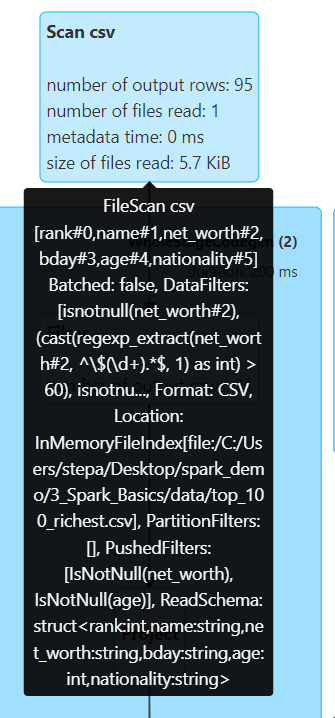
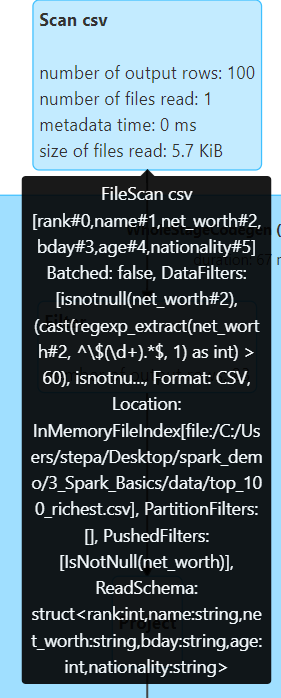
И так, раз вы здесь значит вы действительно пришли посмотреть на крутые штуки, поэтому не буду задерживать.

Начнём с самого простого что уже есть в коде. Думаю вы помните строку

df\_last = df\_prelast.filter(col('age').isNotNull()).select(col('rank'), col('name'), col('net\_worth'), col('bday'), col('age'), col('nationality')). Она была чуть ли не в самом конце. И мы тут видим фильтр, а именно чтобы поле col(‘age’) не было Null. Но что же если мы посмотрим в execution plan?



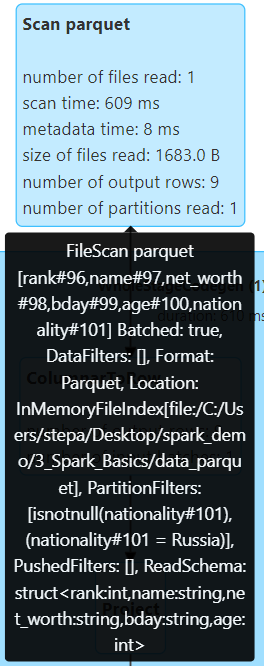
Думаю вы видите, что у нас есть PushedFilters, где есть две проверки на не Null и одна из как раз таки для поля age. Это означает, что Spark пропихнул фильтрацию по этому полю прямо на этап чтения данных из файла. Вы видите что количество выходных строк 95, спойлер: всего строк в файле 100. Получается что спарк реально провел эту фильтрацию, хотя мы и писали её аж в самом конце даже после Join. Вот пруф на счёт 100 строк(тут нет фильтрации по годам и те 5 строк снова грузятся):



Как видим, нет фильтра и уже 100 строк. Это тот самый predicate pushdown и здесь он в своей красе: строки даже не попадают в оперативку по сути ибо фильтрацию моментально. Не всегда так это работает, что все фильтры пропихиваются как можно ближе к считыванию с файла. Существует ряд трансформаций, которые не дадут спарку пропихнуть фильтры так далеко, ибо эти трансформации могут нуждаться в тех строках которые фильтр уберёт. Ещё один немаловажный момент, это типы данных. Например у нас есть фильтр, который фильтрует по net\_worth. Тем не менее, перед этим фильтром идёт достаточно сложное преобразование этого же поля, которое спарк не может сделать на лету. Хотите чтобы ваши фильтры проталкивались ближе к чтению с файла? Делайте фильтрацию с нужными типами полей, условно в файле INT значит и сравнивайте INT а не со строкой(хотя в таком случае спарк всё равно справится, но с более сложной ситуацией уже могут быть нюансы). Пример: age имеет тип int. Если написать col(‘age’)< 40 оно сработает и пропихнёт его, если сделать col(‘age’)<’40’ тоже сработает. Но иногда типы явно не конвертируются друг в друга и тогда будет не оч круто(хотя спарк конечно всё равно будет пытаться конвертнуть прям на лету всеми возможными способами, всё во имя оптимизаций).

Поговорим теперь про вторую вещь, которые вы скорее всего заметили: PartitionFilters. Работает великолепно и сейчас объясню суть на примере df\_parquet. Вот что мы видим в нашем коде:

df\_parquet\_filtered = df\_parquet.filter(col('nationality') == 'Russia'). Мы фильтруем наши данные из паркета по полю национальность. Если вы зайдёте в data\_parquet, то увидите что там куча папок с названиями как раз национальностей. Это означает что перед сохранением данные были разбиты по партициям через PartitionBy(позже узнаете зачем почему и как) по полю национальность. И если обратите ещё больше внимания, то спарк всегда когда что-то пишет, он пишет ещё доп файлы, как раз указываю что где и как хранится(этакая мета-информация). Так вот, спарк даже не будет проверять строки из этих файлов, он просто глянет сразу на мета-информацию и увидит что все строки в этой партиции не подходят. Профит? Офигеть какой профит!!! Вот как это выглядит когда есть фильтрация:



Считана всего лишь одна партиция. И чтобы вы убедились что я не обманываю, вот вывод когда нет фильтра:

