***Задача\_1***

***Таб.1 users***

CREATE DATABASE mydb;

USE mydb;

CREATE TABLE user (

user\_id INT PRIMARY KEY AUTO\_INCREMENT,

created\_date DATE NOT NULL,

registration\_type VARCHAR(50) NOT NULL,

contact VARCHAR(255),

country VARCHAR(50) NOT NULL

);

INSERT INTO user (created\_date, registration\_type, contact, country) VALUES

('2022-03-01', 'PHONE', '+79999999999', 'Russia'),

('2023-10-02', 'VK', 'aaaaaaa', 'Russia'),

('2022-09-04', 'EMAIL', 'aaaa@aaa.ru', 'Belarus'),

('2022-11-02', 'VK', 'bbbbbbb', 'Armenia');

SET SQL\_SAFE\_UPDATES = 0;

UPDATE user

SET contact = CONCAT('+7', FLOOR(RAND() \* 9000000000) + 1000000000)

WHERE contact IS NULL;

UPDATE user

SET registration\_type = 'PHONE'

WHERE registration\_type IS NULL;

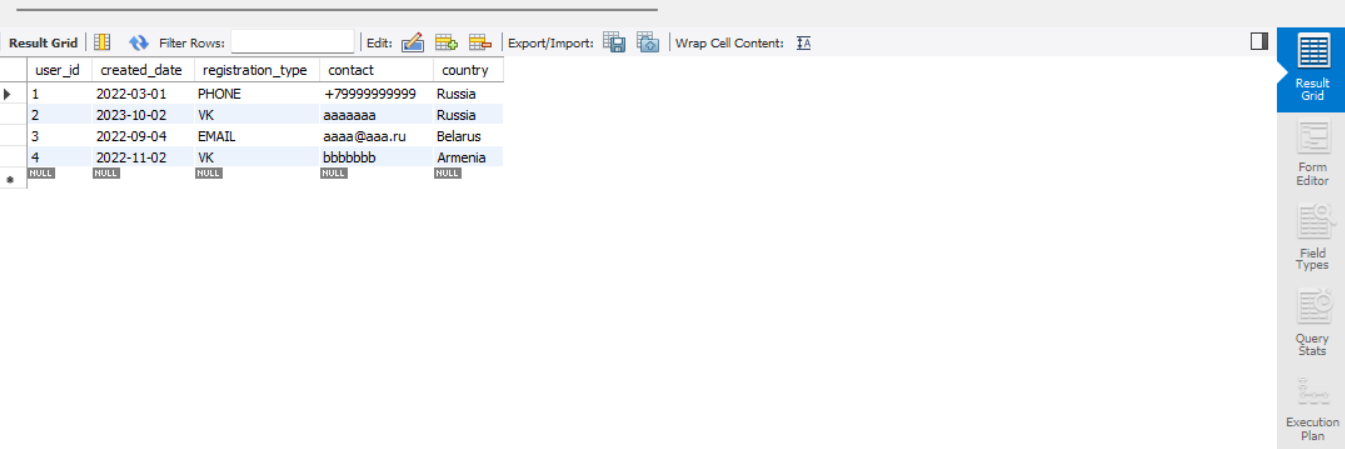
UPDATE user

SET country = 'Unknown Country'

WHERE country IS NULL;

SELECT \*

FROM user



***Таб.2 clickstream***

CREATE DATABASE mydb;

USE mydb;

CREATE TABLE clickstream (

event\_id INT PRIMARY KEY AUTO\_INCREMENT,

datetime TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

user\_id INT,

event VARCHAR(50) NOT NULL,

os VARCHAR(50),

screen VARCHAR(50),

video\_id INT,

install\_channel VARCHAR(50),

buy\_type VARCHAR(50),

FOREIGN KEY (user\_id) REFERENCES user(user\_id)

);

INSERT INTO clickstream (user\_id, event, os, screen, video\_id, install\_channel, buy\_type) VALUES

(1, 'video\_start', 'IOS', 'SEARCH', 10233, NULL, NULL),

(2, 'install', 'ANDROID', 'MAIN', NULL, 'STORE', NULL),

(2, 'buy', 'ANDROID', 'MAIN', NULL, NULL, 'TRIAL'),

(2, 'video\_start', 'ANDROID', 'COLLECTION', 12412, NULL, NULL),

(4, 'install', 'IOS', 'MAIN', NULL, 'LANDING', NULL),

(2, 'buy', 'ANDROID', 'MAIN', NULL, NULL, 'PREMIUM');

UPDATE clickstream

SET video\_id = FLOOR(RAND() \* 20000) + 1000

WHERE video\_id IS NULL;

UPDATE clickstream

SET install\_channel = 'ORGANIC'

WHERE install\_channel IS NULL;

