Save Data

Batch

CSV

In [ ]:
```python
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Save to CSV") \
    .getOrCreate()

df = spark.read.csv("csv_data", header=True)
df.write.csv("output", header=True, mode="overwrite")
```

Json

In [ ]:
```python
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Save to json") \
    .getOrCreate()

df = spark.read.csv("csv_data", header=True)
df.write.json("output.json", mode="overwrite")
```

Excel

In [ ]:
```python
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Save to Excel") \
    .config('spark.jars.packages', 'com.crealytics:spark-excel_2.12:3.2.1_0.17.1') \
    .getOrCreate()

df = spark.read.csv("csv_data", header=True)

df.write.format("com.crealytics.spark.excel") \
    .option("header", "true") \
    .mode("overwrite") \
    .save("output.xlsx")
```

Mysql

- MySQL (Version: 8.0.32)

In [ ]:
```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Save Stream from MySQL") \
    .config('spark.jars.packages', 'mysql:mysql-connector-java:8.0.32') \
    .getOrCreate()

mysql_host = "mysql_host"
mysql_port = "3306"
mysql_database = "mysql_batch_db"
mysql_username = "root"
mysql_password = "secret"
mysql_table = "mysql_batch"

df = spark.read.csv("csv_data", header=True)

df.write.format("jdbc") \
    .option("url", f"jdbc:mysql://{mysql_host}:{mysql_port}/{mysql_database}") \
    .option("driver", "com.mysql.jdbc.Driver") \
    .option("dbtable", mysql_table) \
    .option("user", mysql_username) \
    .option("password", mysql_password) \
    .mode("overwrite") \
    .save()
```

Postgres

- PostgreSQL (Version: 42.5.4)

In [ ]:
```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Save to PostgreSQL") \
    .config('spark.jars.packages', 'org.postgresql:postgresql:42.5.4') \
    .getOrCreate()

postgresql_host = "postgres_host"
postgresql_port = "5432"
postgresql_database = "postgres_batch_db"
postgresql_table = "postgres_batch"
postgresql_username = "postgres"
postgresql_password = "secret"

df = spark.read.csv("csv_data", header=True)

df.write.format("jdbc") \
    .option("url", f"jdbc:postgresql://{postgresql_host}:{postgresql_port}/{postgresql_database}") \
    .option("driver", "org.postgresql.Driver") \
    .option("dbtable", postgresql_table) \
    .option("user", postgresql_username) \
    .option("password", postgresql_password) \
    .mode("overwrite") \
    .save()
```

Mongo

- MongoDB (Version: 3.0.2)

In [ ]:
```python
from pyspark.sql import SparkSession

spark = SparkSession.builder \
    .appName("Save to MongoDB") \
    .config('spark.jars.packages', 'org.mongodb.spark:mongo-spark-connector_2.12:3.0.2') \
    .getOrCreate()

mongo_host = "mongo_host"
mongo_port = "27017"
mongo_database = "mongo_batch_db"
mongo_collection = "mongo_batch"
mongo_username = "mongo"
mongo_password = "mongo"

df = spark.read.csv("csv_data", header=True)

df.write.format("mongo") \
    .option("uri", f"mongodb://{mongo_username}:{mongo_password}@{mongo_host}:{mongo_port}/") \
    .option("database", mongo_database) \
    .option("collection", mongo_collection) \
    .mode("overwrite") \
    .save()
```

Cassandra

- Cassandra (Version: 3.2.0)

In [ ]:
```python
from pyspark.sql import SparkSession

cassandra_host = "cassandra_host"
cassandra_port = "9042"
cassandra_keyspace = "cassandra_batch_db"
cassandra_table = "cassandra_batch"
cassandra_username = "cassandra"
cassandra_password = "cassandra"

spark = SparkSession.builder \
    .appName("Save to Cassandra") \
    .config("spark.jars.packages", "com.datastax.spark:spark-cassandra-connector_2.12:3.2.0") \
    .config("spark.cassandra.connection.host", cassandra_host) \
    .config("spark.cassandra.connection.port", cassandra_port) \
    .config("spark.cassandra.auth.username", cassandra_username) \
    .config("spark.cassandra.auth.password", cassandra_password) \
    .getOrCreate()

df = spark.read.csv("csv_data", header=True)

df.write.format("org.apache.spark.sql.cassandra") \
    .options(table=cassandra_table, keyspace=cassandra_keyspace) \
    .mode("overwrite") \
    .option("confirm.truncate", "true") \
    .save()
```

Stream

The same code used in batch mode can be used as a function in streaming mode by simply replacing the "overwrite" mode with "append" mode.

