**ЗАДАНИЕ**

1. Создать таблицы в форматах PARQUET/ORC/AVRO c компрессией и без оной. (Выберите один вариант, например ORC с компрессией)

2. Заполнить данными из большой таблицы hive\_db.citation\_data

3. Посмотреть на получившийся размер данных

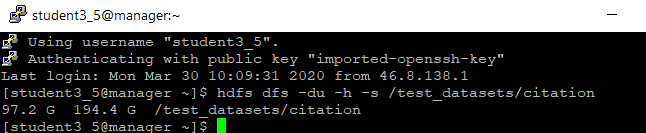
4. Сделать выводы о эффективности хранения и компрессии.

**ВЫПОЛНЕНИЕ**

Согласно заданию имеется большой табличный датасет:

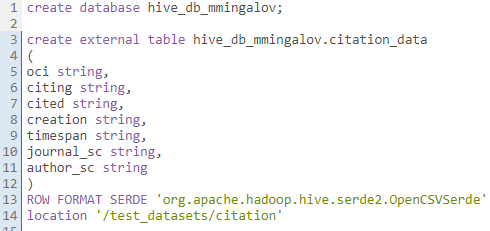
Проверим его размер **в Putty** командой

hdfs dfs -du -h -s /test\_datasets/citation

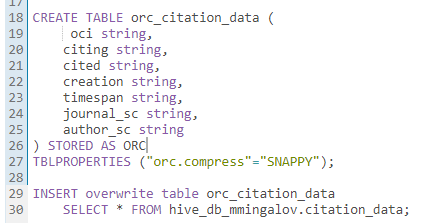


**Запустим HUE**

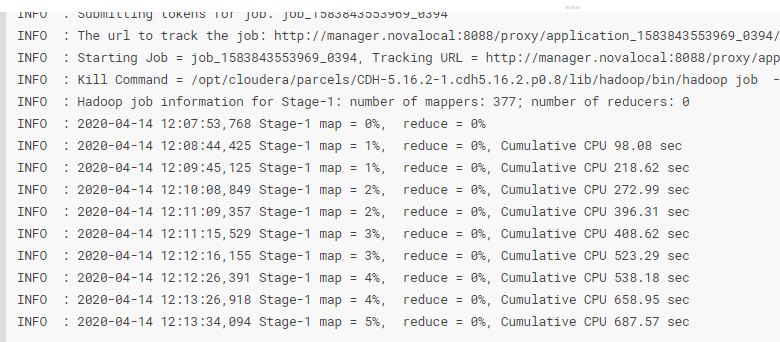
Создадим новую базу данных и таблицу в ней

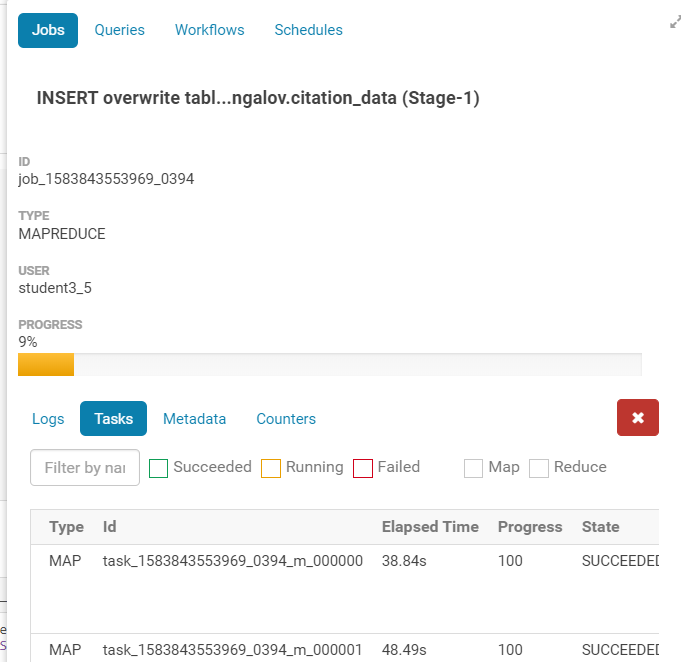


Далее создадим таблицу в формате ORC с компрессией, заполним ее данными из вышесозданной таблицы:



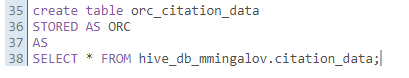
Видим, что процесс пошел:





Процесс завис спустя 2часа на 92%

Перезапустил запрос, изменив код на:



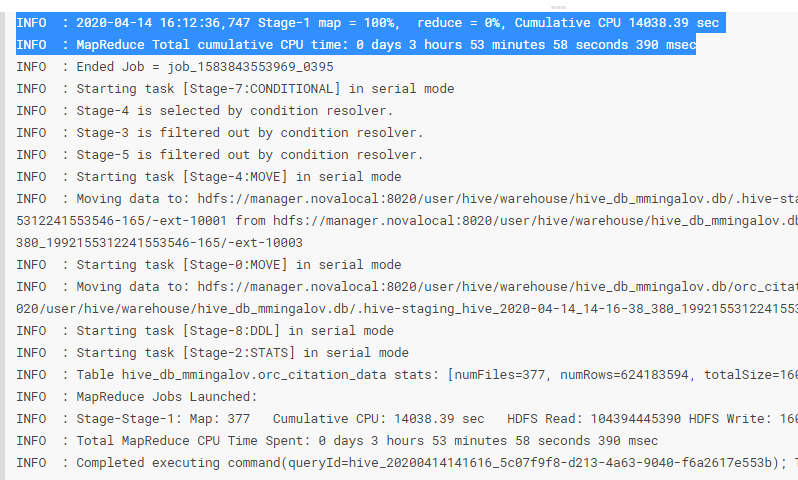
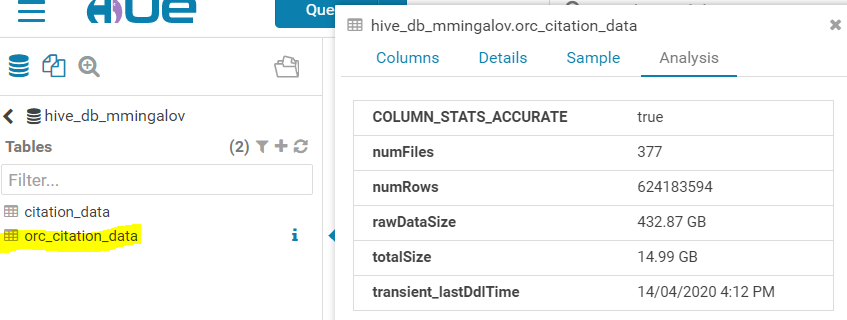
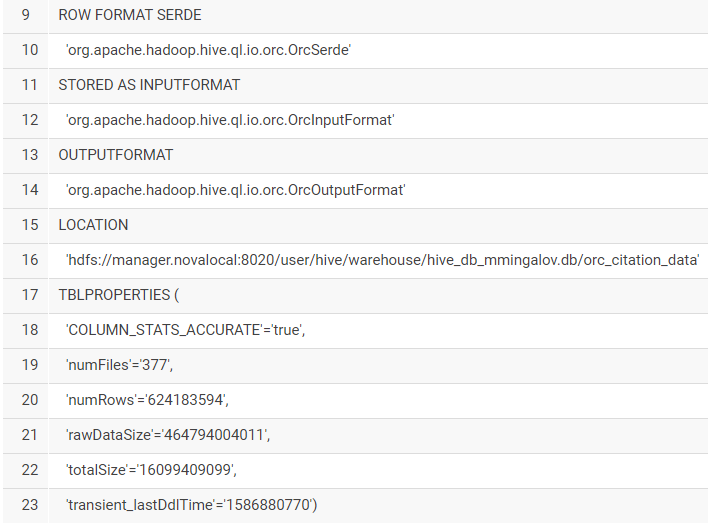


Таблица появилась:

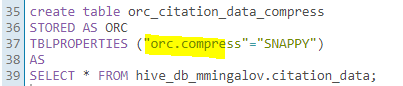


Посмотрим параметры через команду

show create table hive\_db\_mmingalov.orc\_citation\_data;



Сделаем аналогичный запрос, но со включенной компрессией у таблицы:



Запрос отработал:

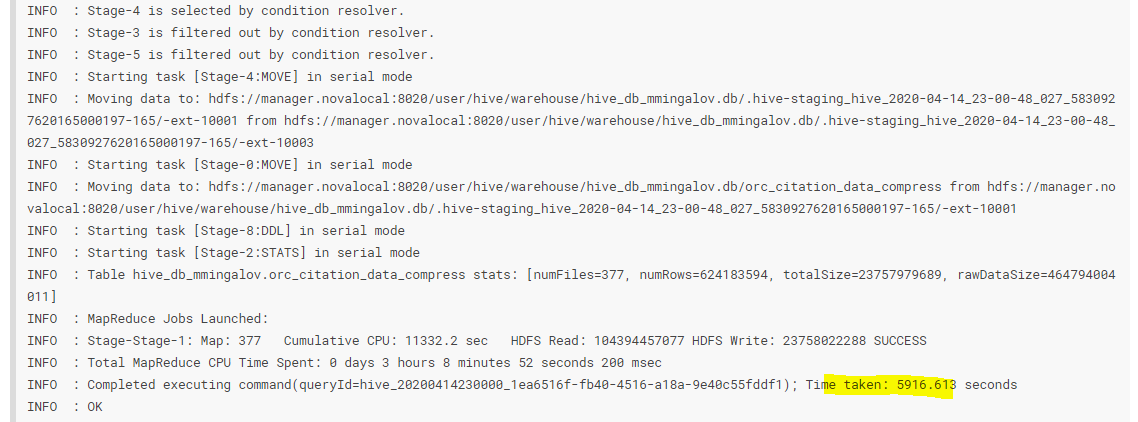
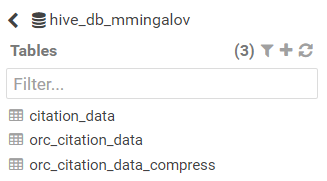
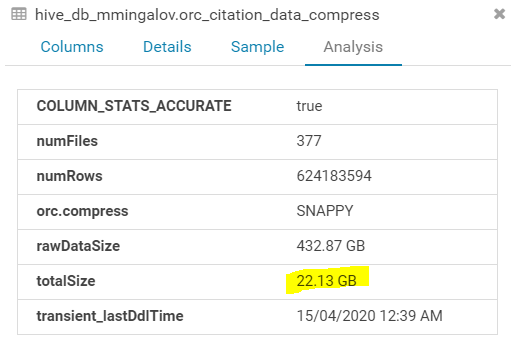


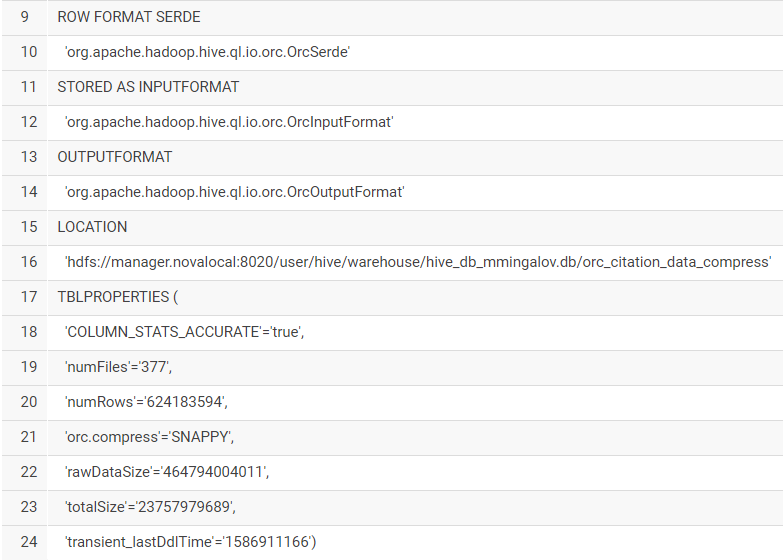
Таблица появилась в БД:





Посмотрим параметры через команду

show create table hive\_db\_mmingalov.orc\_citation\_data;



**Вывод:**

С установленным свойством компрессии созданная таблица получилась больше (22.1 Gb) чем без него (14.9 Gb) в первом случае.

Согласно документации, <https://orc.apache.org/docs/hive-config.html>, стало понятно, что метод компрессии **ZLIB** будет применяться по умолчанию, если это не указано явно.

В наших примерах мы видим, что он наиболее эффективен с точки зрения степени сжатия. Однако выполнялся он дольше (3ч 53 мин) чем SNAPPY (3ч 08 мин).

**ЗАДАНИЕ**

1. Создать отдельную БД в HIve

2. Посмотреть при помощи SQOOP содержимое БД в POsgresql

3. Импортировать в нее три любые таблицы из базы pg\_db в Postgresql используя SQOOP. Для каждой таблице используйте отдельный формат хранения -- ORC/Parquet/AVRO Рекомендую захватить таблицу sales\_large -- там порядка 10 миллионов записей, она будет достаточно репрезентативна для проверки компрессии.

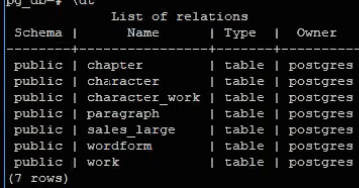
4. Найдите папки на файловой системе куда были сохранены данные. Посмотрите их размер.

5. Сделайте несколько произвольных запросов к этим таблицам.

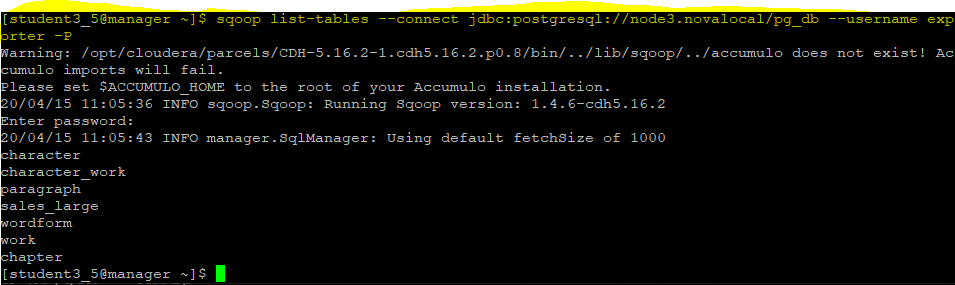
**ВЫПОЛНЕНИЕ**

У нас уже имеется база данных в HIVE с названием hive\_db\_mmingalov

БД в PostgreSQL имеет следующие таблицы:

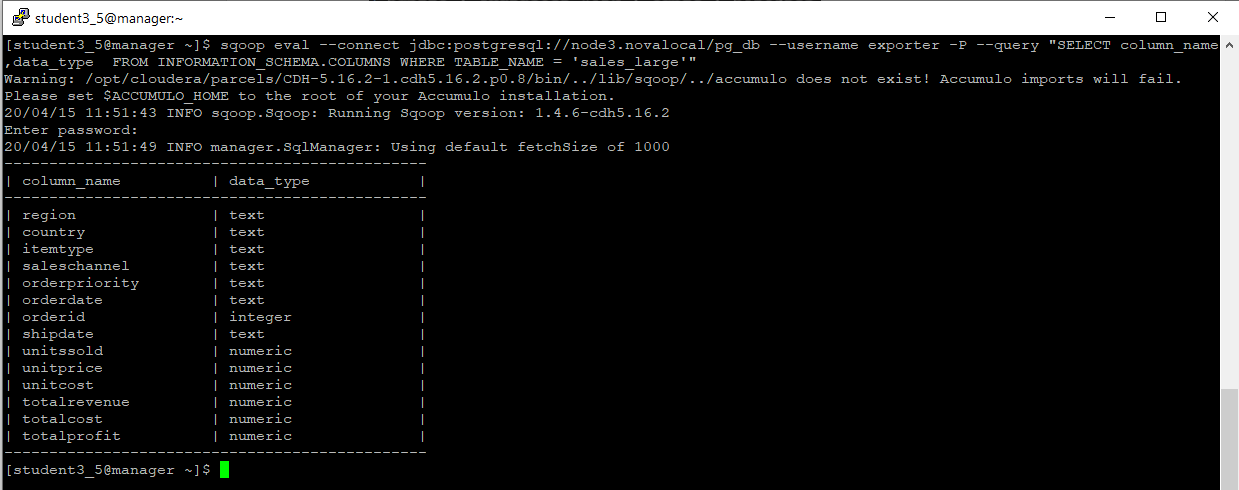


Отобразить список таблиц у БД можно командой **list-tables**:



Структуру интересной нам таблицы можно посмотреть такой командой:

sqoop eval --connect jdbc:postgresql://node3.novalocal/pg\_db --username exporter -P --query "SELECT column\_name FROM INFORMATION\_SCHEMA.COLUMNS WHERE TABLE\_NAME = ‘sales\_large’"



