Gimel_Notebook-Copy1

November 13, 2020

[1]:
```
!scala -version
```

```
Scala code runner version 2.12.10 -- Copyright 2002-2019, LAMP/EPFL and
Lightbend, Inc.
```

Initiated Spark Session with Gimel Libraries

[ ]:
```
from pyspark.sql import SparkSession
spark = SparkSession.builder \
   .appName('Gimel_Demo_Dataproc_1.5x')\
   .config('spark.jars', 'gs://demc-test/lib/gimel-core-2.4.7-SNAPSHOT-uber.
 ↪jar,gs://spark-lib/bigquery/spark-bigquery-latest_2.12.jar') \
   .config('spark.driver.extraClassPath','gimel-core-2.4.7-SNAPSHOT-uber.jar:
 ↪spark-bigquery-latest_2.12.jar') \
   .config('spark.executor.extraClassPath','gimel-core-2.4.7-SNAPSHOT-uber.jar:
 ↪spark-bigquery-latest_2.12.jar') \
   .getOrCreate()
```

[2]:
```
spark.version
```

[2]:
```
'2.4.7'
```

Initiated Gimel Data / SQL API

```
// ------ SCALA --------

val dataset = com.paypal.gimel.DataSet(spark)
```

```
## ------- PYTHON -------

# import DataFrame and SparkSession
from pyspark.sql import DataFrame, SparkSession, SQLContext

# fetch reference to the class in JVM
ScalaDataSet = sc._jvm.com.paypal.gimel.DataSet
```

```
# fetch reference to java SparkSession
jspark = spark._jsparkSession

# initiate dataset
dataset = ScalaDataSet.apply(jspark)
```

Read, Transform, Write ______________________________

[ ]:
```
    ## ---------- Unified Data API ---------- ##


    dataFrame = dataset.read("bigquery_dataset",option)

    dataFrame = dataset.read("hive_dataset",option)

    dataFrame = dataset.read("kafka_dataset",option)

    dataFrame = dataset.read("mysql_dataset",option)

    dataFrame = dataset.read("hdfs_dataset",option)

    dataFrame = dataset.read("gcs_dataset",option)

    dataFrame = dataset.read("s3_dataset",option)


    # ---------- WRITE API ---------- ##

    dataset.write("bigquery_dataset",dataFrame,option)

    dataset.write("elastic_dataset",dataFrame,option)
```

[ ]:
```


    ## ----------  SQL API ---------- ##

    gsql("select * from udc.hive.cluster1.edw.cust_dim")

    gsql("select * from udc.kafka.tracking_cluster1.namespace1.cust_activity")

    gsql("""

    set gimel.kafka.batch.reader.save.checkpoint=true;
    set gimel.kafka.checkpoint.zk.path=/user/checkpoints/app_name;

    insert into udc.hive.cluster1.edw.cust_dim
```

```
select
 k.id
,k.cust_id
,k.activity_type
,m.risk_score
from udc.kafka.tracking_cluster1.namespace1.cust_activity k
join cust_risk_score_from_txn m
where <>

""")
```

Catalog Provider ____

[ ]:
```
## ---- Catalog Provider can be HIVE, USER or external Catalog
```

[ ]:
```
## ---- CatalogProvider = UDC | Distributed Catalog API in PayPal

spark.sql("set gimel.catalog.provider=UDC")

dataFrame = dataset.read("A_dataset",option)
    ## ---- Or ---- ##
gsql("select * from A_dataset")
```

[ ]:
```
## ---- CatalogProvider = HIVE | Distributed Catalog with in a Hive Metastore

spark.sql("""
create external table default.my_dataset
(cols String)
location "gs://demc-test/pp-devcos-dataproc-gcs.BQ_benchmark.date_dim"
TBLPROPERTIES(
'gimel.storage.type'='bigquery',
'table'='pp-devcos-dataproc-gcs.BQ_benchmark.date_dim',
'datasetName'='my_cloud_table',
'bucket'='gs://demc-test/pp-devcos-dataproc-gcs.BQ_benchmark.date_dim',
'format'='parquet',
'abc'='xyz'
)
""")

spark.sql("set gimel.catalog.provider=HIVE")
dataFrame = dataset.read("default.my_dataset",option)
## ---- Or ---- ##
gsql("select * from default.my_dataset")
```

[ ]:
```
## - CatalogProvider = USER | Runtime config (ephemeral)

val dataSetProperties = s"""
  { "datasetName":"bg_dataset",
      "datasetType": "bigquery",
      "fields": [],
      "partitionFields": [],
      "props": {
            "gimel.storage.type":"bigquery",
            "datasetName":"my_cloud_table",
            "bucket":"gs://demc-test/pp-devcos-dataproc-gcs.BQ_benchmark.
↪date_dim",
            "format":"parquet",
            "table" : "pp-devcos-dataproc-gcs.BQ_benchmark.date_dim",
            "abc" : "xyz"
         }
  }"""

spark.sql("set gimel.catalog.provider=USER")
options = Map("bg_dataset.dataSetProperties" -> dataSetProperties)
val dataFrame = dataset.read("bg_dataset",option)
        ## ---- Or ---- ##
gsql("select * from A_dataset")
```

[ ]:

[ ]: