

## 5. Spark Streaming. Stateful streams

Загрузить в топик kafka свои данные, прочитать их в потоке, применить watermark и window. Повторить шаги выполненные на занятии.

Дополнительно, объединить статичный и динамичный потоки. Задание на повышенный бал: Написать скрипт на python для конвертации файла csv в json.

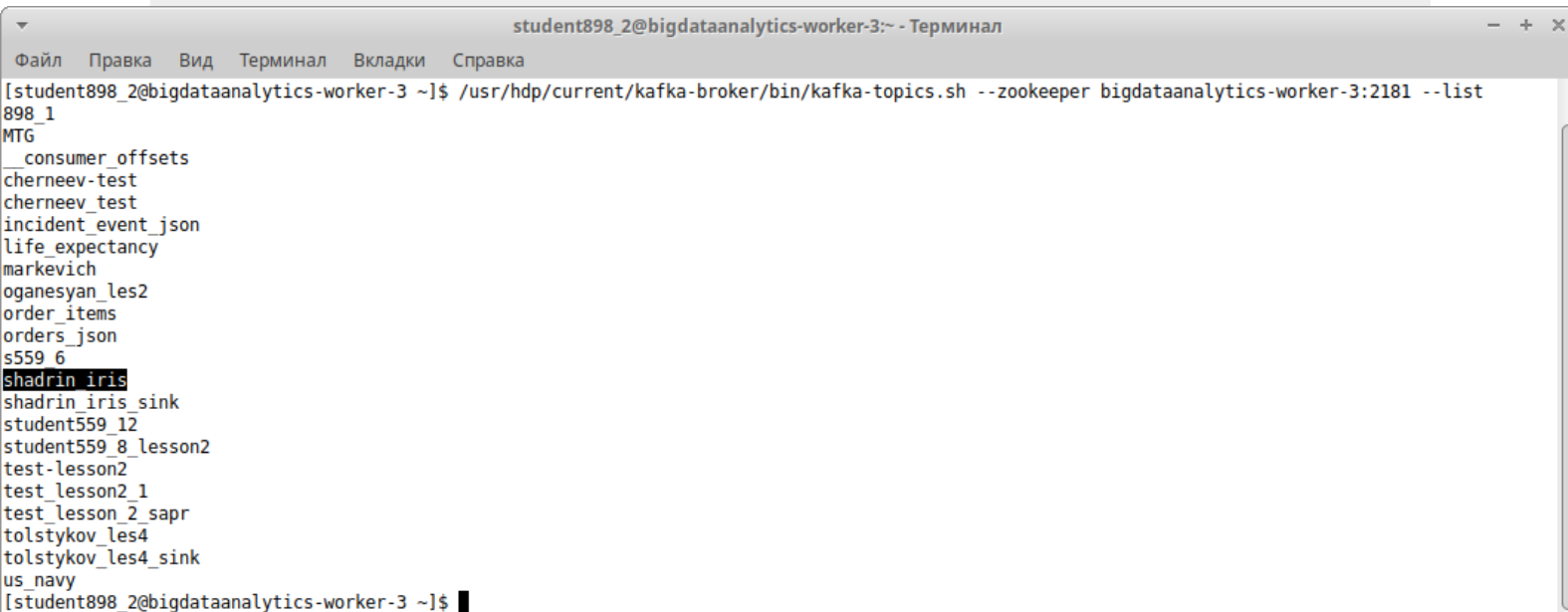
#### Задание 1. Запустить агрегацию по временному окну в разных режимах.

1\.1\.. Подключаемся к серверу

```
ssh -i ~/.ssh/id_rsa_student898_2 student898\_2@37.139.41.176
```

смотрим лист топиков

```
/usr/hdp/current/kafka-broker/bin/kafka-topics.sh --zookeeper bigdataanalytics-worker-3:2181 --list
```



```
student898_2@bigdataanalytics-worker-3:~ - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
[student898_2@bigdataanalytics-worker-3 ~]$ /usr/hdp/current/kafka-broker/bin/kafka-topics.sh --zookeeper bigdataanalytics-worker-3:2181 --list
898_1
MTG
__consumer_offsets
cherneev-test
cherneev_test
incident_event_json
life_expectancy
markevich
oganesyan_les2
order_items
orders_json
s559 6
shadrin_iris
shadrin_iris_sink
student559_12
student559_8_lesson2
test-lesson2
test_lesson2_1
test_lesson_2_sapr
tolstykov_les4
tolstykov_les4_sink
us_navy
[student898_2@bigdataanalytics-worker-3 ~]$
```

Прочитать топик shadrin\_iris

```
/usr/hdp/current/kafka-broker/bin/kafka-console-consumer.sh --topic shadrin_iris --from-beginning --bootstrap-server bigdataanalytics-worker-3:6667 --max-messages 15
```

```
student898_2@bigdataanalytics-worker-3:~ - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
[student898_2@bigdataanalytics-worker-3 ~]$ /usr/hdp/current/kafka-broker/bin/kafka-console-consumer.sh --topic shadrin_iris --from-beginning --bootstrap-server bigdataanalytics-worker-3:6667 --max-messages 15
[
{"sepalLength": 5.1, "sepalWidth": 3.5, "petalLength": 1.4, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 4.9, "sepalWidth": 3.0, "petalLength": 1.4, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 4.7, "sepalWidth": 3.2, "petalLength": 1.3, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 4.6, "sepalWidth": 3.1, "petalLength": 1.5, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 5.0, "sepalWidth": 3.6, "petalLength": 1.4, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 5.4, "sepalWidth": 3.9, "petalLength": 1.7, "petalWidth": 0.4, "species": "setosa"},
{"sepalLength": 4.6, "sepalWidth": 3.1, "petalLength": 1.4, "petalWidth": 0.3, "species": "setosa"},
{"sepalLength": 5.0, "sepalWidth": 3.4, "petalLength": 1.5, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 4.4, "sepalWidth": 2.9, "petalLength": 1.4, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 4.9, "sepalWidth": 3.1, "petalLength": 1.5, "petalWidth": 0.1, "species": "setosa"},
{"sepalLength": 5.4, "sepalWidth": 3.7, "petalLength": 1.5, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 4.8, "sepalWidth": 3.4, "petalLength": 1.6, "petalWidth": 0.2, "species": "setosa"},
{"sepalLength": 4.8, "sepalWidth": 3.0, "petalLength": 1.4, "petalWidth": 0.1, "species": "setosa"},
{"sepalLength": 4.3, "sepalWidth": 3.0, "petalLength": 1.1, "petalWidth": 0.1, "species": "setosa"},
Processed a total of 15 messages
[student898_2@bigdataanalytics-worker-3 ~]$
```

Запускаем `pyspark`

```
export SPARK_KAFKA_VERSION=0.10
```

```
/opt/spark-2.4.8/bin/pyspark --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 --driver-memory 512m --master local[1]
```

```
student898_2@bigdataanalytics-worker-3:~ - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
|      default      | 6 | 0 | 0 | 0 || 6 | 0 |
-----
:: retrieving :: org.apache.spark#spark-submit-parent-07c4eae8-8f5f-460d-97ba-2a696c8f49a5
  confs: [default]
    0 artifacts copied, 6 already retrieved (0kB/6ms)
22/01/21 17:22:05 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
22/01/21 17:22:06 WARN util.Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
Welcome to

  ____      __
 / ___ |    /  \
| |  \| |  /    \
| |___| | /  /\
|  ___  ||  /\
|_|   |_| \_/_/

version 2.4.8

Using Python version 2.7.5 (default, Nov 16 2020 22:23:17)
SparkSession available as 'spark'.
>>>
```