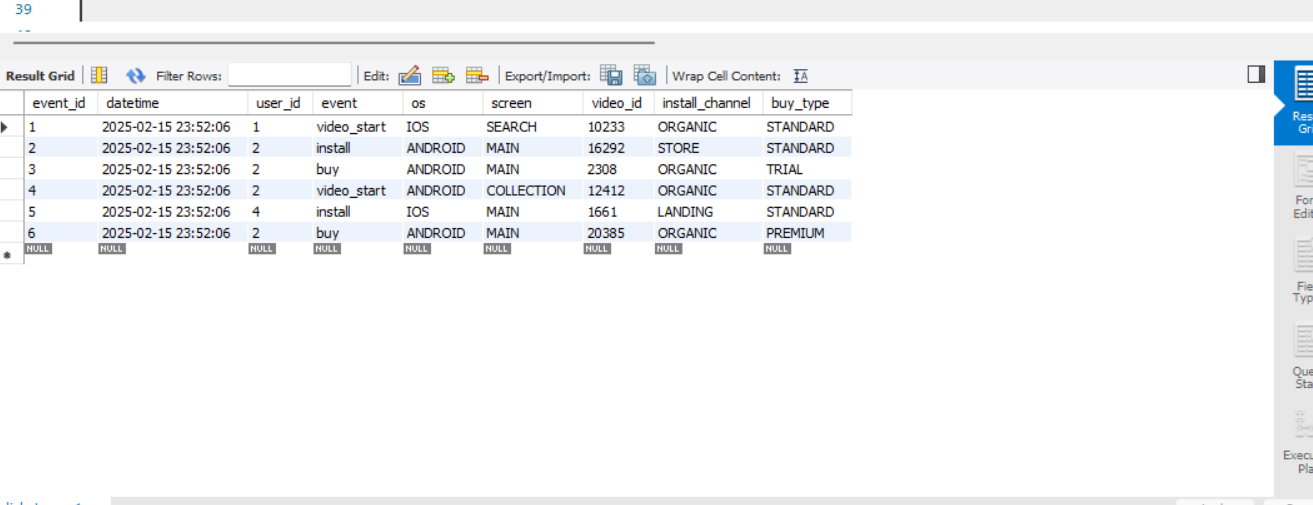
UPDATE clickstream

SET buy\_type = 'STANDARD'

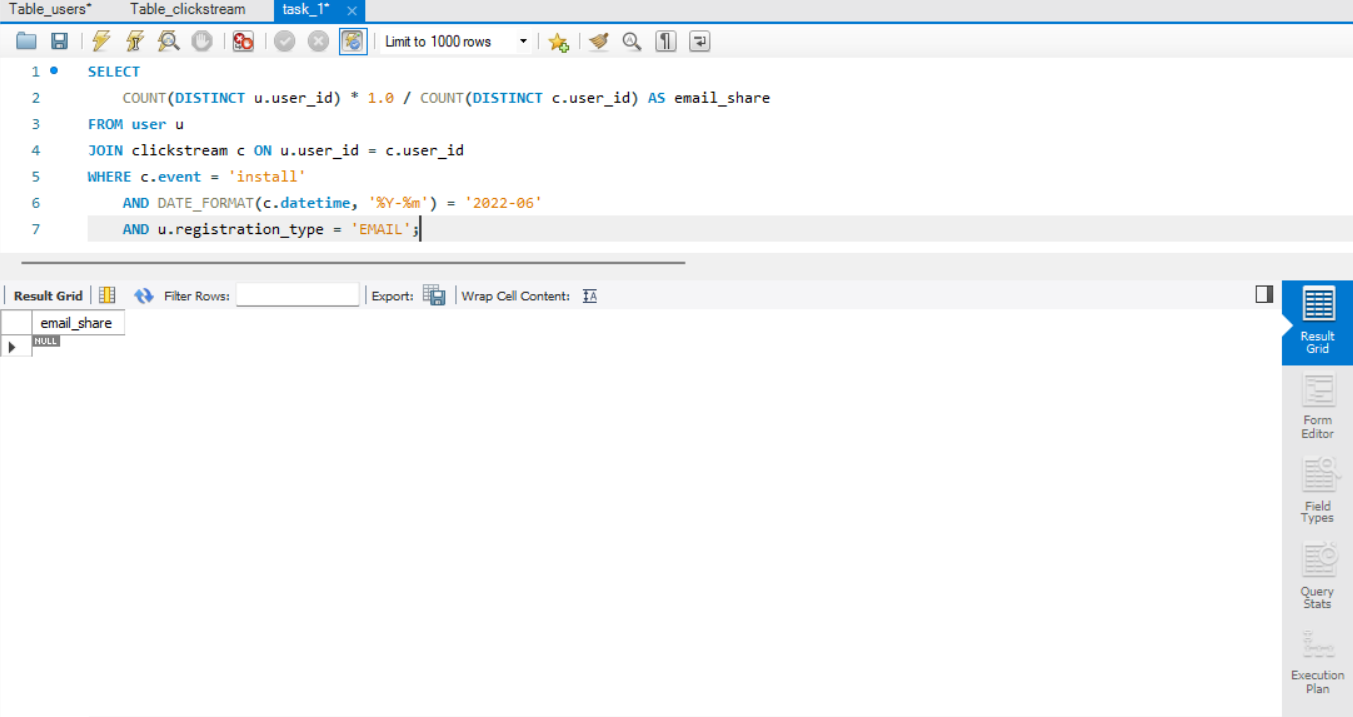
WHERE buy\_type IS NULL;

