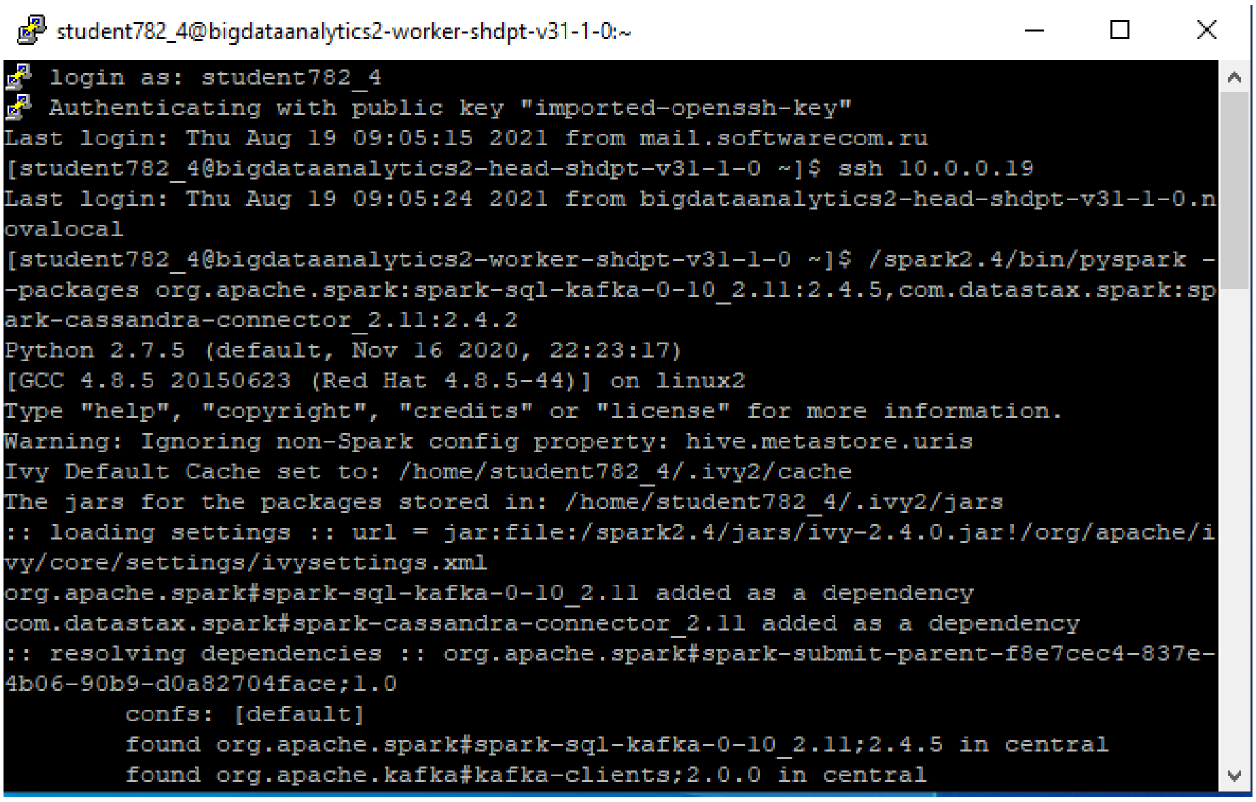
Практическое задание №6

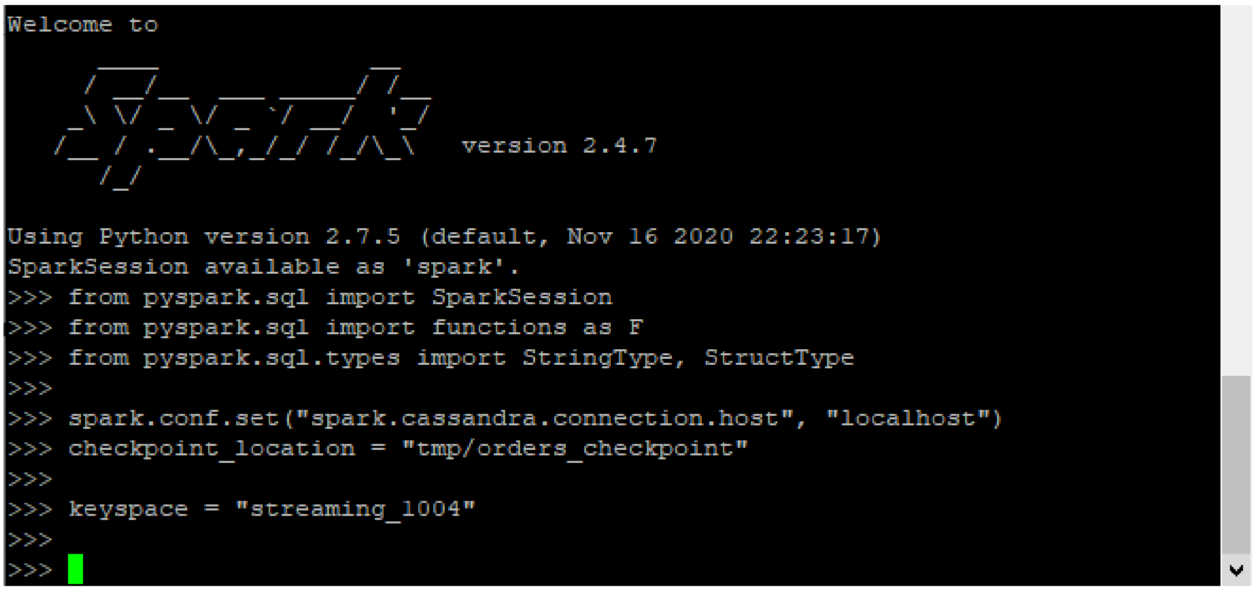
Запускаем spark, добавляем пакет cassandra-connector



student782_4@bigdataanalytics2-worker-shdpt-v31-1-0:~

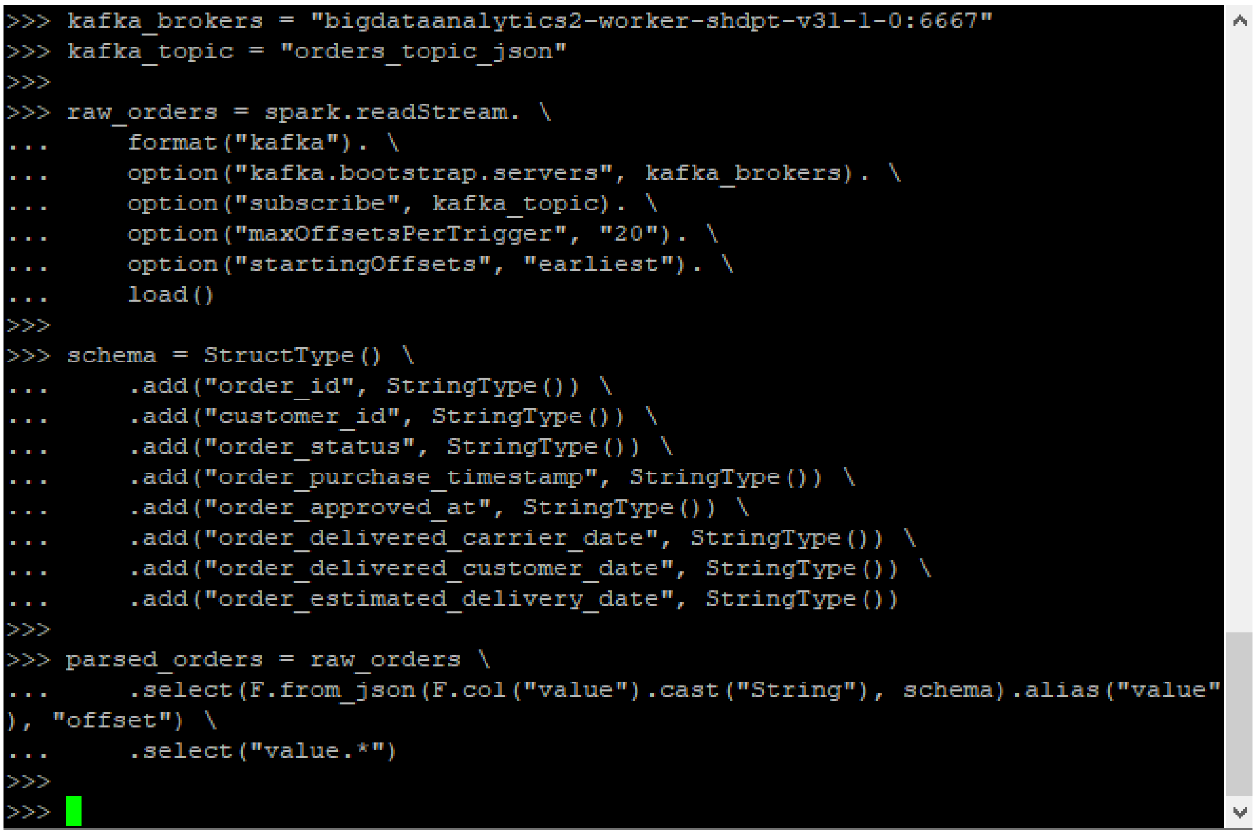
```
login as: student782_4
Authenticating with public key "imported-openssh-key"
Last login: Thu Aug 19 09:05:15 2021 from mail.softwarecom.ru
[student782_4@bigdataanalytics2-head-shdpt-v31-1-0 ~]$ ssh 10.0.0.19
Last login: Thu Aug 19 09:05:24 2021 from bigdataanalytics2-head-shdpt-v31-1-0.n
ovalocal
[student782_4@bigdataanalytics2-worker-shdpt-v31-1-0 ~]$ /spark2.4/bin/pyspark -
-packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5,com.datastax.spark:sp
ark-cassandra-connector_2.11:2.4.2
Python 2.7.5 (default, Nov 16 2020, 22:23:17)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-44)] on linux2
Type "help", "copyright", "credits" or "license" for more information.
Warning: Ignoring non-Spark config property: hive.metastore.uris
Ivy Default Cache set to: /home/student782_4/.ivy2/cache
The jars for the packages stored in: /home/student782_4/.ivy2/jars
:: loading settings :: url = jar:file:/spark2.4/jars/ivy-2.4.0.jar!/org/apache/i
vy/core/settings/ivysettings.xml
org.apache.spark#spark-sql-kafka-0-10_2.11 added as a dependency
com.datastax.spark#spark-cassandra-connector_2.11 added as a dependency
:: resolving dependencies :: org.apache.spark#spark-submit-parent-f8e7cec4-837e-
4b06-90b9-d0a82704face;1.0
        confs: [default]
        found org.apache.spark#spark-sql-kafka-0-10_2.11;2.4.5 in central
        found org.apache.kafka#kafka-clients;2.0.0 in central
```

