In [2]:

```
import os, io, fastavro
from pyspark.sql import SparkSession, Row
from pyspark.sql import Row
from pyspark.sql.functions import *
from pyspark.sql.types import *
from pyspark.sql.window import Window
from pyspark.sql.avro.functions import from_avro, to_avro
import pyspark.sql.functions as psf


spark = SparkSession.builder.appName("Avro")\
            .config("spark.jars.packages", "org.apache.spark:spark-sql-kafka-0-10_2.
            .getOrCreate()
spark
```

```
/Users/aakash10975/spark3/python/pyspark/context.py:227: DeprecationWa
rning: Support for Python 2 and Python 3 prior to version 3.6 is depre
cated as of Spark 3.0. See also the plan for dropping Python 2 support
at https://spark.apache.org/news/plan-for-dropping-python-2-support.ht
ml. (https://spark.apache.org/news/plan-for-dropping-python-2-support.
html.)
  DeprecationWarning)
```

Out[2]:

SparkSession - in-memory

SparkContext

Spark UI (http://192.168.31.143:4040)

Version

v3.0.2

Master

local[*]

AppName

Avro

In [4]:

```
structureData = [
    (("James","","Smith"),"36636","M",3100),
    (("Michael","Rose",""),"40288","M",4300),
    (("Robert","","Williams"),"42114","M",1400),
    (("Maria","Anne","Jones"),"39192","F",5500),
    (("Jen","Mary","Brown"),"","F",-1)
  ]

structureSchema = StructType([
        StructField('name', StructType([
             StructField('firstname', StringType(), True),
             StructField('middlename', StringType(), True),
             StructField('lastname', StringType(), True)
             ])),
         StructField('id', StringType(), True),
         StructField('gender', StringType(), True),
         StructField('salary', IntegerType(), True)
         ])
df2 = spark.createDataFrame(data=structureData,schema=structureSchema)
df2.show()
df2.printSchema()
dfx = df2.select(struct(col("name.firstname").alias("firstname"), col("name.middlena
                        ,col("name.lastname").alias("lastname")).alias("val"))
dfx.printSchema()
dfx.show()
dfx.select("val.firstname").show()
```

```
+--------------------+-----+------+------+
|                name|   id|gender|salary|
+--------------------+-----+------+------+
|    [James, , Smith]|36636|     M|  3100|
|   [Michael, Rose, ]|40288|     M|  4300|
|[Robert, , Williams]|42114|     M|  1400|
|[Maria, Anne, Jones]|39192|     F|  5500|
|  [Jen, Mary, Brown]|     |     F|    -1|
+--------------------+-----+------+------+

root
 |-- name: struct (nullable = true)
 |    |-- firstname: string (nullable = true)
 |    |-- middlename: string (nullable = true)
 |    |-- lastname: string (nullable = true)
 |-- id: string (nullable = true)
 |-- gender: string (nullable = true)
 |-- salary: integer (nullable = true)

root
 |-- val: struct (nullable = false)
 |    |-- firstname: string (nullable = true)
 |    |-- middlename: string (nullable = true)
 |    |-- lastname: string (nullable = true)

+--------------------+
|                 val|
+--------------------+
|    [James, , Smith]|
|   [Michael, Rose, ]|
|[Robert, , Williams]|
|[Maria, Anne, Jones]|
```

```
|  [Jen, Mary, Brown]|
+--------------------+

+---------+
|firstname|
+---------+
|    James|
|  Michael|
|   Robert|
|    Maria|
|      Jen|
+---------+
```

In [5]:

```
jsonFormatSchema = """{
  "namespace": "example.avro",
  "type": "record",
  "name": "User",
  "fields": [
    {"name": "firstname", "type": ["string", "null"]},
    {"name": "middlename", "type": "string"},
    {"name": "lastname", "type": "string"}
  ]
}
"""

jsonFormatSchema
```

Out[5]:

```
'{\n  "namespace": "example.avro",\n  "type": "record",\n  "name": "Us
er",\n  "fields": [\n    {"name": "firstname", "type": ["string", "nul
l"]},\n    {"name": "middlename", "type": "string"},\n    {"name": "la
stname", "type": "string"}\n  ]\n}\n'
```

In [7]:

```
dfx.select(lit("1").alias("key"), to_avro("val", jsonFormatSchema).alias("value")) \
.write \
.format("kafka") \
.option("kafka.bootstrap.servers", "localhost:9092") \
.option("topic", "avro_spark") \
.save()
```

In [8]:

```
dfconsume = spark \
  .read \
  .format("kafka") \
  .option("kafka.bootstrap.servers", "localhost:9092") \
  .option("subscribe", "avro_spark") \
  .load().select("value")
```

In [9]:

```
def deserialize_avro(serialized_msg):
    bytes_io = io.BytesIO(serialized_msg)
    bytes_io.seek(1)
    avro_schema = {
                    "type": "record",
                    "name": "struct",
                    "fields": [
                      {"name": "firstname", "type": "string"},
                      {"name": "middlename", "type": "string"},
                      {"name": "lastname", "type": "string"}
                    ]
                  }

    deserialized_msg = fastavro.schemaless_reader(bytes_io, avro_schema)

    return (deserialized_msg["firstname"],deserialized_msg["middlename"],deserialize



df_schema = StructType([StructField("firstname", StringType(), True),
                        StructField("middlename", StringType(), True),
                        StructField("lastname", StringType(), True)])

avro_deserialize_udf = psf.udf(deserialize_avro, returnType=df_schema)
parsed_df = dfconsume.withColumn("avro", avro_deserialize_udf(psf.col("value"))).sel

parsed_df.show()
```

```
+---------+----------+--------+
|firstname|middlename|lastname|
+---------+----------+--------+
|   Robert|          |Williams|
|    Maria|      Anne|   Jones|
|  Michael|      Rose|        |
|      Jen|      Mary|   Brown|
|    James|          |   Smith|
|   Robert|          |Williams|
|  Michael|      Rose|        |
|    James|          |   Smith|
|    Maria|      Anne|   Jones|
|      Jen|      Mary|   Brown|
+---------+----------+--------+
```

In [ ]:

In [ ]: