**СОЗДАНИЕ БАЗЫ ДАННЫХ И ТАБЛИЦ В НЕЙ**

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

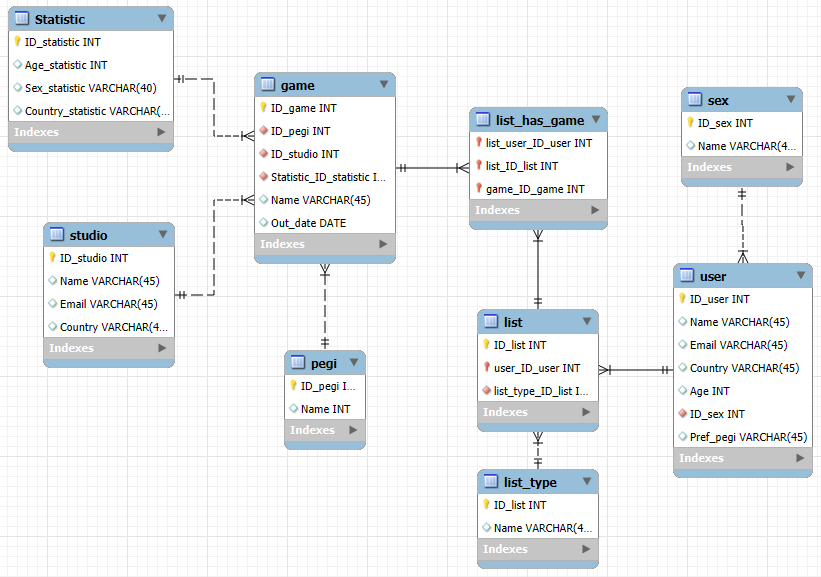


Рисунок 1 – Физическая модель реализуемой базы данных

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.

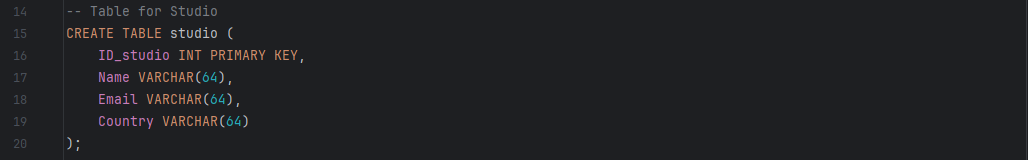
****

Рисунок 4 – Процесс создания таблицы «Studio»

Создадим таблицу PEGI.

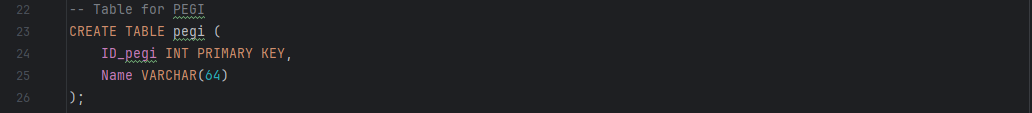
****

Рисунок 5 – Процесс создания таблицы «PEGI»

Создадим таблицу Sex.

****

Рисунок 6 – Процесс создания таблицы «Sex»

Создадим таблицу Game.

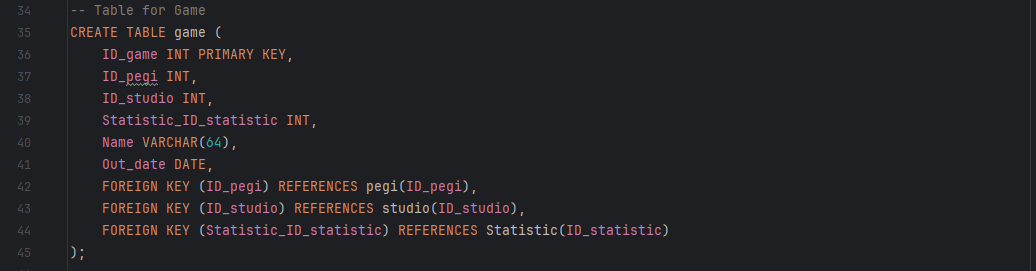
****

Рисунок 7 – Процесс создания таблицы «Game»

Создадим таблицу List Type.

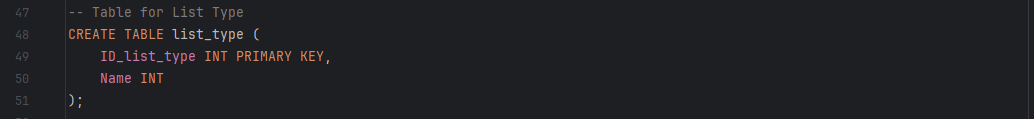
****

Рисунок 8 – Процесс создания таблицы «List Type»

Создадим таблицу User.

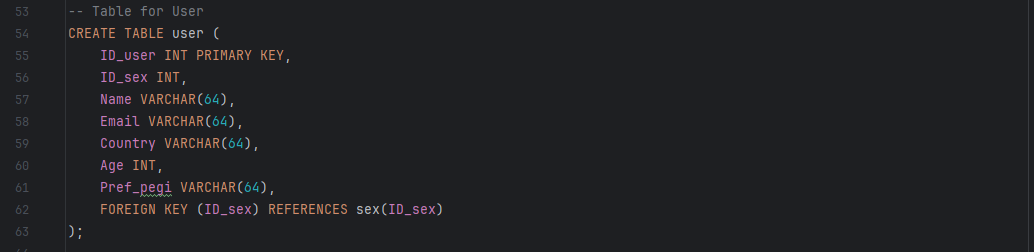
****

Рисунок 9 – Процесс создания таблицы «User»

Создадим таблицу List.

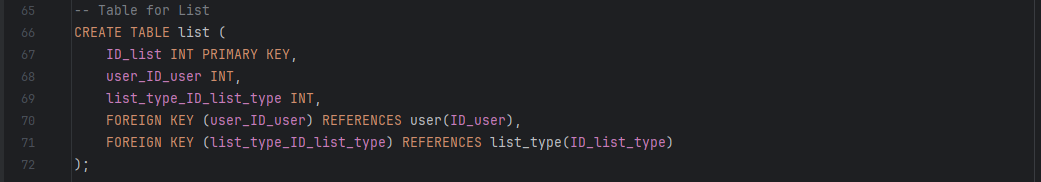
****

Рисунок 10 – Процесс создания таблицы «List»

Создадим таблицу List Has Game.

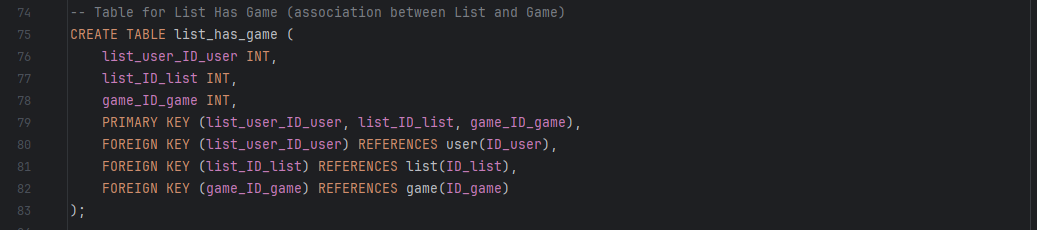
****

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.

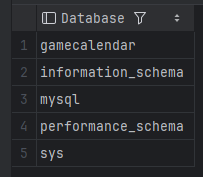
****

Рисунок 12 – Описание всех имеющиеся БД

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

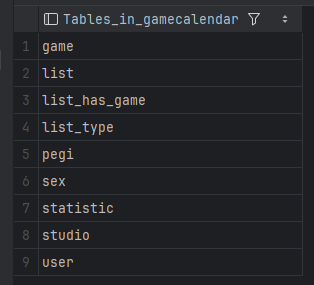
****

Рисунок 13 – Cписок таблиц БД «GameCalendar»

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

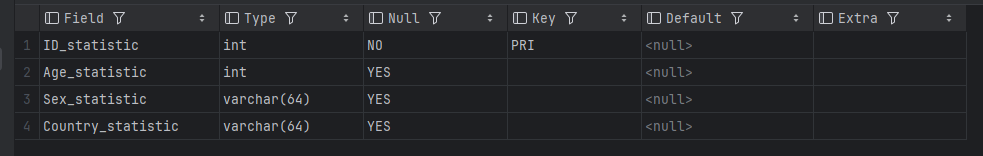
****

Рисунок 14 – Описание столбцов в таблице «statistic»

Просмотрим таблицу studio.

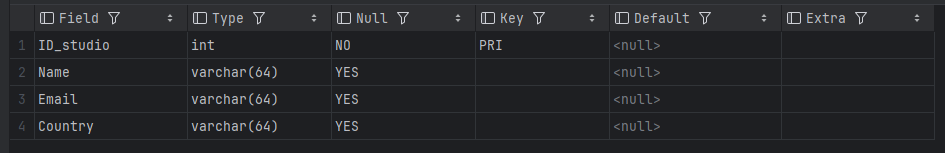
****

Рисунок 15 – Описание столбцов в таблице «studio»

Просмотрим таблицу pegi.

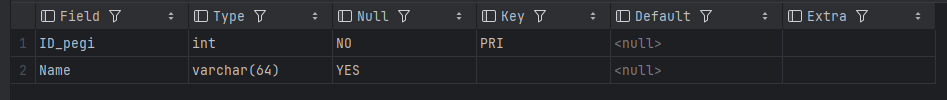
****

Рисунок 16 – Описание столбцов в таблице «pegi»

Просмотрим таблицу sex.

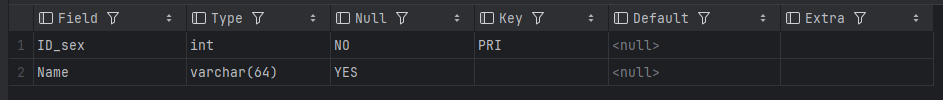
****

Рисунок 17 – Описание столбцов в таблице «sex»

Просмотрим таблицу game.

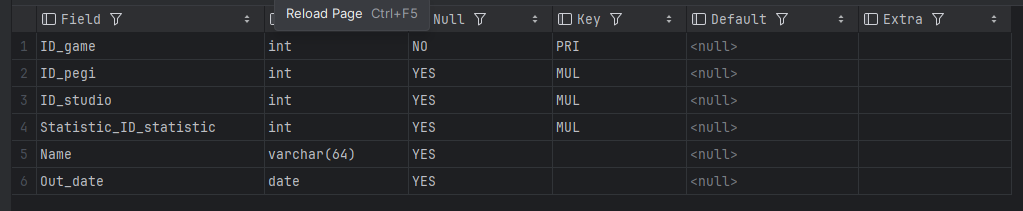
****

Рисунок 18 – Описание столбцов в таблице «game»

Просмотрим таблицу list\_type.

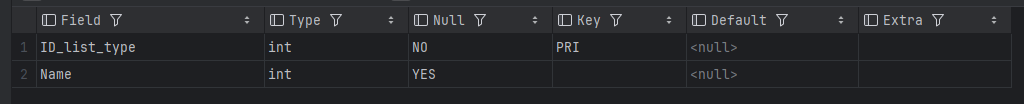
****

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

Просмотрим таблицу user.

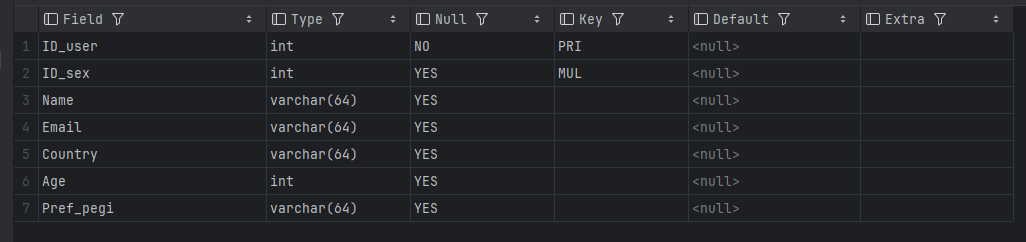
****

Рисунок 20 – Описание столбцов в таблице «user»

Просмотрим таблицу list.

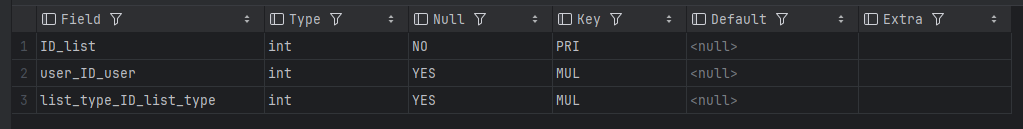
****

Рисунок 21 – Описание столбцов в таблице «list»

Просмотрим таблицу list\_has\_game.

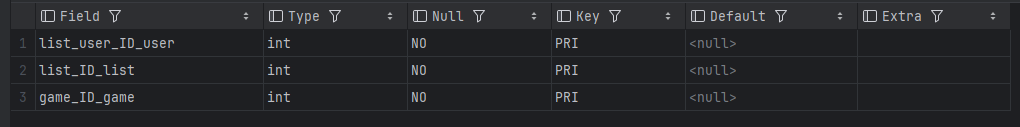
****

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.

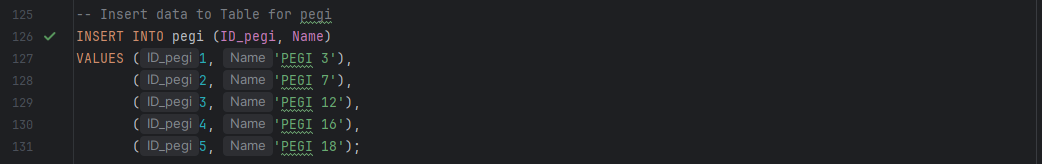
****

Рисунок 3 – Добавление данных в таблицу «pegi»

Добавим данные в нашу таблицу sex.

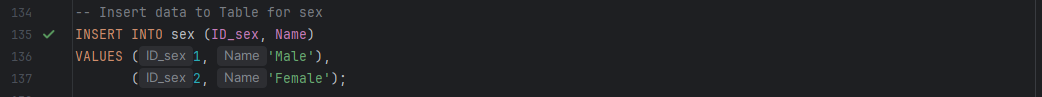
****

Рисунок 4 – Добавление данных в таблицу «sex»

Добавим данные в нашу таблицу game.

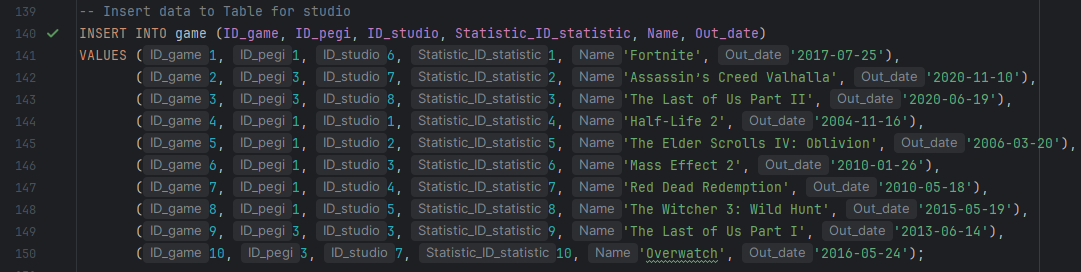
****

Рисунок 5 – Добавление данных в таблицу «game»

Добавим данные в нашу таблицу list\_type.

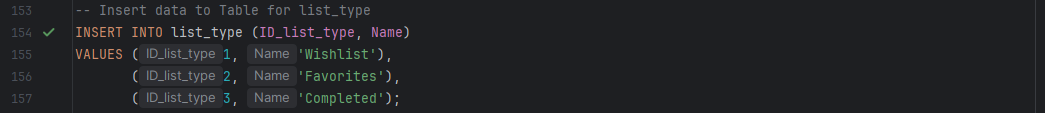
****

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

Добавим данные в нашу таблицу user.

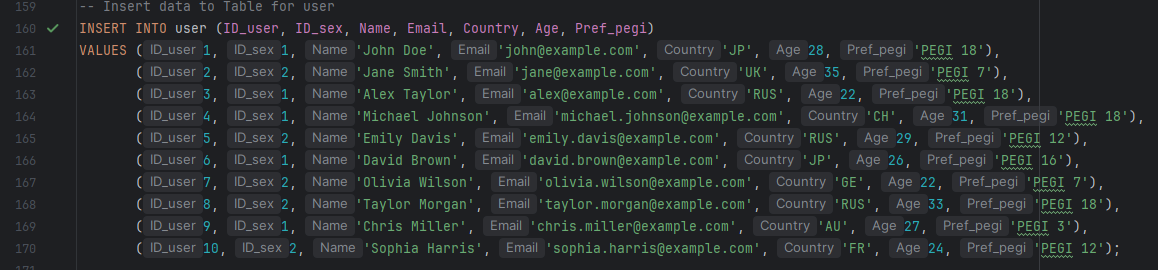
****

Рисунок 7 – Добавление данных в таблицу «user»

Добавим данные в нашу таблицу list.

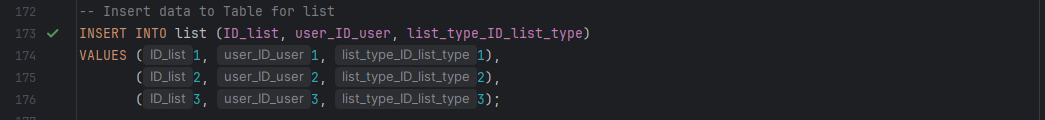
****

Рисунок 8 – Добавление данных в таблицу «list»

Добавим данные в нашу таблицу list\_has\_game.

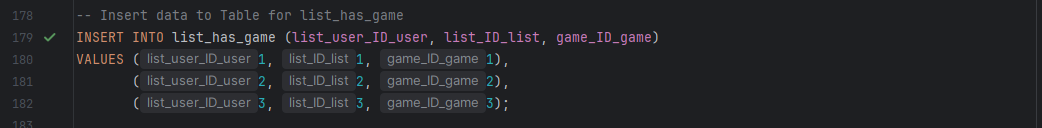
****

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.

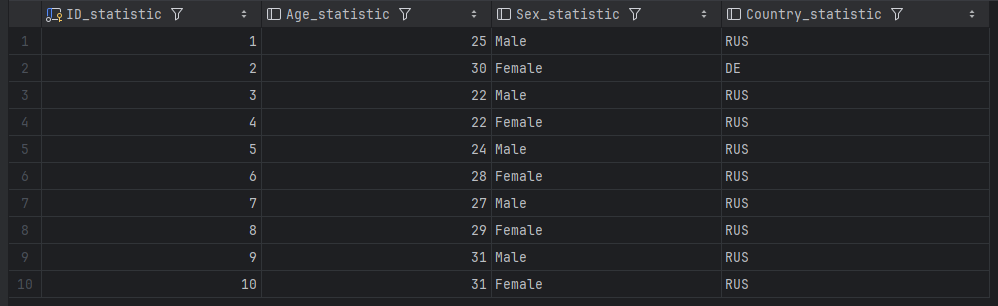
****

Рисунок 10 – Добавленные данные в таблицу «statistic»

Просмотрим данные, которые мы добавили в таблицу studio.

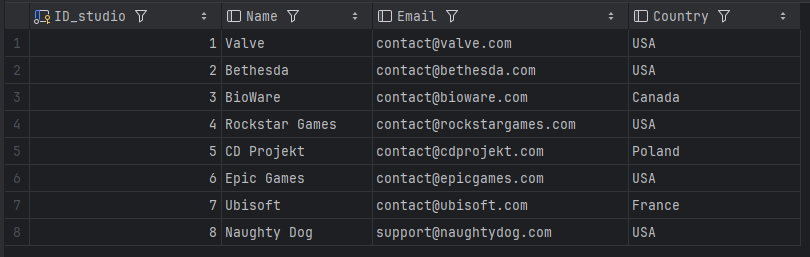
****

Рисунок 11 – Добавленные данные в таблицу «studio»

Просмотрим данные, которые мы добавили в таблицу pegi.

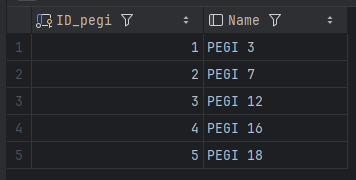
****

Рисунок 12 – Добавленные данные в таблицу «pegi»

Просмотрим данные, которые мы добавили в таблицу sex.

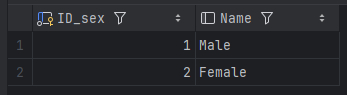
****

Рисунок 13 – Добавленные данные в таблицу «sex»

Просмотрим данные, которые мы добавили в таблицу game.

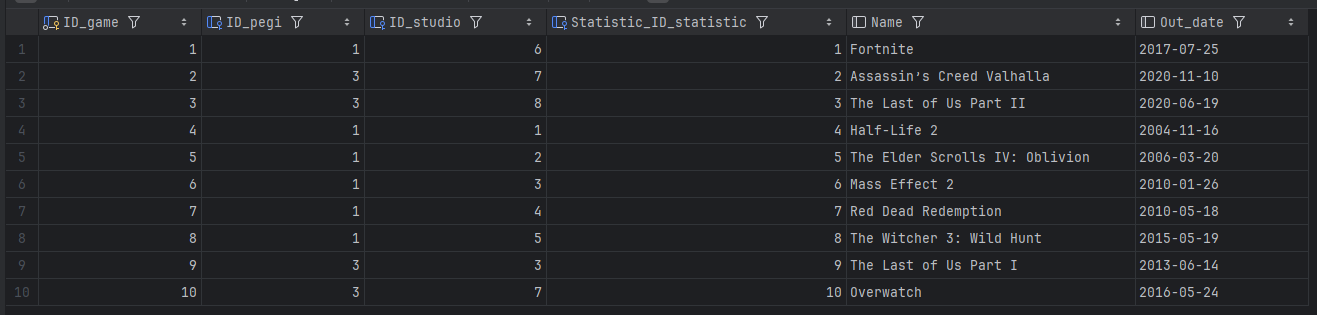
****

Рисунок 14 – Добавленные данные в таблицу «game»

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

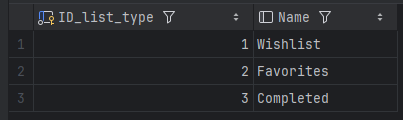
****

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

Просмотрим данные, которые мы добавили в таблицу user.

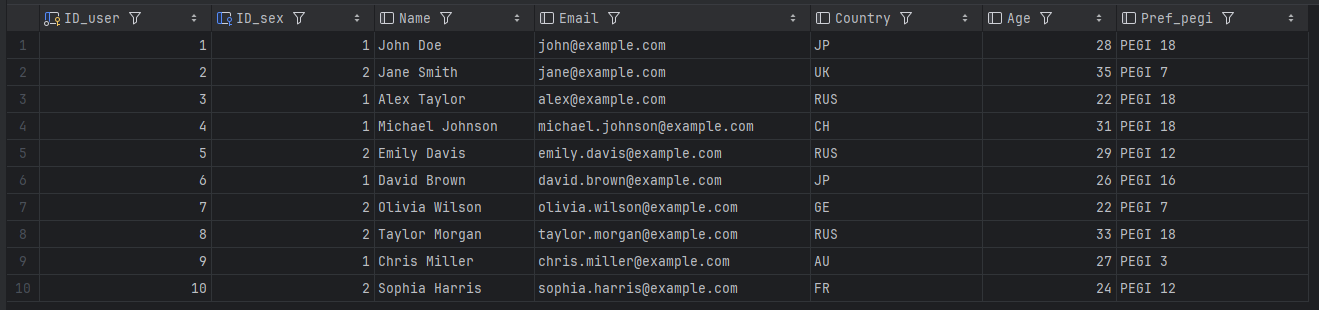
****

Рисунок 16 – Добавленные данные в таблицу «user»

Просмотрим данные, которые мы добавили в таблицу list.

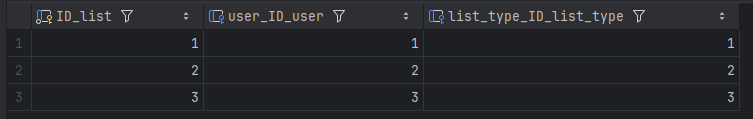
****

Рисунок 17 – Добавленные данные в таблицу «list»

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

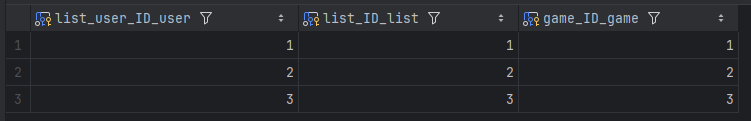
****

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.

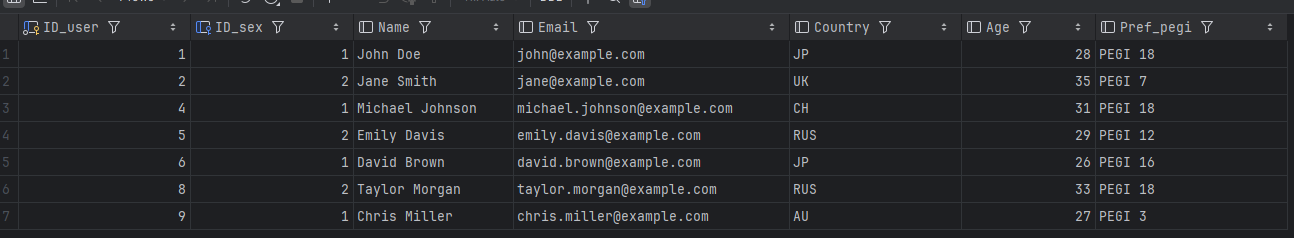


Рисунок 2 – Просмотр таблицы user с применением оператора Age > 25

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.

