**Задание 1**

**SQL-запрос находится в файле «Task1».**

**Задание 2**

**SET max\_parallel\_workers = 8;**

**SET max\_parallel\_workers\_per\_gather = 3;**

**SHOW max\_parallel\_workers;**

**SHOW max\_parallel\_workers\_per\_gather;**

CREATE OR REPLACE FUNCTION previous\_date (date DATE) RETURNS DATE

LANGUAGE sql

**PARALLEL SAFE**

AS

$$

SELECT date - 1;

$$;

Ещё вариант **ALTER TABLE large\_table SET (parallel\_workers = 3);**

EXPLAIN analyze SELECT

previous\_date(date)

FROM

large\_table;

Оптимизатор сам определяет, как эффективнее произвести сканирование последовательно или параллельно.

*Но стоит обратить внимание на следующие параметры:*

**max\_parallel\_workers\_per\_**gather — это наименьшее ограничение на количество воркеров. Во-вторых, исполнитель запросов берет воркеры из пула, ограниченного размером **max\_parallel\_workers**. Наконец, предел верхнего уровня — это **max\_worker\_processes**: общее количество фоновых процессов.

**min\_parallel\_table\_scan\_size**='8MB'

8MB table => 1 worker

24MB table => 2 workers

72MB table => 3 workers

Каждый раз, когда таблица в 3 раза больше, чем min\_parallel\_(index|table)\_scan\_size, postgres добавляет worker.

Помимо длинного списка ограничений параллельного выполнения, PostgreSQL проверяет стоимость:

**parallel\_setup\_cost**, чтобы избежать параллельного выполнения коротких запросов. Он моделирует время, затрачиваемое на настройку памяти, запуск процесса и первоначальную связь.

**parallel\_tuple\_cost**: Общение между лидером и работниками может занять много времени. Время пропорционально количеству кортежей, отправленных воркерами. Параметр моделирует стоимость связи.

**Задание 3**

До конца задание не сделала. Сделала как первое задание, но в разрезе install\_date, platform, is\_paid.

С использованием оконных функций - «Task3\_part1»

Без использования оконных функций - «Task3\_part2»

**Задание 4**

Проверку качества данных я буду проводить по следующим параметрам:

1. Отсутствующие значения;
2. дублированные записи;
3. достоверность данных:

* значения в определенном столбце должны иметь определенный тип данных;
* ограничения диапазона как правило, числа или даты должны попадать в определенный диапазон;
* некоторые столбцы не могут быть пустыми;
* значения столбца происходят из набора дискретных значений, например, пол человека может быть «мужской» или «женский». Т.е. фиксированный, редко изменяемый набор значений (или никогда не изменяющийся);
* данные имеют смысл;

***Часть 1***

Анализ данных (поиск ненормальных данных) я проводила с помощью языка программирования python и библиотеки pandas. Pandas- это самая популярная библиотека для анализа и обработки данных.

Дополнительно использовала PowerBI для просмотра файла и уточнения некоторых моментов. Плюс сам подсказывает ошибки в файле при несовпадении типов данных в колонках.

***Анализ с объяснением можно посмотреть в файле «Task4.ipynb».***

***Часть 2***

1. В колонке «country» 995 отсутствующих значения.

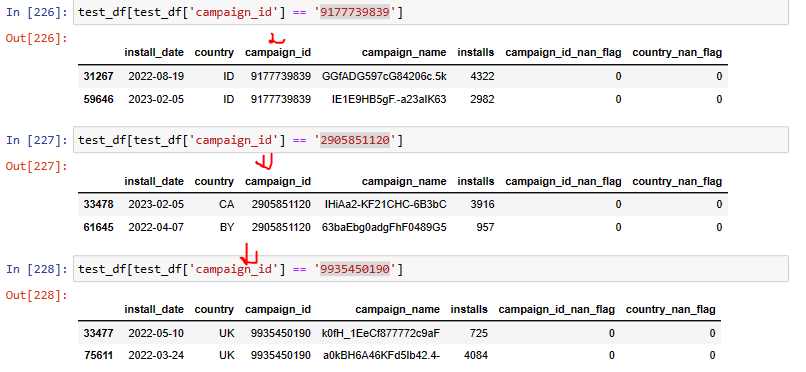
Я их заменила на 0 и установила бинарный флаг «country\_nan\_flag».

1. В колонке «campaign\_id» 2 отсутствующих значения.

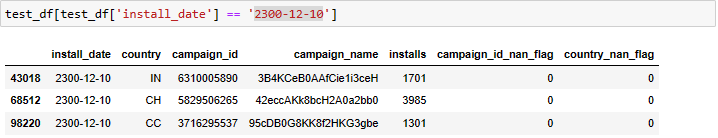
Я их заменила на 0 и установила бинарный флаг «campaign\_id\_nan\_flag».

