Data Correction and Load

Updates parsed files from DataBricks cluster and writes to blob storage

In [ ]:
```
from pyspark.sql.window import Window
from pyspark.sql.functions import desc, row_number
```

In [ ]:
```
# Azure credentials
storageAccountName = 'exchangedata1'
storageAccountAccessKey = '<your-access-key>'
ContainerName = 'source-container'
spark.conf.set(f'fs.azure.account.key.{storageAccountName}.blob.core.windows.net', storageAccountAccessKey)
```

In [ ]:
```
#finding directory paths and contents
import os

#method one is using databricks utilities
print('DataBricks Utility')
print(dbutils.fs.ls('/output/parsed_data/'))
print('\n')
#method two is using os module with databricks filesystem as root folder
print('OS Module')
for i in os.listdir('/dbfs/output/parsed_data/2020-08-05/partition=Q'):
    print(i)
```

```
DataBricks Utility
[FileInfo(path='dbfs:/output/parsed_data/2020-08-05/', name='2020-08-05/', size=0), FileInfo(path='dbfs:/output/parsed_data/2020-08-06/', name='2020-08-06/', size=0)]


OS Module
_committed_1063823211978745755
_committed_787419416886541593
_committed_7977079052928358081
_committed_8179522139142287879
_committed_vacuum112269526786033157
_started_7977079052928358081
_started_8179522139142287879
part-00000-tid-8179522139142287879-243546f6-5765-4e82-93fa-6eeadf5c8279-73-1.c000.snappy.parquet
part-00001-tid-8179522139142287879-243546f6-5765-4e82-93fa-6eeadf5c8279-74-1.c000.snappy.parquet
part-00002-tid-8179522139142287879-243546f6-5765-4e82-93fa-6eeadf5c8279-75-1.c000.snappy.parquet
part-00003-tid-8179522139142287879-243546f6-5765-4e82-93fa-6eeadf5c8279-76-1.c000.snappy.parquet
```

Create Dataframe from parsed parquet files

In [ ]:
```
def parsed_parquet_path(date,type):
    # read file as RDD
        parquet_list=[]
        if type in ('Q','T'):
            dir_path='/output/parsed_data/{}/partition={}/'.format(date,type)
            dir_list=os.listdir('/dbfs'+dir_path)
            for parquet in dir_list:
                if parquet.endswith('.parquet'):
                    parquet_list.append(dir_path+parquet)
            df = spark.read.parquet(*parquet_list)
            return df
        else:
            print('Parition type is incorrect')
            return
```

In [ ]:
```
# read data for 2020-08-05
common_quote_df_85=parsed_parquet_path('2020-08-05','Q')
common_trade_df_85=parsed_parquet_path('2020-08-05','T')

# read data for 2020-08-06
common_quote_df_86=parsed_parquet_path('2020-08-06','Q')
common_trade_df_86=parsed_parquet_path('2020-08-06','T')
```

In [ ]:
```
# veryifying trade data
print('2020-08-05 Trade DF')
print(common_trade_df_85.show(2,truncate=False))
print('2020-08-05 Trade Count')
print(common_trade_df_85.count())
```

```
2020-08-05 Trade DF
```

```
+----------+--------+------+--------+----------------------+------------+------------------+---------+------+--------+------+--------+
trade_dt  |rec_type|symbol|exchange|event_tm              |event_seq_nb|arrival_tm        |trade_pr |bid_pr|bid_size|ask_pr|ask_size|
+----------+--------+------+--------+----------------------+------------+------------------+---------+------+--------+------+--------+
2020-08-05|T       |SYMB  |NASDAQ  |2020-08-05 16:29:56.837|60          |2020-08-05 09:30:00|34.867146|null  |null    |null  |null    |
2020-08-05|T       |SYMB  |NASDAQ  |2020-08-05 17:42:00.878|70          |2020-08-05 09:30:00|36.291695|null  |null    |null  |null    |
+----------+--------+------+--------+----------------------+------------+------------------+---------+------+--------+------+--------+
only showing top 2 rows

None
2020-08-05 Trade Count
60
```

In [ ]:
```
# veryifying quote data
print('2020-08-05 Quotes DF')
print(common_quote_df_85.show(2,truncate=False))
print('2020-08-05 Quotes Count')
print(common_quote_df_85.count())
```

```
2020-08-05 Quotes DF
+----------+--------+------+--------+----------------------+------------+------------------+--------+--------+--------+--------+--------+
trade_dt  |rec_type|symbol|exchange|event_tm              |event_seq_nb|arrival_tm        |trade_pr|bid_pr  |bid_size|ask_pr  |ask_size|
+----------+--------+------+--------+----------------------+------------+------------------+--------+--------+--------+--------+--------+
2020-08-05|Q       |SYMA  |NASDAQ  |2020-08-05 09:36:55.284|1           |2020-08-05 09:30:00|null    |76.10017|100     |77.9648 |100     |
2020-08-05|Q       |SYMA  |NASDAQ  |2020-08-05 09:42:32.247|2           |2020-08-05 09:30:00|null    |75.44373|100     |75.94453|100     |
+----------+--------+------+--------+----------------------+------------+------------------+--------+--------+--------+--------+--------+
only showing top 2 rows

None
2020-08-05 Quotes Count
540
```