SELECT \*

FROM clickstream



Задание\_1



**🔹 Краткое пояснение логики запроса**

Этот SQL-запрос вычисляет **долю пользователей, установивших приложение в июне 2022 года, которые зарегистрировались через EMAIL**.

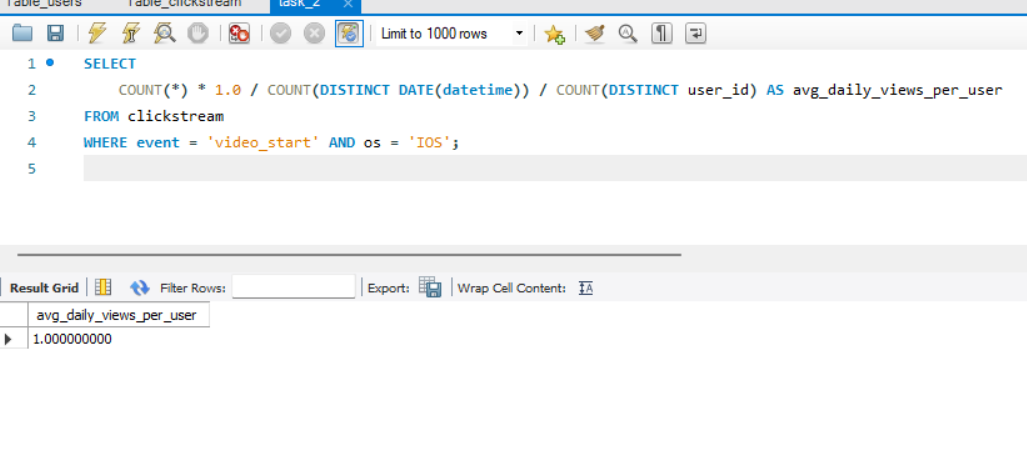
**🔍 Разбор запроса:**

1. **Объединяем таблицы user и clickstream по user\_id** → JOIN clickstream c ON u.user\_id = c.user\_id
2. **Фильтруем только события установки приложения (install)** → WHERE c.event = 'install'
3. **Фильтруем установки, произошедшие в июне 2022 года** → DATE\_FORMAT(c.datetime, '%Y-%m') = '2022-06'
4. **Фильтруем пользователей, зарегистрированных через EMAIL** → u.registration\_type = 'EMAIL'
5. **Считаем количество уникальных пользователей, зарегистрированных через EMAIL и установивших приложение** → COUNT(DISTINCT u.user\_id)
6. **Считаем общее количество пользователей, установивших приложение** → COUNT(DISTINCT c.user\_id)
7. **Делим первое число на второе, чтобы получить долю (email\_share)**.

**📌 Итог:**

Запрос показывает, **какая часть установивших приложение пользователей зарегистрировалась через EMAIL**.

Задание\_2



### ****🔹 Краткое пояснение логики запроса****

Этот SQL-запрос вычисляет **среднее количество просмотров видео в день на одного iOS-пользователя**.

