**JOIN**

**Если нужно найти не совпадения по 2 таблицам**

Suppose that a website contains two tables, the Customers table and the Orders table. Write a SQL query to find all customers who never order anything.

Table: Customers.

+----+-------+

| Id | Name |

+----+-------+

| 1 | Joe |

| 2 | Henry |

| 3 | Sam |

| 4 | Max |

+----+-------+

Table: Orders.

+----+------------+

| Id | CustomerId |

+----+------------+

| 1 | 3 |

| 2 | 1 |

+----+------------+

Using the above tables as example, return the following:

+-----------+

| Customers |

+-----------+

| Henry |

| Max |

+-----------+

Используя левый джоин – данные покупателей будут заполнены все, а вот данные покупок, там где не было покупателя – будут заполнены NULL. Сделать выборку по NULL

SELECT Name AS 'Customers'

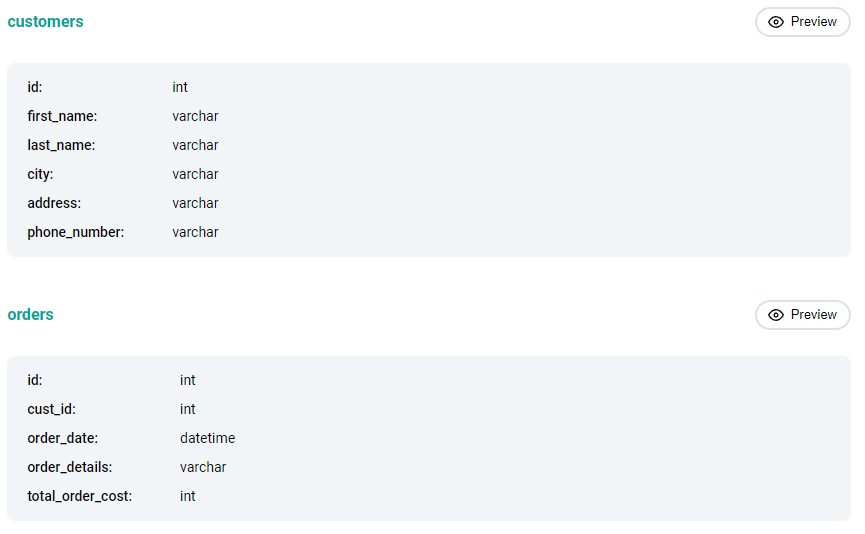
FROM Customers c LEFT JOIN Orders o

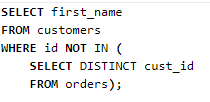
ON c.Id = o.CustomerId

WHERE o.CustomerId IS NULL

**ИЛИ**

Find customers who have never made an order. Output the first name of the customer.





**Если нужно что то сравнить в рамках 1 таблицы – когда данные внутри одной таблицы ссылаются друг на друга**

The Employee table holds all employees including their managers. Every employee has an Id, and there is also a column for the manager Id.

+----+-------+--------+-----------+

| Id | Name | Salary | ManagerId |

+----+-------+--------+-----------+

| 1 | Joe | 70000 | 3 |

| 2 | Henry | 80000 | 4 |

| 3 | Sam | 60000 | NULL |

| 4 | Max | 90000 | NULL |

+----+-------+--------+-----------+

**Given the**Employee**table, write a SQL query that finds out employees who earn more than their managers. For the above table, Joe is the only employee who earns more than his manager.**

+----------+

| Employee |

+----------+

| Joe |

В начале создаем 2 таблицы, что бы их сравнивать

SELECT

a.Name AS 'Employee'

FROM

Employee AS a,

Employee AS b

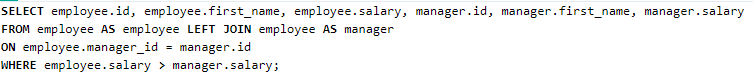
После нужно прилинковать айдишник менеджера к айдишнику сотрудника к кому он привязан

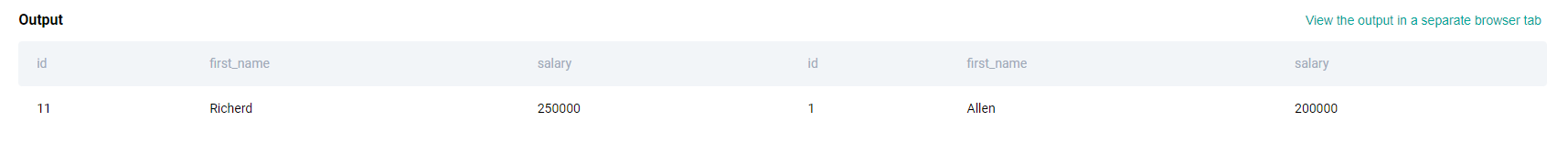
WHERE a.ManagerId = b.Id

После сравниваем их зарплаты

AND a.Salary > b.Salary

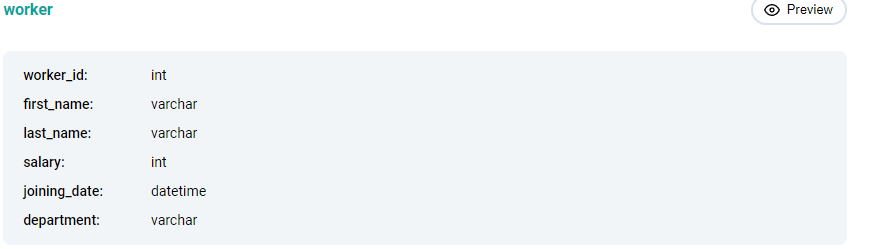
**Или так**

****

****

Мы подсоединили к сотрудника его менеджера. И теперь можем сравнивать их данные

**Find employees with the same salary. Output the worker id along with the first name and the salary.**





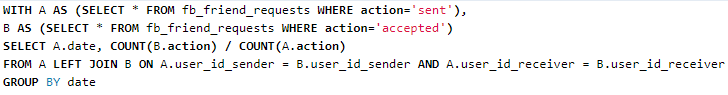
Соединяем работников по зарплате (так как нужно найти именно совпадения по ней), но worker\_id должен не совпадать (что бы не было соединения работника с самим собой)

