dsti-a19/ref/lab3

FINISHED

Link to resources

Documentation of Python script: https://github.com/adaltas/ece-spark/tree/master/structured-streaming (https://github.com/adaltas/ece-spark/tree/master/structured-streaming)
Spark Streaming slides link: http://bit.ly/2DjKjxL (http://bit.ly/2DjKjxL)

Took 0 sec. Last updated by gauthier at March 18 2020, 1:25:18 PM.

Steps to launch the socket: FINISHED

1. SSH to the edge server
2. Get the code from HDFS

```
hdfs dfs -get /learning/code/spark .
```

3. Cd to the code folder: `cd spark/structured-streaming`
4. Run the Python script (choose an unused port number):

```
python3 stream_taxi_data.py edge-1.au.adaltas.cloud 11111 fares
```

Took 0 sec. Last updated by gauthier at March 18 2020, 4:23:30 PM.

FINISHED

```
%pyspark
# Create a socket readStream
fares_raw = spark \
    .readStream \
    .format("socket") \
    .option("host", "edge-1.au.adaltas.cloud") \
    .option("port", 11333) \
    .load()
```

Took 0 sec. Last updated by gauthier at March 18 2020, 2:03:17 PM.

FINISHED

```
%pyspark
from pyspark.sql.functions import explode
from pyspark.sql.functions import split
from pyspark.sql.functions import window

# Parse the socket message "manually"
fares = fares_raw.select(
    split(fares_raw.value, ',')[0].alias('ride_id').cast('int'),
    split(fares_raw.value, ',')[1].alias('taxi_id').cast('int'),
    split(fares_raw.value, ',')[2].alias('driver_id').cast('int'),
    split(fares_raw.value, ',')[3].alias('start_time').cast('timestamp'),
    split(fares_raw.value, ',')[4].alias('payment_type'),
    split(fares_raw.value, ',')[5].alias('tip').cast('float'),
    split(fares_raw.value, ',')[6].alias('tolls').cast('float'),
    split(fares_raw.value, ',')[7].alias('total_fare').cast('float')
)
```

Took 0 sec. Last updated by gauthier at March 18 2020, 2:03:19 PM.

FINISHED

```
%pyspark
# Write all fares events to an in-memory table named "fares"
fares_query = fares \
    .writeStream \
    .outputMode("append") \
    .format("memory") \
    .queryName("fares") \
    .start()
```

Took 0 sec. Last updated by gauthier at March 18 2020, 1:59:51 PM.

FINISHED

```
%pyspark
# Pretty print the result table
z.show(spark.table("fares"))
```

settings

ride_id	taxi_id	driver_id	start_time	payment_type	tip
1	2013000001	2013000001	2020-03-18 12:43:39.0	CSH	0
2	2013000002	2013000002	2020-03-18 12:43:39.0	CSH	0
3	2013000003	2013000003	2020-03-18 12:43:39.0	CRD	2.2
4	2013000004	2013000004	2020-03-18 12:43:39.0	CRD	1.7
5	2013000005	2013000005	2020-03-18 12:43:39.0	CRD	4.65
6	2013000006	2013000006	2020-03-18 12:43:39.0	CSH	0

7	2013000007	2013000007	2020-03-18 12:43:39.0	CRD	1.9
8	2013000008	2013000008	2020-03-18 12:43:39.0	CSH	0

dsti-a19/ref/lab3

Output is truncated to 1000 rows. Learn more about **zeppelin.spark.maxResult**

Took 0 sec. Last updated by gauthier at March 18 2020, 1:49:23 PM. (outdated)

```
%pyspark
spark.table("fares").count()
```

FINISHED

```
209
```

Took 0 sec. Last updated by gauthier at March 18 2020, 2:01:12 PM.

```
%pyspark
fares_query.stop()
```

FINISHED

Took 0 sec. Last updated by gauthier at March 18 2020, 2:02:25 PM.

```
%pyspark
from pyspark.sql.functions import explode
from pyspark.sql.functions import split
from pyspark.sql.functions import window

# Create a socket readStream
fares_raw = spark \
    .readStream \
    .format("socket") \
    .option("host", "edge-1.au.adaltas.cloud") \
    .option("port", 11333) \
    .load()

# Parse the socket message "manually"
fares = fares_raw.select(
    split(fares_raw.value, ',')[0].alias('ride_id').cast('int'),
    split(fares_raw.value, ',')[1].alias('taxi_id').cast('int'),
    split(fares_raw.value, ',')[2].alias('driver_id').cast('int'),
    split(fares_raw.value, ',')[3].alias('start_time').cast('timestamp'),
    split(fares_raw.value, ',')[4].alias('payment_type'),
    split(fares_raw.value, ',')[5].alias('tip').cast('float'),
    split(fares_raw.value, ',')[6].alias('tolls').cast('float'),
    split(fares_raw.value, ',')[7].alias('total_fare').cast('float')
)
```

FINISHED

Took 0 sec. Last updated by gauthier at March 18 2020, 2:34:13 PM.

```
%pyspark
# Compute the number of rides and the average total fare
fares_count = fares \
    .withWatermark('start_time', '5 minutes') \
    .groupBy(window(fares.start_time, '2 minutes', '2 minutes')) \
    .agg({'ride_id': 'count', 'total_fare': 'mean'})

# Start writting the stream to an in-memory table
fares_count_query = fares_count \
    .writeStream \
    .outputMode("complete") \
    .format("memory") \
    .queryName("fares_count") \
    .start()
```

FINISHED

Took 0 sec. Last updated by gauthier at March 18 2020, 2:25:53 PM.

```
%pyspark
z.show(spark.table("fares_count").orderBy('window'))
```

SPARK JOB (http://wrk-1.au.adaltas.cloud:39901/jobs/job?id=86) FINISHED

settings

window	count(ride_id)	avg(to
[2020-03-18 13:24:00.0,2020-03-18 13:26:00.0]	72	14.2549
[2020-03-18 13:26:00.0,2020-03-18 13:28:00.0]	6	13.8833

Took 2 sec. Last updated by gauthier at March 18 2020, 2:26:12 PM.

dsll-a19/ref/lab3

```
%pyspark
rides_count_query.stop()
```

FINISHED

Took 1 sec. Last updated by gauthier at March 18 2020, 2:26:20 PM.

```
%pyspark
rides_raw = spark \
    .readStream \
    .format("socket") \
    .option("host", "edge-1.au.adaltas.cloud") \
    .option("port", 11666) \
    .load()
```

FINISHED

Took 0 sec. Last updated by gauthier at March 18 2020, 2:34:25 PM.

```
%pyspark
rides = rides_raw.select(
    split(rides_raw.value, ',')[0].alias('ride_id').cast('int'),
    split(rides_raw.value, ',')[1].alias('is_start'),
    split(rides_raw.value, ',')[2].alias('start_time').cast('timestamp'),
    split(rides_raw.value, ',')[3].alias('end_time').cast('timestamp'),
    split(rides_raw.value, ',')[4].alias('start_lon').cast('float'),
    split(rides_raw.value, ',')[5].alias('start_lat').cast('float'),
    split(rides_raw.value, ',')[6].alias('end_lon').cast('float'),
    split(rides_raw.value, ',')[7].alias('end_lat').cast('float'),
    split(rides_raw.value, ',')[7].alias('passenger_count').cast('int'),
    split(rides_raw.value, ',')[8].alias('taxi_id').cast('int'),
    split(rides_raw.value, ',')[9].alias('driver_id').cast('int')
)

# Filter the rides dataset to keep only "END" events
rides_end = rides.where(rides.is_start == 'END')
```

FINISHED

Took 0 sec. Last updated by gauthier at March 18 2020, 2:34:30 PM.

```
%pyspark
rides_query = rides_end \
    .writeStream \
    .outputMode("append") \
    .format("memory") \
    .queryName("rides") \
    .start()
```

READY

```
%pyspark
rides_query.stop()
```

READY

```
%pyspark
rides_with_watermark = rides_end.withWatermark('start_time', '1 hour')
fares_with_watermark = fares.withWatermark('start_time', '1 hour')

full_rides = rides_with_watermark \
    .join(
        fares_with_watermark,
        rides_with_watermark.ride_id == fares_with_watermark.ride_id
    )
```

FINISHED

Took 0 sec. Last updated by gauthier at March 18 2020, 2:38:24 PM.

```
%pyspark
full_rides_query = full_rides \
    .writeStream \
    .outputMode("append") \
    .format("memory") \
    .queryName("full_rides") \
    .start()
```

FINISHED

Took 0 sec. Last updated by gauthier at March 18 2020, 2:38:26 PM.

```
%pyspark
z.show(spark.table("full_rides"))
```

FINISHED

settings

ride_id	is_start	start_time	end_time	start_lon	start_lat	end_lon
65	END	2020-03-18 13:37:48.0	2020-03-18 13:37:50.0	-73.99221	40.725124	-73.991646
137	END	2020-03-18 13:38:39.0	2020-03-18 13:38:41.0	0	0	0
77	END	2020-03-18 13:37:59.0	2020-03-18 13:38:01.0	-73.9701	40.768005	-73.96977
94	END	2020-03-18 13:38:19.0	2020-03-18 13:38:21.0	-74.005165	40.72053	-74.00393

70	END	2020-03-18	2020-03-18	-73.97544	40.749657	-73.97733

dsti-a19/ref/lab3

Took 0 sec. Last updated by gauthier at March 18 2020, 2:40:37 PM.