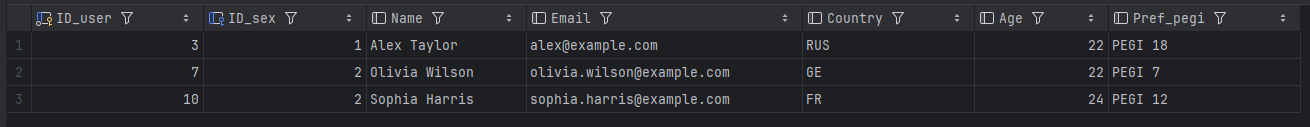
****

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.

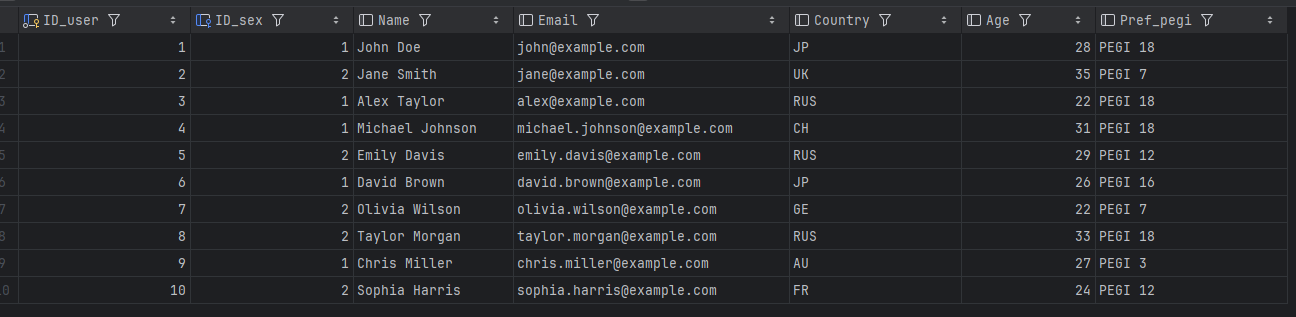
****

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.

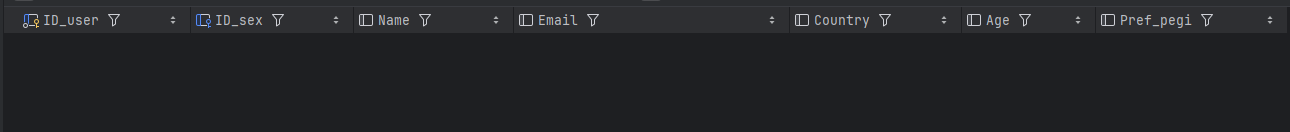
****

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.

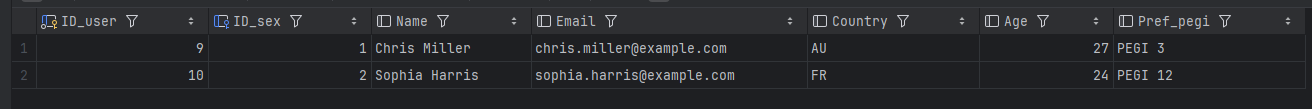
****

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.

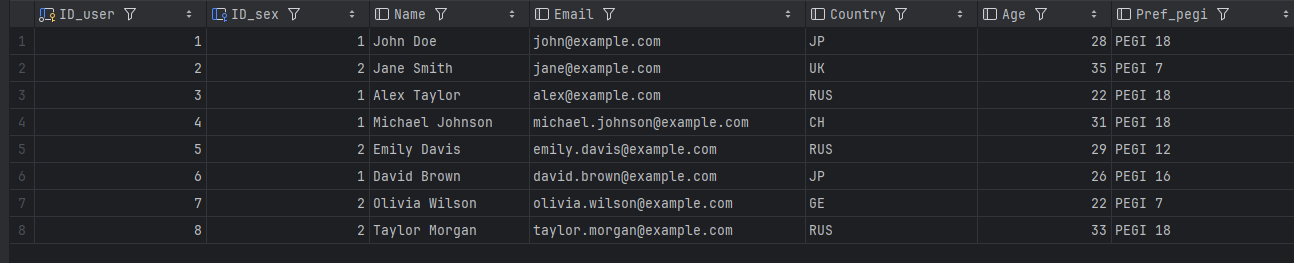
****

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.

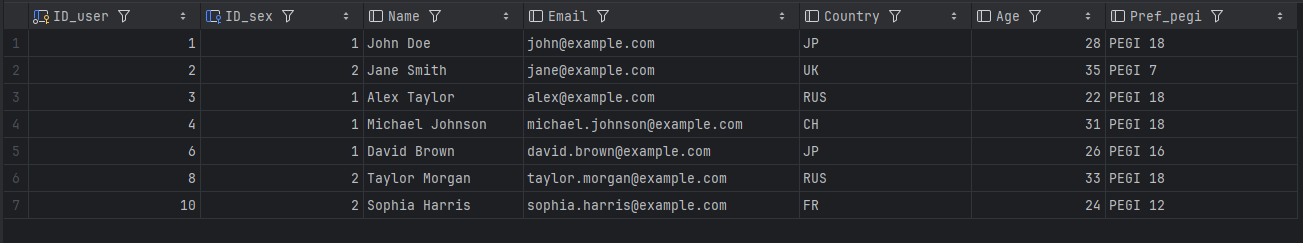
****

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.

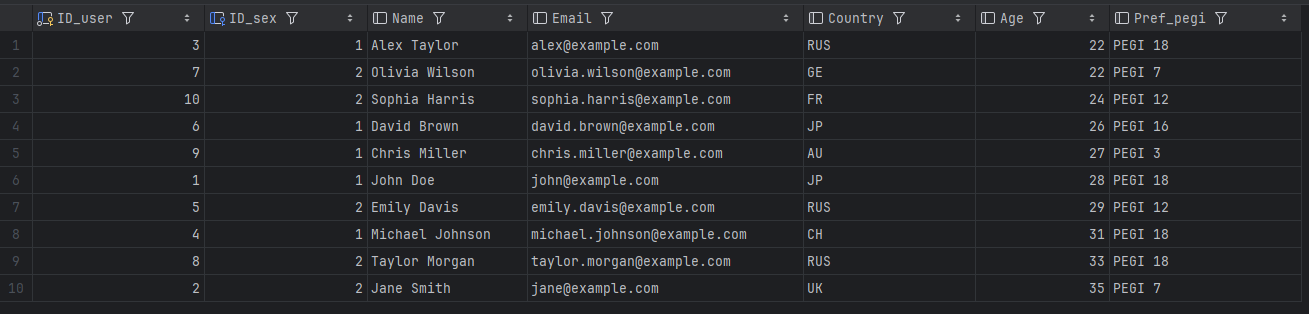
****

Рисунок 13 – Отсортированные данные по возрастанию по столбцу Age

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

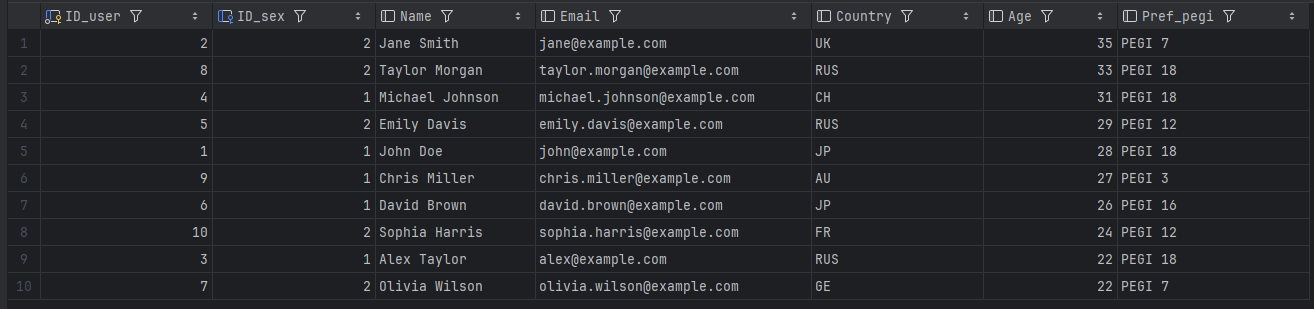
****

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.