Creating Dataframe with columns specific to Quote or Trade

In [ ]:
```
#getting columns pertaining specifically to quotes or trades only since previous dataframe include columns related to both quote and trade data
# 2020-08-05
specific_trade_df_85=common_trade_df_85.select('trade_dt', 'symbol', 'exchange', 'event_tm', 'event_seq_nb', 'arrival_tm', 'trade_pr')
specific_quote_df_85=common_quote_df_85.select('trade_dt', 'symbol', 'exchange', 'event_tm', 'event_seq_nb', 'arrival_tm', 'bid_pr','bid_size','ask_pr','ask_size')
```

In [ ]:
```
#getting columns pertaining specifically to quotes or trades only since previous dataframe include columns related to both quote and trade data
# 2020-08-06
specific_trade_df_86=common_trade_df_86.select('trade_dt', 'symbol', 'exchange', 'event_tm', 'event_seq_nb', 'arrival_tm', 'trade_pr')
specific_quote_df_86=common_quote_df_86.select('trade_dt', 'symbol', 'exchange', 'event_tm', 'event_seq_nb', 'arrival_tm', 'bid_pr','bid_size','ask_pr','ask_size')
```

Correcting Data Function

In [ ]:
```
#same records can by uniquely identifed by columns trade_dt,symbol,event_tm, and event_seq_nb
#since some records may be sent by exchanges in later batches to correct for initial data, we partition by unique identfiers and order by arrival time
#function applies row_number function over defined window to identify and keep most recent recieved records
def apply_latest_data(df):
    WindowSpec=Window.partitionBy('trade_dt','symbol','event_tm','event_seq_nb').orderBy(desc('arrival_tm'))
    corrected_df=df.withColumn('row_number',row_number().over(WindowSpec)).where('row_number == 1').drop('row_number')
    return corrected_df
```

Testing code in apply_latest_data functions

In [ ]:
```
# add in duplicated test data to verify function code works
#test data has the same unique record identifers as another field, but a later arrival time and different trade_pr
test_schema = ['trade_dt', 'symbol', 'exchange', 'event_tm', 'event_seq_nb', 'arrival_tm', 'trade_pr']
test_data=[("2020-08-05","SYMA","NASDAQ","2020-08-05 10:38:50.046",10,"2020-08-05 09:45:00.0",82.11)]
test_df = spark.createDataFrame(test_data, schema=test_schema)

# union test_df with original trade_df
df_with_testdata=specific_trade_df_85.union(test_df)

#creates column with row numbers that partitions by unique identifers and orders by latest arrival time
WindowSpec=Window.partitionBy('trade_dt','symbol','event_tm','event_seq_nb').orderBy(desc('arrival_tm'))
df_with_testdata=df_with_testdata.withColumn('row_number',row_number().over(WindowSpec))


# filtering for records with row number 2 or greater
df_with_testdata.filter(df_with_testdata['row_number']>='2').select('*').show()

# since our test data had event_tm=2020-08-05 10:38:50.046, we filter for this to confirm the later arrival time has row number 1
df_with_testdata.filter(df_with_testdata['event_tm']=='2020-08-05 10:38:50.046').select('*').show(truncate=False)
```

```
+----------+------+--------+--------------------+------------+------------------+-----------------+----------+
  trade_dt|symbol|exchange|            event_tm|event_seq_nb|         arrival_tm|         trade_pr|row_number|
+----------+------+--------+--------------------+------------+------------------+-----------------+----------+
```

```
2020-08-05|  SYMA|  NASDAQ|2020-08-05 10:38:...|          10|2020-08-05 09:30:00|77.77570343017578|         2|
+----------+------+--------+--------------------+------------+-------------------+-----------------+----------+

+----------+------+--------+----------------------+------------+--------------------+-----------------+----------+
trade_dt  |symbol|exchange|event_tm              |event_seq_nb|arrival_tm          |trade_pr         |row_number|
+----------+------+--------+----------------------+------------+--------------------+-----------------+----------+
2020-08-05|SYMA  |NASDAQ  |2020-08-05 10:38:50.046|10          |2020-08-05 09:45:00.0|82.11            |1         |
2020-08-05|SYMA  |NASDAQ  |2020-08-05 10:38:50.046|10          |2020-08-05 09:30:00  |77.77570343017578|2         |
+----------+------+--------+----------------------+------------+--------------------+-----------------+----------+
```

Apply correction to data

In [ ]:
```
#data correction for 2020-08-05
trade_corrected_85=apply_latest_data(specific_trade_df_85)
quote_corrected_85=apply_latest_data(specific_quote_df_85)

#data correction for 2020-08-06
trade_corrected_86=apply_latest_data(specific_trade_df_86)
quote_corrected_86=apply_latest_data(specific_quote_df_86)
```

EOD Load

In [ ]:
```
# Quote and Trade paritions include NYSE and NASDAQ data together in each partition
def write_to_blob(df,date,type):
    blob_path='wasbs://{}@{}.blob.core.windows.net'.format(ContainerName,storageAccountName)
    df.write.parquet(blob_path+'/output/EOD_corrected/{}/partition={}'.format(date,type))
    return
```

In [ ]:
```
#2020-08-05
write_to_blob(trade_corrected_85,'2020-08-05','T')
write_to_blob(quote_corrected_85,'2020-08-05','Q')
#2020-08-06
write_to_blob(trade_corrected_86,'2020-08-06','T')
write_to_blob(quote_corrected_86,'2020-08-06','Q')
```

In [ ]:
```
dbutils.notebook.exit('SUCCESS')
```