

DR. ALVIN'S PUBLICATIONS

SPARK STREAMING

DR. ALVIN ANG



CONTENTS

I. Structured Streaming	3
A. In a Nutshell, How it Works... ..	3
B. Import Libraries	6
C. Start a Spark Session	6
D. Simulating Streaming Data	6
E. Creating a Schema	7
F. ReadStream	8
G. WriteStream	8
H. SQL.....	10
I. Continuing.... 2nd Dataframe Streamed In.... ..	10
J. SQL again.... ..	11
About Dr. Alvin Ang	12

I. STRUCTURED STREAMING

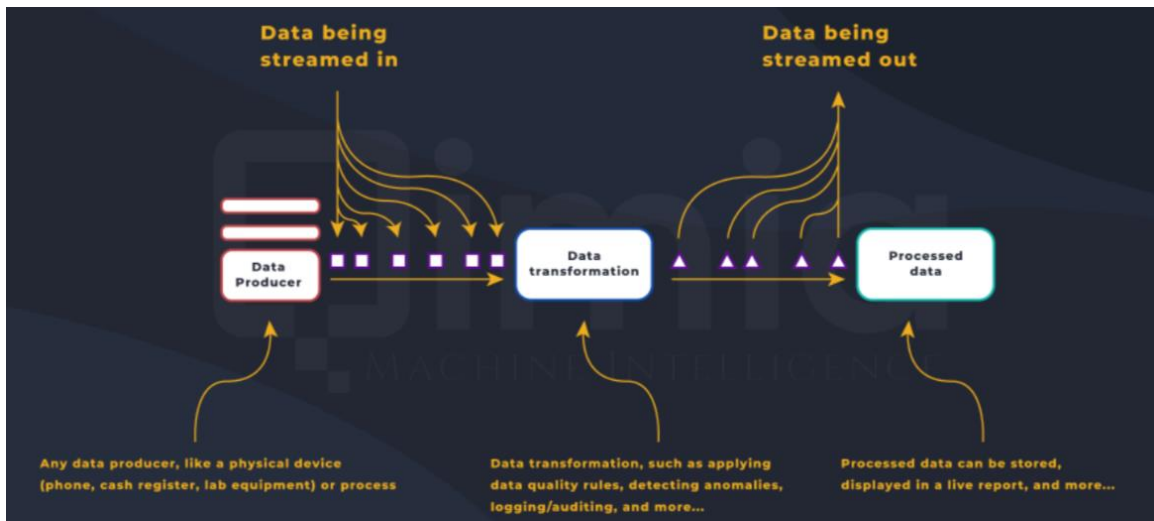
Most of the stuff are referenced from here:

<https://www.amazon.com/Learn-PySpark-Python-based-Machine-Learning/dp/1484249607>

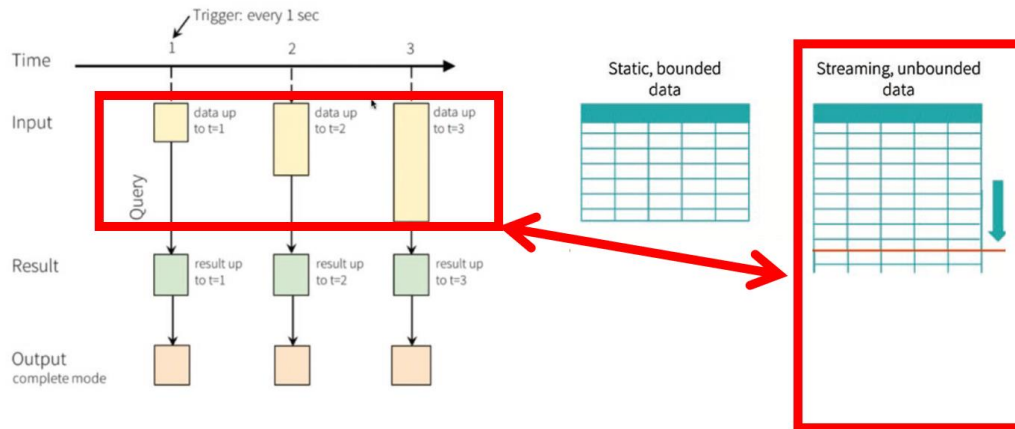
IPYNB

[https://www.alvinang.sg/s/Structured Streaming with PySpark.ipynb](https://www.alvinang.sg/s/Structured%20Streaming%20with%20PySpark.ipynb)