Далее делаем стандартные импорты и определяем константы, схему данных и сам стрим, который читает топик `shadrin\_iris` из Кафки.

```
from pyspark.sql import functions as F
```

```
from pyspark.sql.types import StructType, StringType, FloatType
```

```
kafka_brokers = "bigdataanalytics-worker-3:6667"
```

```
raw_data = spark.readStream. \
```

```
    format("kafka"). \
```

```

option("kafka.bootstrap.servers", kafka_brokers). \

option("subscribe", "shadrin_iris"). \

option("startingOffsets", "earliest"). \

option("maxOffsetsPerTrigger", "6"). \

load()

schema = StructType() \

    .add("sepalLength", FloatType()) \

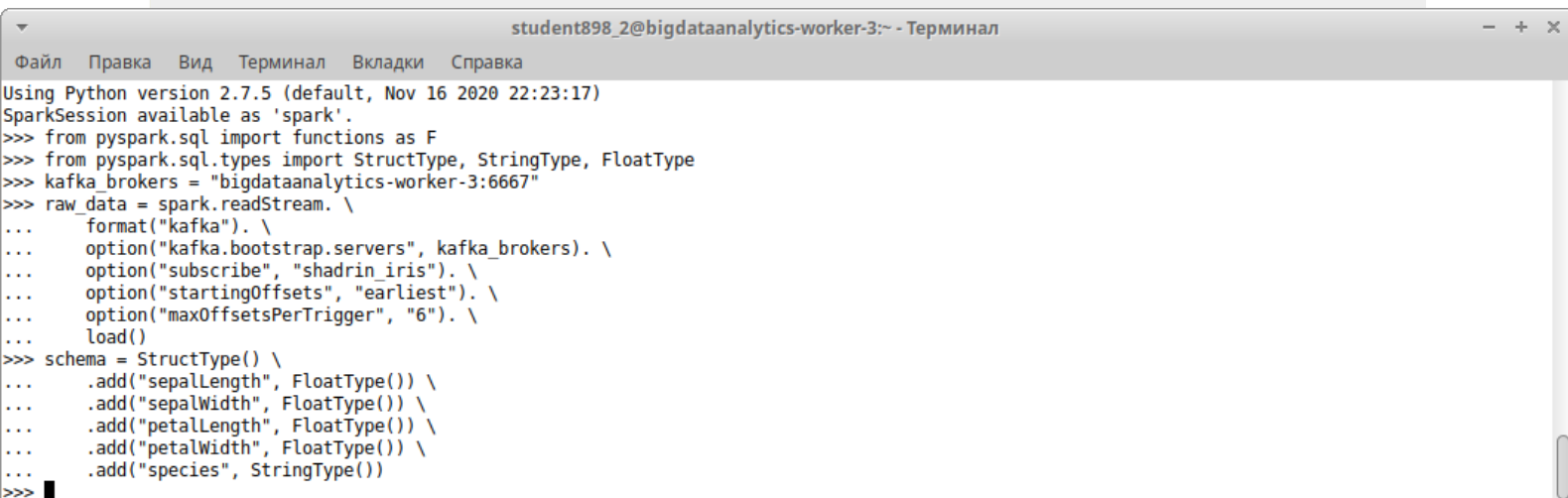
    .add("sepalWidth", FloatType()) \

    .add("petalLength", FloatType()) \

    .add("petalWidth", FloatType()) \

    .add("species", StringType())

```



```

student898_2@bigdataanalytics-worker-3:~ - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
Using Python version 2.7.5 (default, Nov 16 2020 22:23:17)
SparkSession available as 'spark'.
>>> from pyspark.sql import functions as F
>>> from pyspark.sql.types import StructType, StringType, FloatType
>>> kafka_brokers = "bigdataanalytics-worker-3:6667"
>>> raw_data = spark.readStream. \
...     format("kafka"). \
...     option("kafka.bootstrap.servers", kafka_brokers). \
...     option("subscribe", "shadrin_iris"). \
...     option("startingOffsets", "earliest"). \
...     option("maxOffsetsPerTrigger", "6"). \
...     load()
>>> schema = StructType() \
...     .add("sepalLength", FloatType()) \
...     .add("sepalWidth", FloatType()) \
...     .add("petalLength", FloatType()) \
...     .add("petalWidth", FloatType()) \
...     .add("species", StringType())
>>>

```

## WATERMARK и дубликаты внутри одного батча.

Watermark - одна из настроек стрима для сохранения состояния этого стрима внутри чекпойнта. Добавим колонку с временем обработки микробатча. Она понадобится для настройки чекпойнта и ватермарки.

```

extended_iris = raw_data \

    .select(F.from_json(F.col("value").cast("String"), schema).alias("value"),
"offset") \

    .select("value.*", "offset") \

```

```
.withColumn("receive_time", F.current_timestamp())
```

```
extended_iris.printSchema()
```

```
student898_2@bigdataanalytics-worker-3:~ - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
>>> extended_iris = raw_data \
...     .select(F.from_json(F.col("value").cast("String"), schema).alias("value"), "offset") \
...     .select("value.*", "offset") \
...     .withColumn("receive_time", F.current_timestamp())
>>> extended_iris.printSchema()
root
|-- sepalLength: float (nullable = true)
|-- sepalWidth: float (nullable = true)
|-- petalLength: float (nullable = true)
|-- petalWidth: float (nullable = true)
|-- species: string (nullable = true)
|-- offset: long (nullable = true)
|-- receive_time: timestamp (nullable = false)
>>> █
```

В методе `console\_output` обязательно указываем папку, в которой будет храниться чекпойнт.

```
def console_output(df, freq):

    return df.writeStream \

        .format("console") \

        .trigger(processingTime='%s seconds' % freq ) \

        .option("checkpointLocation", "checkpoints/duplicates_console_chk") \

        .options(truncate=False) \

        .start()
```

```
student898_2@bigdataanalytics-worker-3:~ - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
>>> def console_output(df, freq):
...     return df.writeStream \
...         .format("console") \
...         .trigger(processingTime='%s seconds' % freq ) \
...         .option("checkpointLocation", "checkpoints/duplicates_console_chk") \
...         .options(truncate=False) \
...         .start()
...
>>> █
```

Запускаем стрим и смотрим как растёт наш чекпойнт (команда `hdfs dfs -du -h checkpoints/duplicates\_console\_chk`).

```
stream = console_output(extended_iris , 5)

stream.stop()
```

```
student898_2@bigdataanalytics-worker-3:~ - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
Batch: 1
-----
+-----+-----+-----+-----+-----+-----+-----+
|sepalLength|sepalWidth|petalLength|petalWidth|species|offset|receive_time|
+-----+-----+-----+-----+-----+-----+-----+
|5.4        |3.9        |1.7        |0.4        |setosa |6      |2022-01-21 17:28:57.352|
|4.6        |3.4        |1.4        |0.3        |setosa |7      |2022-01-21 17:28:57.352|
|5.0        |3.4        |1.5        |0.2        |setosa |8      |2022-01-21 17:28:57.352|
|4.4        |2.9        |1.4        |0.2        |setosa |9      |2022-01-21 17:28:57.352|
|4.9        |3.1        |1.5        |0.1        |setosa |10     |2022-01-21 17:28:57.352|
|5.4        |3.7        |1.5        |0.2        |setosa |11     |2022-01-21 17:28:57.352|
+-----+-----+-----+-----+-----+-----+-----+

Batch: 2
-----
+-----+-----+-----+-----+-----+-----+-----+
|sepalLength|sepalWidth|petalLength|petalWidth|species|offset|receive_time|
+-----+-----+-----+-----+-----+-----+-----+
|4.8        |3.4        |1.6        |0.2        |setosa |12     |2022-01-21 17:29:00.003|
|4.8        |3.0        |1.4        |0.1        |setosa |13     |2022-01-21 17:29:00.003|
|4.3        |3.0        |1.1        |0.1        |setosa |14     |2022-01-21 17:29:00.003|
|5.8        |4.0        |1.2        |0.2        |setosa |15     |2022-01-21 17:29:00.003|
|5.7        |4.4        |1.5        |0.4        |setosa |16     |2022-01-21 17:29:00.003|
|5.4        |3.9        |1.3        |0.4        |setosa |17     |2022-01-21 17:29:00.003|
+-----+-----+-----+-----+-----+-----+-----+

stream.stop()
>>> stream.stop()
>>> █
```