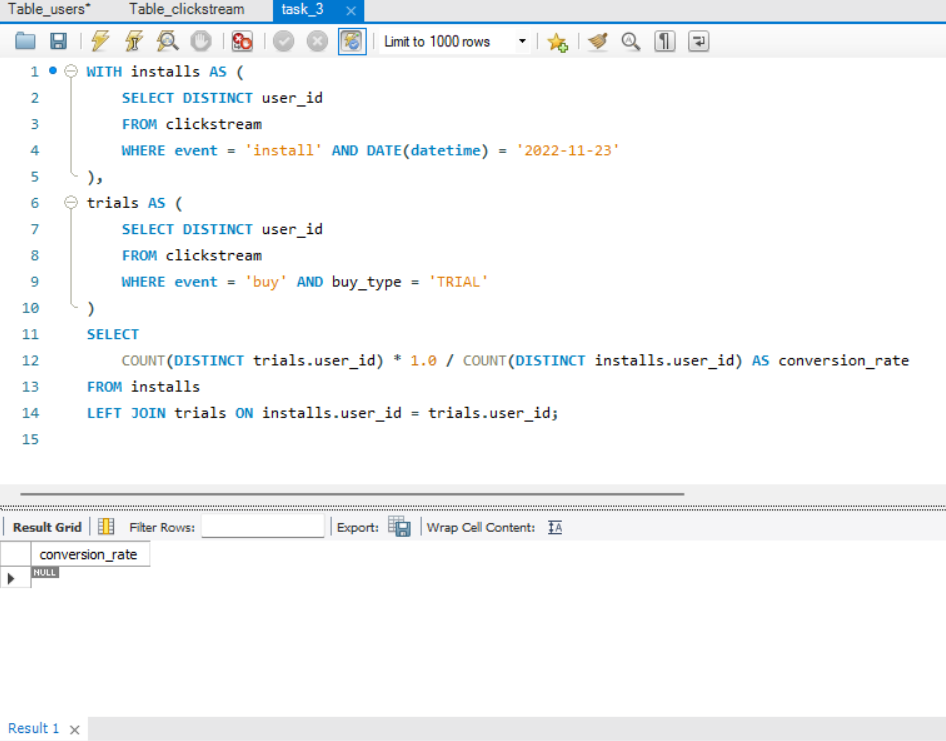
#### ****🔍 Разбор запроса:****

1. **Фильтруем только события video\_start для iOS-пользователей** → WHERE event = 'video\_start' AND os = 'IOS'
2. **Считаем общее количество просмотров видео (COUNT(\*))**
3. **Считаем количество уникальных дней (COUNT(DISTINCT DATE(datetime)))**
4. **Считаем количество уникальных пользователей (COUNT(DISTINCT user\_id))**
5. **Вычисляем среднее число просмотров в день на одного пользователя**:
   1. Делим общее количество просмотров на **количество дней**
   2. Затем делим на **количество уникальных пользователей**

#### ****📌 Итог:****

Запрос показывает **сколько видео в среднем смотрит один iOS-пользователь за день**.

Задание\_3



**🔹 Краткое пояснение логики запроса**

Этот SQL-запрос вычисляет **конверсию из установки приложения в покупку триальной (пробной) подписки для пользователей, установивших приложение 23 ноября 2022 года**.

**🔍 Разбор запроса:**

1. **Создаём временную таблицу (installs)**

* Выбираем **уникальных пользователей (user\_id)**, которые **установили приложение 23 ноября 2022** → WHERE event = 'install' AND DATE(datetime) = '2022-11-23'

1. **Создаём временную таблицу (trials)**

* Выбираем **уникальных пользователей (user\_id)**, которые **оформили триальную подписку (buy\_type = 'TRIAL')**

1. **Соединяем пользователей из installs с trials через LEFT JOIN**

* Это позволит сохранить **всех пользователей, установивших приложение**, даже если они **не оформили подписку**

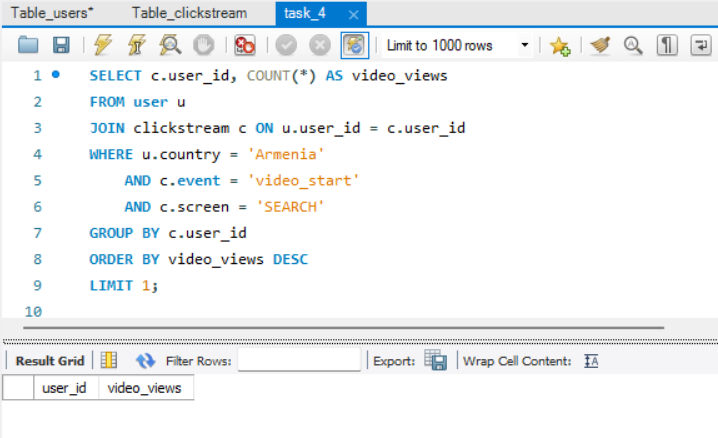
1. **Вычисляем конверсию:**

* COUNT(DISTINCT trials.user\_id) → **Считаем, сколько пользователей оформили триал**
* COUNT(DISTINCT installs.user\_id) → **Считаем, сколько пользователей установили приложение 23 ноября**
* **Делим первое число на второе**, чтобы получить **долю установивших, которые оформили триал**

**📌 Итог:**

Запрос показывает **процент пользователей, которые оформили пробную подписку после установки приложения 23 ноября 2022 года**.

Задание\_4



### ****🔹 Краткое пояснение логики запроса****

Этот SQL-запрос **определяет пользователя из Армении, который совершил наибольшее количество просмотров видео на экране поиска**.

