SCRIPTS EXECUTION

Load card_transactions.csv into MongoDB

```
[hadoop@ip-172-31-66-21 ~]$ aws s3 cp s3://history-transactions/card_transactions.csv - | mongoimport --db transaction_db --collection card_transactions --type csv --headerline
2023-05-13T18:13:42.314+0000    connected to: mongodb://localhost/
2023-05-13T18:13:43.037+0000    53292 document(s) imported successfully. 0 document(s) failed to import.
```

```
[hadoop@ip-172-31-74-203 ~]$ mongosh
Current Mongosh Log ID: 645d819a9891c8d64e68a09d
Connecting to:          mongodb://127.0.0.1:27017/?directConnection=true&serverSelectionTimeoutMS=2000&appName=mongosh+1.8.2
Using MongoDB:          6.0.5
Using Mongosh:          1.8.2

For mongosh info see: https://docs.mongodb.com/mongodb-shell/


To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

------
   The server generated these startup warnings when booting
   2023-05-12T00:00:00.266+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
   2023-05-12T00:00:00.266+00:00: vm.max_map_count is too low
------

------
   Enable MongoDB's free cloud-based monitoring service, which will then receive and display
   metrics about your deployment (disk utilization, CPU, operation statistics, etc).

   The monitoring data will be available on a MongoDB website with a unique URL accessible to you
   and anyone you share the URL with. MongoDB may use this information to make product
   improvements and to suggest MongoDB products and deployment options to you.

   To enable free monitoring, run the following command: db.enableFreeMonitoring()
   To permanently disable this reminder, run the following command: db.disableFreeMonitoring()
------

test> use transaction_db
switched to db transaction_db
transaction_db> db.card_transactions.findOne()
{
  _id: ObjectId("645d8194389154a318c0c087"),
  card_id: Long("348702330256514"),
  member_id: Long("37495066290"),
  amount: 4310362,
  postcode: 33946,
  pos_id: Long("614677375609919"),
  transaction_dt: '11-02-2018 00:00:00',
  status: 'GENUINE'
}
```

Ingest RDBMS - Load card_member.csv and member_score.csv into Spark data frame and table.

```
>>> df_card = spark.read.options(inferSchema='True',header='True').csv('s3://history-transactions/card_member.csv')
>>> df_card.printSchema()
root
 |-- card_id: long (nullable = true)
 |-- member_id: long (nullable = true)
 |-- member_joining_dt: timestamp (nullable = true)
 |-- card_purchase_dt: string (nullable = true)
 |-- country: string (nullable = true)
 |-- city: string (nullable = true)

>>> df_score = spark.read.options(inferSchema='True',header='True').csv('s3://history-transactions/member_score.csv'
...
... )
>>> df_score.printSchema()
root
 |-- member_id: long (nullable = true)
 |-- score: integer (nullable = true)
```

```
>>> df_card.createOrReplaceTempView('tb_card')
>>> spark.sql('SELECT * FROM tb_card LIMIT 3').show()
+---------------+---------------+-------------------+----------------+-------------+----------+
|        card_id|      member_id|  member_joining_dt|card_purchase_dt|      country|      city|
+---------------+---------------+-------------------+----------------+-------------+----------+
|340028465709212|  9250698176266|2012-02-08 06:04:13|           05/13|United States| Barberton|
|340054675199675|835873341185231|2017-03-10 09:24:44|           03/17|United States|Fort Dodge|
|340082915339645|512969555857346|2014-02-15 06:30:30|           07/14|United States|    Graham|
+---------------+---------------+-------------------+----------------+-------------+----------+

>>> df_score.createOrReplaceTempView('tb_score')
>>> spark.sql('SELECT * FROM tb_score LIMIT 3').show()
+-------------+-----+
|    member_id|score|
+-------------+-----+
|  37495066290|  339|
| 117826301530|  289|
|1147922084344|  393|
+-------------+-----+
```