Создадим конфигурацию spark

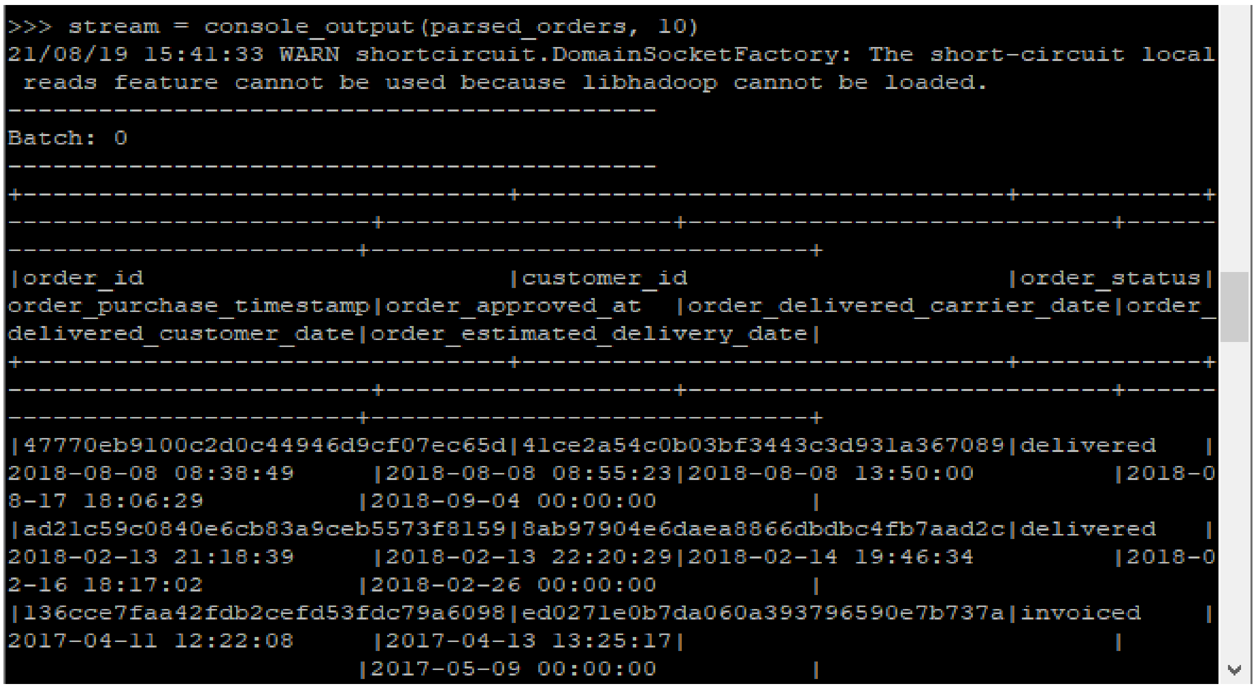


```
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /__ / .__/\_,_/_/ /_/\_\   version 2.4.7
      /_/

Using Python version 2.7.5 (default, Nov 16 2020 22:23:17)
SparkSession available as 'spark'.
>>> from pyspark.sql import SparkSession
>>> from pyspark.sql import functions as F
>>> from pyspark.sql.types import StringType, StructType
>>>
>>> spark.conf.set("spark.cassandra.connection.host", "localhost")
>>> checkpoint_location = "tmp/orders_checkpoint"
>>>
>>> keyspace = "streaming_1004"
>>>
>>>
```

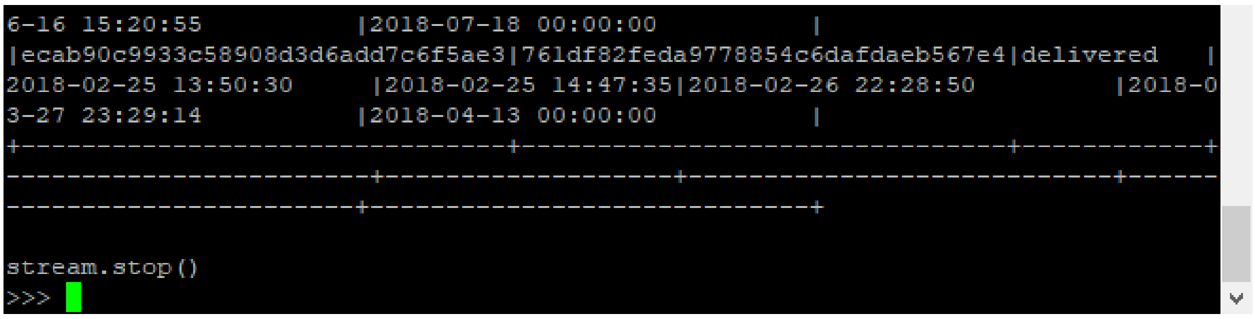
Вывод на консоль, создаем поток



```
>>> kafka_brokers = "bigdataanalytics2-worker-shdpt-v31-1-0:6667"
>>> kafka_topic = "orders_topic_json"
>>>
>>> raw_orders = spark.readStream. \
...     format("kafka"). \
...     option("kafka.bootstrap.servers", kafka_brokers). \
...     option("subscribe", kafka_topic). \
...     option("maxOffsetsPerTrigger", "20"). \
...     option("startingOffsets", "earliest"). \
...     load()
>>>
>>> schema = StructType() \
...     .add("order_id", StringType()) \
...     .add("customer_id", StringType()) \
...     .add("order_status", StringType()) \
...     .add("order_purchase_timestamp", StringType()) \
...     .add("order_approved_at", StringType()) \
...     .add("order_delivered_carrier_date", StringType()) \
...     .add("order_delivered_customer_date", StringType()) \
...     .add("order_estimated_delivery_date", StringType())
>>>
>>> parsed_orders = raw_orders \
...     .select(F.from_json(F.col("value").cast("String"), schema).alias("value"
), "offset") \
...     .select("value.*")
>>>
>>>
```