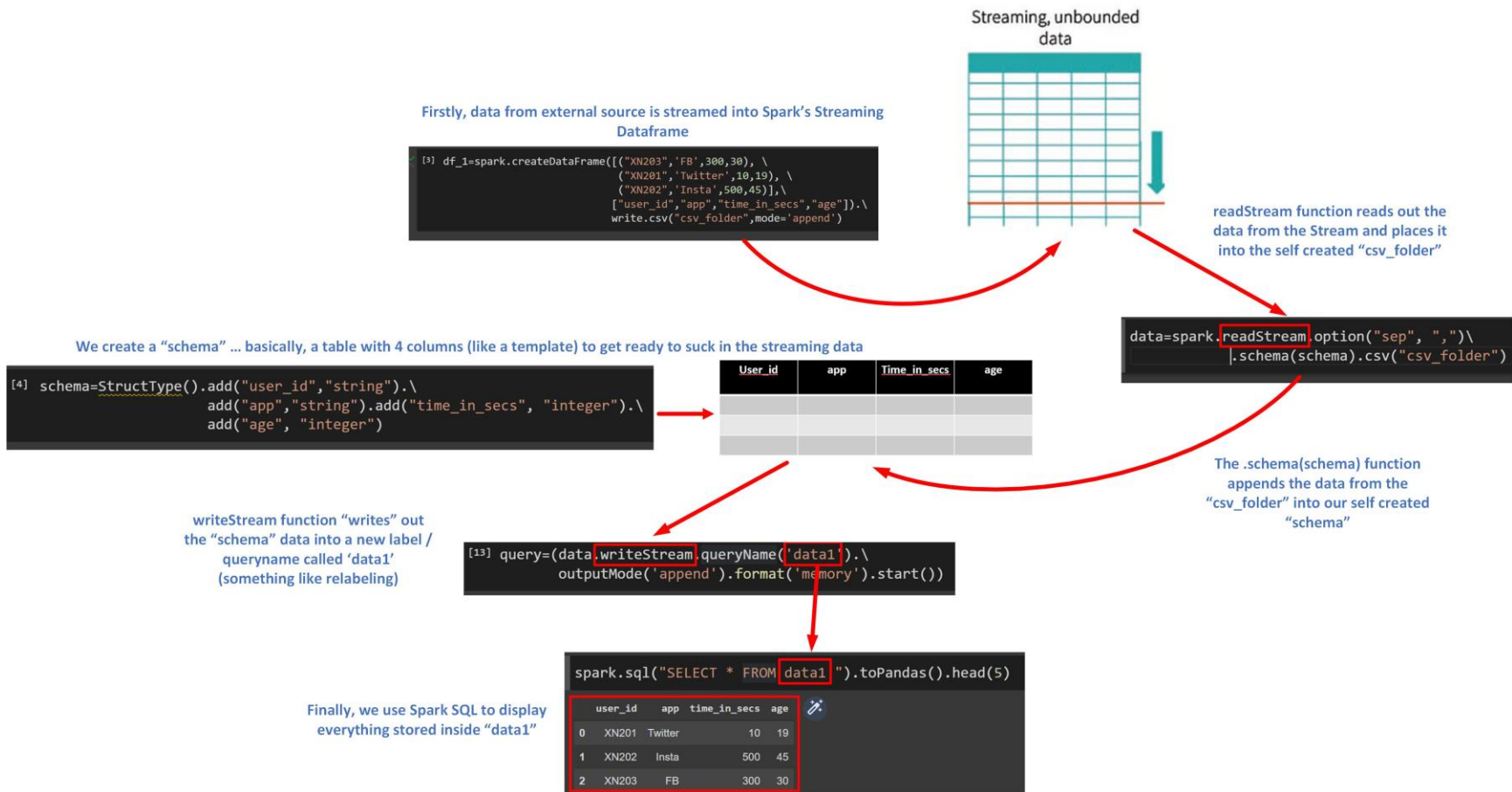
A. IN A NUTSHELL, HOW IT WORKS...



<https://qimia.io/en/blog/Developing-Streaming-Applications-Spark-Structured-Streaming>



- At $t = 1$, data streams in and is stored in the Spark Dataframe.
- We may query it and extract it out to store it someplace else.
- At $t = 2$, this process continues. Data is constantly appended into the initial Dataframe.
- We may query it every second if we want, but each time we query, the dataframe is constantly expanding.
- There are 3 main areas (refer to figure above):
 - INPUT \rightarrow where data is appended to the dataframe
 - QUERY \rightarrow .readStream; where the streaming data is extracted using SQL
 - OUTPUT \rightarrow .writeStream; where the data is stored in another dataframe someplace else



B. IMPORT LIBRARIES

Import Libraries

```
▶ from pyspark.sql import SparkSession
   spark=SparkSession.builder.appName('structured_streaming').getOrCreate()

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

C. START A SPARK SESSION

- Please refer here: <https://www.alvinang.sg/s/Installing-Spark-on-Colab-by-Dr-Alvin-Ang.pdf>

D. SIMULATING STREAMING DATA

Simulating INPUT