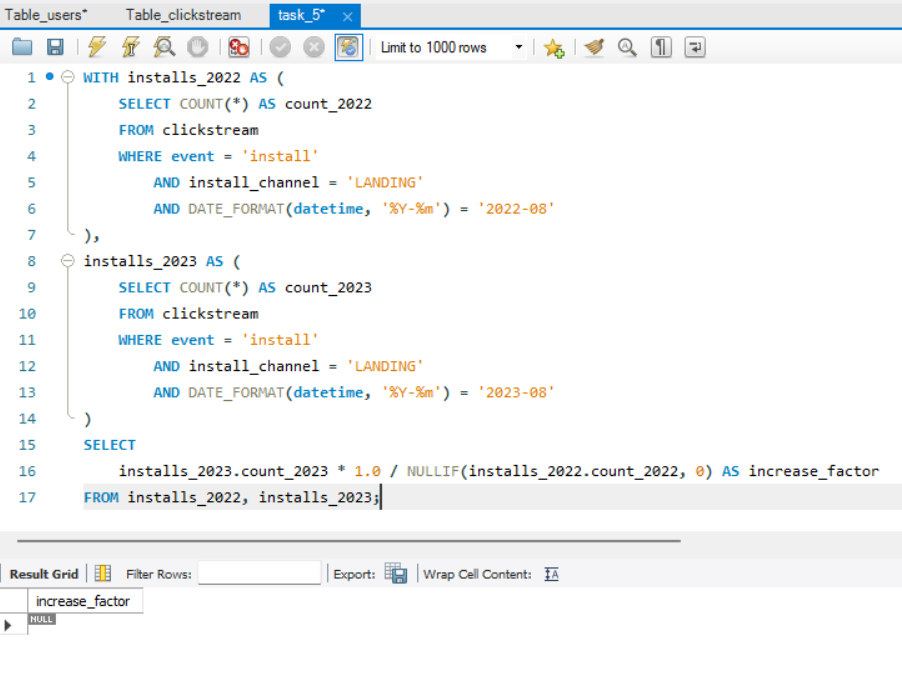
#### ****🔍 Разбор запроса:****

1. **Объединяем таблицы user и clickstream по user\_id** → JOIN clickstream c ON u.user\_id = c.user\_id
2. **Фильтруем только пользователей из Армении** → WHERE u.country = 'Armenia'
3. **Оставляем только просмотры видео (video\_start) на экране поиска (SEARCH)**
4. **Группируем данные по пользователю** → GROUP BY c.user\_id
5. **Считаем количество просмотров для каждого пользователя** → COUNT(\*) AS video\_views
6. **Сортируем пользователей по убыванию количества просмотров** → ORDER BY video\_views DESC
7. **Берём только одного пользователя с максимальным количеством просмотров** → LIMIT 1

#### ****📌 Итог:****

Запрос возвращает **ID пользователя из Армении, который чаще всего смотрел видео на экране поиска**.

Задание\_5



**🔹 Краткое пояснение логики запроса**

Этот SQL-запрос **вычисляет, во сколько раз увеличилось количество установок с лендингов в августе 2023 года по сравнению с августом 2022 года**.

**🔍 Разбор запроса:**

1. **Создаём временную таблицу installs\_2022**, которая:

* Считает **количество установок (install) с лендингов (install\_channel = 'LANDING') в августе 2022 года**
* Фильтрует по дате → DATE\_FORMAT(datetime, '%Y-%m') = '2022-08'
* Сохраняет результат в count\_2022

1. **Создаём временную таблицу installs\_2023**, которая:

* Аналогично считает **установки с лендингов, но уже за август 2023 года**
* Сохраняет результат в count\_2023

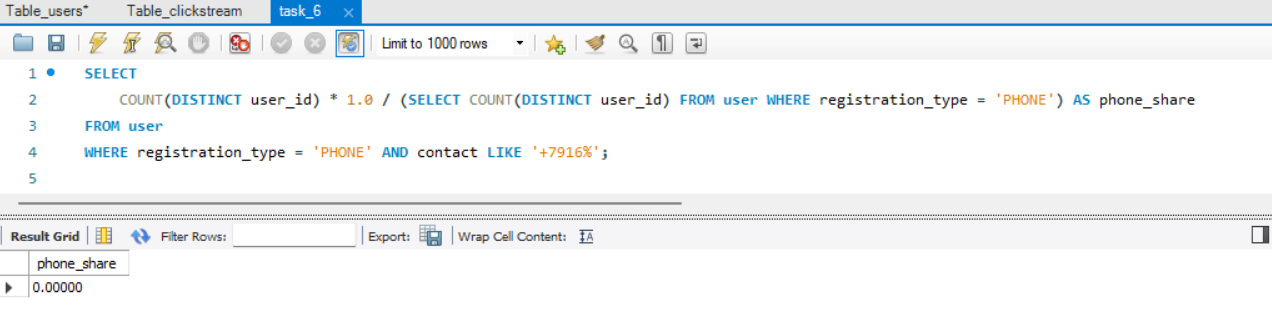
1. **В финальном SELECT-запросе:**

* Делим **количество установок в 2023 году** на **количество установок в 2022 году**
* Используем NULLIF(installs\_2022.count\_2022, 0), чтобы избежать деления на 0 (если в 2022 году не было установок)

**📌 Итог:**

Запрос показывает **во сколько раз увеличилось количество установок с лендингов в августе 2023 года по сравнению с августом 2022 года**.

Задача\_6



**🔹 Краткое пояснение логики запроса**

Этот SQL-запрос **вычисляет долю пользователей, зарегистрировавшихся через телефон, чей номер начинается с кода +7916**.

**🔍 Разбор запроса:**

1. **Фильтруем только пользователей, зарегистрированных через телефон** → WHERE registration\_type = 'PHONE'
2. **Из них выбираем тех, чей номер начинается с +7916** → AND contact LIKE '+7916%'
3. **Считаем количество таких пользователей (COUNT(DISTINCT user\_id))**
4. **Делим на общее количество пользователей с регистрацией по телефону**:

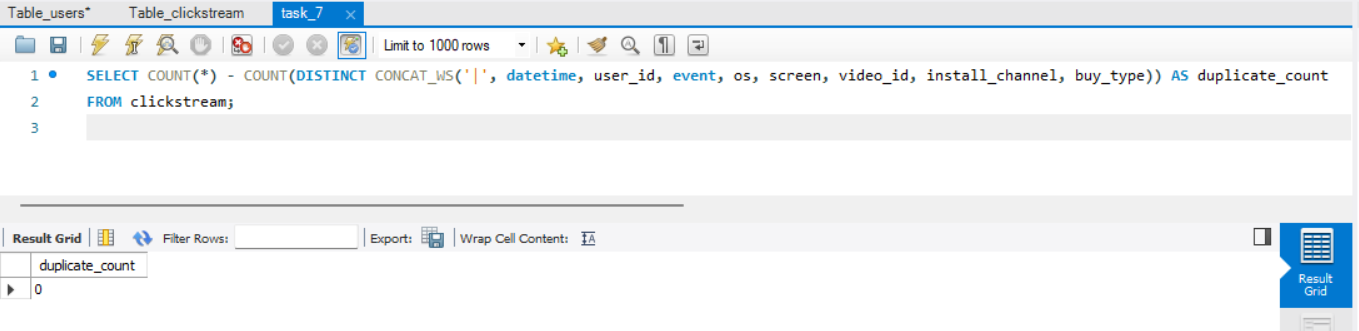
* В подзапросе (SELECT COUNT(DISTINCT user\_id) FROM user WHERE registration\_type = 'PHONE') считаем **всех пользователей, зарегистрировавшихся через телефон**

1. **Домножаем на 1.0, чтобы получить результат в виде десятичной дроби (долю, а не целое число)**

**📌 Итог:**

Запрос возвращает **процент (долю) пользователей, зарегистрированных через телефон, чей номер начинается с +7916**.

Задание\_7



**🔹 Краткое пояснение логики запроса**

Этот SQL-запрос **определяет количество дубликатов (повторяющихся строк) в таблице clickstream**.

**🔍 Разбор запроса:**

1. **Считаем общее количество строк в таблице** → COUNT(\*)
2. **Определяем количество уникальных строк**:

* Используем COUNT(DISTINCT ...) для подсчёта **только уникальных комбинаций значений**
* CONCAT\_WS('|', datetime, user\_id, event, os, screen, video\_id, install\_channel, buy\_type) объединяет все столбцы в одну строку
* Это позволяет считать **дублирующимися те строки, у которых все значения совпадают**

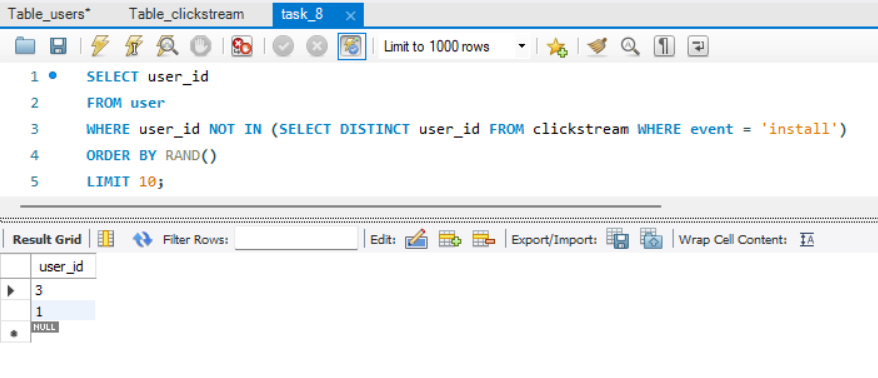
1. **Вычисляем разницу** между общим числом строк и числом уникальных строк:

* COUNT(\*) - COUNT(DISTINCT ...)
* Если разница больше 0, значит, в таблице есть дубликаты

**📌 Итог:**

Запрос возвращает **количество повторяющихся записей (дубликатов) в таблице clickstream**.