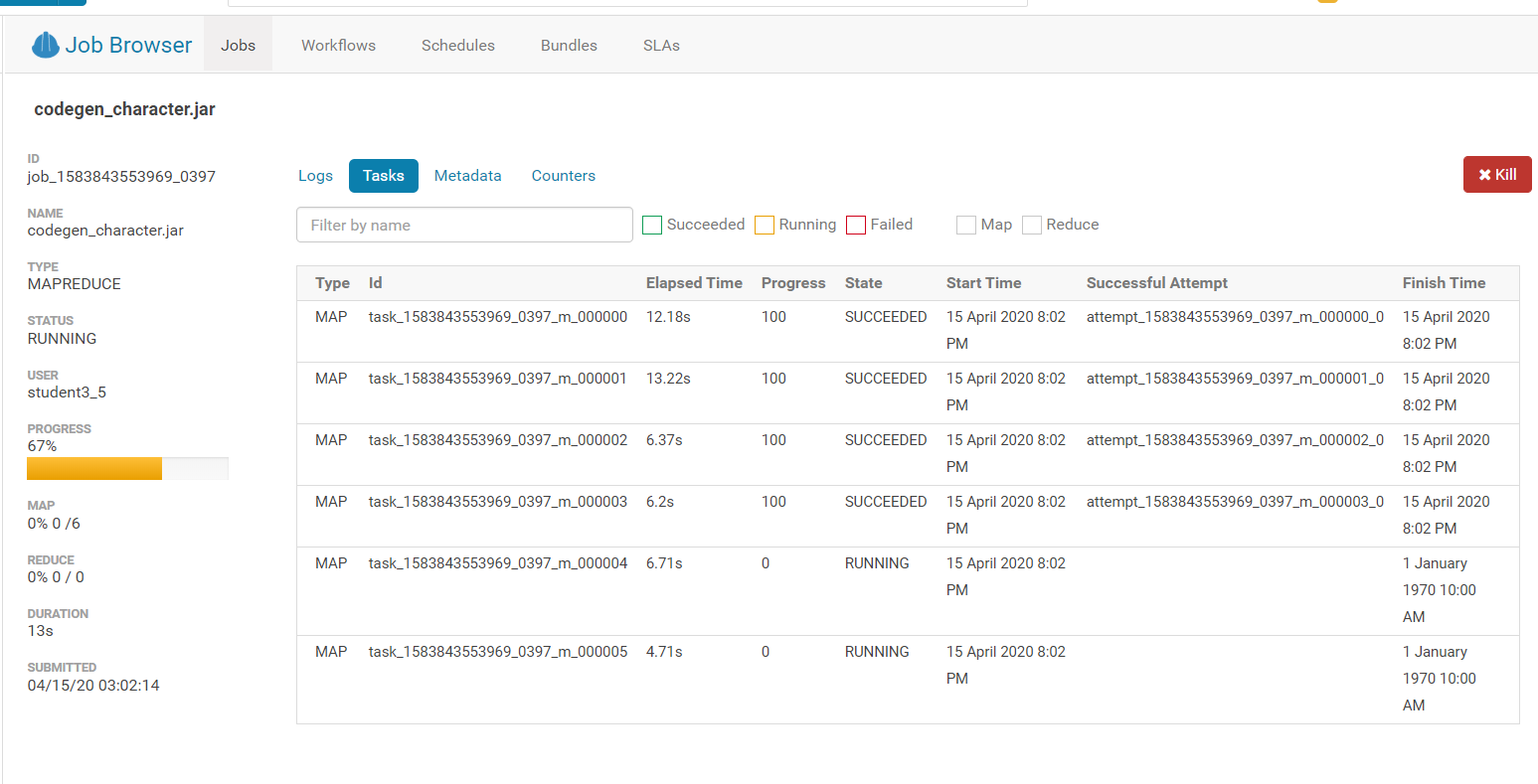
**ЗАГРУЗИМ ТАБЛИЦУ 1**

Шаблон запроса может выглядеть так (из вебинара):

sqoop import --connect jdbc:postgresql://node3.novalocal /pg\_db --username exporter -P --table character --hive-import --hive-database hi ve\_db\_mmingalov --hive-table character\_parquet --as-parquetfile

пароль: exporter\_pass

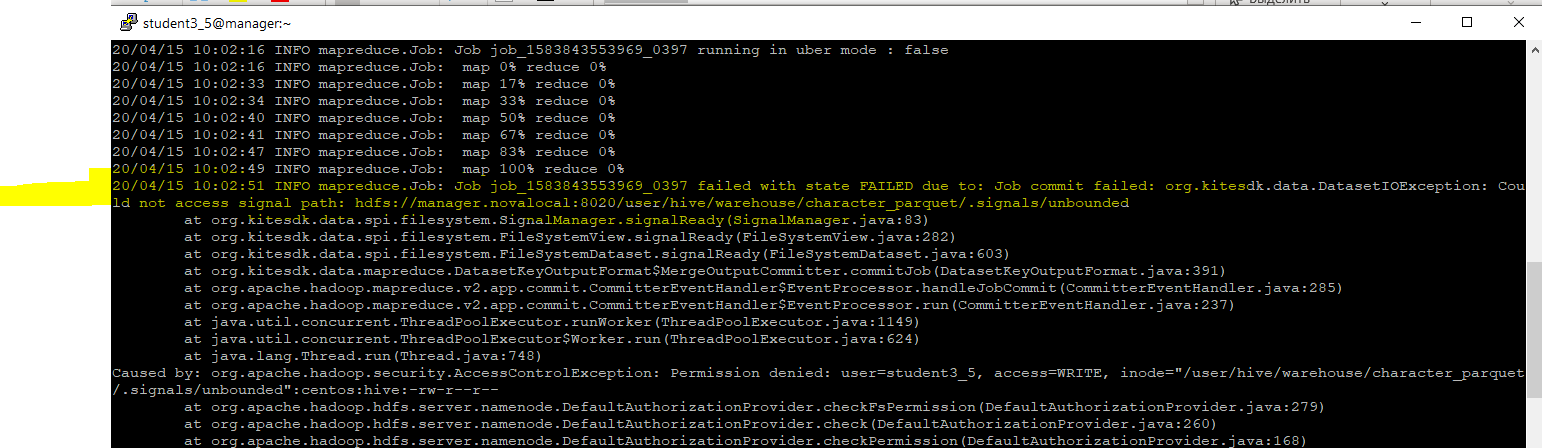
Запускаем, видим, что job работает:



Мы предположили, что таблица будет создана на стороне Hive автоматически…

Однако таблица не появилась

Причина описана на скриншоте:



Попробуем предварительно создать таблицу на стороне HIVE с форматом хранения PARQUET

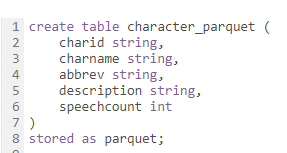
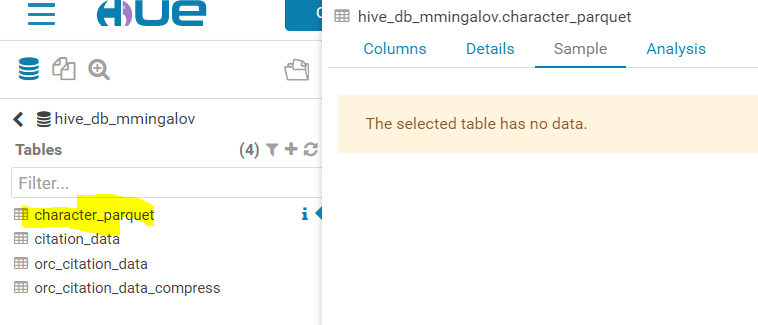


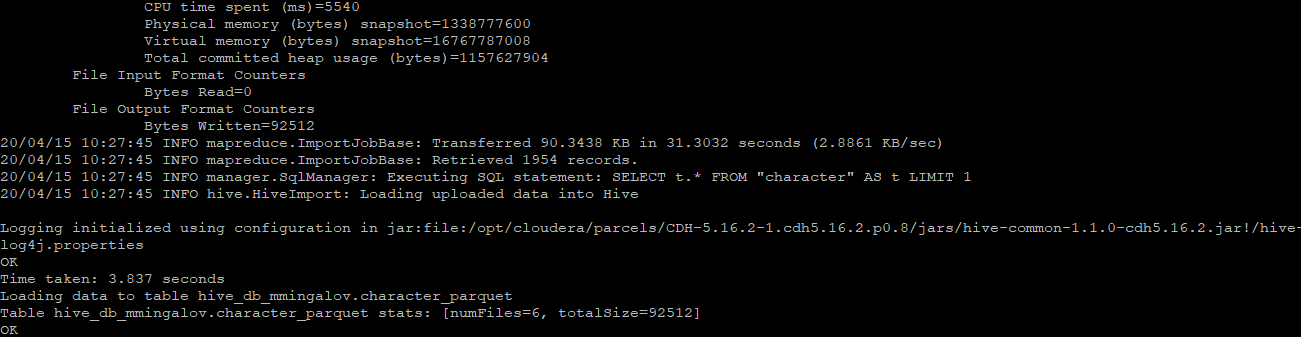
Таблица видна:



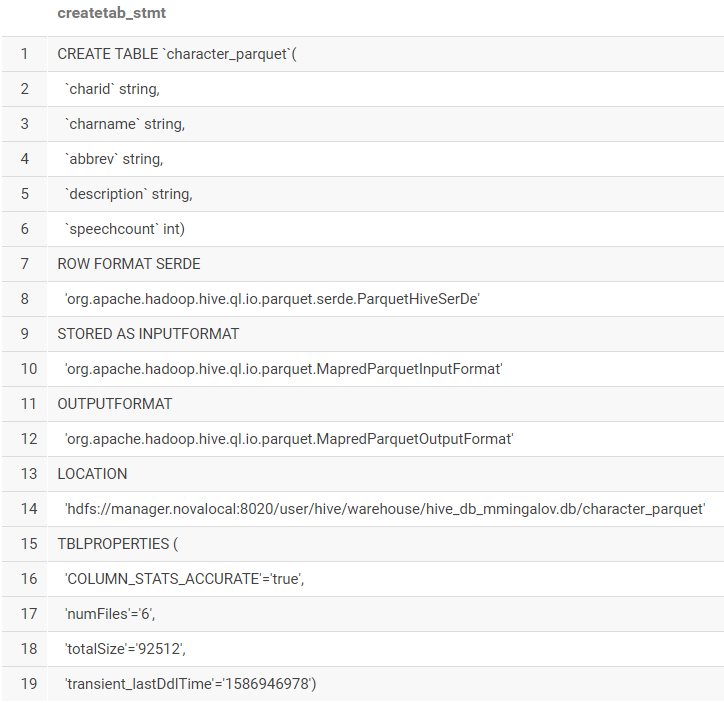
Далее повторим наш запрос в **Putty но без указания типа таблицы as parquetfile**

sqoop import --connect jdbc:postgresql://node3.novalocal/pg\_db --username exporter -P --table character --hive-import --hive-database hive\_db\_mmingalov --hive-table character\_parquet

Запрос отработал:



show create table character\_parquet;



ДАЛЕЕ

**ЗАГРУЗИМ ТАБЛИЦУ 2**

Попробуем создать еще одну таблицу на стороне HIVE с форматом хранения ORC

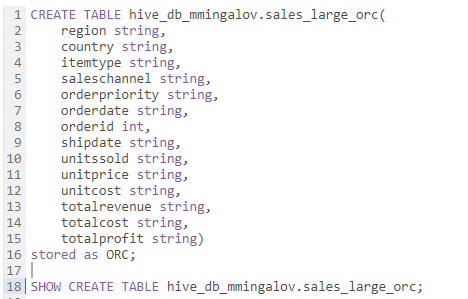


Таблица создана



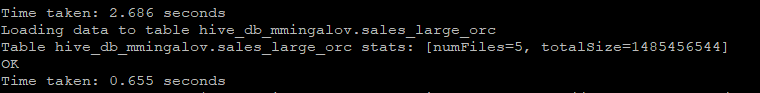
Заполним ее данными:

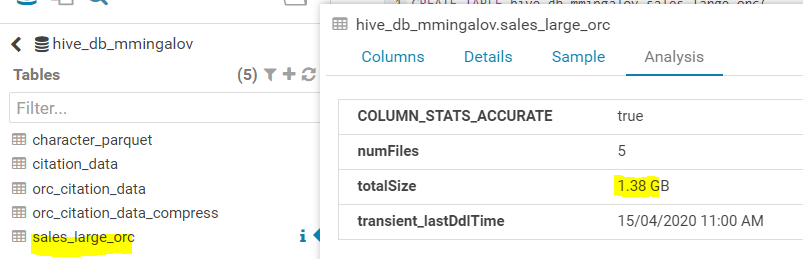
sqoop import --connect jdbc:postgresql://node3.novalocal/pg\_db --username exporter -P --table sales\_large --hive-import --hive-database hive\_db\_mmingalov --hive-table sales\_large\_orc –split-by country

**Примечание: было добавлено –split-by поскольку без него запрос не выполнился и была рекомендация использовать данный параметр**

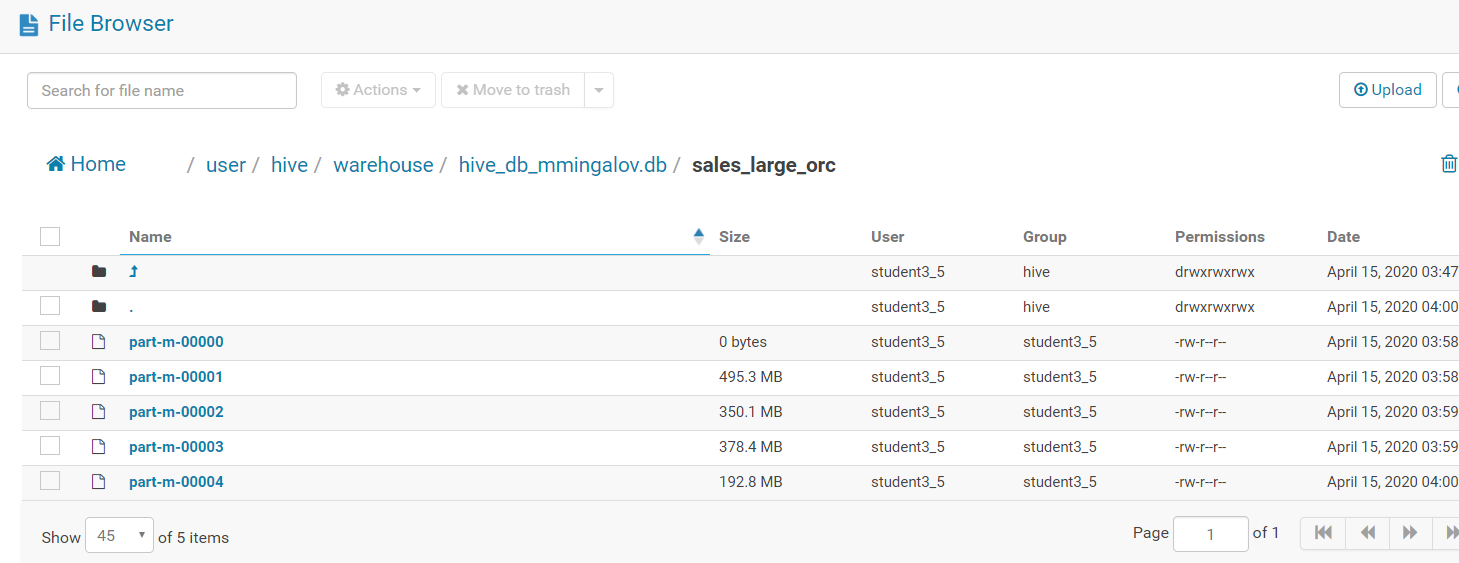
**Я выбрал поле COUNTRY**

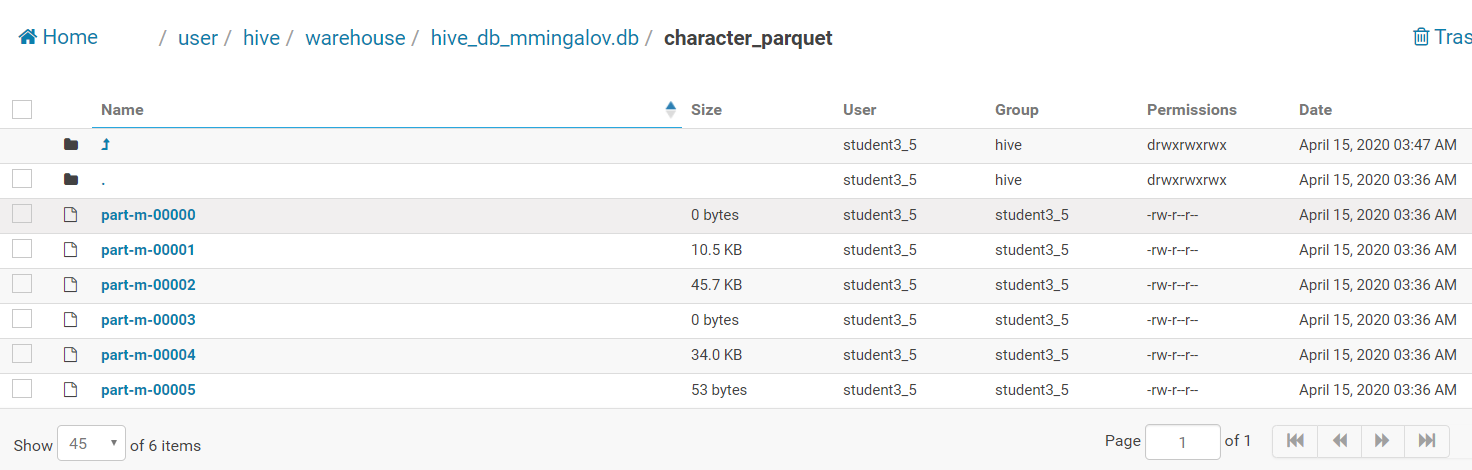
**Видим, что запрос отработал**





Найдем папки, куда были сохранены файлы:





По какой-то причине некоторые файлы имеют длину 0 байт…

**ЗАГРУЗИМ ТАБЛИЦУ 3**

Таблицу avro можно создать следующим образом:

sqoop import --connect jdbc:postgresql://node3.novalocal/pg\_db --username exporter -P --table sales\_large --hive-import --hive-database hive\_db\_mmingalov --hive-table sales\_large\_avro –split-by country

**ЗАПРОСЫ К ТАБЛИЦАМ**

SELECT count(\*) FROM character\_parquet

SELECT country, count(\*) FROM sales\_large\_orc GROUP BY country