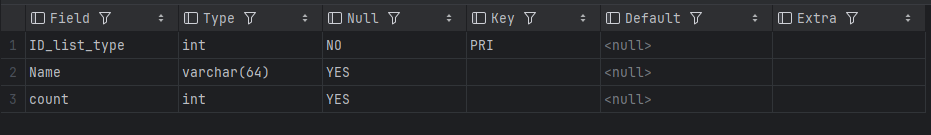
****

Рисунок 1 – Добавление столбца count в таблицу list\_type

Для изменения имени существующего столбца используется оператор CHANGE.

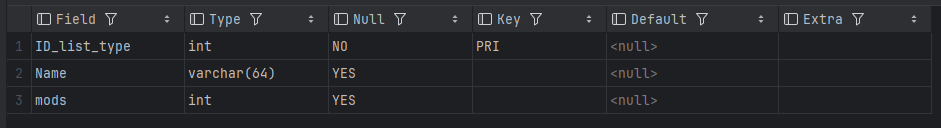
****

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

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

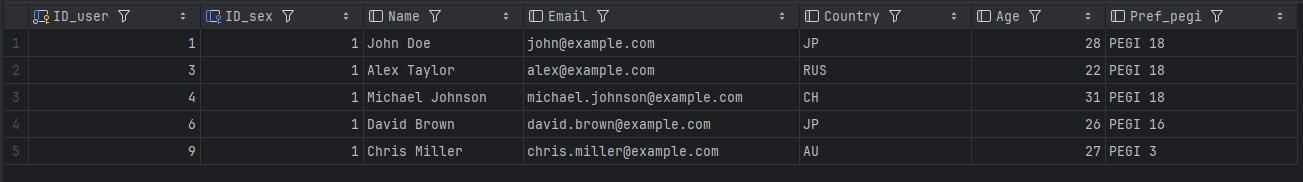
****

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

**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.

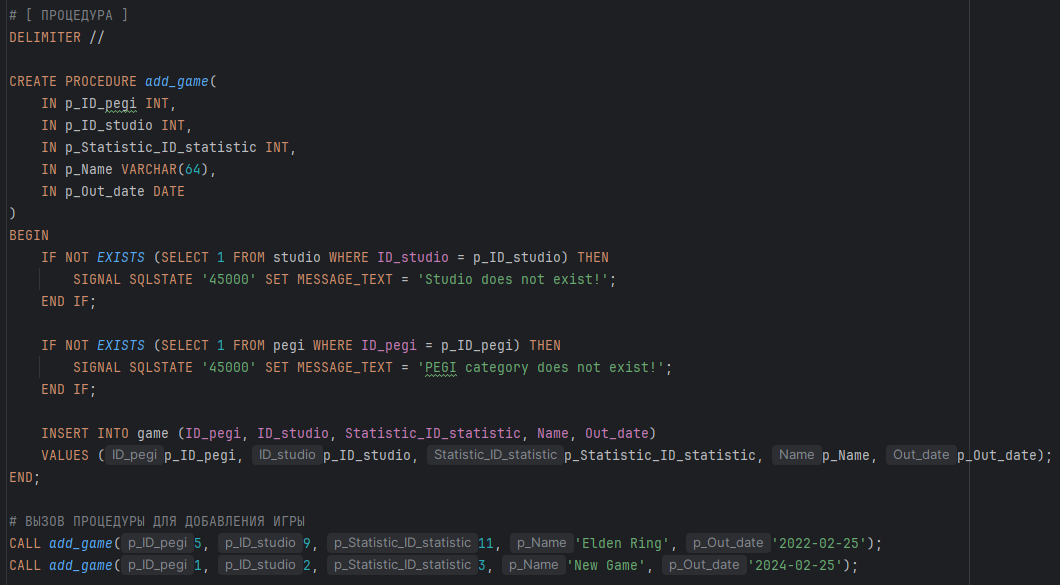
****

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

Вызовем написанную нами процедуру используя оператор CALL.

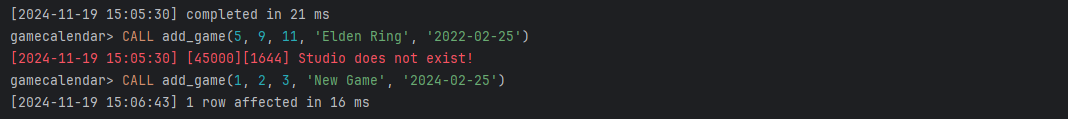


Рисунок 2 – Результат отработки процедуры

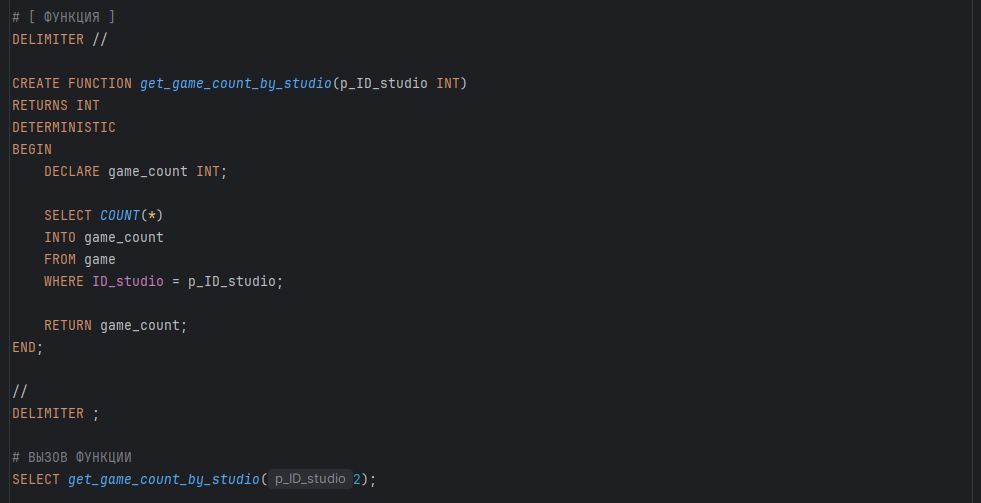


Рисунок 3 – Код функции для просмотра количества игр у студии и ее вызов

Вызовем написанную нами функцию.

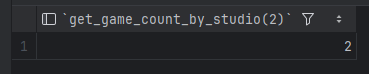


Рисунок 4 – Результат отработки функции

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 5 – Создание таблицы для логирования изменений

Напишем триггер, который будет срабатывать при добавление новых данных в таблицу game.

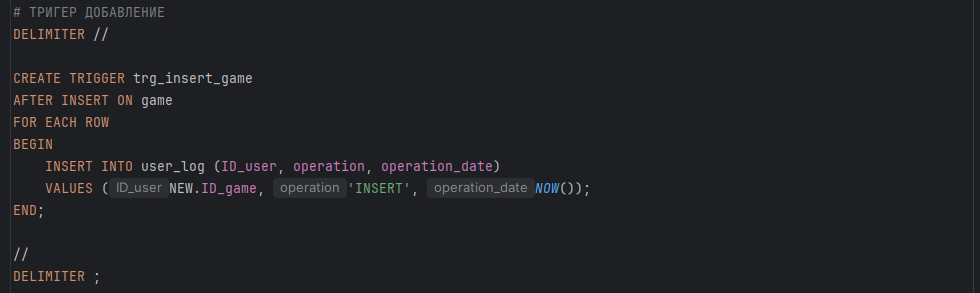
****

Рисунок 6 – Триггер на добавление новой игры

При добавлении данных в нашу таблицу новых данных у нас вызывается триггер, который записывает изменения в таблицу логирования, созданную ранее.

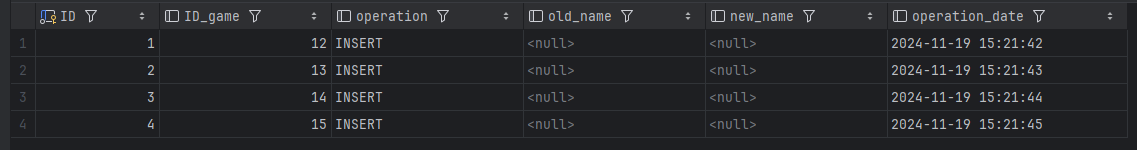
****

Рисунок 7 – Срабатывание триггера на добавление новой игры

Напишем триггер, который будет срабатывать при удалении новых данных в таблицу game.

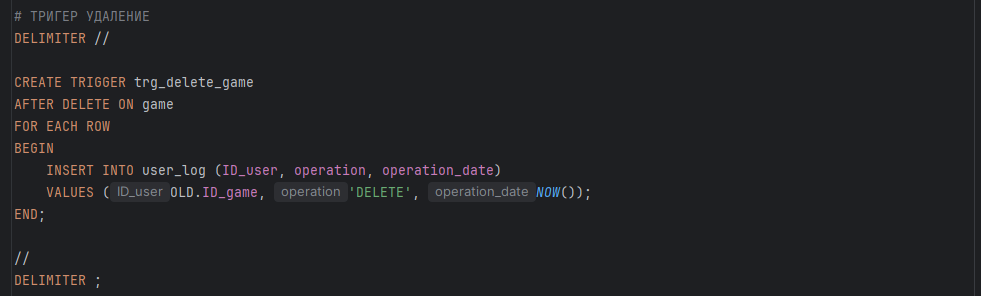
****

Рисунок 8 – Триггер на удаление игры

При удалении данных из нашу таблицу новых данных у нас вызывается триггер, который записывает изменения в таблицу логирования, созданную ранее.

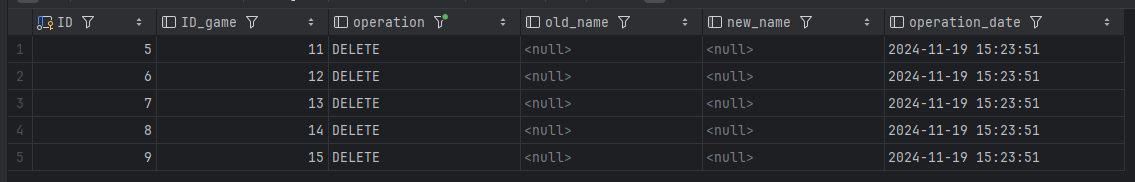
****

Рисунок 9 – Срабатывание триггера на удаление игры

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

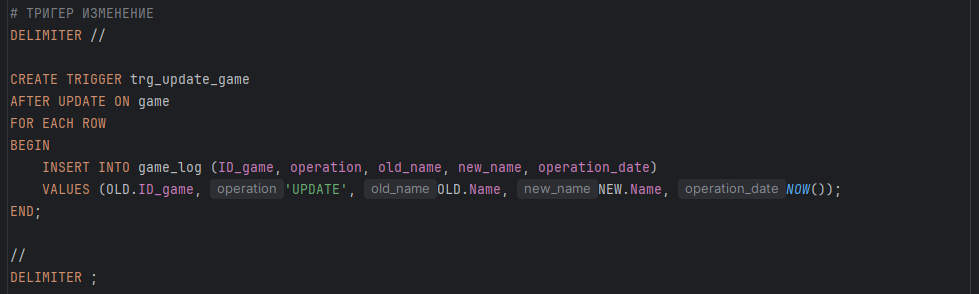
****

Рисунок 10 – Триггер на изменение игры

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

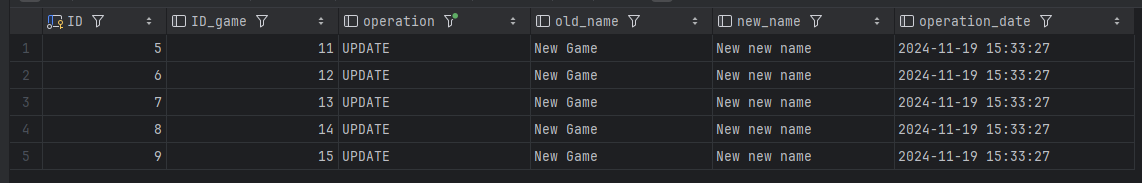
****

Рисунок 9 – Срабатывание триггера на изменение 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)

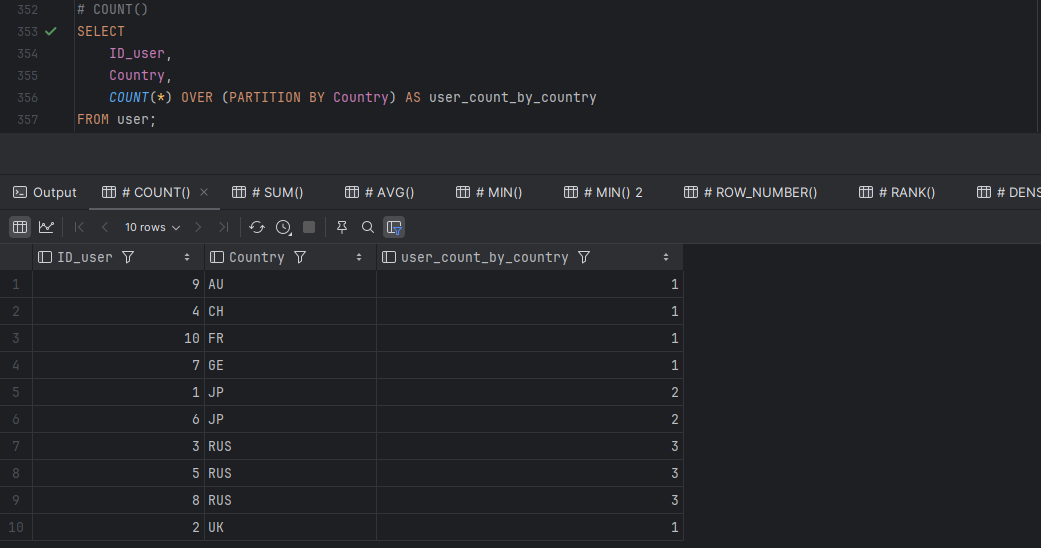


Рисунок 1 – Код и выполнение агрегатной функции COUNT

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

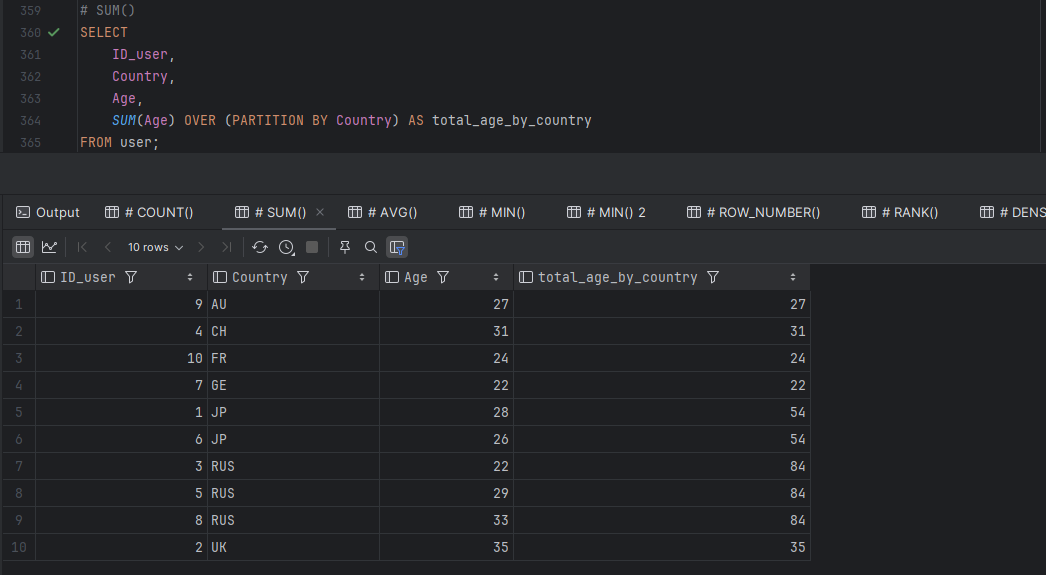


Рисунок 2 – Код и выполнение агрегатной функции SUM

AVG(\*) — определяет среднее значение в столбце

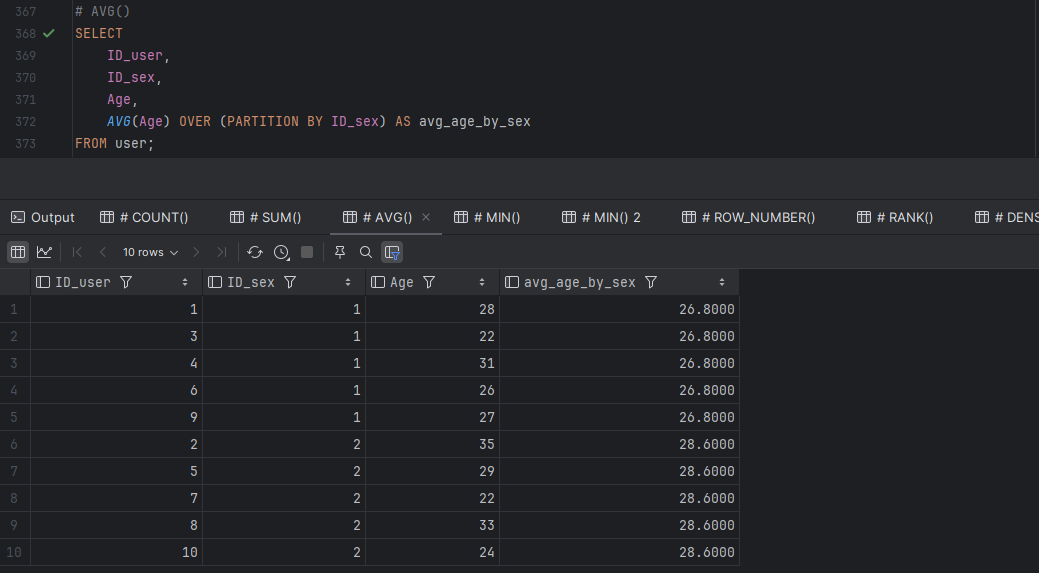


Рисунок 3 – Код и выполнение агрегатной функции AVG

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

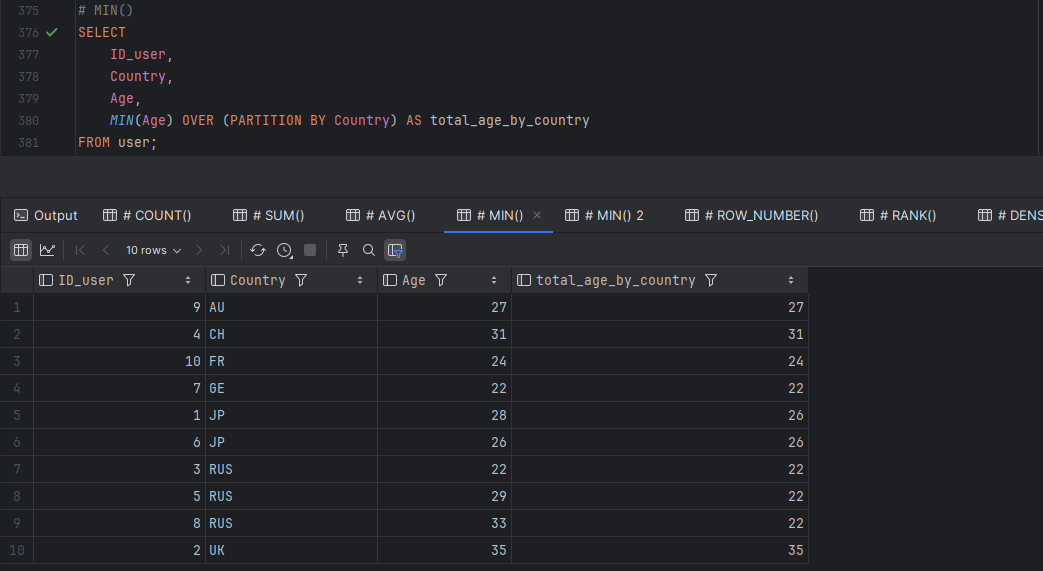


Рисунок 4 – Код и выполнение агрегатной функции MIN

MAX(\*) — определяет максимальное значение в столбце

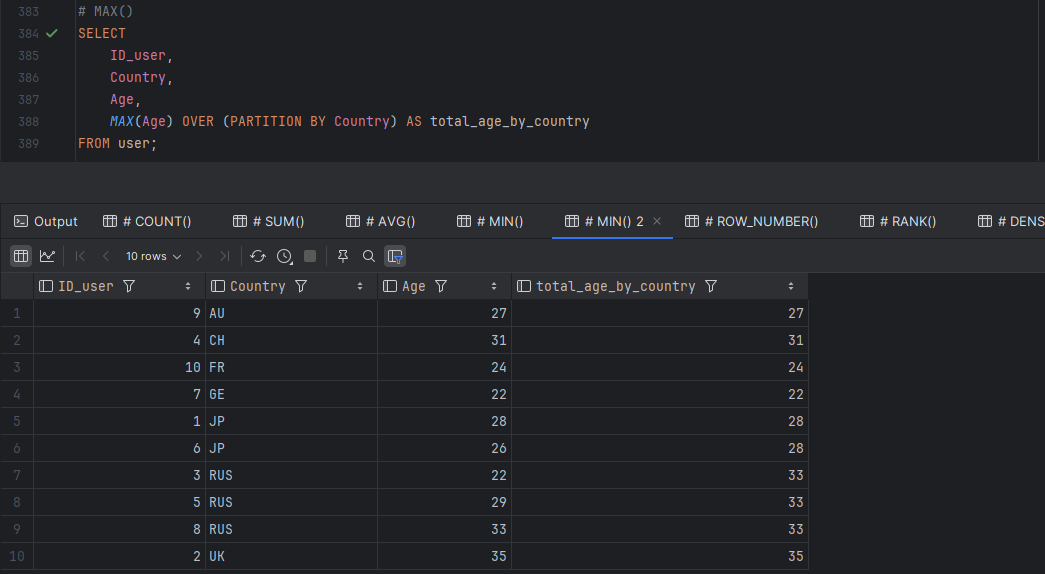


Рисунок 5 – Код и выполнение агрегатной функции MAX

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.

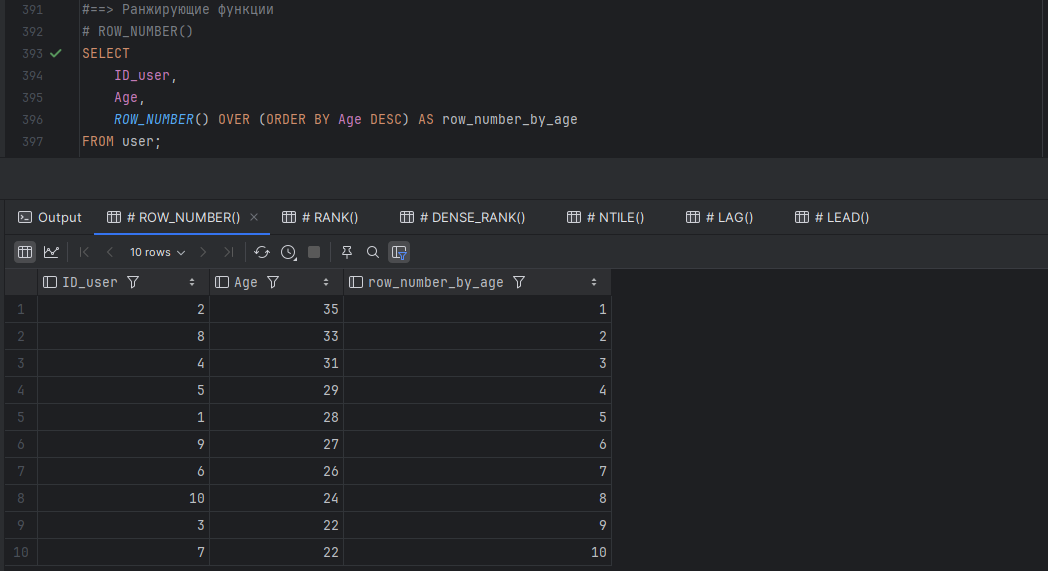


Рисунок 6 – Код и выполнение ранжирующей функции ROW\_NUMBER()

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).

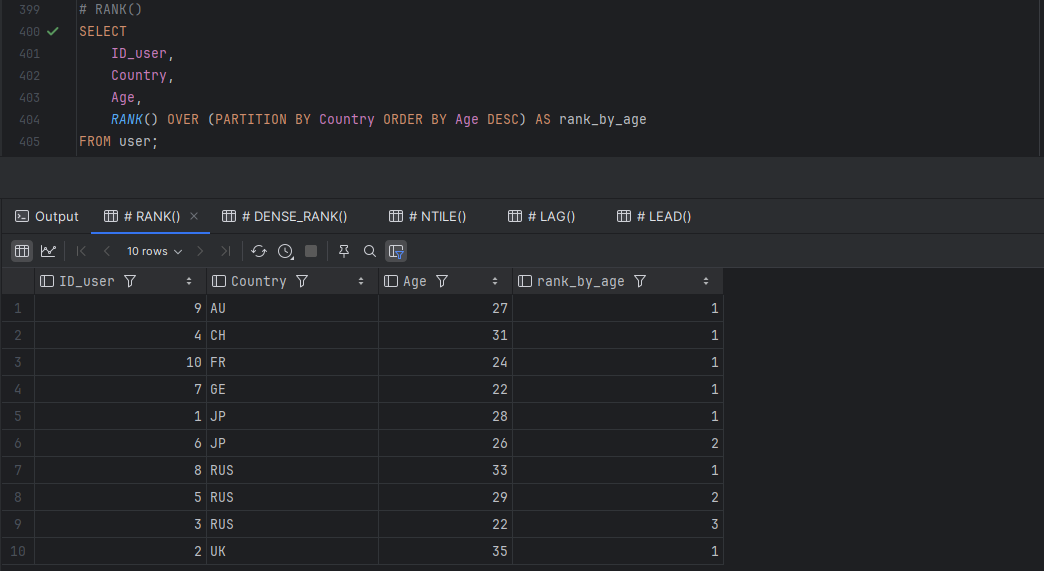


Рисунок 7 – Код и выполнение ранжирующей функции RANK()