Load card_transaction from MongoDB to Spark data frame and table.

```
>>> df_tran = spark.read.format("mongodb").load()
>>> df_tran.printSchema()
root
 |-- _id: string (nullable = true)
 |-- amount: integer (nullable = true)
 |-- card_id: long (nullable = true)
 |-- member_id: long (nullable = true)
 |-- pos_id: long (nullable = true)
 |-- postcode: integer (nullable = true)
 |-- status: string (nullable = true)
 |-- transaction_dt: string (nullable = true)

>>> df_tran.createOrReplaceTempView('tb_tran')
>>> spark.sql('SELECT * FROM tb_tran LIMIT 3').show()
+--------------------+-------+---------------+-----------+---------------+--------+-------+-------------------+
|                 _id| amount|        card_id|  member_id|         pos_id|postcode| status|     transaction_dt|
+--------------------+-------+---------------+-----------+---------------+--------+-------+-------------------+
|645fd356ab3b87d97...| 330148|348702330256514|37495066290|614677375609919|   33946|GENUINE|11-02-2018 00:00:00|
|645fd356ab3b87d97...|9084849|348702330256514|37495066290|614677375609919|   33946|GENUINE|11-02-2018 00:00:00|
|645fd356ab3b87d97...| 136052|348702330256514|37495066290|614677375609919|   33946|GENUINE|11-02-2018 00:00:00|
+--------------------+-------+---------------+-----------+---------------+--------+-------+-------------------+
```

Create Look-up table

```
>>> spark.sql("CREATE TABLE tb_lookup (card_id STRING, ucl DOUBLE, postcode STRING, transaction_dt STRING, score INT)")
23/05/13 19:06:41 WARN ResolveSessionCatalog: A Hive serde table will be created as there is no table provider specifie
d. You can set spark.sql.legacy.createHiveTableByDefault to false so that native data source table will be created inst
ead.
23/05/13 19:06:41 INFO SQLStdHiveAccessController: Created SQLStdHiveAccessController for session context : HiveAuthzSe
ssionContext [sessionString=a301c5c0-3a43-4f5a-b571-8b5727916ece, clientType=HIVECLI]
23/05/13 19:06:41 WARN SessionState: METASTORE_FILTER_HOOK will be ignored, since hive.security.authorization.manager i
s set to instance of HiveAuthorizerFactory.
23/05/13 19:06:41 INFO metastore: Mestastore configuration hive.metastore.filter.hook changed from org.apache.hadoop.hi
ve.metastore.DefaultMetaStoreFilterHookImpl to org.apache.hadoop.hive.ql.security.authorization.plugin.AuthorizationMet
aStoreFilterHook
23/05/13 19:06:41 INFO metastore: Closed a connection to metastore, current connections: 0
23/05/13 19:06:41 INFO metastore: Trying to connect to metastore with URI thrift://ip-172-31-66-21.ec2.internal:9083
23/05/13 19:06:41 INFO metastore: Opened a connection to metastore, current connections: 1
23/05/13 19:06:41 INFO metastore: Connected to metastore.
23/05/13 19:06:41 INFO metastore: Trying to connect to metastore with URI thrift://ip-172-31-66-21.ec2.internal:9083
23/05/13 19:06:41 INFO metastore: Opened a connection to metastore, current connections: 2
23/05/13 19:06:41 INFO metastore: Connected to metastore.
DataFrame[]
```

```
>>> spark.sql('SELECT * FROM tb_lookup')
DataFrame[card_id: string, ucl: double, postcode: string, transaction_dt: string, score: int]
```