```
▶ df_1=spark.createDataFrame([("XN203", 'FB', 300, 30), \
                              ("XN201", 'Twitter', 10, 19), \
                              ("XN202", 'Insta', 500, 45)], \
                              ["user_id", "app", "time_in_secs", "age"]).\
   write.csv("csv_folder", mode='append')
```

once the dataframe is created, colab creates the "csv_folder" as instructed....and somehow breaks it down into 2 CSVs...
double click it to view th CSV...

- Since we are just doing simulation, we pretend that the data is streamed in through “creating a dataframe”
- By right, df_1 should contain the table as below, but note that streaming data is unlike Pandas dataframe (you can’t just show the dataframe like that, you need to use readStream and writeStream).

	user_id	app	time_in_secs	age
0	XN201	Twitter	10	19
1	XN202	Insta	500	45
2	XN203	FB	300	30

E. CREATING A SCHEMA

```

Creating a Schema, getting ready to store the data

[4] schema=StructType().add("user_id","string").\
    add("app","string").add("time_in_secs", "integer").\
    add("age", "integer")

```

F. READSTREAM

QUERY: readStream to read in data from the Data Frame

```
[5] data=spark.readStream.option("sep", ",")\
    .schema(schema).csv("csv_folder")
```

- readStream grabs the data out of the “csv_folder” and stores it into the “schema”

```
✓ [6] data
0s
DataFrame[user_id: string, app: string, time_in_secs: int, age: int]

✓ data.printSchema()
0s
root
 |-- user_id: string (nullable = true)
 |-- app: string (nullable = true)
 |-- time_in_secs: integer (nullable = true)
 |-- age: integer (nullable = true)
```

G. WRITESTREAM

```
✓ [13] query=(data.writeStream.queryName('data1').\
0s      outputMode('append').format('memory').start())
there are a few output
modes.....

✓ [14] query
0s
'append' 'complete'

<pyspark.sql.streaming.StreamingQuery at 0x7fc160883c90>
```



```
query=(data.writeStream.queryName('data1').\
outputMode('complete').format('memory').start())
```

```
-----  
AnalysisException                               Traceback (most recent call last)  
<ipython-input-16-877534bc43ea> in <module>()  
----> 1 query=(data.writeStream.queryName('data1').          outputMode('complete').format('memory').start())  
  
      2 frames  
-----  
/content/spark-3.2.1-bin-hadoop3.2/python_pyspark/sql/utils.py in deco(*a, **kw)  
    115         # Hide where the exception came from that shows a non-Pythonic  
    116         # JVM exception message.  
--> 117         raise converted from None  
    118     else:  
    119         raise
```

```
AnalysisException: Complete output mode not supported when there are no streaming aggregations on streaming  
DataFrames/Datasets;  
FileSource[csv_folder]
```

- Can you see the difference in the above 2 pictures?
- One is using “append”, while the other is using “complete” output mode.
- As can be seen, the “complete” mode gives an error.
- To give a brief explanation, is because “complete” mode only allows the data to be “written out” IF you have performed AGGREGATIONS on it...
- It doesn't allow you to “extract out” the entire “streamed data”...
- Below gives an exmple using “complete” mode....

▼ Demonstrating Output Mode "complete"using Aggregations...

```
[13] app_count=data.groupby('app').count()
```

```
[14] query=(app_count.writeStream.queryName('count_query').\
outputMode('complete').format('memory').start())
```

```
spark.sql("select * from count_query ").toPandas().head(5)
```

	app	count
0	Insta	1
1	FB	1
2	Twitter	1

as can be seen, you can only use "complete" mode if you do Aggregation i.e. Groupby....

- We shall only explain “append” in this manuscript (to keep things simple).

H. SQL

```
spark.sql("SELECT * FROM data1 ").toPandas().head(5)
```

	user_id	app	time_in_secs	age
0	XN201	Twitter	10	19
1	XN202	Insta	500	45
2	XN203	FB	300	30

- Finally, we get the very first streamed table!
- We have to use Spark SQL to query and display the data out of data1.

I. CONTINUING.... 2ND DATAFRAME STREAMED IN....

```
df_2=spark.createDataFrame([("XN206", 'BABA', 50, 80), \
                             ("XN207", 'BOOBOO', 88, 44), \
                             ("XN208", 'GUNDOO', 222, 555)], \
                             ["user_id", "app", "time_in_secs", "age"]).\
write.csv("csv_folder", mode='append')
```

- We simulate another new data streaming in.. this time is df2....

J. SQL AGAIN....

```
spark.sql("SELECT * FROM data1 ").toPandas().head(10)
```

	user_id	app	time_in_secs	age
0	XN202	Twitter	10	19
1	XN203	Insta	500	45
2	XN201	FB	300	30
3	XN207	BOoboo	88	44
4	XN208	GUNDOO	222	555
5	XN206	BABA	50	80

3 new rows from the streamed data has now been appended to data1

- This time, when you run the exact SQL code again, you realized that df2 has been appended on top of df1.....

ABOUT DR. ALVIN ANG



Dr. Alvin Ang earned his Ph.D., Masters and Bachelor degrees from NTU, Singapore. He is a scientist, entrepreneur, as well as a personal/business advisor. More about him at www.AlvinAng.sg.