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

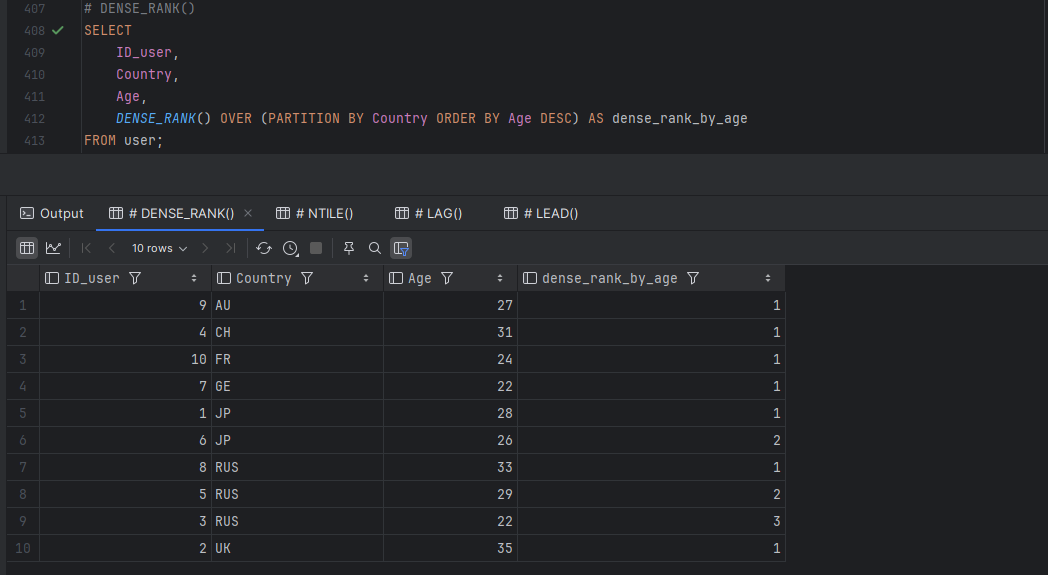


Рисунок 8 – Код и выполнение ранжирующей функции DENSE\_RANK()

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

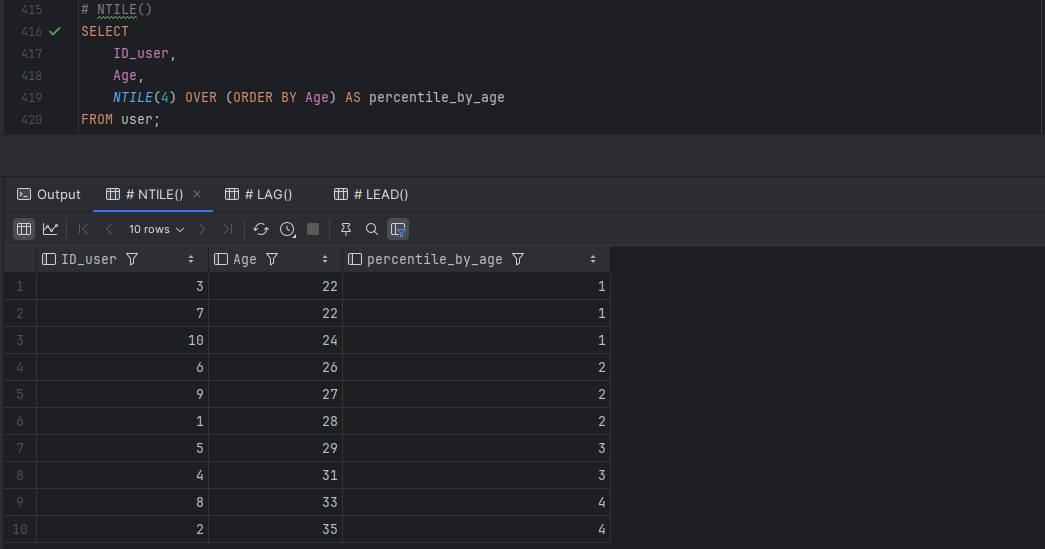


Рисунок 9 – Код и выполнение ранжирующей функции NTILE()

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;

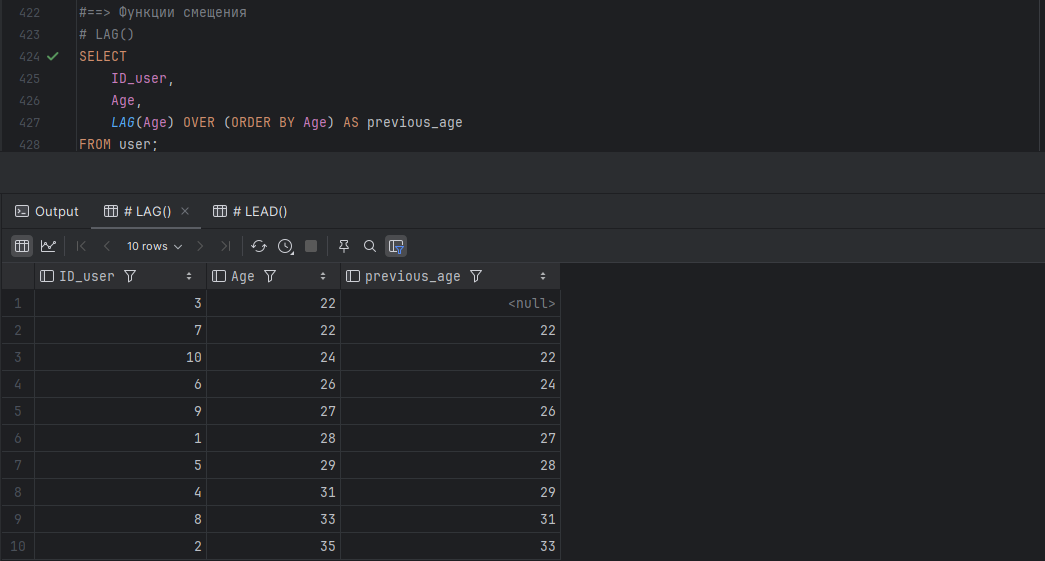


Рисунок 10 – Код и выполнение функции смещения LAG()

Теперь отобразим функцию смещения LEAD

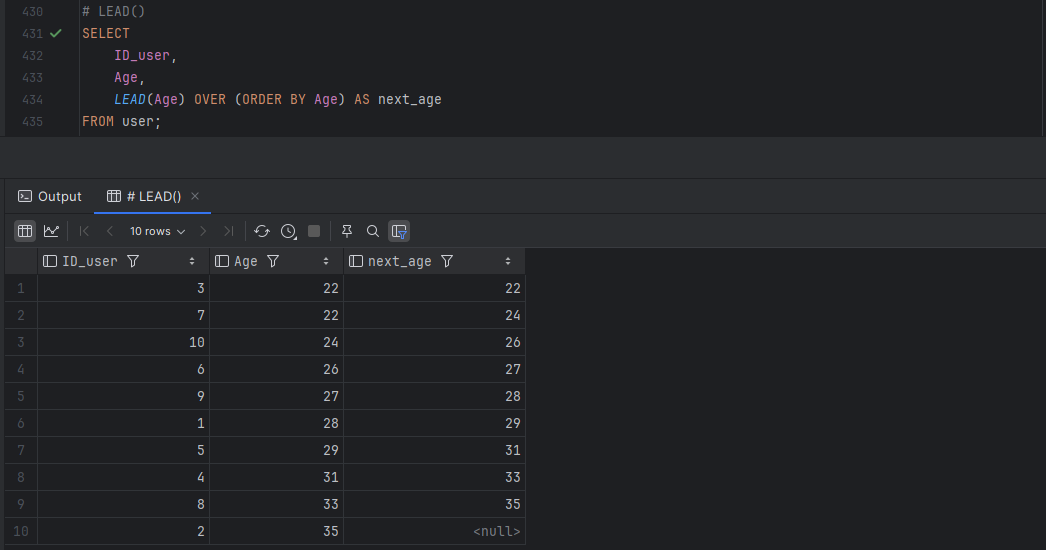


Рисунок 11 – Код и выполнение функции смещения LAG()