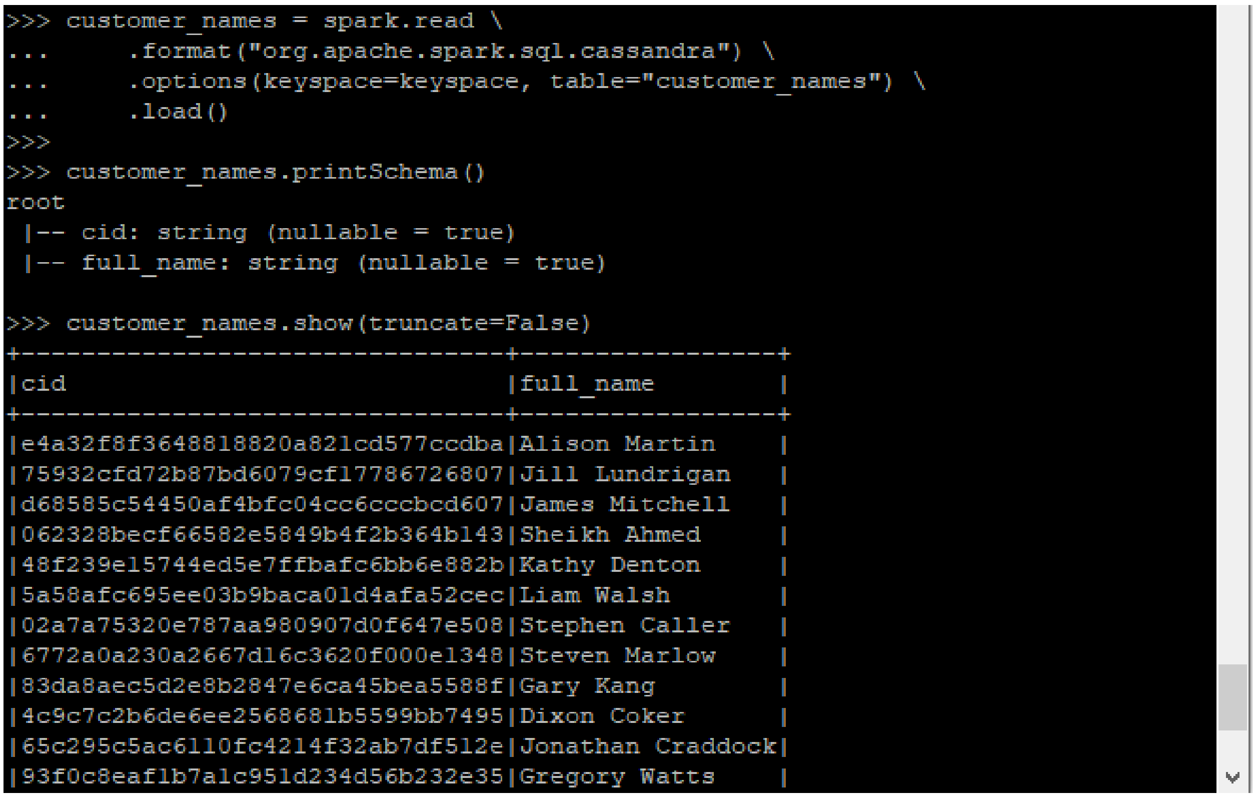


```
>>> stream = console_output(parsed_orders, 10)
21/08/19 15:41:33 WARN shortcircuit.DomainSocketFactory: The short-circuit local
 reads feature cannot be used because libhadoop cannot be loaded.
-------------------------------------------
Batch: 0
-------------------------------------------
+--------------------------------+--------------------------------+------------+
------------------------+-------------------+----------------------------+------
-----------------------+-----------------------------+
|order_id                        |customer_id                     |order_status|
order_purchase_timestamp|order_approved_at  |order_delivered_carrier_date|order_
delivered_customer_date|order_estimated_delivery_date|
+--------------------------------+--------------------------------+------------+
------------------------+-------------------+----------------------------+------
-----------------------+-----------------------------+
|47770eb9100c2d0c44946d9cf07ec65d|41ce2a54c0b03bf3443c3d931a367089|delivered   |
2018-08-08 08:38:49     |2018-08-08 08:55:23|2018-08-08 13:50:00         |2018-0
8-17 18:06:29          |2018-09-04 00:00:00          |
|ad21c59c0840e6cb83a9ceb5573f8159|8ab97904e6daea8866dbdbc4fb7aad2c|delivered   |
2018-02-13 21:18:39     |2018-02-13 22:20:29|2018-02-14 19:46:34         |2018-0
2-16 18:17:02          |2018-02-26 00:00:00          |
|136cce7faa42fdb2cefd53fdc79a6098|ed0271e0b7da060a393796590e7b737a|invoiced    |
2017-04-11 12:22:08     |2017-04-13 13:25:17|                            |
                       |2017-05-09 00:00:00          |
```



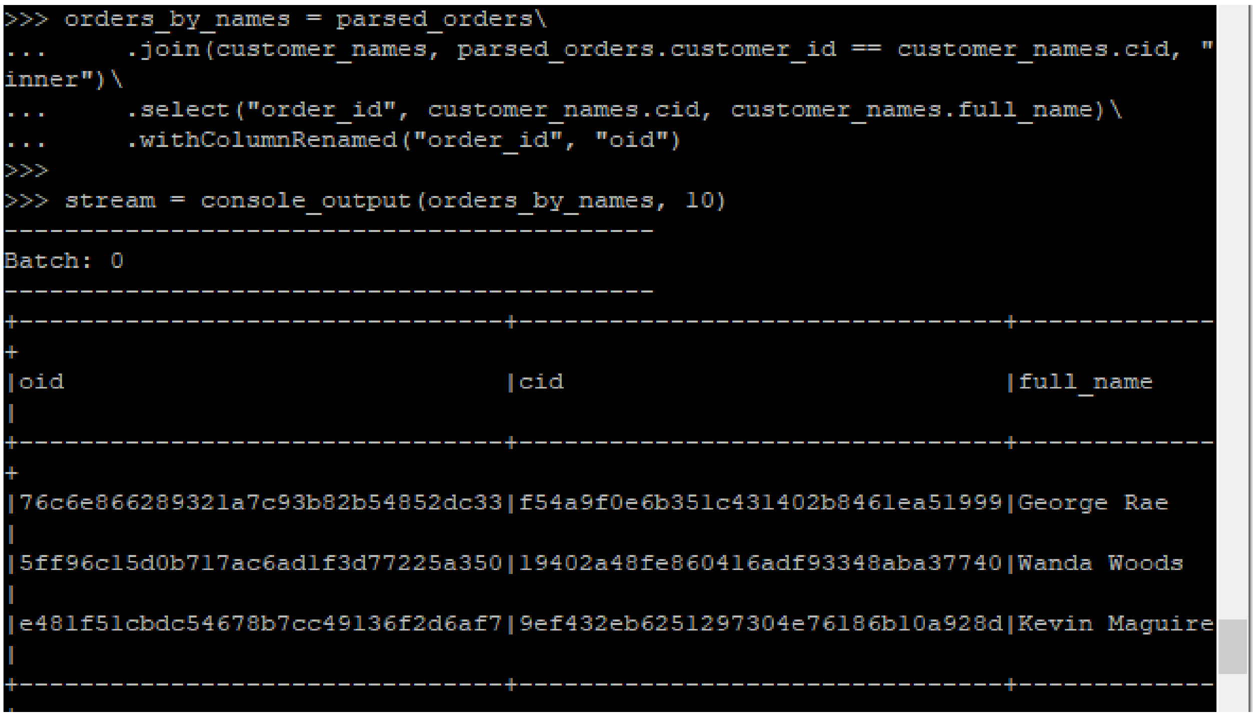
```
6-16 15:20:55          |2018-07-18 00:00:00          |
|ecab90c9933c58908d3d6add7c6f5ae3|761df82feda9778854c6dafdaeb567e4|delivered   |
2018-02-25 13:50:30     |2018-02-25 14:47:35|2018-02-26 22:28:50         |2018-0
3-27 23:29:14          |2018-04-13 00:00:00          |
+--------------------------------+--------------------------------+------------+
------------------------+-------------------+----------------------------+------
-----------------------+-----------------------------+

stream.stop()
>>>
```

Читаем из Cassandra, выводим схему и таблицу



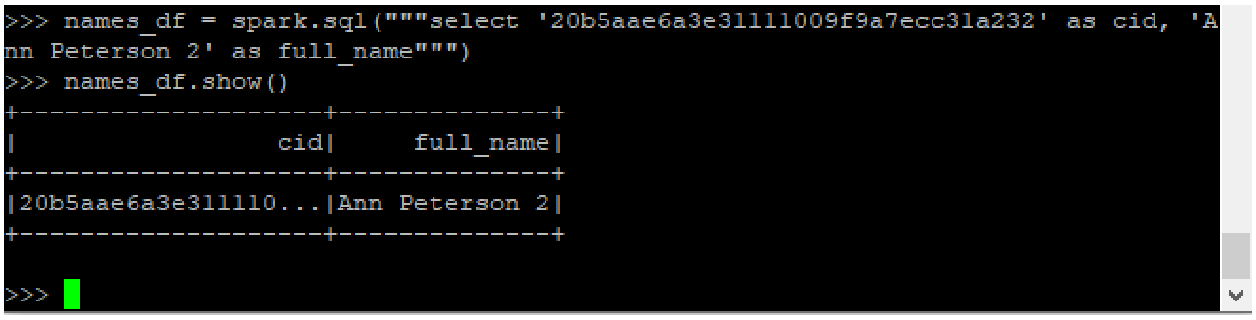
```
>>> customer_names = spark.read \
...     .format("org.apache.spark.sql.cassandra") \
...     .options(keyspace=keyspace, table="customer_names") \
...     .load()
>>>
>>> customer_names.printSchema()
root
 |-- cid: string (nullable = true)
 |-- full_name: string (nullable = true)

>>> customer_names.show(truncate=False)
+--------------------------------+-----------------+
|cid                             |full_name        |
+--------------------------------+-----------------+
|e4a32f8f3648818820a821cd577ccdba|Alison Martin    |
|75932cfd72b87bd6079cf17786726807|Jill Lundrigan   |
|d68585c54450af4bfc04cc6cccbcd607|James Mitchell   |
|062328becf66582e5849b4f2b364b143|Sheikh Ahmed     |
|48f239e15744ed5e7ffbafc6bb6e882b|Kathy Denton     |
|5a58afc695ee03b9baca01d4afa52cec|Liam Walsh       |
|02a7a75320e787aa980907d0f647e508|Stephen Caller   |
|6772a0a230a2667d16c3620f000e1348|Steven Marlow    |
|83da8aec5d2e8b2847e6ca45bea5588f|Gary Kang        |
|4c9c7c2b6de6ee2568681b5599bb7495|Dixon Coker      |
|65c295c5ac6110fc4214f32ab7df512e|Jonathan Craddock|
|93f0c8eaf1b7a1c951d234d56b232e35|Gregory Watts    |
```

Связываем таблицы и выводим



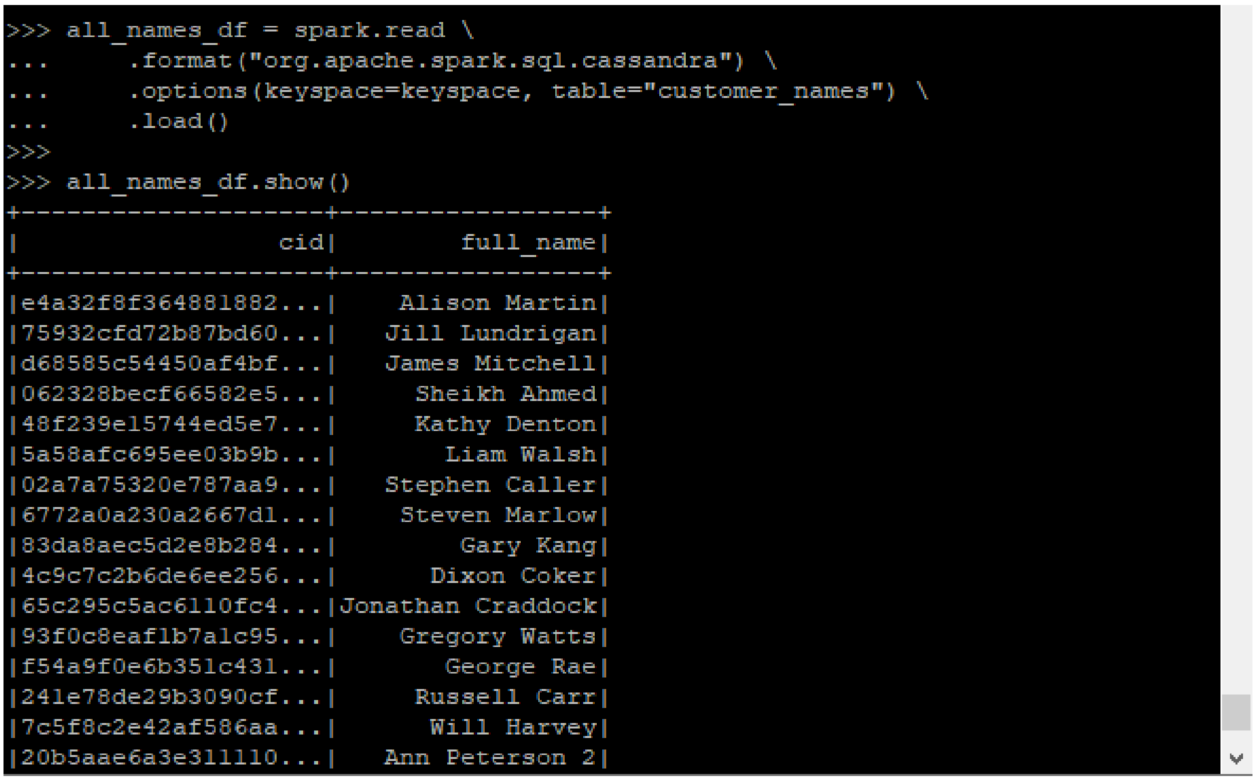
```
>>> orders_by_names = parsed_orders\
...     .join(customer_names, parsed_orders.customer_id == customer_names.cid, "
inner")\
...     .select("order_id", customer_names.cid, customer_names.full_name)\
...     .withColumnRenamed("order_id", "oid")
>>>
>>> stream = console_output(orders_by_names, 10)
-------------------------------------------
Batch: 0
-------------------------------------------
+--------------------------------+--------------------------------+-------------
+
|oid                             |cid                             |full_name
|
+--------------------------------+--------------------------------+-------------
+
|76c6e866289321a7c93b82b54852dc33|f54a9f0e6b351c431402b8461ea51999|George Rae
|
|5ff96c15d0b717ac6ad1f3d77225a350|19402a48fe860416adf93348aba37740|Wanda Woods
|
|e481f51cbdc54678b7cc49136f2d6af7|9ef432eb6251297304e76186b10a928d|Kevin Maguire
|
+--------------------------------+--------------------------------+-------------
```

Делаем выборку



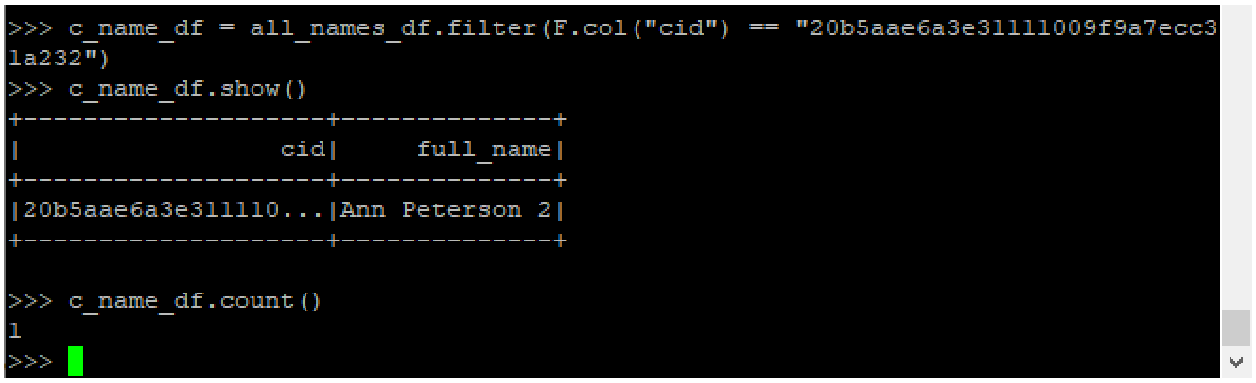
```
>>> names_df = spark.sql("""select '20b5aae6a3e31111009f9a7ecc31a232' as cid, 'A
nn Peterson 2' as full_name""")
>>> names_df.show()
+--------------------+--------------+
|                 cid|     full_name|
+--------------------+--------------+
|20b5aae6a3e311110...|Ann Peterson 2|
+--------------------+--------------+

>>> 
```