Задание\_8



**🔹 Краткое пояснение логики запроса**

Этот SQL-запрос **выбирает 10 случайных пользователей, которые никогда не устанавливали приложение**.

**🔍 Разбор запроса:**

1. **Выбираем пользователей из таблицы user**
2. **Фильтруем тех, у кого user\_id отсутствует в clickstream для события install**:

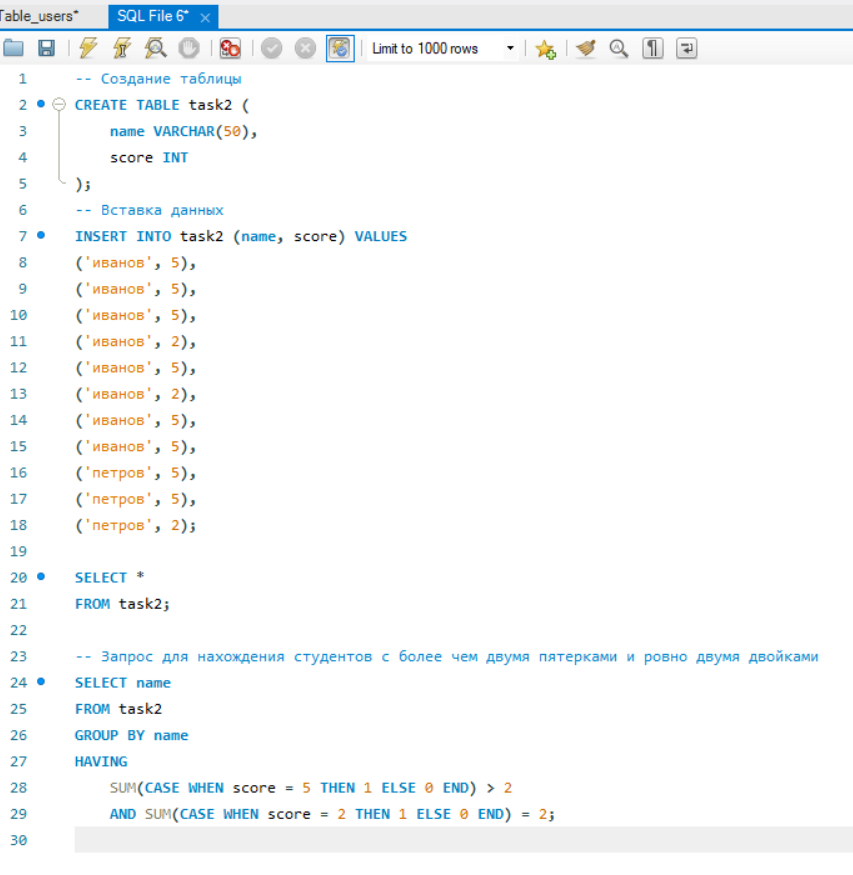
* Используем NOT IN (SELECT DISTINCT user\_id FROM clickstream WHERE event = 'install')
* Это означает, что пользователь **никогда не совершал установку (install)**

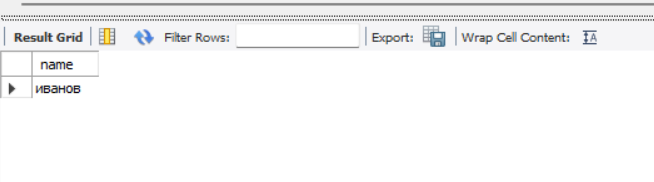
1. **Сортируем пользователей случайным образом** → ORDER BY RAND()
2. **Ограничиваем результат 10 строками** → LIMIT 10

**📌 Итог:**

Запрос выбирает **10 случайных пользователей, которые никогда не устанавливали приложение**.

***Задача\_2***





**🔍 Разбор:**

**1. Группировка (GROUP BY)**

Запрос группирует данные по name (то есть по студентам). Это значит, что все оценки одного студента объединяются в одну строку.

**2. Подсчёт пятёрок и двоек (SUM(CASE WHEN … THEN 1 ELSE 0 END))**

* SUM(CASE WHEN score = 5 THEN 1 ELSE 0 END) – считает количество оценок 5 у каждого студента.
* SUM(CASE WHEN score = 2 THEN 1 ELSE 0 END) – считает количество оценок 2 у каждого студента.

