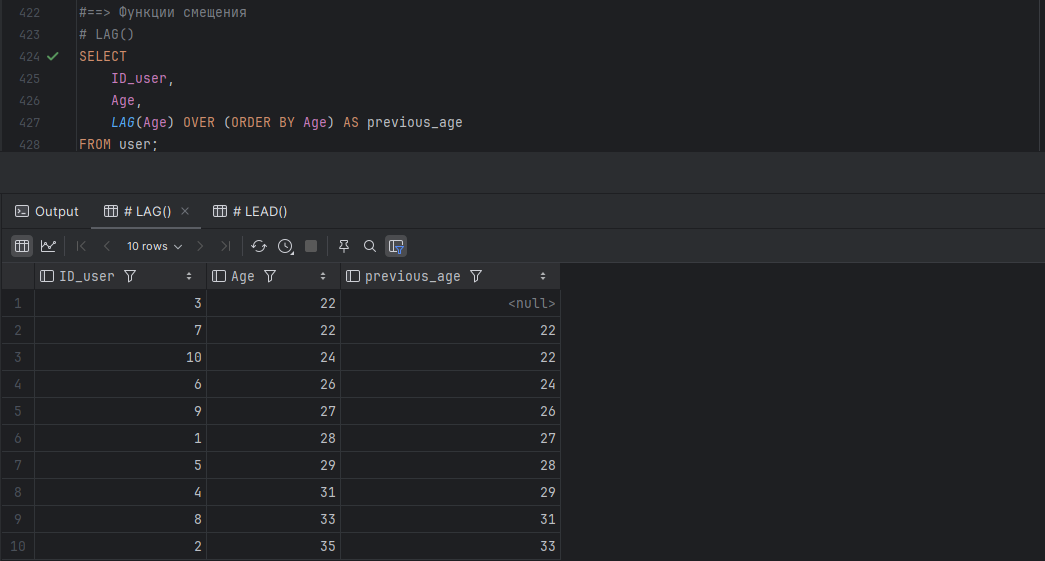


Рисунок 10 – Код и выполнение функции смещения LAG()

Теперь отобразим функцию смещения LEAD

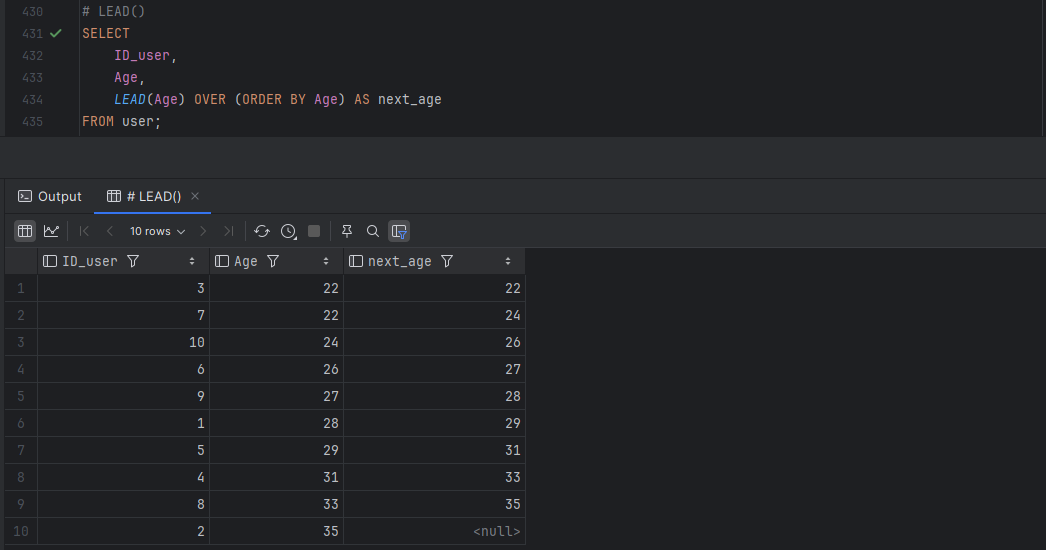


Рисунок 11 – Код и выполнение функции смещения LAG()

FIRST\_VALUE(\*) и LAST\_VALUE(\*) — функция FIRST\_VALUE позволяет получить первое значение в окне, а LAST\_VALUE, соответственно, последнее значение.

