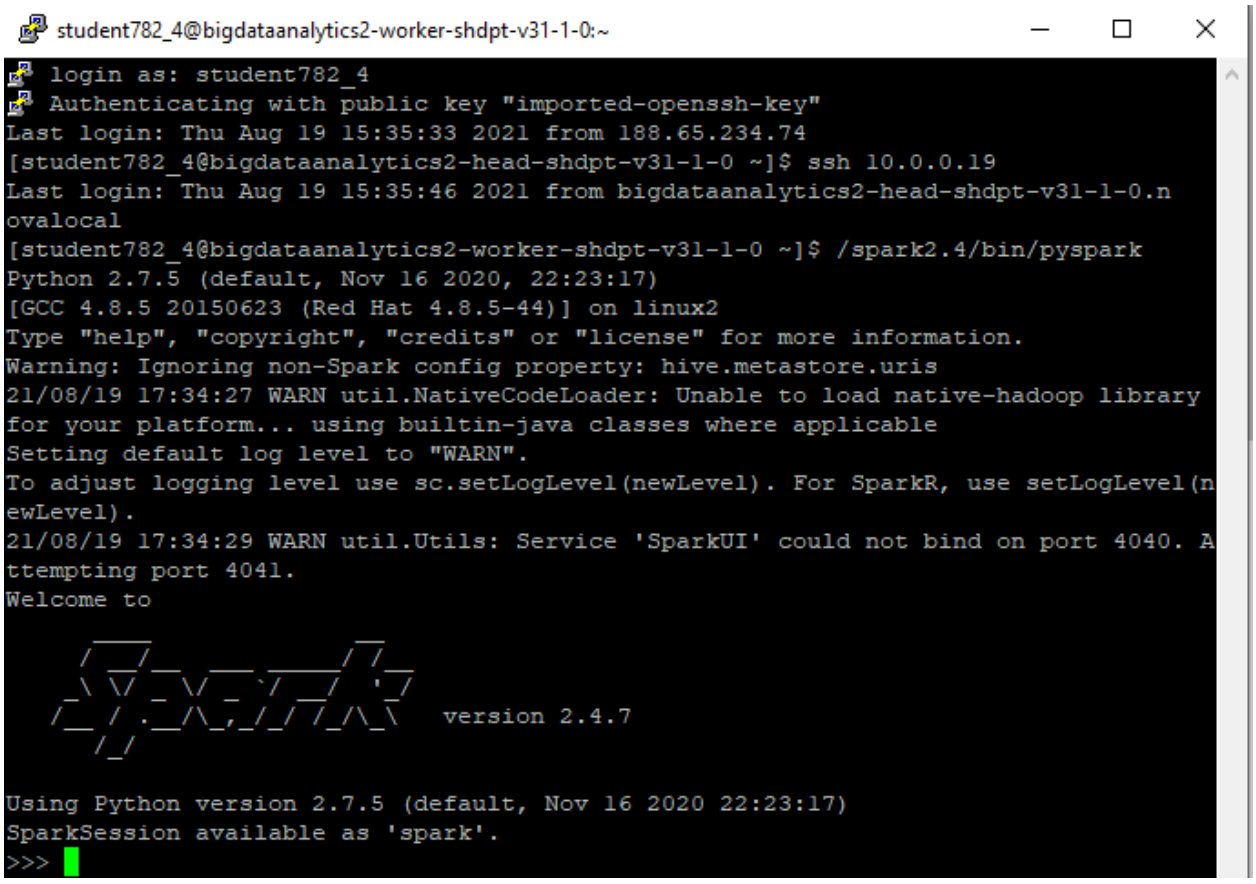
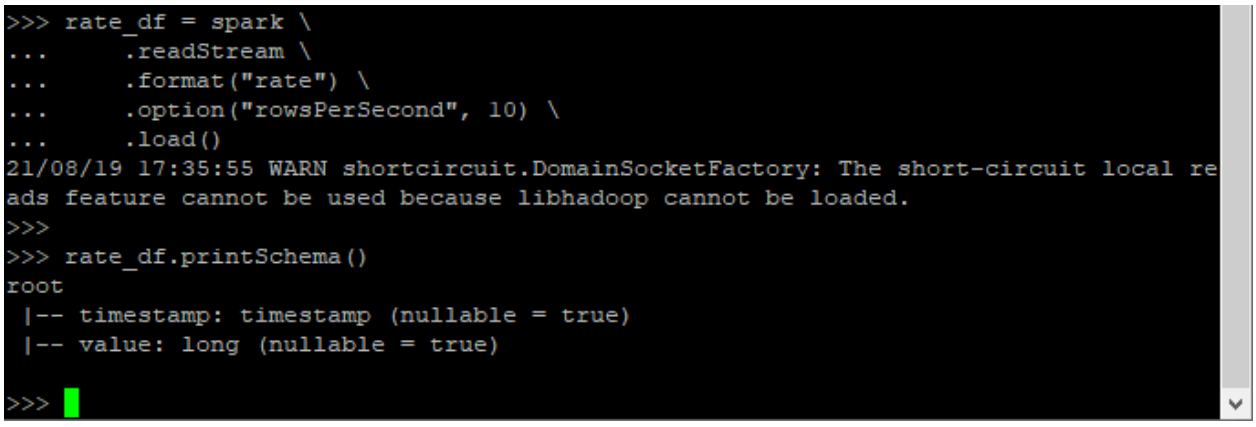
Практическое задание №1