**3. Фильтрация (HAVING)**

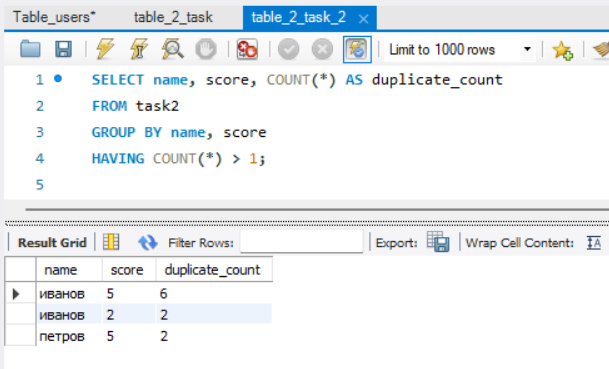
* > 2 — оставляет только тех, у кого **больше двух пятёрок**.
* = 2 — оставляет только тех, у кого **ровно две двойки**.

**4. Итог**

Запрос возвращает только тех студентов, у которых **более двух пятёрок** и **ровно две двойки**.

Результат: **Иванов**.

***Задача\_3***

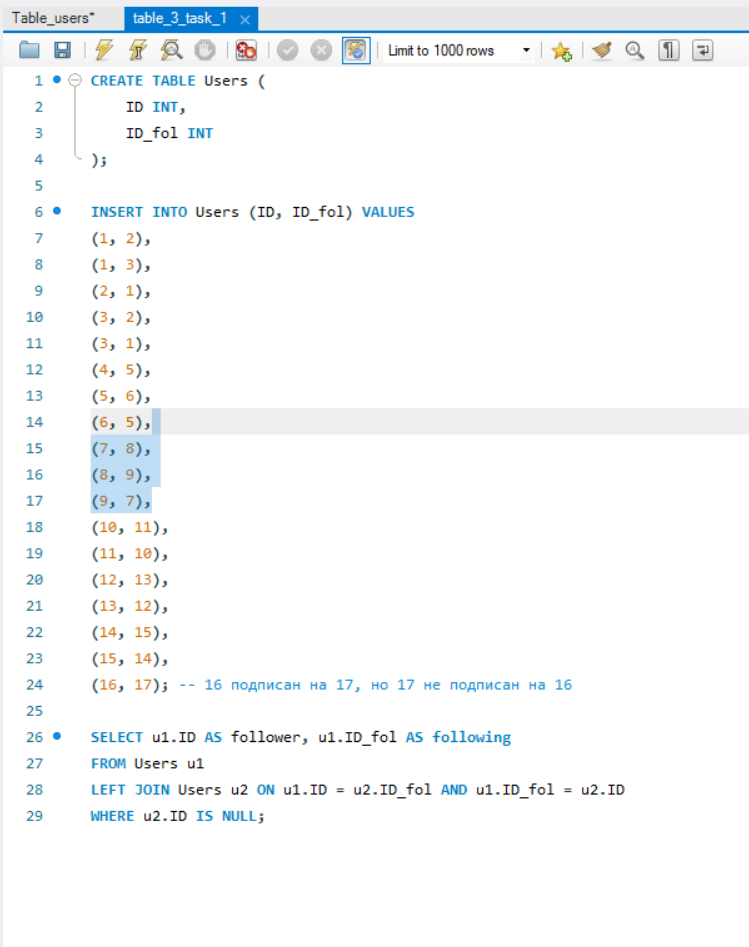


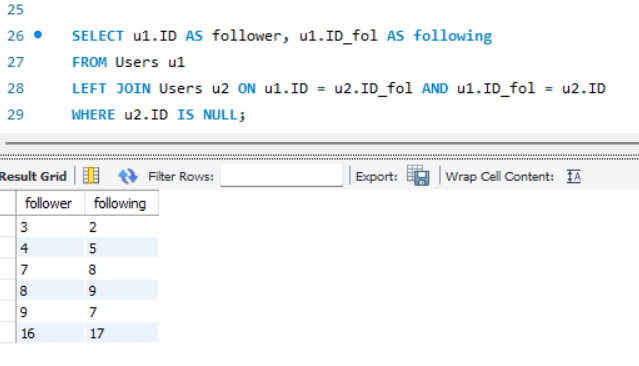
**🔍 Разбор:**

* GROUP BY name, score группирует строки по name и score.
* COUNT(\*) считает количество повторений каждой уникальной комбинации name и score.
* HAVING COUNT(\*) > 1 оставляет только те записи, которые встречаются более одного раза (то есть дубликаты).

**Вывод:**  
Этот запрос покажет, какие комбинации (name, score) повторяются и сколько раз. Если duplicate\_count = 2, значит такая комбинация встречается дважды, если duplicate\_count = 5, то пять раз и т. д.

***Задача\_3***





### 🔍 Разбор:

1. Users u1 — основная таблица, из которой берем все подписки.
2. LEFT JOIN Users u2 ON u1.ID = u2.ID\_fol AND u1.ID\_fol = u2.ID — ищем обратную подписку (взаимную).
3. WHERE u2.ID IS NULL — оставляем только те случаи, когда обратной подписки нет (то есть NULL).

**Вывод:**  
Этот запрос покажет всех пользователей, которые подписаны на кого-то, но не получили ответной подписки.