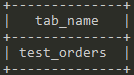
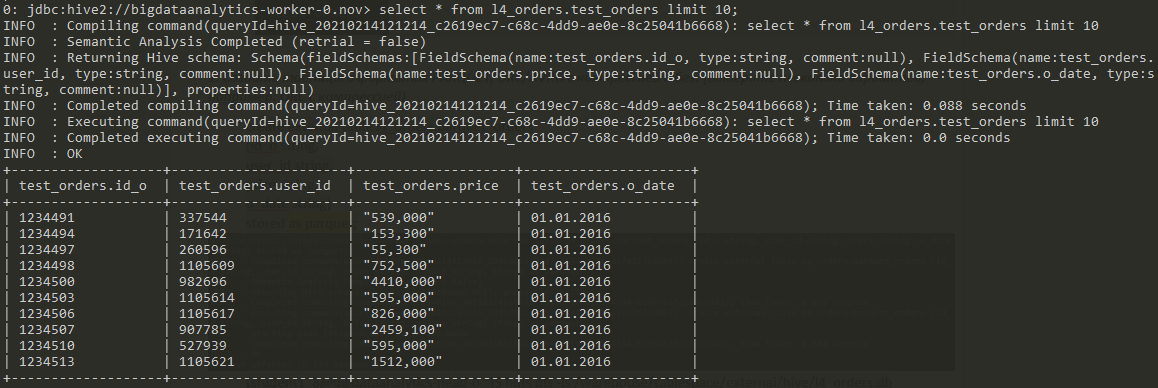
*Использую базу данных и датасет, с которыми работала на прошлом уроке*

**[student1\_8@bigdataanalytics-head-0 ~]$ hive**

**0: jdbc:hive2://bigdataanalytics-worker-0.nov> use l4\_orders;**

**0: jdbc:hive2://bigdataanalytics-worker-0.nov> show tables;**





***1. Создать таблицы в форматах PARQUET/ORC/AVRO c компрессией и без. (Выберите несколько вариантов, например: ORC с компрессией)***

**create external table l4\_orders.parquet\_orders**

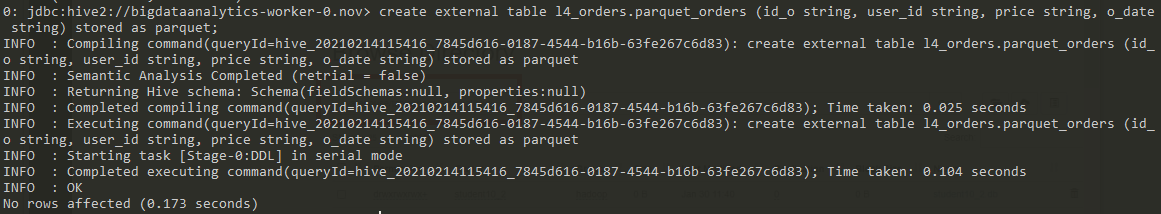
**(id\_o string,**

**user\_id string,**

**price string,**

**o\_date string)**

**stored as parquet;**



**create external table l4\_orders.avro\_orders**

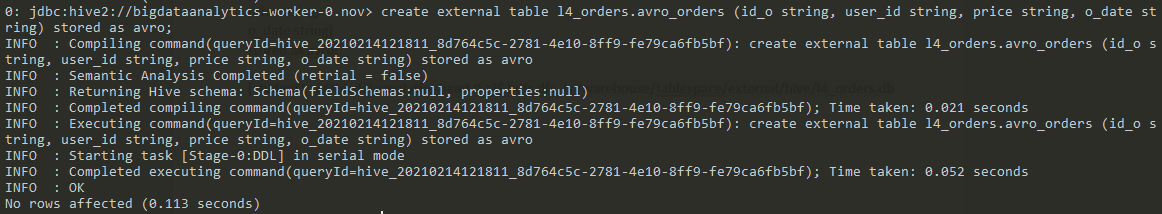
**(id\_o string,**

**user\_id string,**

**price string,**

**o\_date string)**

**stored as avro;**



**create external table l4\_orders.orc\_orders**

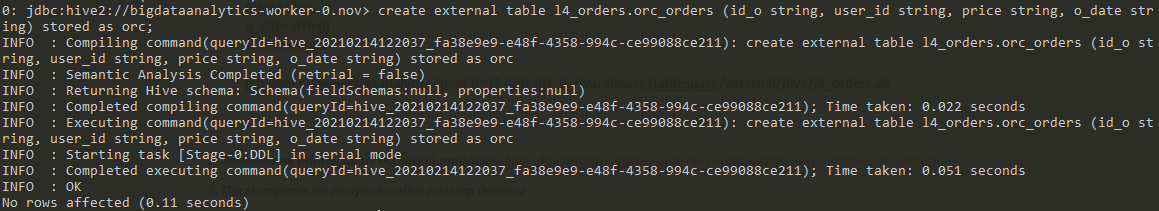
**(id\_o string,**

**user\_id string,**

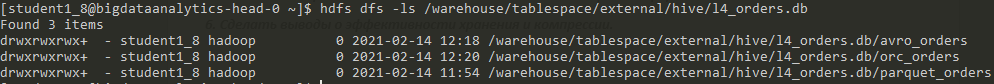
**price string,**

**o\_date string)**

**stored as orc;**



**[student1\_8@bigdataanalytics-head-0 ~]$ hdfs dfs -ls /warehouse/tablespace/external/hive/l4\_orders.db**

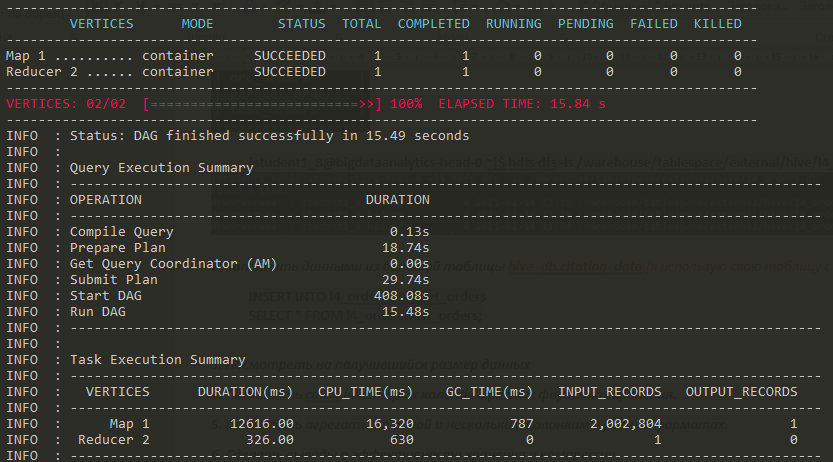


***2. Заполнить данными из большой таблицы hive\_db.citation\_data*** *(я использую свою таблицу с прошлого урока)*

**INSERT INTO l4\_orders.parquet\_orders**

**SELECT \* FROM l4\_orders.test\_orders;** *No rows affected (472.537 seconds)*

*Вставка данных из рабочей таблицы в parquet выполнялась достаточно долго, вот некоторые статистические данные по времени*



*Забыла создать таблицы со сжатием – делаю это сейчас*

**SET parquet.compression=SNAPPY;**

**create external table l4\_orders.parquet\_orders\_snappy**

**(id\_o string,**

**user\_id string,**

**price string,**

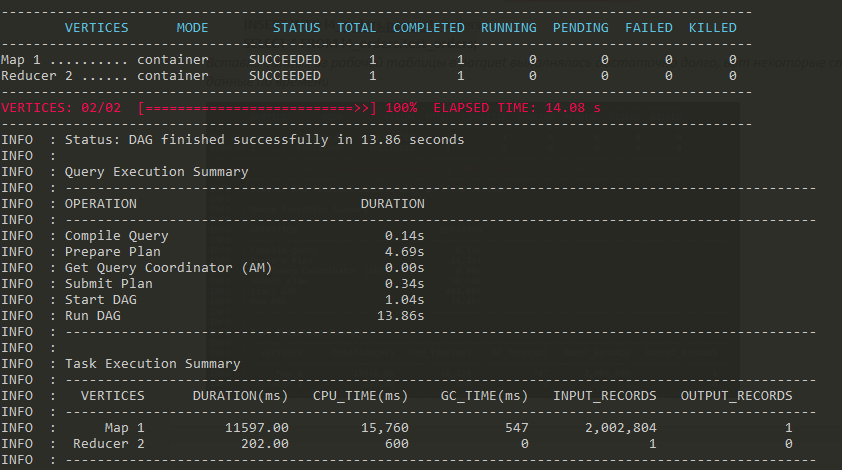
**o\_date string)**

**stored as parquet;**

**INSERT INTO l4\_orders.parquet\_orders\_snappy**

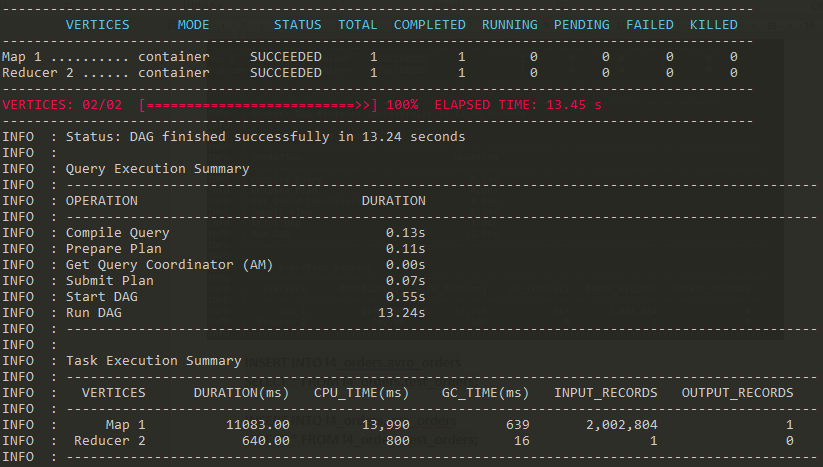
**SELECT \* FROM l4\_orders.test\_orders;** *No rows affected (20.298 seconds)*

*Вставка данных из рабочей таблицы в parquet со сжатием выполнилась намного быстрее, вот некоторые статистические данные по времени*



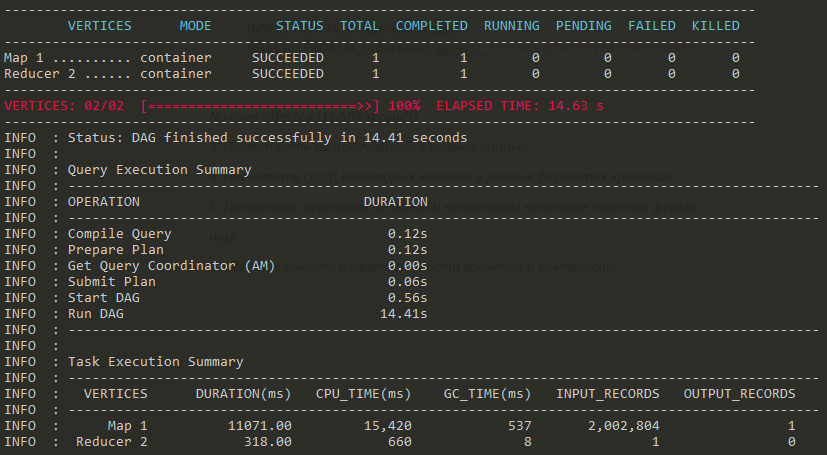
**INSERT INTO l4\_orders.avro\_orders**

**SELECT \* FROM l4\_orders.test\_orders;** *No rows affected (14.317 seconds)*



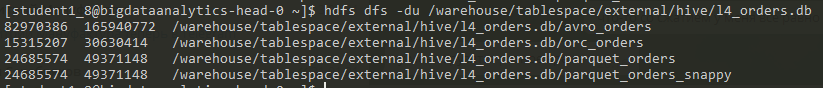
**INSERT INTO l4\_orders.** **orc\_orders**

**SELECT \* FROM l4\_orders.test\_orders;** *No rows affected (15.504 seconds)*



***3. Посмотреть на получившийся размер данных***

*Вижу, что сжатие snappy для parquet у меня почему-то не сработало*



*Проделала процедуру сжатия для avro и создала еще одну таблицу avro*

**SET avro.output.codec=snappy;**

**create external table l4\_orders.avro\_orders\_snappy**

**(id\_o string,**

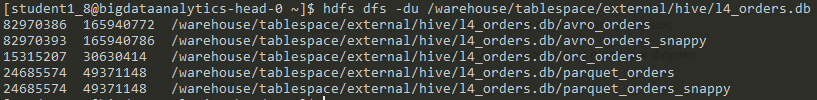
**user\_id string,**

**price string,**

**o\_date string)**

**stored as avro;**

*заполнила таблицу данными и снова такая же история, только сжатая таблица даже чуть больше*



*Я, к сожалению, не поняла как работает сжатие(((*

***4. Посчитать count некоторых колонок в разных форматах хранения.***

***5. Посчитать агрегаты по одной и нескольким колонкам в разных форматах.***

*Я решила в сравнении посчитать COUNT и SUM*

|  |  |
| --- | --- |
| **select count(id\_o) as count from avro\_orders;** | **select sum(id\_o) as total from avro\_orders;** |
| **select count(id\_o) as count from avro\_orders\_snappy;** | **select sum(id\_o) as total from avro\_orders\_snappy;** |
| **select count(id\_o) as count from orc\_orders;** | **select sum(id\_o) as total from orc\_orders;** |
| **select count(id\_o) as count from parquet\_orders;** | **select sum(id\_o) as total from parquet\_orders;** |
| **select count(id\_o) as count from parquet\_orders\_snappy;** | **select sum(id\_o) as total from parquet\_orders\_snappy;** |

***6. Сделать выводы о эффективности хранения и компрессии.***

*К сожалению, я не могу сделать выводы о компрессии, т.к. я так и не поняла, как она работает.*

*Касательно эффективности хранения, если сравнивать количество занимаемого места и скорость обработки, то выигрывает orc, а avro на последнем месте.*