Входим на сервер 10.0.0.19, запускаем spark.



Загружаем датафрейм в формате rate и просматриваем схему



Убеждаемся, что датафрейм потоковый

Запускаем поток с интервалом по умолчанию

```
>>> rate_stream = rate_df \
...     .writeStream \
...     .format("console") \
...     .start()
>>> -------------------------------------------
Batch: 0
-------------------------------------------
+---------+-----+
|timestamp|value|
+---------+-----+
+---------+-----+


-------------------------------------------
Batch: 1
-------------------------------------------
+--------------------+-----+
|           timestamp|value|
+--------------------+-----+
|2021-08-19 17:47:...|    0|
|2021-08-19 17:47:...|    4|
|2021-08-19 17:47:...|    8|
|2021-08-19 17:47:...|    1|
|2021-08-19 17:47:...|    5|
|2021-08-19 17:47:...|    9|
|2021-08-19 17:47:...|    2|
|2021-08-19 17:47:...|    6|
```

Останавливаем поток

```
rate_stream.stop()-------------------------------------------
Batch: 19
-------------------------------------------
+--------------------+-----+
|           timestamp|value|
+--------------------+-----+
|2021-08-19 17:48:...|  180|
|2021-08-19 17:48:...|  184|
|2021-08-19 17:48:...|  188|
|2021-08-19 17:48:...|  181|
|2021-08-19 17:48:...|  185|
|2021-08-19 17:48:...|  189|
|2021-08-19 17:48:...|  182|
|2021-08-19 17:48:...|  186|
|2021-08-19 17:48:...|  183|
|2021-08-19 17:48:...|  187|
+--------------------+-----+


>>>
```

Запускаем поток с опциями (триггер – 30 секунд)

```
>>> stream = rate_df \
...     .writeStream \
...     .trigger(processingTime='30 seconds') \
...     .format("console") \
...     .option("truncate", False) \
...     .start()
>>> -------------------------------------------
Batch: 0
-------------------------------------------
+---------+-----+
|timestamp|value|
+---------+-----+
+---------+-----+


-------------------------------------------
Batch: 1
-------------------------------------------
+-----------------------+-----+
|timestamp              |value|
+-----------------------+-----+
|2021-08-19 17:56:48.516|0    |
|2021-08-19 17:56:48.916|4    |
|2021-08-19 17:56:49.316|8    |
|2021-08-19 17:56:49.716|12   |
|2021-08-19 17:56:50.116|16   |
|2021-08-19 17:56:50.516|20   |
|2021-08-19 17:56:50.916|24   |
|2021-08-19 17:56:51.316|28   |
```

Просматриваем метрики и статусы

```
stream.explain()
== Physical Plan ==
WriteToDataSourceV2 org.apache.spark.sql.execution.streaming.sources.MicroBatchWrit
er@7135773a
+- *(1) Project [timestamp#346, value#347L]
   +- *(1) ScanV2 rate[timestamp#346, value#347L] (Options: [rowsPerSecond=10])
>>> stream.isActive
True
>>> stream.lastProgress
{u'stateOperators': [], u'name': None, u'timestamp': u'2021-08-19T18:03:30.000Z', u
'processedRowsPerSecond': 1635.2201257861634, u'inputRowsPerSecond': 9.971619237554
652, u'numInputRows': 260, u'batchId': 1, u'sources': [{u'description': u'RateStrea
mV2[rowsPerSecond=10, rampUpTimeSeconds=0, numPartitions=default', u'endOffset': 26
, u'processedRowsPerSecond': 1635.2201257861634, u'inputRowsPerSecond': 9.971619237
554652, u'numInputRows': 260, u'startOffset': 0}], u'durationMs': {u'queryPlanning'
: 16, u'walCommit': 38, u'getEndOffset': 0, u'addBatch': 73, u'getBatch': 0, u'setO
ffsetRange': 0, u'triggerExecution': 159}, u'runId': u'36a4d10d-6782-48df-b1d2-f36e
2ac05722', u'id': u'6152f0fd-c8e8-4e44-a486-152818f67a7f', u'sink': {u'description'
: u'org.apache.spark.sql.execution.streaming.ConsoleSinkProvider@6e321e24'}}
>>> stream.status
{u'message': u'Waiting for next trigger', u'isTriggerActive': False, u'isDataAvaila
ble': True}
```

Выводим каждые 5 секунд по 5 записей

```
>>> def rate_source(rps=1):
...     return spark \
...         .readStream \
...         .format("rate") \
...         .option("rowsPerSecond", rps) \
...         .load()
...
>>> def console_output(df, freq):
...     return df \
...         .writeStream \
...         .format("console") \
...         .trigger(processingTime='%s seconds' % freq) \
...         .option("truncate", False) \
...         .start()
...
>>> stream_source = rate_source(5)
>>> stream_sink = console_output(stream_source, 5)
>>> -------------------------------------------
Batch: 0
-------------------------------------------
+---------+-----+
|timestamp|value|
+---------+-----+
+---------+-----+

-------------------------------------------
Batch: 1
-------------------------------------------
+-----------------------+-----+
|timestamp              |value|
+-----------------------+-----+
|2021-08-19 18:11:54.458|35   |
|2021-08-19 18:11:55.258|39   |
|2021-08-19 18:11:56.058|43   |
|2021-08-19 18:11:56.858|47   |
|2021-08-19 18:11:57.658|51   |
|2021-08-19 18:11:58.458|55   |
|2021-08-19 18:11:59.258|59   |
|2021-08-19 18:11:54.658|36   |
|2021-08-19 18:11:55.458|40   |
|2021-08-19 18:11:56.258|44   |
|2021-08-19 18:11:57.058|48   |
|2021-08-19 18:11:57.858|52   |
|2021-08-19 18:11:58.658|56   |
|2021-08-19 18:11:54.858|37   |
|2021-08-19 18:11:55.658|41   |
|2021-08-19 18:11:56.458|45   |
|2021-08-19 18:11:57.258|49   |
|2021-08-19 18:11:58.058|53   |
|2021-08-19 18:11:58.858|57   |
|2021-08-19 18:11:55.058|38   |
+-----------------------+-----+
only showing top 20 rows

stream_sink.stop()
>>>
>>>
```

Задаем фильтрацию по полю value и выводим каждые 5 секунд

```
>>> filtered_rate = rate_df \
...     .filter(F.col("value") % F.lit("2") == 0)
>>>
>>> stream = console_output(filtered_rate, 5)
>>> -------------------------------------------
Batch: 0
-------------------------------------------
+---------+-----+
|timestamp|value|
+---------+-----+
+---------+-----+


-------------------------------------------
Batch: 1
-------------------------------------------
+-----------------------+-----+
|timestamp              |value|
+-----------------------+-----+
|2021-08-19 18:16:04.303|0    |
|2021-08-19 18:16:04.703|4    |
|2021-08-19 18:16:05.103|8    |
|2021-08-19 18:16:05.503|12   |
|2021-08-19 18:16:05.903|16   |
|2021-08-19 18:16:06.303|20   |
|2021-08-19 18:16:06.703|24   |
|2021-08-19 18:16:07.103|28   |
|2021-08-19 18:16:07.503|32   |
|2021-08-19 18:16:07.903|36   |
```

Добавляем новый столбец для вывода

```
>>> extra_rate = filtered_rate \
...     .withColumn("is_jubilee", F.when(
...         (F.col("value") % F.lit(10) == 0), F.lit("true")
...     ).otherwise(F.lit("false")))
>>>
>>> stream = console_output(extra_rate, 5)
>>> -------------------------------------------
Batch: 0
-------------------------------------------
+---------+-----+----------+
|timestamp|value|is_jubilee|
+---------+-----+----------+
+---------+-----+----------+


-------------------------------------------
Batch: 1
-------------------------------------------
+-----------------------+-----+----------+
|timestamp              |value|is_jubilee|
+-----------------------+-----+----------+
|2021-08-19 18:21:32.792|0    |true      |
|2021-08-19 18:21:33.192|4    |false     |
|2021-08-19 18:21:33.592|8    |false     |
|2021-08-19 18:21:33.992|12   |false     |
|2021-08-19 18:21:34.392|16   |false     |
|2021-08-19 18:21:32.992|2    |false     |
|2021-08-19 18:21:33.392|6    |false     |
|2021-08-19 18:21:33.792|10   |true      |
```