Выводим таблицу



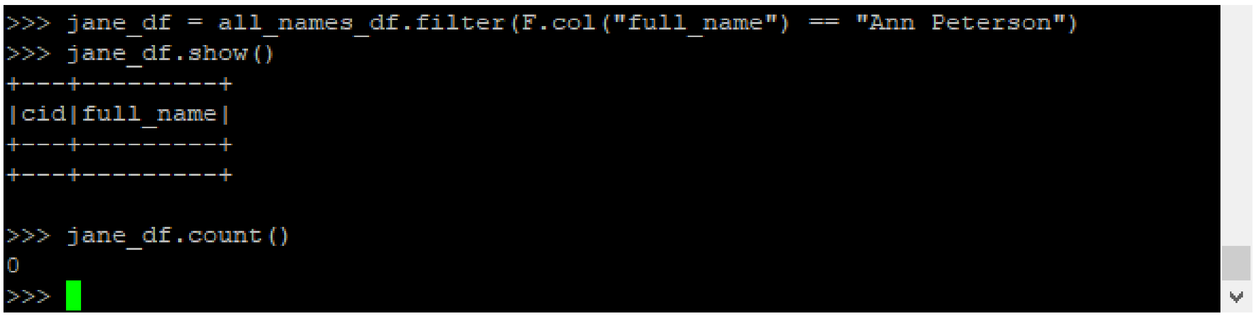
```
>>> all_names_df = spark.read \
...     .format("org.apache.spark.sql.cassandra") \
...     .options(keyspace=keyspace, table="customer_names") \
...     .load()
>>>
>>> all_names_df.show()
+--------------------+------------------+
|                 cid|         full_name|
+--------------------+------------------+
|e4a32f8f364881882...|     Alison Martin|
|75932cfd72b87bd60...|    Jill Lundrigan|
|d68585c54450af4bf...|    James Mitchell|
|062328becf66582e5...|      Sheikh Ahmed|
|48f239e15744ed5e7...|      Kathy Denton|
|5a58afc695ee03b9b...|        Liam Walsh|
|02a7a75320e787aa9...|    Stephen Caller|
|6772a0a230a2667d1...|     Steven Marlow|
|83da8aec5d2e8b284...|         Gary Kang|
|4c9c7c2b6de6ee256...|       Dixon Coker|
|65c295c5ac6110fc4...|Jonathan Craddock|
|93f0c8eaf1b7a1c95...|     Gregory Watts|
|f54a9f0e6b351c431...|        George Rae|
|241e78de29b3090cf...|      Russell Carr|
|7c5f8c2e42af586aa...|       Will Harvey|
|20b5aae6a3e311110...|    Ann Peterson 2|
```

Фильтруем по полю cid и подсчитываем количество



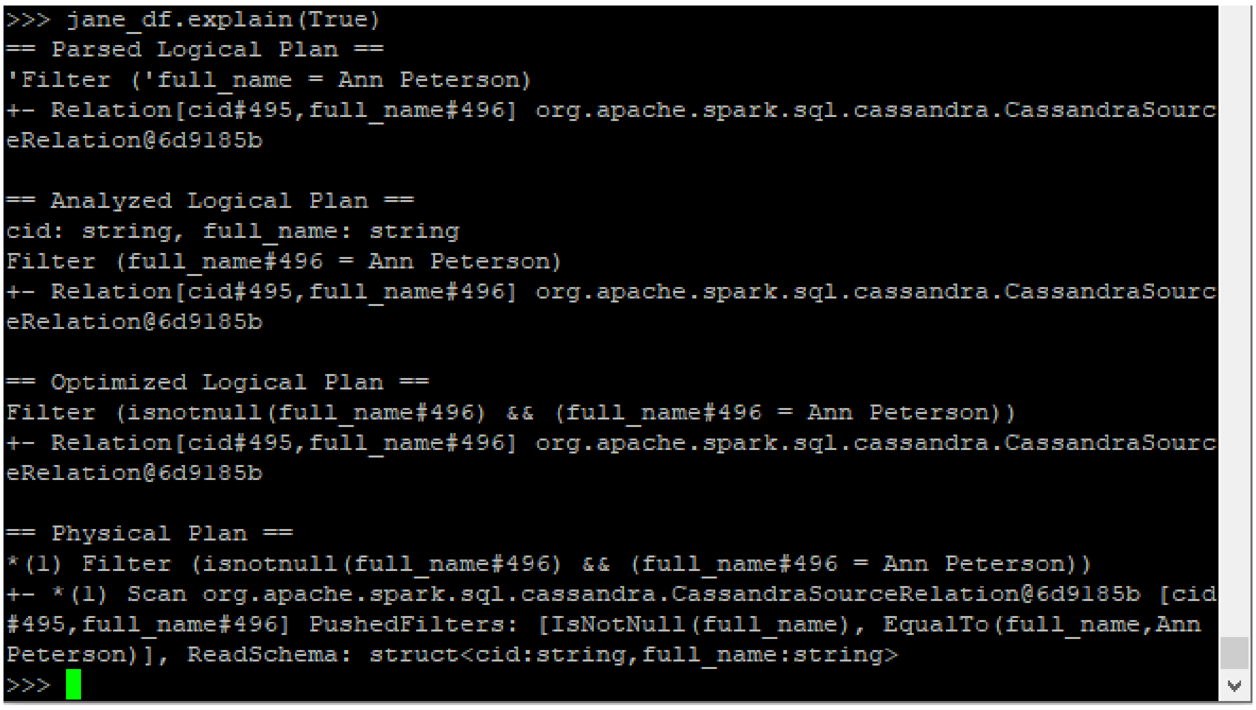
```
>>> c_name_df = all_names_df.filter(F.col("cid") == "20b5aae6a3e31111009f9a7ecc3
1a232")
>>> c_name_df.show()
+--------------------+--------------+
|                 cid|     full_name|
+--------------------+--------------+
|20b5aae6a3e311110...|Ann Peterson 2|
+--------------------+--------------+

>>> c_name_df.count()
1
>>> 
```

Фильтруем по полю full_name и подсчитываем количество



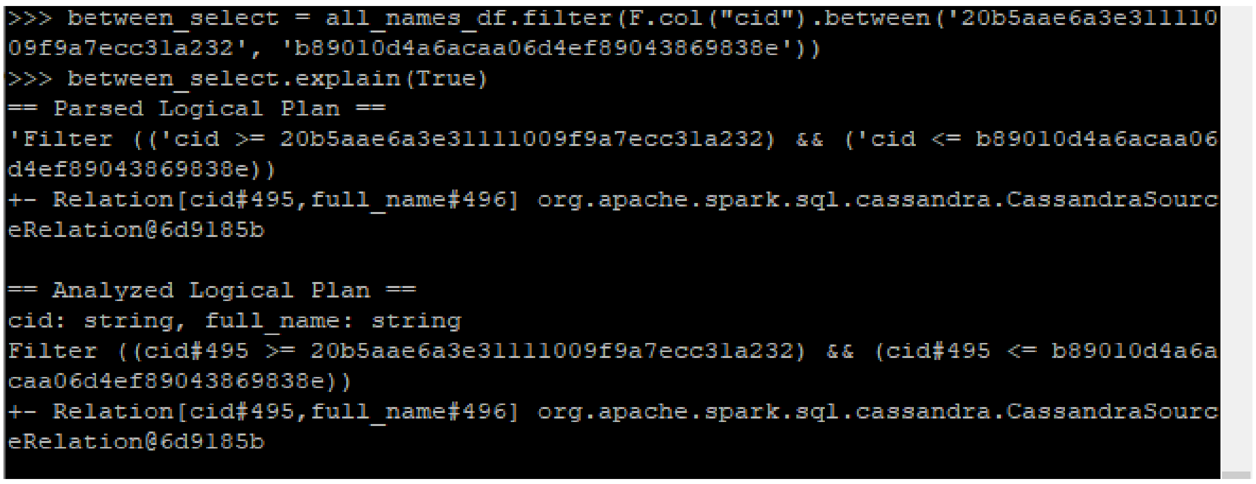
```
>>> jane_df = all_names_df.filter(F.col("full_name") == "Ann Peterson")
>>> jane_df.show()
+---+---------+
|cid|full_name|
+---+---------+
+---+---------+

>>> jane_df.count()
0
>>>
```

Просматриваем план запроса



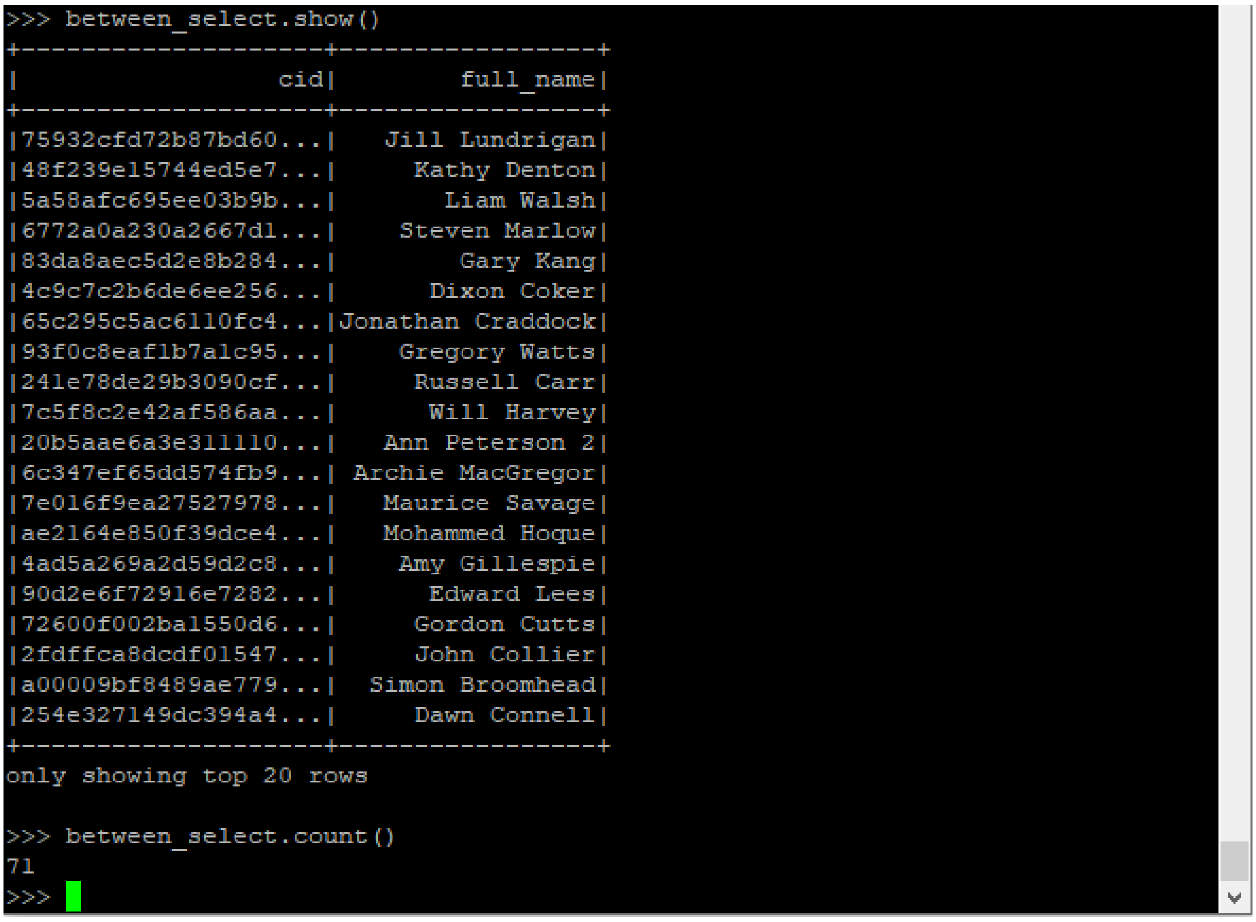
```
>>> jane_df.explain(True)
== Parsed Logical Plan ==
'Filter ('full_name = Ann Peterson)
+- Relation[cid#495,full_name#496] org.apache.spark.sql.cassandra.CassandraSourc
eRelation@6d9185b

== Analyzed Logical Plan ==
cid: string, full_name: string
Filter (full_name#496 = Ann Peterson)
+- Relation[cid#495,full_name#496] org.apache.spark.sql.cassandra.CassandraSourc
eRelation@6d9185b

== Optimized Logical Plan ==
Filter (isnotnull(full_name#496) && (full_name#496 = Ann Peterson))
+- Relation[cid#495,full_name#496] org.apache.spark.sql.cassandra.CassandraSourc
eRelation@6d9185b

== Physical Plan ==
*(1) Filter (isnotnull(full_name#496) && (full_name#496 = Ann Peterson))
+- *(1) Scan org.apache.spark.sql.cassandra.CassandraSourceRelation@6d9185b [cid
#495,full_name#496] PushedFilters: [IsNotNull(full_name), EqualTo(full_name,Ann
Peterson)], ReadSchema: struct<cid:string,full_name:string>
>>>
```

Задаем фильтр для набора данных и просматриваем план запроса



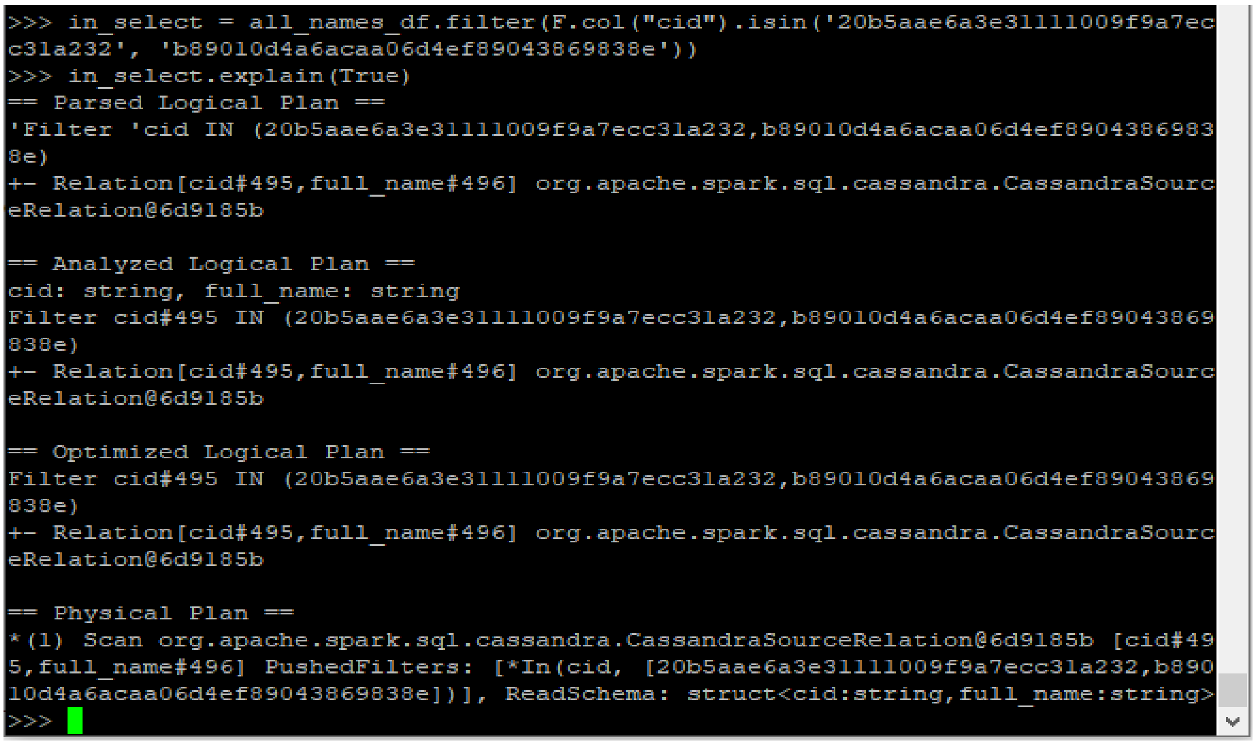
```
>>> between_select = all_names_df.filter(F.col("cid").between('20b5aae6a3e311110
09f9a7ecc31a232', 'b89010d4a6acaa06d4ef89043869838e'))
>>> between_select.explain(True)
== Parsed Logical Plan ==
'Filter (('cid >= 20b5aae6a3e31111009f9a7ecc31a232) && ('cid <= b89010d4a6acaa06
d4ef89043869838e))
+- Relation[cid#495,full_name#496] org.apache.spark.sql.cassandra.CassandraSourc
eRelation@6d9185b

== Analyzed Logical Plan ==
cid: string, full_name: string
Filter ((cid#495 >= 20b5aae6a3e31111009f9a7ecc31a232) && (cid#495 <= b89010d4a6a
caa06d4ef89043869838e))
+- Relation[cid#495,full_name#496] org.apache.spark.sql.cassandra.CassandraSourc
eRelation@6d9185b
```

Выводим набор данных и подсчитываем количество



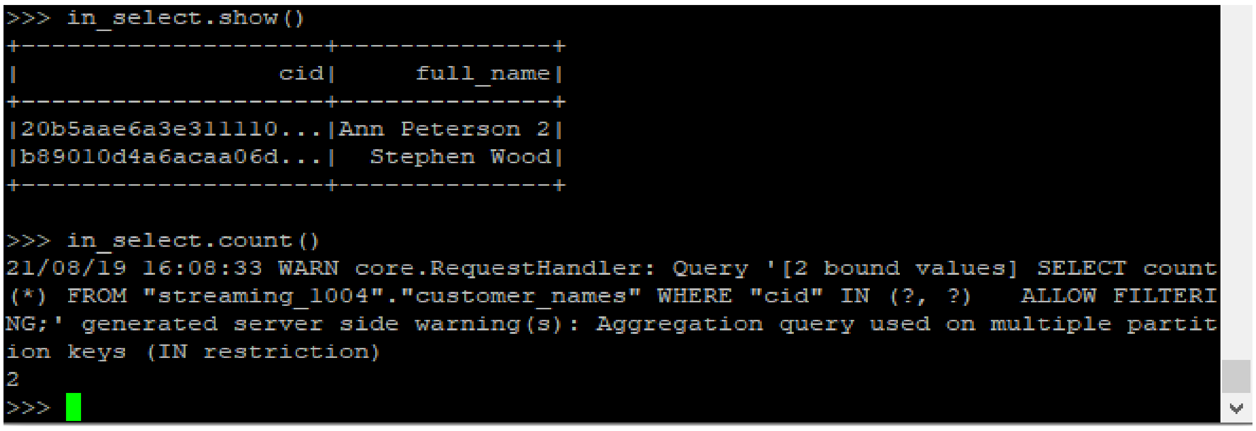
```
>>> between_select.show()
+--------------------+------------------+
|                 cid|         full_name|
+--------------------+------------------+
|75932cfd72b87bd60...|   Jill Lundrigan|
|48f239e15744ed5e7...|     Kathy Denton|
|5a58afc695ee03b9b...|       Liam Walsh|
|6772a0a230a2667d1...|    Steven Marlow|
|83da8aec5d2e8b284...|        Gary Kang|
|4c9c7c2b6de6ee256...|      Dixon Coker|
|65c295c5ac6110fc4...|Jonathan Craddock|
|93f0c8eaf1b7a1c95...|    Gregory Watts|
|241e78de29b3090cf...|     Russell Carr|
|7c5f8c2e42af586aa...|      Will Harvey|
|20b5aae6a3e311110...|   Ann Peterson 2|
|6c347ef65dd574fb9...|  Archie MacGregor|
|7e016f9ea27527978...|   Maurice Savage|
|ae2164e850f39dce4...|   Mohammed Hoque|
|4ad5a269a2d59d2c8...|    Amy Gillespie|
|90d2e6f72916e7282...|      Edward Lees|
|72600f002ba1550d6...|     Gordon Cutts|
|2fdffca8dcdf01547...|     John Collier|
|a00009bf8489ae779...|  Simon Broomhead|
|254e327149dc394a4...|     Dawn Connell|
+--------------------+------------------+
only showing top 20 rows

>>> between_select.count()
71
>>> 
```

Формируем следующий набор данных и просматриваем план запроса



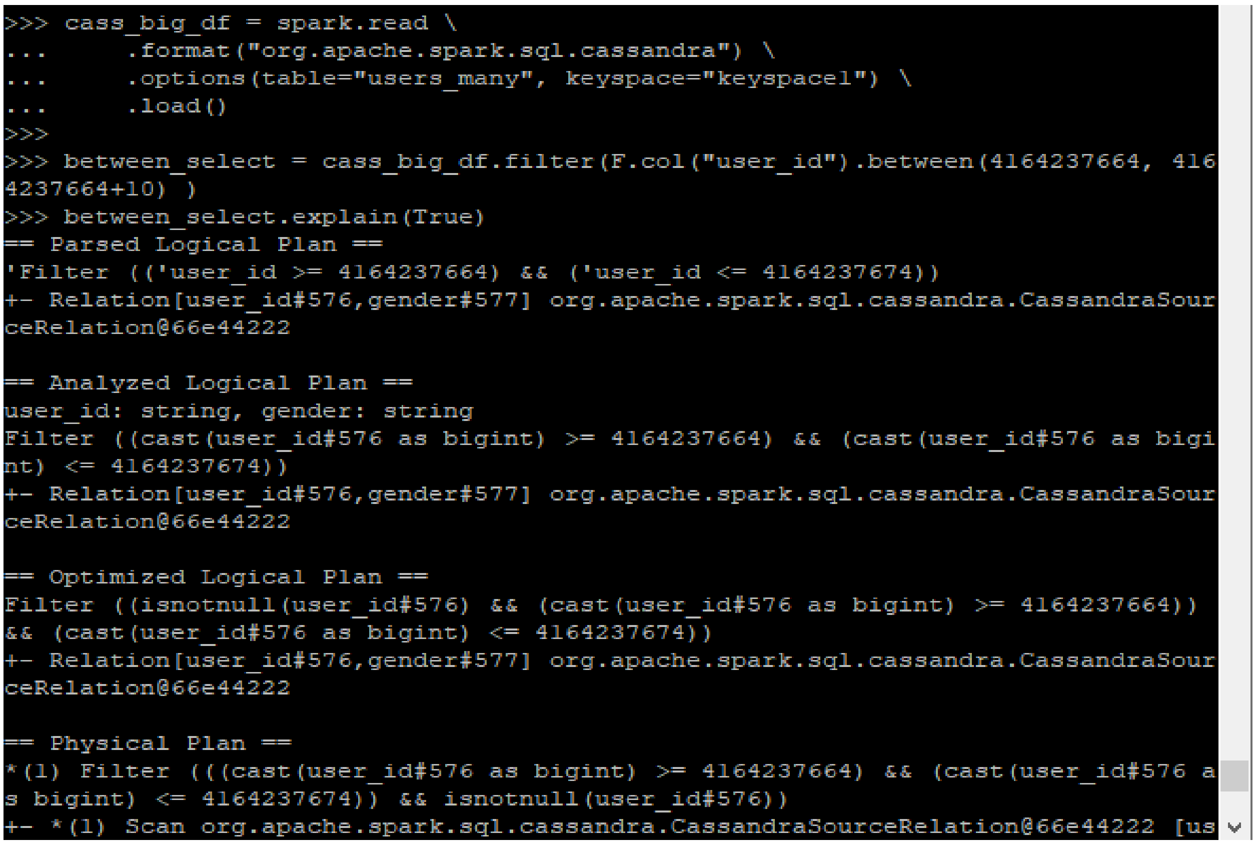
```
>>> in_select = all_names_df.filter(F.col("cid").isin('20b5aae6a3e31111009f9a7ec
c31a232', 'b89010d4a6acaa06d4ef89043869838e'))
>>> in_select.explain(True)
== Parsed Logical Plan ==
'Filter 'cid IN (20b5aae6a3e31111009f9a7ecc31a232,b89010d4a6acaa06d4ef8904386983
8e)
+- Relation[cid#495,full_name#496] org.apache.spark.sql.cassandra.CassandraSourc
eRelation@6d9185b

== Analyzed Logical Plan ==
cid: string, full_name: string
Filter cid#495 IN (20b5aae6a3e31111009f9a7ecc31a232,b89010d4a6acaa06d4ef89043869
838e)
+- Relation[cid#495,full_name#496] org.apache.spark.sql.cassandra.CassandraSourc
eRelation@6d9185b

== Optimized Logical Plan ==
Filter cid#495 IN (20b5aae6a3e31111009f9a7ecc31a232,b89010d4a6acaa06d4ef89043869
838e)
+- Relation[cid#495,full_name#496] org.apache.spark.sql.cassandra.CassandraSourc
eRelation@6d9185b

== Physical Plan ==
*(1) Scan org.apache.spark.sql.cassandra.CassandraSourceRelation@6d9185b [cid#49
5,full_name#496] PushedFilters: [*In(cid, [20b5aae6a3e31111009f9a7ecc31a232,b890
10d4a6acaa06d4ef89043869838e])], ReadSchema: struct<cid:string,full_name:string>
>>> 
```

Выводим набор данных и подсчитываем количество записей



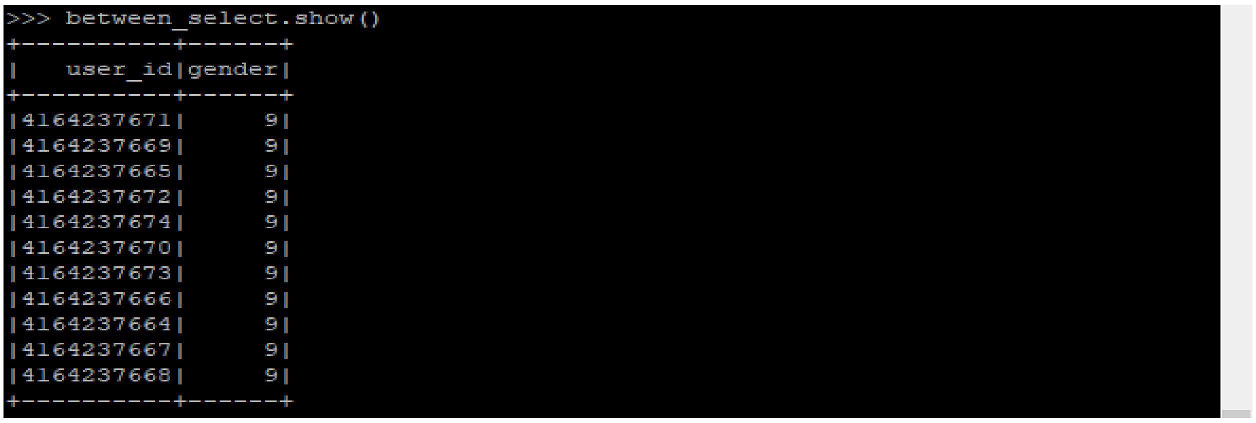
```
>>> in_select.show()
+--------------------+--------------+
|                 cid|     full_name|
+--------------------+--------------+
|20b5aae6a3e311110...|Ann Peterson 2|
|b89010d4a6acaa06d...|  Stephen Wood|
+--------------------+--------------+

>>> in_select.count()
21/08/19 16:08:33 WARN core.RequestHandler: Query '[2 bound values] SELECT count
(*) FROM "streaming_1004"."customer_names" WHERE "cid" IN (?, ?)   ALLOW FILTERI
NG;' generated server side warning(s): Aggregation query used on multiple partit
ion keys (IN restriction)
2
>>>
```

