Week 6_Lab01_Simonsen

June 16, 2024

[ ]:
```
'''
In this notebook, you'll notice my comments in the code. Ultimately, I had to␣
 ↪use two different strategies for exporting the csv, and for writing it to a␣
 ↪specified path in dbfs. I tried to use the Databricks CLI to export from dbfs␣
 ↪to local, but was unsucessful in the installation so I went with the below␣
 ↪code. Let me know if you have questions.
'''
```

[ ]:
```
from pyspark.sql.types import StructType, StructField, StringType , IntegerType,␣
 ↪FloatType, TimestampType, DoubleType
from pyspark.sql import functions as F
spark.conf.set("spark.sql.legacy.timeParserPolicy", "LEGACY")


schema = StructType([
                     StructField('device_id',IntegerType()),
                     StructField('device_name',StringType()),
                     StructField('ip',StringType()),
                     StructField('cca2',StringType()),
                     StructField('cca3',StringType()),
                     StructField('cn',StringType()),
                     StructField('latitude',FloatType()),
                     StructField('longitude',FloatType()),
                     StructField('scale',StringType()),
                     StructField('temp',IntegerType()),
                     StructField('humidity',IntegerType()),
                     StructField('battery_level',IntegerType()),
                     StructField('c02_level',IntegerType()),
                     StructField('lcd',StringType()),
                     StructField('timestamp',TimestampType()),


])
```

[ ]:
```
%sql

drop schema if exists bronze CASCADE ;
```

```
create schema if not exists bronze ;
drop table if exists bronze.countries;
use bronze
```

[ ]:
```
dbutils.fs.rm("dbfs:/FileStore/Merrimack/Week_6/bronzecheckpoint", recurse= True)
dbutils.fs.rm("dbfs:/FileStore/Merrimack/Week_6/silvercheckpoint", recurse= True)
```

[ ]:
```
False
```

[ ]:
```
source_dir = 'dbfs:/FileStore/Merrimack/Week_6'
countries_bronze= (spark.readStream.format("json")
                   .option('header', 'true')
                   .schema(schema)
                   .load(source_dir)
)
```

[ ]:
```
WriteStream = (countries_bronze.writeStream
               .option('checkpointLocation',f'{source_dir}/bronzecheckpoint')
               .partitionBy('cn')
               .outputMode("append")
               .queryName('AppendBronze')
               .toTable("bronze.countries")

)
```

[ ]:
```
%sql
drop schema if exists silver CASCADE ;
create schema if not exists silver ;
drop table if exists silver.countries;
use silver ;
```

[ ]:
```
%sql

select * from bronze.countries limit 10
```

[ ]:
```
bronze_location = 'dbfs:/user/hive/warehouse/bronze.db/countries'
dbutils.fs.ls(f'{bronze_location}')
```

[ ]:
```
[FileInfo(path='dbfs:/user/hive/warehouse/bronze.db/countries/_delta_log/',
name='_delta_log/', size=0, modificationTime=1718561184000),
 FileInfo(path='dbfs:/user/hive/warehouse/bronze.db/countries/cn=Austria/',
name='cn=Austria/', size=0, modificationTime=1718561189000),
 FileInfo(path='dbfs:/user/hive/warehouse/bronze.db/countries/cn=Canada/',
name='cn=Canada/', size=0, modificationTime=1718561189000),
 FileInfo(path='dbfs:/user/hive/warehouse/bronze.db/countries/cn=China/',
name='cn=China/', size=0, modificationTime=1718561189000),
 FileInfo(path='dbfs:/user/hive/warehouse/bronze.db/countries/cn=India/',
```

```
name='cn=India/', size=0, modificationTime=1718561189000),
 FileInfo(path='dbfs:/user/hive/warehouse/bronze.db/countries/cn=Italy/',
name='cn=Italy/', size=0, modificationTime=1718561189000),
 FileInfo(path='dbfs:/user/hive/warehouse/bronze.db/countries/cn=Japan/',
name='cn=Japan/', size=0, modificationTime=1718561189000),
 FileInfo(path='dbfs:/user/hive/warehouse/bronze.db/countries/cn=Netherlands/',
name='cn=Netherlands/', size=0, modificationTime=1718561189000),
 FileInfo(path='dbfs:/user/hive/warehouse/bronze.db/countries/cn=Norway/',
name='cn=Norway/', size=0, modificationTime=1718561189000),
 FileInfo(path='dbfs:/user/hive/warehouse/bronze.db/countries/cn=Philippines/',
name='cn=Philippines/', size=0, modificationTime=1718561189000),
 FileInfo(path='dbfs:/user/hive/warehouse/bronze.db/countries/cn=Republic of
Korea/', name='cn=Republic of Korea/', size=0, modificationTime=1718561189000),
 FileInfo(path='dbfs:/user/hive/warehouse/bronze.db/countries/cn=United
States/', name='cn=United States/', size=0, modificationTime=1718561189000)]
```

[ ]:
```
silver_countries = (spark.readStream
                    .format('delta')
                    .option("delta.format", "parquet")
                    .option("delta.inferColumnTypes", "true")
                    .option('header', 'true')
                    .load(bronze_location)
)
```

[ ]:
```
'''
Here, I had to toggle between the different group by options to be able to write␣
 ↪to stream and print to csv, and use the display function in order to show it␣
 ↪in the notebook and export to my local machine. The code below shows the last␣
 ↪update I made to write it to a csv in dbfs by including the timestamp in the␣
 ↪group by clause.
'''
silver_countries = (silver_countries
                    .withWatermark("timestamp", "10 minutes")
                    .groupBy("cn", "timestamp")
                    .agg(F.avg(F.col("temp")).alias('average_temp'), F.
 ↪count("*").alias('count'))
)
```

[ ]:
```
'\nsilver_countries = (silver_countries\n
.withWatermark("timestamp", "10 minutes")\n                    .groupBy("cn",
"timestamp")\n
.agg(F.avg(F.col("temp")).alias(\'average_temp\'),
F.count("*").alias(\'count\'))\n
#.orderBy(\'count\',ascending=False)\n)\n'
```

[ ]:
```
#Here is how I was able to download the CSV to my local machine.
display(silver_countries)
```

[ ]:
```
#Here is the code I used to write the csv to the specifed DBFS directory. Again,␣
↪I could not download from dbfs to my local machine, hence my use of display()␣
↪above.
silver_countries = silver_countries.withWatermark("timestamp", "10 minutes")

schema = StructType([
    StructField("cn", StringType(), True),
    StructField("average_temp", DoubleType(), True),
    StructField("count", DoubleType(), True),
    StructField("timestamp", TimestampType(), True)
])

silver_countries_df = spark.read.schema(schema).parquet(f'{source_dir}/silver.
↪parquet')

silver_countries_df.write.csv(f'{source_dir}/silver.countries', header=True, ␣
↪mode="overwrite")
```

[ ]:
```
'\nsilver_countries_csv = (silver_countries\n    .writeStream\n
.format("csv")\n    #.option("format", "append")\n
.trigger(processingTime="30 seconds")\n    .option("checkpointLocation",
f\'{source_dir}/silvercheckpoint\')\n    .option("path",
f\'{source_dir}/silver.countries\')\n    .option("header", \'true\')\n
.outputMode("complete")\n    .start()\n)\n'
```

[ ]:
```
WriteStream.stop()
```