- Append:

If the output table or file already exists, new data is added to the end of it.

- Overwrite:

If the output table or file already exists, it is replaced by the new data. This means that the old content is removed.

- Ignore:

If the output table or file already exists, new data is not written, and no modification is made to the existing data source. No error is raised.

- ErrorIfExists (Default):

This raises an error if the output table or file already exists. This is the default behavior if no mode is specified.

CSV

In [ ]:
```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import from_json, col
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, FloatType

schema = StructType([
    StructField("sale_id", StringType()),
    StructField("pos_id", IntegerType()),
    StructField("pos_name", StringType()),
    StructField("article", StringType()),
    StructField("quantity", FloatType()),
    StructField("prix", FloatType()),
    StructField("total", FloatType()),
    StructField("sale_type", StringType()),
    StructField("payment_mode", StringType()),
    StructField("sale_time", StringType()),
])


def save_to_csv(df, batch_id):

    df.write.csv("output", header=True, mode="append")

spark = SparkSession.builder \
    .appName("KafkaConsumer") \
    .config("spark.jars.packages",
            "org.apache.spark:spark-sql-kafka-0-10_2.12:3.2.1," +
            "org.apache.kafka:kafka-clients:3.4.1") \
    .getOrCreate()

input_df = spark \
    .readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "kafka:9092") \
    .option("subscribe", "kairouan_sales") \
    .option("startingOffsets", "earliest") \
    .load()

df = input_df \
    .selectExpr("CAST(value AS STRING)") \
    .select(from_json(col("value"), schema).alias("kairouan_sales")) \
    .select("kairouan_sales.*") \

query1 = df \
    .writeStream \
    .trigger(processingTime="1 seconds") \
    .foreachBatch(save_to_csv) \
    .outputMode("update") \
    .start()

query1.awaitTermination()
```

Excel

In [ ]:

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import from_json, col
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, FloatType

schema = StructType([
    StructField("sale_id", StringType()),
    StructField("pos_id", IntegerType()),
    StructField("pos_name", StringType()),
    StructField("article", StringType()),
    StructField("quantity", FloatType()),
    StructField("prix", FloatType()),
    StructField("total", FloatType()),
    StructField("sale_type", StringType()),
    StructField("payment_mode", StringType()),
    StructField("sale_time", StringType()),
])


def save_to_excel(df, batch_id):

    df.write.format("com.crealytics.spark.excel") \
        .option("header", "true") \
        .mode("append") \
        .save("output.xlsx")

spark = SparkSession.builder \
    .appName("KafkaConsumer") \
    .config("spark.jars.packages",
            "org.apache.spark:spark-sql-kafka-0-10_2.12:3.2.1," +
            "org.apache.kafka:kafka-clients:3.4.1," +
            "com.crealytics:spark-excel_2.12:3.2.1_0.17.1") \
    .getOrCreate()

input_df = spark \
    .readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "kafka:9092") \
    .option("subscribe", "kairouan_sales") \
    .option("startingOffsets", "earliest") \
    .load()

df = input_df \
    .selectExpr("CAST(value AS STRING)") \
    .select(from_json(col("value"), schema).alias("kairouan_sales")) \
    .select("kairouan_sales.*") \

query1 = df \
    .writeStream \
    .trigger(processingTime="1 seconds") \
    .foreachBatch(save_to_excel) \
    .outputMode("update") \
    .start()

query1.awaitTermination()
```

Mysql

- MySQL (Version: 8.0.32)

In [ ]:

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import from_json, col
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, FloatType

schema = StructType([
    StructField("sale_id", StringType()),
    StructField("pos_id", IntegerType()),
    StructField("pos_name", StringType()),
    StructField("article", StringType()),
    StructField("quantity", FloatType()),
    StructField("prix", FloatType()),
    StructField("total", FloatType()),
    StructField("sale_type", StringType()),
    StructField("payment_mode", StringType()),
    StructField("sale_time", StringType()),
])

mysql_host = "mysql_host"
mysql_port = "3306"
mysql_database = "mysql_stream_db"
mysql_username = "root"
mysql_password = "secret"
mysql_table = "mysql_stream"

def save_to_mysql(df, batch_id):

    df.write.format("jdbc") \
        .option("url", f"jdbc:mysql://{mysql_host}:{mysql_port}/{mysql_database}") \
        .option("driver", "com.mysql.jdbc.Driver") \
        .option("dbtable", mysql_table) \
        .option("user", mysql_username) \
        .option("password", mysql_password) \
        .mode("append") \
        .save()

spark = SparkSession.builder \
    .appName("KafkaConsumer") \
    .config("spark.jars.packages",
            "org.apache.spark:spark-sql-kafka-0-10_2.12:3.2.1," +
            "org.apache.kafka:kafka-clients:3.4.1," +
            "mysql:mysql-connector-java:8.0.32") \
    .getOrCreate()

input_df = spark \
    .readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "kafka:9092") \
    .option("subscribe", "kairouan_sales") \
    .option("startingOffsets", "earliest") \
    .load()

df = input_df \
    .selectExpr("CAST(value AS STRING)") \
    .select(from_json(col("value"), schema).alias("kairouan_sales")) \
    .select("kairouan_sales.*") \

query1 = df \
    .writeStream \
    .trigger(processingTime="1 seconds") \
    .foreachBatch(save_to_mysql) \
    .outputMode("update") \
    .start()

query1.awaitTermination()
```

Postgres

PostgreSQL

- PostgreSQL (Version: 42.5.4)

In [ ]:

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import from_json, col
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, FloatType

schema = StructType([
    StructField("sale_id", StringType()),
    StructField("pos_id", IntegerType()),
    StructField("pos_name", StringType()),
    StructField("article", StringType()),
    StructField("quantity", FloatType()),
    StructField("prix", FloatType()),
    StructField("total", FloatType()),
    StructField("sale_type", StringType()),
    StructField("payment_mode", StringType()),
    StructField("sale_time", StringType()),
])

postgresql_host = "postgres_host"
postgresql_port = "5432"
postgresql_database = "postgres_stream_db"
postgresql_table = "postgres_stream"
postgresql_username = "postgres"
postgresql_password = "secret"

def save_to_postgres(df, batch_id):

    df.write.format("jdbc") \
    .option("url", f"jdbc:postgresql://{postgresql_host}:{postgresql_port}/{postgresql_database}") \
    .option("driver", "org.postgresql.Driver") \
    .option("dbtable", postgresql_table) \
    .option("user", postgresql_username) \
    .option("password", postgresql_password) \
    .mode("append") \
    .save()

spark = SparkSession.builder \
    .appName("KafkaConsumer") \
    .config("spark.jars.packages",
            "org.apache.spark:spark-sql-kafka-0-10_2.12:3.2.1," +
            "org.apache.kafka:kafka-clients:3.4.1," +
            "org.postgresql:postgresql:42.5.4") \
    .getOrCreate()

input_df = spark \
    .readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "kafka:9092") \
    .option("subscribe", "kairouan_sales") \
    .option("startingOffsets", "earliest") \
    .load()

df = input_df \
    .selectExpr("CAST(value AS STRING)") \
    .select(from_json(col("value"), schema).alias("kairouan_sales")) \
    .select("kairouan_sales.*") \

query1 = df \
    .writeStream \
    .trigger(processingTime="1 seconds") \
    .foreachBatch(save_to_postgres) \
    .outputMode("update") \
    .start()

query1.awaitTermination()
```

Mongo

- MongoDB (Version: 3.0.2)

In [ ]:
```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import from_json, col
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, FloatType