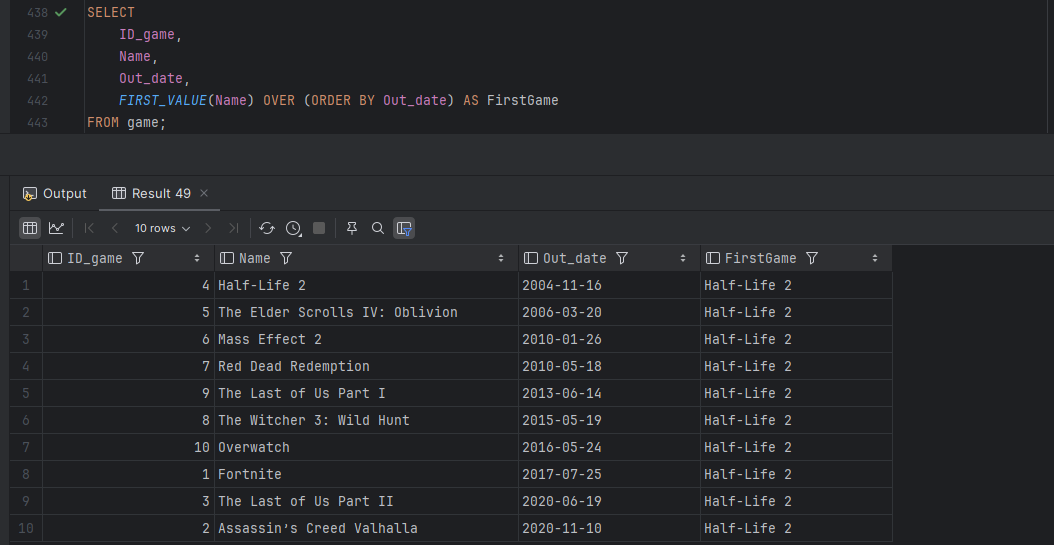


Рисунок 12 – Код и выполнение функции смещения FIRST\_VALUE ()

**ВЫВОД**

В ходе изучения работы с базами данных на языке MySQL мы освоили множество ключевых аспектов, которые позволяют эффективно управлять данными и их обработкой. Мы научились проектировать структуры данных, создавая базы данных и таблицы с использованием различных типов данных, что дало возможность организовать хранение информации и оптимизировать доступ к ней. При создании таблиц мы учитывали важные элементы, такие как первичные ключи, индексы и связи между таблицами, что является основой нормализации базы данных и позволяет обеспечивать целостность и правильную организацию данных.

Кроме того, были освоены методы добавления данных в таблицы с помощью SQL-запросов, включая как одиночные, так и массовые вставки. Это значительно упрощает процесс наполнения таблиц данными и их дальнейшее управление, что крайне важно для работы с динамично изменяющимися информационными потоками в базе данных.

Хранимые процедуры и функции открыло перед нами возможность создания повторно используемых блоков кода, что значительно упрощает выполнение сложных операций с данными и способствует улучшению производительности базы данных. Триггеры же научили нас автоматически выполнять действия при наступлении определенных событий в базе данных, таких как добавление, обновление или удаление данных, что повысило уровень автоматизации процессов и безопасности при работе с данными.

В результате, эти навыки и знания позволили нам значительно расширить возможности работы с базами данных, улучшив как процесс разработки, так и управление данными на всех этапах их жизненного цикла. Мы научились эффективно проектировать, наполнять, извлекать, изменять и автоматизировать операции с данными, что является основой успешного и безопасного использования реляционных баз данных.