Load data into Look-up table

Take last 10 transactions for each card

```
>>> df_10trans = spark.sql("\
... SELECT card_id, amount, postcode, transaction_dt, status, rn \
... FROM (\
... SELECT card_id, amount, postcode, transaction_dt, status, ROW_NUMBER() OVER (PARTITION BY card_id ORDER BY unix_tim
estamp(transaction_dt,'dd-MM-yyyy hh:mm:ss') DESC) AS rn \
... FROM tb_tran \
... WHERE status = 'GENUINE') a \
... WHERE a.rn <= 10")
>>> df_10trans.createOrReplaceTempView('tb_10trans')
>>> spark.sql('SELECT * FROM tb_10trans LIMIT 3').show()
+--------------+-------+--------+------------------+-------+---+
|        card_id| amount|postcode|     transaction_dt| status| rn|
+--------------+-------+--------+------------------+-------+---+
|340028465709212|8696557|   24658|02-01-2018 03:25:35|GENUINE|  1|
|340028465709212| 430409|   58270|15-11-2017 01:59:54|GENUINE|  2|
|340028465709212|6503191|   84776|09-11-2017 07:18:21|GENUINE|  3|
+--------------+-------+--------+------------------+-------+---+

>>> spark.sql('SELECT * FROM tb_10trans LIMIT 20').show()
+--------------+-------+--------+------------------+-------+---+
|        card_id| amount|postcode|     transaction_dt| status| rn|
+--------------+-------+--------+------------------+-------+---+
|340028465709212|8696557|   24658|02-01-2018 03:25:35|GENUINE|  1|
|340028465709212| 430409|   58270|15-11-2017 01:59:54|GENUINE|  2|
|340028465709212|6503191|   84776|09-11-2017 07:18:21|GENUINE|  3|
|340028465709212|8884049|   25537|07-10-2017 09:17:12|GENUINE|  4|
|340028465709212|9291309|   31322|12-08-2017 08:29:54|GENUINE|  5|
|340028465709212|8370505|   84056|12-07-2017 02:51:29|GENUINE|  6|
|340028465709212|9687739|   51542|05-07-2017 11:05:55|GENUINE|  7|
|340028465709212|6500086|   25040|24-06-2017 01:13:31|GENUINE|  8|
|340028465709212| 581323|   46182|17-05-2017 12:36:12|GENUINE|  9|
|340028465709212|5118701|   12045|30-03-2017 04:09:10|GENUINE| 10|
|340054675199675|  29445|   50140|15-01-2018 10:56:43|GENUINE|  1|
|340054675199675|9728785|   77373|10-01-2018 02:47:11|GENUINE|  2|
|340054675199675|2223104|   35973|09-01-2018 10:59:10|GENUINE|  3|
|340054675199675|1201277|   84530|28-12-2017 05:48:04|GENUINE|  4|
|340054675199675|6140357|   40023|18-12-2017 10:33:04|GENUINE|  5|
|340054675199675|7914699|   41844|12-12-2017 07:04:51|GENUINE|  6|
|340054675199675|7573707|   12024|06-12-2017 08:52:38|GENUINE|  7|
|340054675199675|2797924|   54141|04-12-2017 12:59:15|GENUINE|  8|
|340054675199675|7876899|   71047|27-11-2017 01:54:59|GENUINE|  9|
|340054675199675|5418389|   21084|05-11-2017 12:00:53|GENUINE| 10|
+--------------+-------+--------+------------------+-------+---+
```

Calculate UCL

```
>>> df_ucl = spark.sql("\
... SELECT a.card_id, (a.avge + (3 * a.std)) as UCL \
... FROM (\
... SELECT t.card_id, AVG(t.amount) AS avge, STDDEV(t.amount) as std \
... FROM tb_10trans t \
... GROUP BY t.card_id) a")
>>> df_ucl.createOrReplaceTempView('tb_ucl')
>>> spark.sql('SELECT * FROM tb_ucl LIMIT 3').show()
+--------------+--------------------+
|        card_id|                 UCL|
+--------------+--------------------+
|340028465709212|1.6685076623853374E7|
|340054675199675|1.5032693399975928E7|
|340082915339645|1.5323729774843596E7|
+--------------+--------------------+
```

Insert data into Look-up table

```
>>> spark.sql("INSERT INTO TABLE tb_lookup \
... SELECT trans.card_id, ucl.ucl, trans.postcode, trans.transaction_dt, cdsc.score \
... FROM  tb_10trans trans \
... JOIN tb_ucl ucl \
... ON ucl.card_id = trans.card_id \
... JOIN (\
... SELECT DISTINCT crd.card_id, scr.score \
... FROM tb_card crd \
... JOIN tb_score scr \
... ON crd.member_id = scr.member_id) AS cdsc \
... ON trans.card_id = cdsc.card_id \
... WHERE trans.rn = 1")
DataFrame[]
>>> spark.sql('SELECT * FROM tb_lookup LIMIT 3').show()
+---------------+------------------+--------+------------------+-----+
|        card_id|               ucl|postcode|    transaction_dt|score|
+---------------+------------------+--------+------------------+-----+
|5411141346922507|1.5997512196104523E7|   25156|20-09-2017 09:59:16|  225|
|4502008970767222| 1.684959540133459E7|   93010|31-01-2018 09:39:57|  657|
|5341963603599990| 1.516390152866365E7|   18419|09-01-2018 07:28:30|  554|
+---------------+------------------+--------+------------------+-----+
```

Save lookup table in MongoDB

```
df_lookup.write.format("mongodb").mode("append").save()
```

```
switched to db transaction_db
transaction_db> db.card_transactions.findOne()
{
  _id: ObjectId("646a94c87760a83f9571a002"),
  card_id: Long("348702330256514"),
  member_id: Long("37495066290"),
  amount: 4310362,
  postcode: 33946,
  pos_id: Long("614677375609919"),
  transaction_dt: '11-02-2018 00:00:00',
  status: 'GENUINE'
}
transaction_db> db.tb_lookup.findOne()
{
  _id: ObjectId("646a9b313ec71934249ddea4"),
  card_id: Long("5411141346922507"),
  ucl: 15997512.196104523,
  postcode: 25156,
  transaction_dt: '20-09-2017 09:59:16',
  score: 225
}
```