FINISHED

```
%pyspark
full_rides_query.stop()
```

Took 1 sec. Last updated by gauthier at March 18 2020, 2:41:18 PM.

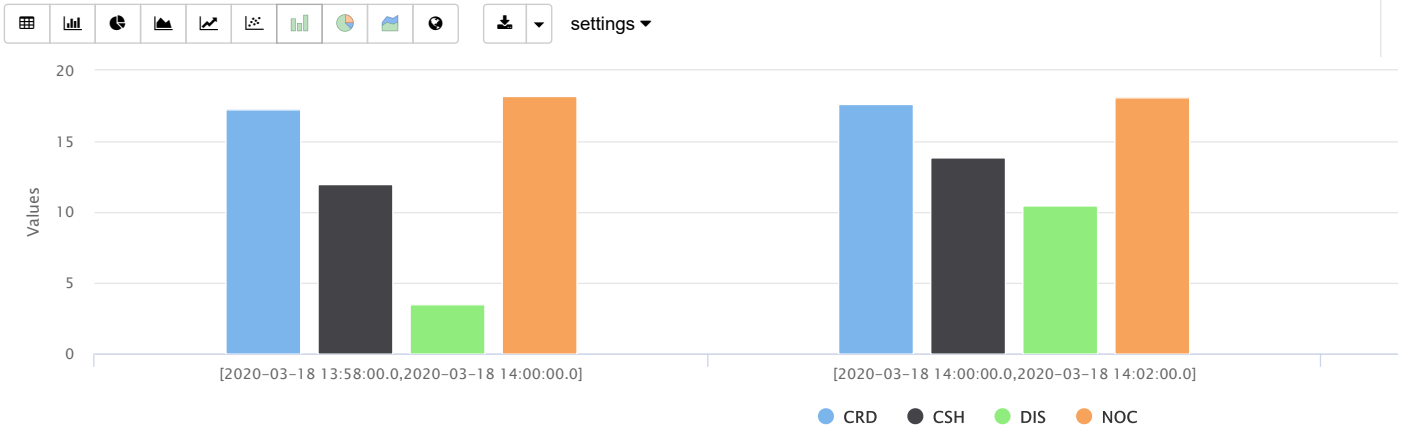
FINISHED

```
%pyspark
# Expected lab code for 1 KPI
from pyspark.sql.functions import explode
from pyspark.sql.functions import split
from pyspark.sql.functions import window

# Create a socket readStream
fares_raw = spark \
    .readStream \
    .format("socket") \
    .option("host", "edge-1.au.adaltas.cloud") \
    .option("port", 11333) \
    .load()


# Parse the socket message "manually"
fares = fares_raw.select(
    split(fares_raw.value, ',')[0].alias('ride_id').cast('int'),
    split(fares_raw.value, ',')[1].alias('taxi_id').cast('int'),
    split(fares_raw.value, ',')[2].alias('driver_id').cast('int'),
    split(fares_raw.value, ',')[3].alias('start_time').cast('timestamp'),
    split(fares_raw.value, ',')[4].alias('payment_type'),
    split(fares_raw.value, ',')[5].alias('tip').cast('float'),
    split(fares_raw.value, ',')[6].alias('tolls').cast('float'),
    split(fares_raw.value, ',')[7].alias('total_fare').cast('float')
)

fares_count = fares \
    .withWatermark('start_time', '5 minutes') \
    .groupBy(window(fares.start_time, '2 minutes', '2 minutes'), fares.payment_type) \
    .agg({'ride_id': 'count', 'total_fare': 'mean', 'tip': 'mean'})

# Start writting the stream to an in-memory table
fares_count_query = fares_count \
    .writeStream \
    .trigger(processingTime='30 seconds') \
    .outputMode("complete") \
    .format("memory") \
    .queryName("fares_count") \
    .start()
```

Took 0 sec. Last updated by gauthier at March 18 2020, 2:58:05 PM.

FINISHED

```
%pyspark
z.show(spark.table("fares_count"))
```



Took 0 sec. Last updated by gauthier at March 18 2020, 3:02:43 PM.

FINISHED

```
%pyspark
fares_count_query.stop()
```

Took 0 sec. Last updated by gauthier at March 18 2020, 3:02:38 PM.

FINISHED

Homework

Display using a graph:

1. The percentage for each type of payment per window
2. The mean ride duration

Took 0 sec. Last updated by gautier at March 18 2020, 2:52:56 PM.

dsti-a19/ref/lab3

```
%md
```

READY