**Если есть 1 таблица и нужно узнать отношение каких то сумм данных после фильтрации к сумме других данных (данные должны быть логически связаны – например действие ‘отправить’ и действие ‘принять’)**

What is the overall friend acceptance rate by date? Your output should have the rate of acceptances by the date the request was sent. Order by the earliest date to latest.

Assume that each friend request starts by a user sending (i.e., user\_id\_sender) a friend request to another user (i.e., user\_id\_receiver) that's logged in the table with action = 'sent'. If the request is accepted, the table logs action = 'accepted'. If the request is not accepted, no record of action = 'accepted' is logged.

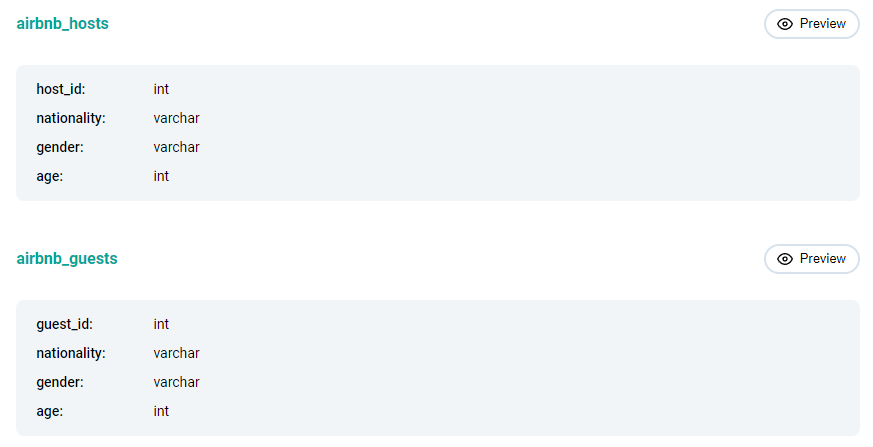
* Assume that each friend request starts by a user sending a friend request that's logged in the table with action = 'sent'
* When a friend request is accepted, a notification is sent to the user that sent the original friend request and action = 'accepted'. This also means that the original user\_id\_sender is now logged as user\_id\_accepted and vice versa for the original receiver. You'll need to filter your tables using this logic.
* Create two tables using subqueries where you're filtering out 'sent' and 'accepted' rows
* Join the two tables using a left join
* The acceptance rate calculation should be (number of acceptance / number of sent requests)
* Group by the friend request sent date and order by DESC

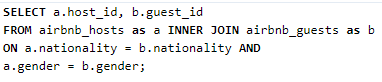
****

1. Создаем 2 таблицы, с отфильтрованными значениями
2. Джойним их так, что бы для юзера кто не принял еще запрос был NULL
3. Суммируем кол-во посланных запросов и принятых (поля где NULL будут считаться как 0)
4. Делим
5. Группируем по дате, что бы результат был за каждое число

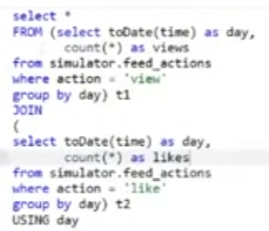
**Если нужно найти совпадения по таблицам, но в качестве ключа используется не их айдишники, а что то другое**

**Find matching hosts and guests pairs in a way that they are both of the same gender and nationality. Output the host id and the guest id of matched pair.**

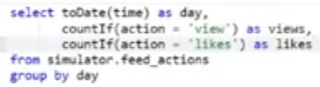




**CLICKHOUSE – найти просмотры и лайки по дням**

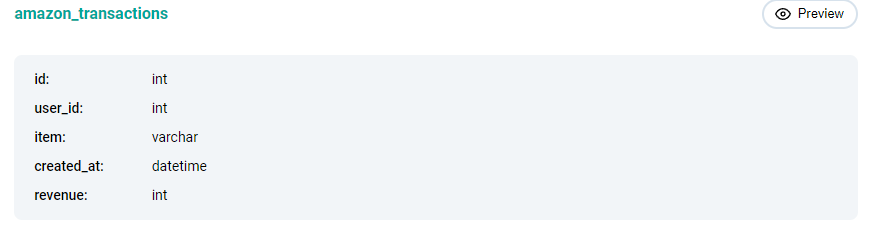


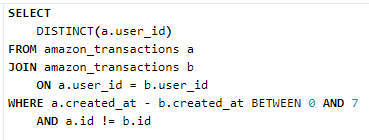
Или так



**Если нужно посчитать какой временной промежуток, и есть 1 таблица**

Write a query that'll identify returning active users. A returning active user is a user that has made a second purchase within 7 days of any other of their purchases. Output a list of user\_ids of these returning active users.





* 1. Джойним таблицу саму с собой, по user\_id
  2. Выполняем фильтрацию высчитываю разницу дат
  3. Что бы одна и та же строка не джойнилась сама с собой, то указываем что айдишники не должны совпадать. Тем самым мы заджойним одинаковых юзеров, но теперь каждый джоин строк не будет совпадать

Можно так же выполнить через оконную функцию **LAG()**