Задаем другой формат вывода, делаем выборку и выводим план запроса



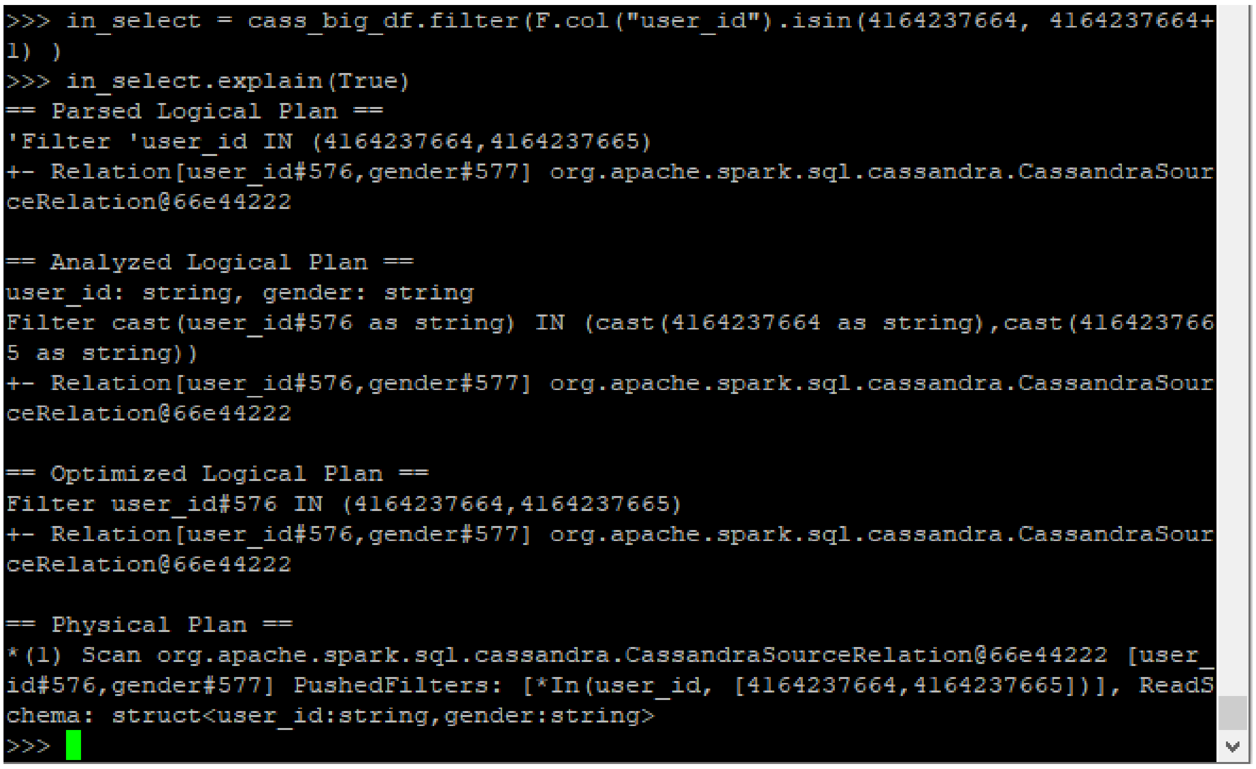
```
>>> cass_big_df = spark.read \
...     .format("org.apache.spark.sql.cassandra") \
...     .options(table="users_many", keyspace="keyspace1") \
...     .load()
>>>
>>> between_select = cass_big_df.filter(F.col("user_id").between(4164237664, 416
4237664+10) )
>>> between_select.explain(True)
== Parsed Logical Plan ==
'Filter (('user_id >= 4164237664) && ('user_id <= 4164237674))
+- Relation[user_id#576,gender#577] org.apache.spark.sql.cassandra.CassandraSour
ceRelation@66e44222

== Analyzed Logical Plan ==
user_id: string, gender: string
Filter ((cast(user_id#576 as bigint) >= 4164237664) && (cast(user_id#576 as bigi
nt) <= 4164237674))
+- Relation[user_id#576,gender#577] org.apache.spark.sql.cassandra.CassandraSour
ceRelation@66e44222

== Optimized Logical Plan ==
Filter ((isnotnull(user_id#576) && (cast(user_id#576 as bigint) >= 4164237664))
&& (cast(user_id#576 as bigint) <= 4164237674))
+- Relation[user_id#576,gender#577] org.apache.spark.sql.cassandra.CassandraSour
ceRelation@66e44222

== Physical Plan ==
*(1) Filter (((cast(user_id#576 as bigint) >= 4164237664) && (cast(user_id#576 a
s bigint) <= 4164237674)) && isnotnull(user_id#576))
+- *(1) Scan org.apache.spark.sql.cassandra.CassandraSourceRelation@66e44222 [us
```

Выводим результат



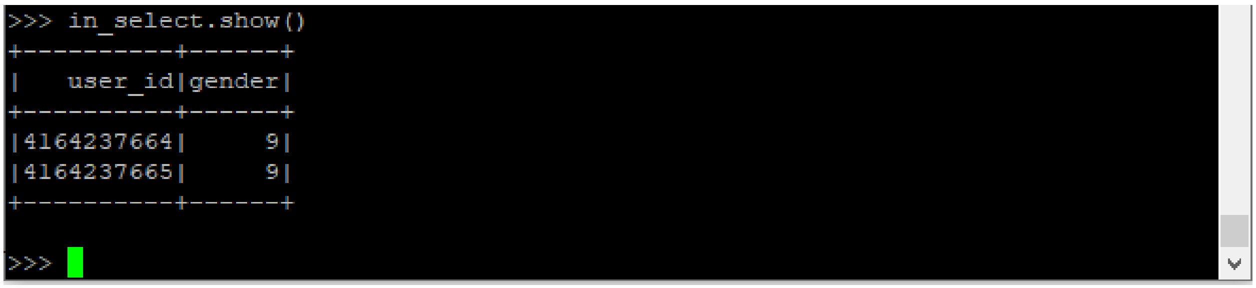
```
>>> between_select.show()
+----------+------+
|   user_id|gender|
+----------+------+
|4164237671|     9|
|4164237669|     9|
|4164237665|     9|
|4164237672|     9|
|4164237674|     9|
|4164237670|     9|
|4164237673|     9|
|4164237666|     9|
|4164237664|     9|
|4164237667|     9|
|4164237668|     9|
+----------+------+
```

Делаем выборку набора данных и просматриваем план запроса



```
>>> in_select = cass_big_df.filter(F.col("user_id").isin(4164237664, 4164237664+
1) )
>>> in_select.explain(True)
== Parsed Logical Plan ==
'Filter 'user_id IN (4164237664,4164237665)
+- Relation[user_id#576,gender#577] org.apache.spark.sql.cassandra.CassandraSour
ceRelation@66e44222

== Analyzed Logical Plan ==
user_id: string, gender: string
Filter cast(user_id#576 as string) IN (cast(4164237664 as string),cast(416423766
5 as string))
+- Relation[user_id#576,gender#577] org.apache.spark.sql.cassandra.CassandraSour
ceRelation@66e44222

== Optimized Logical Plan ==
Filter user_id#576 IN (4164237664,4164237665)
+- Relation[user_id#576,gender#577] org.apache.spark.sql.cassandra.CassandraSour
ceRelation@66e44222

== Physical Plan ==
*(1) Scan org.apache.spark.sql.cassandra.CassandraSourceRelation@66e44222 [user_
id#576,gender#577] PushedFilters: [*In(user_id, [4164237664,4164237665])], ReadS
chema: struct<user_id:string,gender:string>
>>>
```

Выводим набор данных



```
>>> in_select.show()
+----------+------+
|   user_id|gender|
+----------+------+
|4164237664|     9|
|4164237665|     9|
+----------+------+

>>>
```