schema = StructType([
    StructField("sale_id", StringType()),
    StructField("pos_id", IntegerType()),
    StructField("pos_name", StringType()),
    StructField("article", StringType()),
    StructField("quantity", FloatType()),
    StructField("prix", FloatType()),
    StructField("total", FloatType()),
    StructField("sale_type", StringType()),
    StructField("payment_mode", StringType()),
    StructField("sale_time", StringType()),
])

mongo_host = "mongo_host"
mongo_port = "27017"
mongo_database = "mongo_stream_db"
mongo_collection = "mongo_stream"
mongo_username = "mongo"
mongo_password = "mongo"

def save_to_mysql(df, batch_id):

    df.write.format("mongo") \
    .option("uri", f"mongodb://{mongo_username}:{mongo_password}@{mongo_host}:{mongo_port}/") \
    .option("database", mongo_database) \
    .option("collection", mongo_collection) \
    .mode("append") \
    .save()

spark = SparkSession.builder \
    .appName("KafkaConsumer") \
    .config("spark.jars.packages",
            "org.apache.spark:spark-sql-kafka-0-10_2.12:3.2.1," +
            "org.apache.kafka:kafka-clients:3.4.1," +
            "org.mongodb.spark:mongo-spark-connector_2.12:3.0.2") \
    .getOrCreate()

input_df = spark \
    .readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "kafka:9092") \
    .option("subscribe", "kairouan_sales") \
    .option("startingOffsets", "earliest") \
    .load()

df = input_df \
    .selectExpr("CAST(value AS STRING)") \
    .select(from_json(col("value"), schema).alias("kairouan_sales")) \
    .select("kairouan_sales.*") \

query1 = df \
    .writeStream \
    .trigger(processingTime="1 seconds") \
    .foreachBatch(save_to_mysql) \
    .outputMode("update") \
    .start()

query1.awaitTermination()
```

Cassandra

- Cassandra (Version: 3.2.0)

In [ ]:

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import from_json, col
from pyspark.sql.types import StructType, StructField, StringType, IntegerType, FloatType

schema = StructType([
    StructField("sale_id", StringType()),
    StructField("pos_id", IntegerType()),
    StructField("pos_name", StringType()),
    StructField("article", StringType()),
    StructField("quantity", FloatType()),
    StructField("prix", FloatType()),
    StructField("total", FloatType()),
    StructField("sale_type", StringType()),
    StructField("payment_mode", StringType()),
    StructField("sale_time", StringType()),
])

cassandra_host = "cassandra_host"
cassandra_port = "9042"
cassandra_keyspace = "cassandra_stream_db"
cassandra_table = "cassandra_stream"
cassandra_username = "cassandra"
cassandra_password = "cassandra"

def save_to_cassandra(df, batch_id):

    df.write.format("org.apache.spark.sql.cassandra") \
    .options(table=cassandra_table, keyspace=cassandra_keyspace) \
    .mode("append") \
    .option("confirm.truncate", "true") \
    .save()

spark = SparkSession.builder \
    .appName("KafkaConsumer") \
    .config("spark.jars.packages",
            "org.apache.spark:spark-sql-kafka-0-10_2.12:3.2.1," +
            "org.apache.kafka:kafka-clients:3.4.1," +
            "com.datastax.spark:spark-cassandra-connector_2.12:3.2.0") \
    .config("spark.cassandra.connection.host", cassandra_host) \
    .config("spark.cassandra.connection.port", cassandra_port) \
    .config("spark.cassandra.auth.username", cassandra_username) \
    .config("spark.cassandra.auth.password", cassandra_password) \
    .getOrCreate()

input_df = spark \
    .readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers", "kafka:9092") \
    .option("subscribe", "kairouan_sales") \
    .option("startingOffsets", "earliest") \
    .load()

df = input_df \
    .selectExpr("CAST(value AS STRING)") \
    .select(from_json(col("value"), schema).alias("kairouan_sales")) \
    .select("kairouan_sales.*") \

query1 = df \
    .writeStream \
    .trigger(processingTime="1 seconds") \
    .foreachBatch(save_to_cassandra) \
    .outputMode("update") \
    .start()

query1.awaitTermination()
```