In [1]:
```
from pyspark.sql import SparkSession
import pyspark.sql.functions as F
from pyspark.sql.types import *

spark = SparkSession\
    .builder\
    .appName("chapter-21-stream")\
    .getOrCreate()

import os
SPARK_BOOK_DATA_PATH = os.environ['SPARK_BOOK_DATA_PATH']
```

In [2]:
```
static = spark.read.json(SPARK_BOOK_DATA_PATH + "/data/activity-data/")
dataSchema = static.schema
```

In [3]:
```
# COMMAND ----------

streaming = spark.readStream.schema(dataSchema).option("maxFilesPerTrig
  .json(SPARK_BOOK_DATA_PATH + "/data/activity-data")
```

In [4]:
```
# COMMAND ----------

activityCounts = streaming.groupBy("gt").count()
```

In [5]:
```
# COMMAND ----------

activityQuery = activityCounts.writeStream.queryName("activity_counts")
  .format("memory").outputMode("complete")\
  .start()
```

In [6]:
```
# COMMAND ----------

from time import sleep
for x in range(5):
    spark.sql("SELECT * FROM activity_counts").show()
    sleep(1)
```

```
+----------+-----+
|        gt|count|
+----------+-----+
|  stairsup|10452|
|       sit|12309|
|     stand|11385|
|      walk|13256|
|      bike|10797|
|stairsdown| 9365|
|      null|10448|
+----------+-----+

+----------+-----+
|        gt|count|
+----------+-----+
|  stairsup|20905|
|       sit|24619|
|     stand|22770|
|      walk|26512|
|      bike|21594|
|stairsdown|18729|
|      null|20895|
+----------+-----+

+----------+-----+
|        gt|count|
+----------+-----+
|  stairsup|31357|
|       sit|36929|
|     stand|34155|
|      walk|39768|
|      bike|32391|
|stairsdown|28094|
|      null|31342|
+----------+-----+

+----------+-----+
|        gt|count|
+----------+-----+
|  stairsup|41809|
|       sit|49238|
|     stand|45539|
|      walk|53024|
|      bike|43187|
|stairsdown|37459|
|      null|41791|
+----------+-----+

+----------+-----+
|        gt|count|
```

```
+----------+-----+
|  stairsup|52262|
|       sit|61545|
|     stand|56924|
|      walk|66280|
|      bike|53985|
|stairsdown|46823|
|      null|52240|
+----------+-----+
```

In [7]:
```
# COMMAND ----------

from time import sleep
for x in range(5):
    spark.sql("SELECT * FROM activity_counts").show()
    sleep(1)
```

```
+----------+------+
|        gt| count|
+----------+------+
|  stairsup|177717|
|       sit|209235|
|     stand|193549|
|      walk|225352|
|      bike|183560|
|stairsdown|159179|
|      null|177612|
+----------+------+

+----------+------+
|        gt| count|
+----------+------+
|  stairsup|188178|
|       sit|221543|
|     stand|204933|
|      walk|238608|
|      bike|194357|
|stairsdown|168539|
|      null|188057|
+----------+------+

+----------+------+
|        gt| count|
+----------+------+
|  stairsup|198636|
|       sit|233851|
|     stand|216319|
|      walk|251864|
|      bike|205154|
|stairsdown|177899|
|      null|198503|
+----------+------+

+----------+------+
|        gt| count|
+----------+------+
|  stairsup|209097|
|       sit|246159|
|     stand|227703|
|      walk|265120|
|      bike|215951|
|stairsdown|187259|
|      null|208949|
+----------+------+

+----------+------+
|        gt| count|
```

```
+----------+------+
|  stairsup|219558|
|       sit|258467|
|     stand|239087|
|      walk|278376|
|      bike|226748|
|stairsdown|196618|
|      null|219395|
+----------+------+
```

In [9]:
```
spark.streams.active
```

Out[9]:
```
[<pyspark.sql.streaming.StreamingQuery at 0x7f25f3acdc88>]
```

In [ ]:
```
# COMMAND ----------

from pyspark.sql.functions import expr
simpleTransform = streaming.withColumn("stairs", expr("gt like '%stairs'
  .where("stairs")\
  .where("gt is not null")\
  .select("gt", "model", "arrival_time", "creation_time")\
  .writeStream\
  .queryName("simple_transform")\
  .format("memory")\
  .outputMode("append")\
  .start()
```

In [ ]:
```
# COMMAND ----------

deviceModelStats = streaming.cube("gt", "model").avg()\
  .drop("avg(Arrival_time)")\
  .drop("avg(Creation_Time)")\
  .drop("avg(Index)")\
  .writeStream.queryName("device_counts")\
  .format("memory")\
  .outputMode("complete")\
  .start()
```

In [ ]:
```
# COMMAND ----------

historicalAgg = static.groupBy("gt", "model").avg()
deviceModelStats = streaming.drop("Arrival_Time", "Creation_Time", "Inde
  .cube("gt", "model").avg()\
  .join(historicalAgg, ["gt", "model"])\
  .writeStream.queryName("device_counts")\
  .format("memory")\
  .outputMode("complete")\
  .start()


# COMMAND ----------
```

In [ ]:
```
# Subscribe to 1 topic
df1 = spark.readStream.format("kafka")\
  .option("kafka.bootstrap.servers", "host1:port1,host2:port2")\
  .option("subscribe", "topic1")\
  .load()
# Subscribe to multiple topics
df2 = spark.readStream.format("kafka")\
  .option("kafka.bootstrap.servers", "host1:port1,host2:port2")\
  .option("subscribe", "topic1,topic2")\
  .load()
# Subscribe to a pattern
df3 = spark.readStream.format("kafka")\
  .option("kafka.bootstrap.servers", "host1:port1,host2:port2")\
  .option("subscribePattern", "topic.*")\
  .load()


# COMMAND ----------

df1.selectExpr("topic", "CAST(key AS STRING)", "CAST(value AS STRING)")\
  .writeStream\
  .format("kafka")\
  .option("kafka.bootstrap.servers", "host1:port1,host2:port2")\
  .option("checkpointLocation", "/to/HDFS-compatible/dir")\
  .start()
df1.selectExpr("CAST(key AS STRING)", "CAST(value AS STRING)")\
  .writeStream\
  .format("kafka")\
  .option("kafka.bootstrap.servers", "host1:port1,host2:port2")\
  .option("checkpointLocation", "/to/HDFS-compatible/dir")\
  .option("topic", "topic1")\
  .start()


# COMMAND ----------

socketDF = spark.readStream.format("socket")\
  .option("host", "localhost").option("port", 9999).load()


# COMMAND ----------

activityCounts.writeStream.trigger(processingTime='5 seconds')\
  .format("console").outputMode("complete").start()


# COMMAND ----------

activityCounts.writeStream.trigger(once=True)\
  .format("console").outputMode("complete").start()


# COMMAND ----------
```