Задаём водтермарку, которая должна очищать чекпойнт. Первый параметр - название колонки, на которую смотрит водтермарка, второй параметр - гарантированное время жизни информации о сообщении в чекпойнте. Именно для этого мы добавляли столбец `receive\_time`.

```
waterwarked_iris = extended_iris.withWatermark("receive_time", "30 seconds")
```

```
waterwarked_iris.printSchema()
```

```
student898_2@bigdataanalytics-worker-3:~ - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
>>> waterwarked_iris = extended_iris.withWatermark("receive_time", "30 seconds")
>>> waterwarked_iris.printSchema()
root
|-- sepalLength: float (nullable = true)
|-- sepalWidth: float (nullable = true)
|-- petalLength: float (nullable = true)
|-- petalWidth: float (nullable = true)
|-- species: string (nullable = true)
|-- offset: long (nullable = true)
|-- receive_time: timestamp (nullable = false)
>>> █
```

Схема не поменялась. Водтермарка только следит за чекпойнтом, но никак не аффецит наши данные.

Теперь данные можно проверить на наличие дубликатов. Дубли проверяем по двум колонкам: `species` и `receive\_time`. Таким образом будут отсеиваться дубли по полю `species` внутри одного микробатча, так как столбец `receive\_time` для всех записей внутри этого микробатча одинаковый.

```
deduplicated_iris = waterwarked_iris.drop_duplicates(["species",
"receive_time"])
```

Чтобы всё заработало, надо во 2 терминале очистить чекпойнт

```
hdfs dfs -rm -r checkpoints/duplicates_console_chk
```

```
stream = console_output(deduplicated_iris , 20)
```

```
stream.stop()
```

```
student898_2@bigdataanalytics-worker-3:~ - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
Batch: 0
-----
+-----+-----+-----+-----+-----+-----+
|sepalLength|sepalWidth|petalLength|petalWidth|species|offset|receive_time|
+-----+-----+-----+-----+-----+-----+
|null       |null      |null      |null      |null   |0      |2022-01-21 17:34:43.259|
|5.1        |3.5       |1.4       |0.2       |setosa |1      |2022-01-21 17:34:43.259|
+-----+-----+-----+-----+-----+-----+
Batch: 1
-----
+-----+-----+-----+-----+-----+-----+
|sepalLength|sepalWidth|petalLength|petalWidth|species|offset|receive_time|
+-----+-----+-----+-----+-----+-----+
|5.4        |3.9       |1.7       |0.4       |setosa |6      |2022-01-21 17:35:00.004|
+-----+-----+-----+-----+-----+-----+
[Stage 8:=====>                                (34 + 1) / 200]
```

Без указания в `drop\_duplicates` столбца с временной меткой, дубликаты отсеивались бы по всем данным. И все эти данные хранились бы в чекпойнте, даже не смотря на то что у нас настроена воте́рмарка. Чтобы не допустить раздувания чекпойнта, указывается столбец с временной меткой. Воте́рмарка ограничивает "глубину" данных не только для поиска дубликатов, но и для любой группирующей функции.

1\5\ WINDOW и дубликаты за периоды времени.

Создаём временное окно. В структуру датафрейма добавился новый столбец.

```
windowed_iris = extended_iris.withColumn("window_time",
F.window(F.col("receive_time"), "2 minutes"))
```

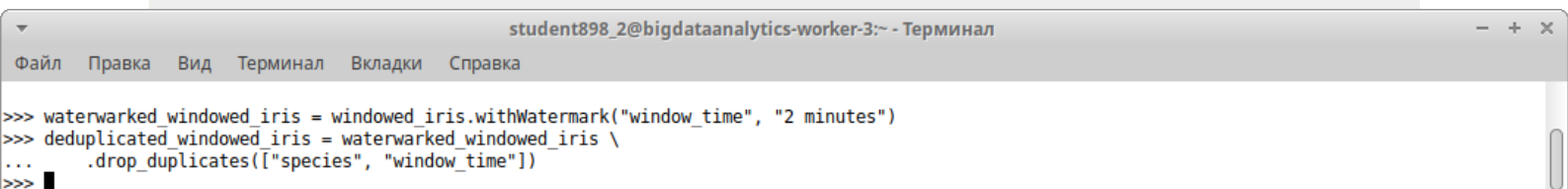
```
windowed_iris.printSchema()
```

```
student898_2@bigdataanalytics-worker-3:~ - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
>>> windowed_iris = extended_iris.withColumn("window_time", F.window(F.col("receive_time"), "2 minutes"))
>>> windowed_iris.printSchema()
root
 |-- sepalLength: float (nullable = true)
 |-- sepalWidth: float (nullable = true)
 |-- petalLength: float (nullable = true)
 |-- petalWidth: float (nullable = true)
 |-- species: string (nullable = true)
 |-- offset: long (nullable = true)
 |-- receive_time: timestamp (nullable = false)
 |-- window_time: struct (nullable = false)
 |     |-- start: timestamp (nullable = true)
 |     |-- end: timestamp (nullable = true)
>>>
```

Устанавливаем воте́рмарку для очистки чекпоинта и удаляем дубли в каждом окне.

```
waterwarnked_windowed_iris = windowed_iris.withWatermark("window_time", "2 minutes")
```

```
deduplicated_windowed_iris = waterwarnked_windowed_iris \  
  
    .drop_duplicates(["species", "window_time"])
```



The screenshot shows a terminal window titled "student898\_2@bigdataanalytics-worker-3:~ - Терминал". The menu bar includes "Файл", "Правка", "Вид", "Терминал", "Вкладки", and "Справка". The terminal content shows the code being executed line by line, with the cursor at the end of the last line.

```
>>> waterwarnked_windowed_iris = windowed_iris.withWatermark("window_time", "2 minutes")  
>>> deduplicated_windowed_iris = waterwarnked_windowed_iris \  
...     .drop_duplicates(["species", "window_time"])  
>>> █
```

Проверяем как удаляются дубли из каждого окна.

```
stream = console_output(deduplicated_windowed_iris , 10)
```

```
stream.stop()
```

```
student898_2@bigdataanalytics-worker-3:~ - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка

Batch: 2
>>> +-----+
|sepalLength|sepalWidth|petalLength|petalWidth|species|offset|receive_time|window_time|
+-----+
|4.8|3.4|1.6|0.2|setosa|12|2022-01-21 17:35:20.004|[2022-01-21 17:34:00, 2022-01-21 17:36:00]|
+-----+

Batch: 3
+-----+
|sepalLength|sepalWidth|petalLength|petalWidth|species|offset|receive_time|window_time|
+-----+
|5.1|3.5|1.4|0.3|setosa|18|2022-01-21 17:41:48.429|[2022-01-21 17:40:00, 2022-01-21 17:42:00]|
+-----+

Batch: 4
+-----+
|sepalLength|sepalWidth|petalLength|petalWidth|species|offset|receive_time|window_time|
+-----+

Batch: 5
+-----+
|sepalLength|sepalWidth|petalLength|petalWidth|species|offset|receive_time|window_time|
+-----+
|4.7|3.2|1.6|0.2|setosa|30|2022-01-21 17:42:00.004|[2022-01-21 17:42:00, 2022-01-21 17:44:00]|
+-----+

Batch: 6
+-----+
|sepalLength|sepalWidth|petalLength|petalWidth|species|offset|receive_time|window_time|
+-----+

Batch: 7
+-----+
|sepalLength|sepalWidth|petalLength|petalWidth|species|offset|receive_time|window_time|
+-----+

Batch: 8
+-----+
|sepalLength|sepalWidth|petalLength|petalWidth|species|offset|receive_time|window_time|
+-----+
|7.0|3.2|4.7|1.4|versicolor|51|2022-01-21 17:42:30.004|[2022-01-21 17:42:00, 2022-01-21 17:44:00]|
+-----+

stream.stop()
>>> stream.stop()
>>> █
```

Видим что в рамках одного окна дубликатов нет.

## 1\6\. SLIDING WINDOW

Аналогично предыдущему пункту создаём дополнительное поле `sliding\_time`. В функции `F.window` первый аргумент это колонка (временная метка), по которой создаётся окно; второй аргумент - ширина окна; третий - сдвиг окна. Добавляем вотермарку и указываем колонки, по которым будем исключать дубли.

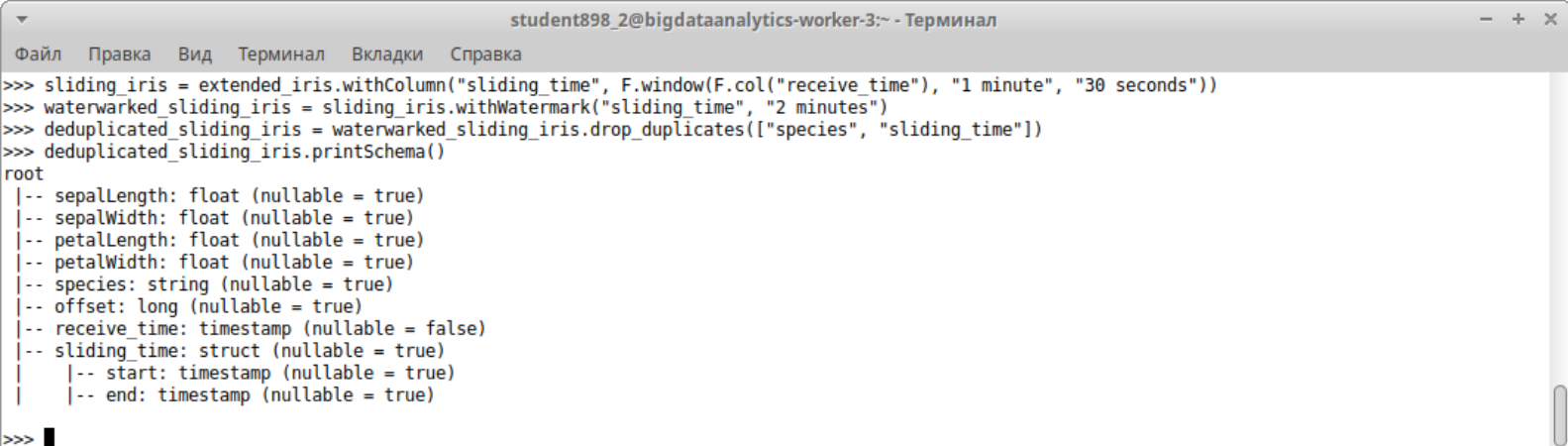


```
sliding_iris = extended_iris.withColumn("sliding_time",
F.window(F.col("receive_time"), "1 minute", "30 seconds"))

waterworked_sliding_iris = sliding_iris.withWatermark("sliding_time", "2
minutes")

deduplicated_sliding_iris =
waterworked_sliding_iris.drop_duplicates(["species", "sliding_time"])

deduplicated_sliding_iris.printSchema()
```



The screenshot shows a terminal window titled "student898\_2@bigdataanalytics-worker-3:~ - Терминал". The terminal contains the following commands and output:

```
>>> sliding_iris = extended_iris.withColumn("sliding_time", F.window(F.col("receive_time"), "1 minute", "30 seconds"))
>>> waterworked_sliding_iris = sliding_iris.withWatermark("sliding_time", "2 minutes")
>>> deduplicated_sliding_iris = waterworked_sliding_iris.drop_duplicates(["species", "sliding_time"])
>>> deduplicated_sliding_iris.printSchema()
root
|-- sepalLength: float (nullable = true)
|-- sepalWidth: float (nullable = true)
|-- petalLength: float (nullable = true)
|-- petalWidth: float (nullable = true)
|-- species: string (nullable = true)
|-- offset: long (nullable = true)
|-- receive_time: timestamp (nullable = false)
|-- sliding_time: struct (nullable = true)
|   |-- start: timestamp (nullable = true)
|   |-- end: timestamp (nullable = true)
>>>
```

```
stream = console_output(deduplicated_sliding_iris , 10)

stream.stop()
```

```
student898_2@bigdataanalytics-worker-3:~ - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка

if needed...Note that this is normal for the first batch of starting query.
[Stage 24:=====>(198 + 1) / 200]22/01/21 17:46:14 WARN state.HDFSBackedStateStoreProvider: The state f
or version 9 doesn't exist in loadedMaps. Reading snapshot file and delta files if needed...Note that this is normal for the first batch of starting q
uery.
-----
Batch: 9
-----
+-----+-----+-----+-----+-----+-----+-----+-----+
|sepalLength|sepalWidth|petalLength|petalWidth|species|offset|receive_time|sliding_time|
+-----+-----+-----+-----+-----+-----+-----+-----+
|5.5|2.3|4.0|1.3|versicolor|54|2022-01-21 17:46:05.677|[2022-01-21 17:45:30, 2022-01-21 17:46:30]|
|5.5|2.3|4.0|1.3|versicolor|54|2022-01-21 17:46:05.677|[2022-01-21 17:46:00, 2022-01-21 17:47:00]|
+-----+-----+-----+-----+-----+-----+-----+-----+

22/01/21 17:46:14 WARN streaming.ProcessingTimeExecutor: Current batch is falling behind. The trigger interval is 10000 milliseconds, but spent 12230
milliseconds
-----
Batch: 10
-----
+-----+-----+-----+-----+-----+-----+-----+-----+
|sepalLength|sepalWidth|petalLength|petalWidth|species|offset|receive_time|sliding_time|
+-----+-----+-----+-----+-----+-----+-----+-----+
|
+-----+-----+-----+-----+-----+-----+-----+-----+

Batch: 11
-----
+-----+-----+-----+-----+-----+-----+-----+-----+
|sepalLength|sepalWidth|petalLength|petalWidth|species|offset|receive_time|sliding_time|
+-----+-----+-----+-----+-----+-----+-----+-----+
|
+-----+-----+-----+-----+-----+-----+-----+-----+

Batch: 12
-----
+-----+-----+-----+-----+-----+-----+-----+-----+
|sepalLength|sepalWidth|petalLength|petalWidth|species|offset|receive_time|sliding_time|
+-----+-----+-----+-----+-----+-----+-----+-----+
|6.1|2.8|4.0|1.3|versicolor|72|2022-01-21 17:46:30.003|[2022-01-21 17:46:30, 2022-01-21 17:47:30]|
+-----+-----+-----+-----+-----+-----+-----+-----+

stream.stop()
>>> stream.stop()
>>> █
```

Тут так же видим что в рамках каждого окна нет дубликатов. В третьем миробатче хорошо видно, как одна и та же запись с `offset = 19` является новой для обоих окон `19:49:30 - 19:50:30` и `19:50:00 - 19:51:00`.

## 1\.7\ . OUTPUT MODES - считаем суммы

Переопределяем метод `console\_output` так, чтобы можно было задавать режим вывода результата работы агрегационных функций.

```
def console_output(df, freq, out_mode):

    return df.writeStream.format("console") \

        .trigger(processingTime='%s seconds' % freq ) \

        .options(truncate=False) \

        .option("checkpointLocation", "checkpoints/watermark_console_chk2") \

        .outputMode(out_mode) \
```

```
.start()
```

Используем ранее созданный датафрейм с водермаркой на 2 минуты. Группируем данные по скользящему окну `window\_time`.

```
count_iris = waterwarked_windowed_iris.groupBy("window_time").count()
```

Перед каждым запуском очищаем чекпойнт командой `hdfs dfs -rm -r checkpoints/watermark\_console\_chk2`.

```
##### update
```

```
stream = console_output(count_iris , 10, "update")
```

```
stream.stop()
```

```
student898_2@bigdataanalytics-worker-3:~ - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
|[2022-01-21 17:50:00, 2022-01-21 17:52:00]|6  |
+-----+
22/01/21 17:51:46 WARN streaming.ProcessingTimeExecutor: Current batch is falling behind. The trigger interval is 10000 milliseconds, but spent 10016 milliseconds
-----
Batch: 1
-----
+-----+
|window_time|count|
+-----+
|[2022-01-21 17:50:00, 2022-01-21 17:52:00]|12  |
+-----+
-----
Batch: 2
-----
+-----+
|window_time|count|
+-----+
|[2022-01-21 17:50:00, 2022-01-21 17:52:00]|18  |
+-----+
-----
Batch: 3
-----
+-----+
|window_time|count|
+-----+
|[2022-01-21 17:52:00, 2022-01-21 17:54:00]|6  |
+-----+
-----
Batch: 4
-----
+-----+
|window_time|count|
+-----+
|[2022-01-21 17:52:00, 2022-01-21 17:54:00]|12  |
+-----+
stream.stop()
>>> stream.stop()
>>>
```

В режиме `outputMode("update")` в консоль пишутся только обновляющиеся записи. Считается `count` по каждому окну и выводятся записи только о тех окнах, в которых значение поменялось.

```
##### complete
```

```
student898_2@bigdataanalytics-worker-3:~ - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка

[Stage 42:=====>(198 + 1) / 200]22/01/21 17:54:11 WARN state.HDFSBackedStateStoreProvider: The state f
or version 5 doesn't exist in loadedMaps. Reading snapshot file and delta files if needed...Note that this is normal for the first batch of starting q
uery.
-----
Batch: 5
-----
+-----+
|window_time|count|
+-----+
|[2022-01-21 17:52:00, 2022-01-21 17:54:00]|12|
|[2022-01-21 17:54:00, 2022-01-21 17:56:00]|6|
|[2022-01-21 17:50:00, 2022-01-21 17:52:00]|18|
+-----+

22/01/21 17:54:11 WARN streaming.ProcessingTimeExecutor: Current batch is falling behind. The trigger interval is 10000 milliseconds, but spent 11204
milliseconds
-----
Batch: 6
-----
+-----+
|window_time|count|
+-----+
|[2022-01-21 17:52:00, 2022-01-21 17:54:00]|12|
|[2022-01-21 17:54:00, 2022-01-21 17:56:00]|12|
|[2022-01-21 17:50:00, 2022-01-21 17:52:00]|18|
+-----+

22/01/21 17:54:23 WARN streaming.ProcessingTimeExecutor: Current batch is falling behind. The trigger interval is 10000 milliseconds, but spent 12153
milliseconds
-----
Batch: 7
-----
+-----+
|window_time|count|
+-----+
|[2022-01-21 17:52:00, 2022-01-21 17:54:00]|12|
|[2022-01-21 17:54:00, 2022-01-21 17:56:00]|18|
|[2022-01-21 17:50:00, 2022-01-21 17:52:00]|18|
+-----+

-----
Batch: 8
-----
+-----+
|window_time|count|
+-----+
|[2022-01-21 17:52:00, 2022-01-21 17:54:00]|12|
|[2022-01-21 17:54:00, 2022-01-21 17:56:00]|24|
|[2022-01-21 17:50:00, 2022-01-21 17:52:00]|18|
+-----+

-----
Batch: 9
-----
+-----+
|window_time|count|
+-----+
|[2022-01-21 17:52:00, 2022-01-21 17:54:00]|12|
|[2022-01-21 17:54:00, 2022-01-21 17:56:00]|30|
|[2022-01-21 17:50:00, 2022-01-21 17:52:00]|18|
+-----+

stream.stop()

stream.stop()
File "<stdin>", line 1
  stream.stop()stream.stop()
    ^
SyntaxError: invalid syntax
>>> stream.stop()
>>>
```

```
stream = console_output(count_iris , 10, "complete")
```

```
stream.stop()
```

```
##### append
```

Пишем все записи только один раз. Информация выводится один раз, когда окно заканчивается.

```
stream = console_output(count_iris , 10, "append")
```

```
stream.stop()
```

Тут не увидел результата все батчи пустые. Скорее всего данный режим не поддерживается для данной агрегирующей функции.

1\8\.. Наблюдаем за суммами в плавающем окне.

```
student898_2@bigdataanalytics-worker-3:~ - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
query.
-----
Batch: 13
-----
+-----+-----+
|sliding_time|count|
+-----+-----+
|[2022-01-21 17:58:00, 2022-01-21 17:59:00]|6|
|[2022-01-21 17:57:30, 2022-01-21 17:58:30]|6|
+-----+-----+

22/01/21 17:58:22 WARN streaming.ProcessingTimeExecutor: Current batch is falling behind. The trigger interval is 10000 milliseconds, but spent 15671 milliseconds

Batch: 14
-----
+-----+-----+
|sliding_time|count|
+-----+-----+
|[2022-01-21 17:58:00, 2022-01-21 17:59:00]|12|
|[2022-01-21 17:57:30, 2022-01-21 17:58:30]|12|
+-----+-----+

Batch: 15
-----
+-----+-----+
|sliding_time|count|
+-----+-----+
|[2022-01-21 17:58:00, 2022-01-21 17:59:00]|18|
|[2022-01-21 17:58:30, 2022-01-21 17:59:30]|6|
+-----+-----+

Batch: 16
-----
+-----+-----+
|sliding_time|count|
+-----+-----+
|[2022-01-21 17:58:00, 2022-01-21 17:59:00]|24|
|[2022-01-21 17:58:30, 2022-01-21 17:59:30]|12|
+-----+-----+

Batch: 17
-----
+-----+-----+
|sliding_time|count|
+-----+-----+
|[2022-01-21 17:58:00, 2022-01-21 17:59:00]|30|
|[2022-01-21 17:58:30, 2022-01-21 17:59:30]|18|
+-----+-----+

stream.stop()
>>> stream.stop()
>>> █
```

```
sliding_iris = waterwarked_sliding_iris.groupBy("sliding_time").count()
```

```
stream = console_output(sliding_iris , 10, "update")
```

```
stream.stop()
```

Наблюдаем только обновляющиеся записи в каждом плавающем окне.

#### ##### Задание 2. Сдвойнить стрим со статикой.

Создадим статический датафрейм, который будет расширять исходный датасет ирисов.

```
static_df_schema = StructType() \

    .add("species", StringType()) \

    .add("description", StringType())

static_df_data = (

    ("setosa", "Iris setosa has a deep violet blue flower. The sepals are deeply-veined dark purple with a yellow-white signal."),

    ("versicolor", "Iris versicolor is a flowering herbaceous perennial plant, growing 10-80 cm high. The well developed blue flower has 6 petals and sepals spread out nearly flat and have two forms."),

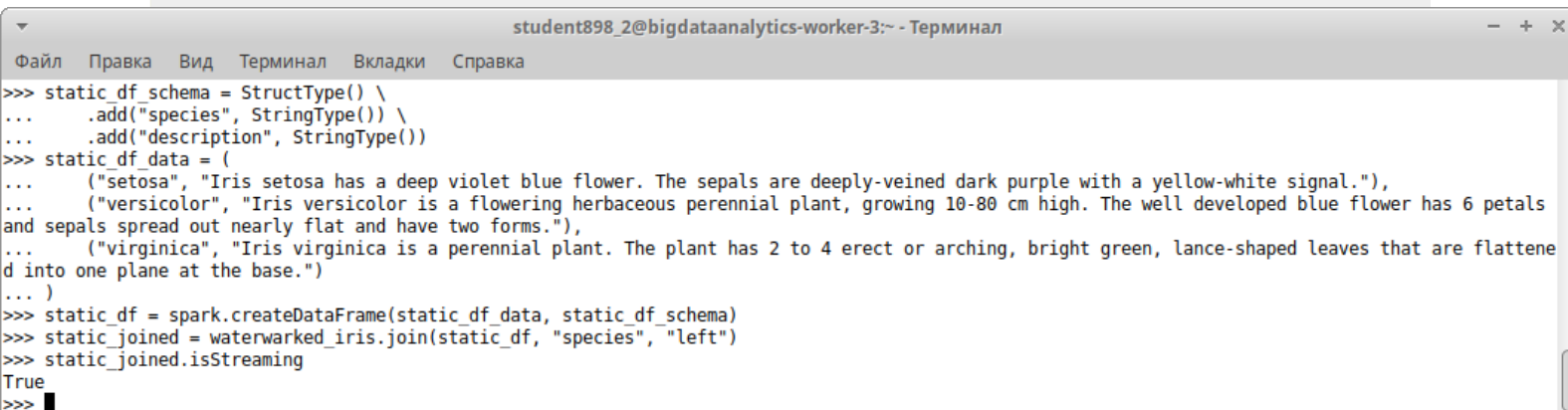
    ("virginica", "Iris virginica is a perennial plant. The plant has 2 to 4 erect or arching, bright green, lance-shaped leaves that are flattened into one plane at the base.")

)

static_df = spark.createDataFrame(static_df_data, static_df_schema)

static_joined = waterwarked_iris.join(static_df, "species", "left")

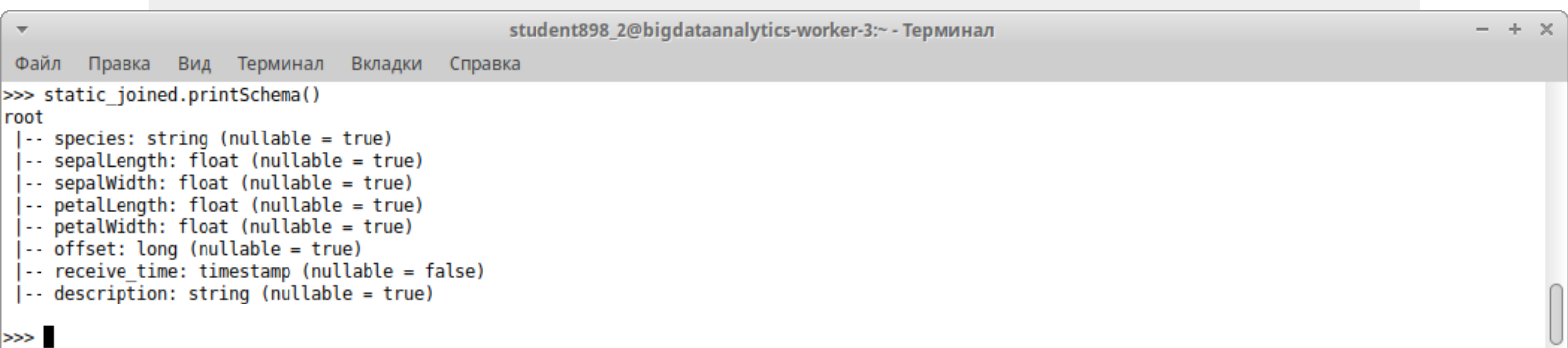
static_joined.isStreaming
```



```
student898_2@bigdataanalytics-worker-3:~ - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
>>> static_df_schema = StructType() \
...     .add("species", StringType()) \
...     .add("description", StringType())
>>> static_df_data = (
...     ("setosa", "Iris setosa has a deep violet blue flower. The sepals are deeply-veined dark purple with a yellow-white signal."),
...     ("versicolor", "Iris versicolor is a flowering herbaceous perennial plant, growing 10-80 cm high. The well developed blue flower has 6 petals and sepals spread out nearly flat and have two forms."),
...     ("virginica", "Iris virginica is a perennial plant. The plant has 2 to 4 erect or arching, bright green, lance-shaped leaves that are flattened into one plane at the base.")
... )
>>> static_df = spark.createDataFrame(static_df_data, static_df_schema)
>>> static_joined = waterwarked_iris.join(static_df, "species", "left")
>>> static_joined.isStreaming
True
>>>
```

После джойна стрима со статикой получаем стрим.

```
static_joined.printSchema()
```

A screenshot of a terminal window titled "student898\_2@bigdataanalytics-worker-3:~ - Терминал". The window has a menu bar with "Файл", "Правка", "Вид", "Терминал", "Вкладки", and "Справка". The terminal shows the command ">>> static\_joined.printSchema()" and its output, which is a schema definition for a DataFrame. The output lists fields: species (string, nullable), sepalLength (float, nullable), sepalWidth (float, nullable), petalLength (float, nullable), petalWidth (float, nullable), offset (long, nullable), receive\_time (timestamp, not nullable), and description (string, nullable). The prompt ">>>" is followed by a cursor.

```
student898_2@bigdataanalytics-worker-3:~ - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
>>> static_joined.printSchema()
root
|-- species: string (nullable = true)
|-- sepalLength: float (nullable = true)
|-- sepalWidth: float (nullable = true)
|-- petalLength: float (nullable = true)
|-- petalWidth: float (nullable = true)
|-- offset: long (nullable = true)
|-- receive_time: timestamp (nullable = false)
|-- description: string (nullable = true)
>>> █
```

Добавлась колонка `description`.

```
stream = console_output(static_joined , 10, "update")
```

```
stream.stop()
```



```

-----+-----
|species |sepalLength|sepalWidth|petalLength|petalWidth|offset|receive_time |description
|-----+-----+-----+-----+-----+-----+-----+
|virginica|7.3      |2.9      |6.3      |1.8      |108    |2022-01-21 18:03:55.443|Iris virginica is a perennial plant. The plant has 2 to 4 erect or arching, bright green, lance-shaped leaves that are flattened into one plane at the base.
|virginica|6.7      |2.5      |5.8      |1.8      |109    |2022-01-21 18:03:55.443|Iris virginica is a perennial plant. The plant has 2 to 4 erect or arching, bright green, lance-shaped leaves that are flattened into one plane at the base.
|virginica|7.2      |3.6      |6.1      |2.5      |110    |2022-01-21 18:03:55.443|Iris virginica is a perennial plant. The plant has 2 to 4 erect or arching, bright green, lance-shaped leaves that are flattened into one plane at the base.
|virginica|6.5      |3.2      |5.1      |2.0      |111    |2022-01-21 18:03:55.443|Iris virginica is a perennial plant. The plant has 2 to 4 erect or arching, bright green, lance-shaped leaves that are flattened into one plane at the base.
|virginica|6.4      |2.7      |5.3      |1.9      |112    |2022-01-21 18:03:55.443|Iris virginica is a perennial plant. The plant has 2 to 4 erect or arching, bright green, lance-shaped leaves that are flattened into one plane at the base.
|virginica|6.8      |3.0      |5.5      |2.1      |113    |2022-01-21 18:03:55.443|Iris virginica is a perennial plant. The plant has 2 to 4 erect or arching, bright green, lance-shaped leaves that are flattened into one plane at the base.
-----+-----

```

Batch: 19

```

>>> -----+-----
|species |sepalLength|sepalWidth|petalLength|petalWidth|offset|receive_time |description
|-----+-----+-----+-----+-----+-----+-----+
|virginica|5.7      |2.5      |5.0      |2.0      |114    |2022-01-21 18:04:00.003|Iris virginica is a perennial plant. The plant has 2 to 4 erect or arching, bright green, lance-shaped leaves that are flattened into one plane at the base.
|virginica|5.8      |2.8      |5.1      |2.4      |115    |2022-01-21 18:04:00.003|Iris virginica is a perennial plant. The plant has 2 to 4 erect or arching, bright green, lance-shaped leaves that are flattened into one plane at the base.
|virginica|6.4      |3.2      |5.3      |2.3      |116    |2022-01-21 18:04:00.003|Iris virginica is a perennial plant. The plant has 2 to 4 erect or arching, bright green, lance-shaped leaves that are flattened into one plane at the base.
|virginica|6.5      |3.0      |5.5      |1.8      |117    |2022-01-21 18:04:00.003|Iris virginica is a perennial plant. The plant has 2 to 4 erect or arching, bright green, lance-shaped leaves that are flattened into one plane at the base.
|virginica|7.7      |3.8      |6.7      |2.2      |118    |2022-01-21 18:04:00.003|Iris virginica is a perennial plant. The plant has 2 to 4 erect or arching, bright green, lance-shaped leaves that are flattened into one plane at the base.
|virginica|7.7      |2.6      |6.9      |2.3      |119    |2022-01-21 18:04:00.003|Iris virginica is a perennial plant. The plant has 2 to 4 erect or arching, bright green, lance-shaped leaves that are flattened into one plane at the base.
-----+-----

```

Batch: 20

```

-----+-----
|species |sepalLength|sepalWidth|petalLength|petalWidth|offset|receive_time |description
|-----+-----+-----+-----+-----+-----+-----+
|virginica|6.0      |2.2      |5.0      |1.5      |120    |2022-01-21 18:04:10.003|Iris virginica is a perennial plant. The plant has 2 to 4 erect or arching, bright green, lance-shaped leaves that are flattened into one plane at the base.
|virginica|6.9      |3.2      |5.7      |2.3      |121    |2022-01-21 18:04:10.003|Iris virginica is a perennial plant. The plant has 2 to 4 erect or arching, bright green, lance-shaped leaves that are flattened into one plane at the base.
|virginica|5.6      |2.8      |4.9      |2.0      |122    |2022-01-21 18:04:10.003|Iris virginica is a perennial plant. The plant has 2 to 4 erect or arching, bright green, lance-shaped leaves that are flattened into one plane at the base.
|virginica|7.7      |2.8      |6.7      |2.0      |123    |2022-01-21 18:04:10.003|Iris virginica is a perennial plant. The plant has 2 to 4 erect or arching, bright green, lance-shaped leaves that are flattened into one plane at the base.
|virginica|6.3      |2.7      |4.9      |1.8      |124    |2022-01-21 18:04:10.003|Iris virginica is a perennial plant. The plant has 2 to 4 erect or arching, bright green, lance-shaped leaves that are flattened into one plane at the base.
|virginica|6.7      |3.3      |5.7      |2.1      |125    |2022-01-21 18:04:10.003|Iris virginica is a perennial plant. The plant has 2 to 4 erect or arching, bright green, lance-shaped leaves that are flattened into one plane at the base.
-----+-----

```

```

stream.stop()
>>> stream.stop()
>>> █

```

### ##### Задание 3. Сдвойнить стрим со стримом.

Это задание сделаем на примере датасетов товаров и заказов.

Датасет, соотносящий товары и заказы читаем из кафки, топик `order\_items`.

```
raw_orders_items = spark.readStream. \
```



```
format("kafka"). \

option("kafka.bootstrap.servers", kafka_brokers). \

option("subscribe", "order_items"). \

option("startingOffsets", "earliest"). \

load()
```

Разбираем value и добавляем окно.

```
schema_orders_items = StructType() \

    .add("order_id", StringType()) \

    .add("order_item_id", StringType()) \

    .add("product_id", StringType()) \

    .add("seller_id", StringType()) \

    .add("shipping_limit_date", StringType()) \

    .add("price", StringType()) \

    .add("freight_value", StringType())

extended_orders_items = raw_orders_items \

    .select(F.from_json(F.col("value").cast("String"),

schema_orders_items).alias("value")) \

    .select("value.*") \

    .withColumn("order_items_receive_time", F.current_timestamp()) \

    .withColumn("window_time", F.window(F.col("order_items_receive_time"), "2

minutes"))

extended_orders_items.printSchema()
```

```
student898_2@bigdataanalytics-worker-3:~ - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
>>> raw_orders_items = spark.readStream. \
...   format("kafka"). \
...   option("kafka.bootstrap.servers", kafka_brokers). \
...   option("subscribe", "order_items"). \
...   option("startingOffsets", "earliest"). \
...   load()
>>> schema_orders_items = StructType() \
...   .add("order_id", StringType()) \
...   .add("order_item_id", StringType()) \
...   .add("product_id", StringType()) \
...   .add("seller_id", StringType()) \
...   .add("shipping_limit_date", StringType()) \
...   .add("price", StringType()) \
...   .add("freight_value", StringType())
>>> extended_orders_items = raw_orders_items \
...   .select(F.from_json(F.col("value").cast("String"), schema_orders_items).alias("value")) \
...   .select("value.*") \
...   .withColumn("order_items_receive_time", F.current_timestamp()) \
...   .withColumn("window_time", F.window(F.col("order_items_receive_time"), "2 minutes"))
>>> extended_orders_items.printSchema()
root
|-- order_id: string (nullable = true)
|-- order_item_id: string (nullable = true)
|-- product_id: string (nullable = true)
|-- seller_id: string (nullable = true)
|-- shipping_limit_date: string (nullable = true)
|-- price: string (nullable = true)
|-- freight_value: string (nullable = true)
|-- order_items_receive_time: timestamp (nullable = false)
|-- window_time: struct (nullable = false)
|   |-- start: timestamp (nullable = true)
|   |-- end: timestamp (nullable = true)
```

Датасет списка заказов читаем из кафки, топик `orders\_json`.

```
raw_orders = spark.readStream. \

    format("kafka"). \

    option("kafka.bootstrap.servers", kafka_brokers). \

    option("subscribe", "orders_json"). \

    option("maxOffsetsPerTrigger", "5"). \

    option("startingOffsets", "earliest"). \

    load()
```

Разбираем value, добавляем колонку со временем получения сообщения, создаём по ней окно и добавляем вотермарку.

```
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())

waterwarked_windowed_orders = raw_orders \

    .select(F.from_json(F.col("value").cast("String"), schema).alias("value"),
"offset") \

    .select("value.order_id", "value.order_status",
"value.order_purchase_timestamp") \

    .withColumn("order_receive_time", F.current_timestamp()) \

    .withColumn("window_time",F.window(F.col("order_receive_time"),"2
minutes")) \

    .withWatermark("window_time", "2 minutes")

waterwarked_windowed_orders.printSchema()

```

```
student898_2@bigdataanalytics-worker-3:~ - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка

>>> raw_orders = spark.readStream. \
...   format("kafka"). \
...   option("kafka.bootstrap.servers", kafka_brokers). \
...   option("subscribe", "orders_json"). \
...   option("maxOffsetsPerTrigger", "5"). \
...   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())
>>> watermarked_windowed_orders = raw_orders \
...   .select(F.from_json(F.col("value").cast("String"), schema).alias("value"), "offset") \
...   .select("value.order_id", "value.order_status", "value.order_purchase_timestamp") \
...   .withColumn("order_receive_time", F.current_timestamp()) \
...   .withColumn("window_time", F.window(F.col("order_receive_time"), "2 minutes")) \
...   .withWatermark("window_time", "2 minutes")
>>> watermarked_windowed_orders.printSchema()
root
|-- order_id: string (nullable = true)
|-- order_status: string (nullable = true)
|-- order_purchase_timestamp: string (nullable = true)
|-- order_receive_time: timestamp (nullable = false)
|-- window_time: struct (nullable = false)
|   |-- start: timestamp (nullable = true)
|   |-- end: timestamp (nullable = true)
>>> █
```

Делаем джойн двух датасетов.

```
streams_joined = watermarked_windowed_orders \

    .join(extended_orders_items, ["order_id", "window_time"] , "inner") \

    .select("order_id", "order_item_id", "product_id", "window_time")
```

Тип отображения `update` не подходит для `inner` джойна.

Очищаем чекпоинты

```
stream = console_output(streams_joined , 10, "append")

stream.stop()
```

```
student898_2@bigdataanalytics-worker-3:~ - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
-----
Batch: 0
-----
+-----+-----+-----+-----+
|order_id|order_item_id|product_id|window_time|
+-----+-----+-----+-----+
|e481f51cbdc54678b7cc49136f2d6af7|1|87285b34884572647811a353c7ac498a|[2022-01-21 18:12:00, 2022-01-21 18:14:00]|
|order_id|order_item_id|product_id|window_time|
+-----+-----+-----+-----+
|47770eb9100c2d0c44946d9cf07ec65d|1|aa4383b373c6aca5d8797843e5594415|[2022-01-21 18:12:00, 2022-01-21 18:14:00]|
|53cdb2fc8bc7dce0b6741e2150273451|1|595fac2a385ac33a80bd5114aec74eb8|[2022-01-21 18:12:00, 2022-01-21 18:14:00]|
|949d5b44dbf5de918fe9c16f97b45f8a|1|d0b61bfb1de832b15ba9d266ca96e5b0|[2022-01-21 18:12:00, 2022-01-21 18:14:00]|
+-----+-----+-----+-----+

22/01/21 18:13:20 WARN streaming.ProcessingTimeExecutor: Current batch is falling behind. The trigger interval is 10000 milliseconds, but spent 26945 milliseconds
-----
Batch: 1
-----
+-----+-----+-----+-----+
|order_id|order_item_id|product_id|window_time|
+-----+-----+-----+-----+
|76c6e866289321a7c93b82b54852dc33|1|ac1789e492dcd698c5c10b97a671243a|[2022-01-21 18:12:00, 2022-01-21 18:14:00]|
|ad21c59c0840e6cb83a9ceb5573f8159|1|65266b2da20d04dbe00c5c2d3bb7859e|[2022-01-21 18:12:00, 2022-01-21 18:14:00]|
|a4591c265e18cb1dcee52889e2d8acc3|1|060cb19345d90064d1015407193c233d|[2022-01-21 18:12:00, 2022-01-21 18:14:00]|
|136cce7faa42fdb2cefd53fdc79a6098|1|a1804276d9941ac0733cfd409f5206eb|[2022-01-21 18:12:00, 2022-01-21 18:14:00]|
|6514b8ad8028c9f2cc2374ded245783f|1|4520766ec412348b8d4caa5e8a18c464|[2022-01-21 18:12:00, 2022-01-21 18:14:00]|
+-----+-----+-----+-----+

22/01/21 18:13:33 WARN streaming.ProcessingTimeExecutor: Current batch is falling behind. The trigger interval is 10000 milliseconds, but spent 12799 milliseconds
-----
Batch: 2
-----
+-----+-----+-----+-----+
|order_id|order_item_id|product_id|window_time|
+-----+-----+-----+-----+
|82566a660a982b15fb86e904c8d32918|1|72a97c271b2e429974398f46b93ae530|[2022-01-21 18:12:00, 2022-01-21 18:14:00]|
|e69bfb5eb88e0ed6a785585b27e16dbf|1|9a78fb9862b10749a117f7fc3c31f051|[2022-01-21 18:12:00, 2022-01-21 18:14:00]|
|34513ce0c4fab462a55830c0989c7edb|1|f7e0fa615b386bc9a8b9eb52bc1fff76|[2022-01-21 18:12:00, 2022-01-21 18:14:00]|
|e6ce16cb79ec1d90b1da9085a6118aeb|1|08574b074924071f4e201e151b152b4e|[2022-01-21 18:12:00, 2022-01-21 18:14:00]|
|e6ce16cb79ec1d90b1da9085a6118aeb|2|08574b074924071f4e201e151b152b4e|[2022-01-21 18:12:00, 2022-01-21 18:14:00]|
|5ff96c15d0b717ac6ad1f3d77225a350|1|10adb53d8faa890ca7c2f0cbcb68d777|[2022-01-21 18:12:00, 2022-01-21 18:14:00]|
+-----+-----+-----+-----+
```

Здесь не увидел результата, так как в топике `order\_items` не было данных. По факту этот топик вычитывается целиком за раз, поэтому в первом окне можно наблюдать микробатчи сдвоенного датасета. Для остальных окон микробатчи пустые, так как `window\_time` уже различаются. Из топика `order\_items` новые данные не приходят.

Написать скрипт на python для конвертации файла csv в json.

```
import csv

import json

with open('test.csv') as f:

    reader = csv.DictReader(f)

    rows = list(reader)
```

```
with open('test.json', 'w') as f:  
    json.dump(rows, f)
```