1. В колонке «campaign\_id» есть повторения значений:

* 9177739839;
* 2905851120;
* 9935450190;



1. В колонке «install\_date» представлено 3 раза значение «2300-12-10». Что выходит из диапазона дат нашего анализа.



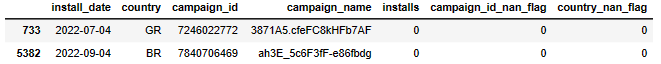
1. В колонке «install\_date» представлено 2 раза значение «1970-01-01». Что выходит из диапазона дат нашего анализа.



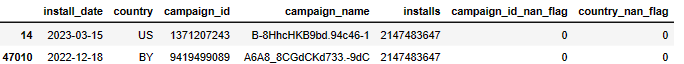
1. В колонке «installs» представлены 973 строки со значением **-1**, что не соответствует смыслу. Количество должно быть положительным числом.



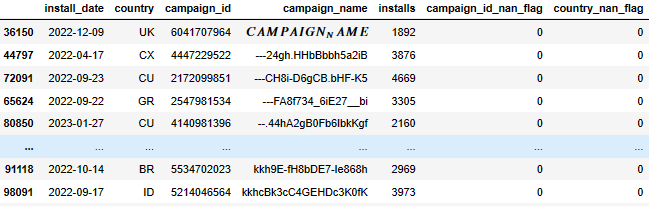
1. В колонке «installs» представлены 12 строк со значением **0**, что, по моему мнению, не нужно для анализа. Уточнить этот вопрос.



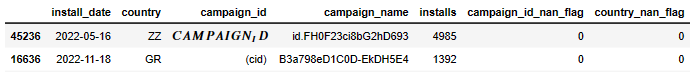
1. В колонке «installs» представлены 2 строки со значением **2147483647**, слишком большое число по сравнению с остальными.



1. В колонке «campaign\_name» представлена строка со значением **«**𝐶𝐴𝑀𝑃𝐴𝐼𝐺𝑁𝑁𝐴𝑀𝐸**»**, что не имеет смысла.
2. Уточнить какие символы разрешены в значениях колонки «campaign\_name» и определить шаблон значения, чтобы потом можно было сравнить с помощью регулярных выражений.

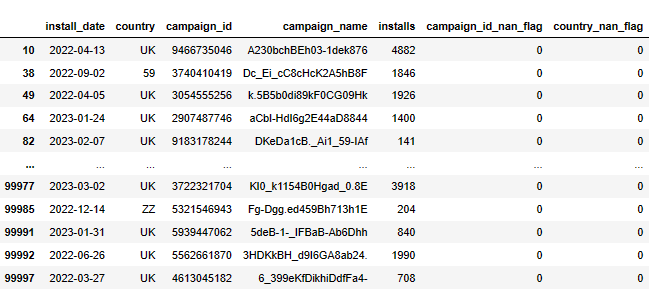


1. В колонке «campaign\_id» представлены 2 строки со строковыми значениями «𝐶𝐴𝑀𝑃𝐴𝐼𝐺𝑁𝐼𝐷» и «(cid)», хотя значения должны содержать только цифры.



1. В колонке «country» есть значения, которые не соответствуют двухбуквенному коду страны в формате ISO 3166.

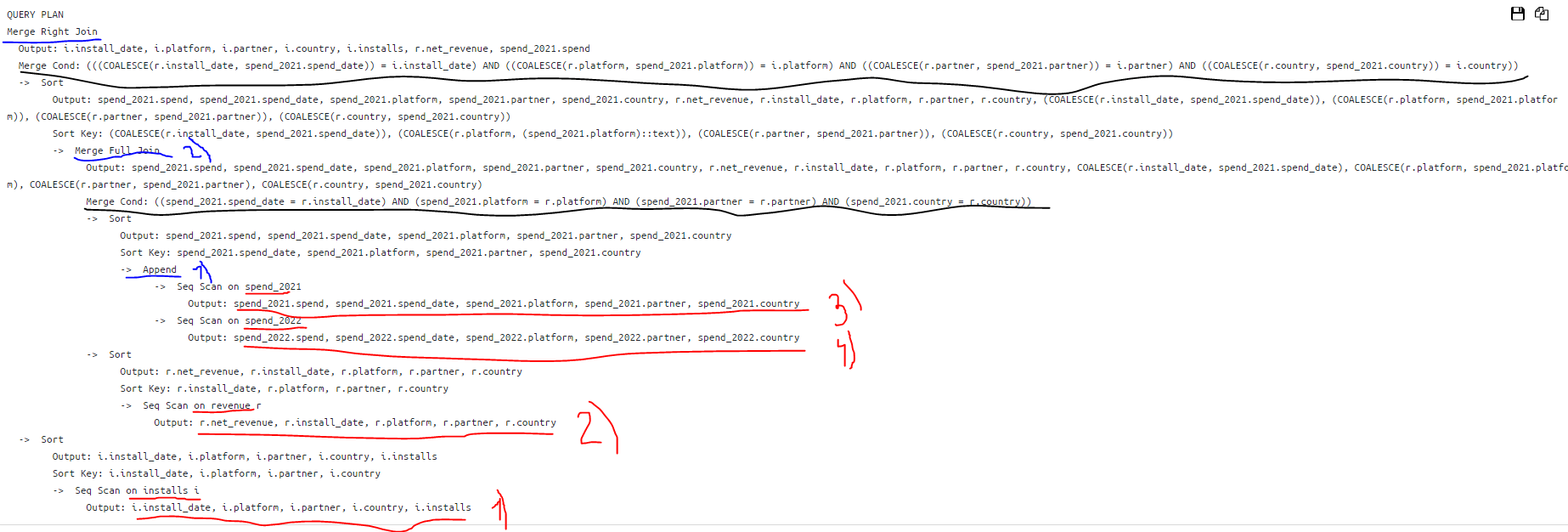
**59, EU, UK, ZZ.** – 12103 строк всего



**Задание 5**

***SQL-запрос можно посмотреть в файле «Task5.txt»***

*Немного объяснения*



Для удобного чтения плана запроса использовала сайт <https://explain.tensor.ru/>, без добавления в архив.

*Сразу можно увидеть, какие таблицы используются и какие у них поля:*

1. Таблица **installs**, присваивается псевдоним **i**

* install\_date;
* platform;
* partner;
* country;
* installs;

1. Таблица **revenue**, присваивается псевдоним **r**

* install\_date;
* platform;
* partner;
* country;
* net\_revenue;

1. Таблица **spend\_2021**

* spend\_date;
* platform;
* partner;
* country;
* spend

1. Таблица **spend\_2022**

* spend\_date;
* platform;
* partner;
* country;
* spend

*Основные действия, которые происходят в запросе:*

Чем правее стрелочка операнда, тем раньше происходит операция.

1. Append- происходит **union all** между таблицами **spend\_2021** и **spend\_2022**.
2. Merge Full Join – происходит **full join** между результатом соединением таблиц **spend\_2021** и **spend\_2022** и таблицей **revenue**.

По условию (**spend\_2021.spend\_date = r.install\_date) AND (spend\_2021.platform = r.platform) AND (spend\_2021.partner = r.partner) AND (spend\_2021.country = r.country)**)

1. Merge Right Join – результат предыдущего действия **right join**

с таблицей installs.

По условию **(((COALESCE(r.install\_date, spend\_2021.spend\_date)) = i.install\_date) AND ((COALESCE(r.platform, spend\_2021.platform)) = i.platform) AND ((COALESCE(r.partner, spend\_2021.partner)) = i.partner) AND ((COALESCE(r.country, spend\_2021.country)) = i.country))**)

**Функция** COALESCE

Функция COALESCE принимает неограниченное количество аргументов. Он возвращает первый аргумент, который не равен нулю. Если все аргументы равны нулю, COALESCE функция вернет значение null.

Функция COALESCE оценивает аргументы слева направо, пока не найдет первый ненулевой аргумент. Все остальные аргументы из первого ненулевого аргумента не оцениваются.

Остальные операнды **Sort и Seq Scan** происходят автоматически при произведении остальных вышесказанных действиях.

Я создала таблицы и далее писала к ним запрос с функцией explain для получения такого же плана запроса.

SQL-запросы для создания таблиц:

create table installs (install\_date date, platform varchar,partner varchar,country varchar, installs int);

create table revenue (install\_date date, platform varchar,partner varchar,country varchar, net\_revenue varchar);

create table spend\_2021 (spend\_date date, platform varchar,partner varchar,country varchar, spend varchar);

create table spend\_2022 (spend\_date date, platform varchar,partner varchar,country varchar, spend varchar);

**Замечание:**

Сам оптимизатор может изменить порядок чтения таблиц revenue и installs и вместо right join выполнить (показать в плане запроса) left join. Это может быть связано с размером таблиц. И таким образом, ПРАВОЕ соединение в этом случае указывает на тот же поток данных, что и ЛЕВОЕ соединение, которое мы определили.