Code Logic - Retail Data Analysis

Logic for Python Script ‘spark-streaming.py’

Importing Dependencies :

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *
from pyspark.sql.functions import from_json
from pyspark.sql.window import Window
```

Writing the Python functions, which contain the logic for the UDFs

is_a_order: To determine if invoice is for an order or not I used an if-else statement

```
def is_a_order(type):
   if type=="ORDER":
      return 1
   else:
      return 0
```

is_a_return: To determine if invoice is for a return or not I used an if-else statement

```
def is_a_return(type):
   if type=="RETURN":
      return 1
   else:
      return 0
```

total_item_count: To calculate the number of products in every invoice I added the quantity ordered of each product in that invoice

```
def total_item_count(items):
   total_count = 0
   for item in items:
      total_count = total_count + item['quantity']
   return total_count
```

total_cost: - To calculate the total income from every invoice I needed to calculate the income from sale of each product, so I multiplied the unit price of the product with the quantity of the product purchased. The sum of this cost across the products in that invoice gives me the total cost of the order. I also made sure that if the transaction is a return transaction, the total cost is negative.

```
def total_cost(items,type):
  total_price = 0
  for item in items:
     total_price = total_price + item['unit_price'] * item['quantity']
  if type=="RETURN":
     return total_price * (-1)
  else:
     return total_price
```

Initializing the Spark session and setting the log level to error as a good practice

```
spark = SparkSession  \
       .builder  \
       .appName("StructuredSocketRead")  \
       .getOrCreate()
spark.sparkContext.setLogLevel('ERROR')
```

Reading input data from Kafka mentioning the details of the Kafka broker, such as bootstrap server, port and topic name

```
raw_order = spark  \
       .readStream  \
       .format("kafka")  \
       .option("kafka.bootstrap.servers","18.211.252.152:9092")  \
       .option("subscribe","real-time-project")  \
       .option("startingOffsets", "latest")  \
       .load()
```

Defining JSON schema of each order, using appropriate datatypes and StrucField in the case of the item attributes

```
JSON_Schema = StructType() \
        .add("invoice_no", LongType()) \
              .add("country",StringType()) \
        .add("timestamp", TimestampType()) \
        .add("type", StringType()) \
        .add("total_items",IntegerType())\
        .add("is_order",IntegerType()) \
        .add("is_return",IntegerType()) \
        .add("items", ArrayType(StructType([
        StructField("SKU", StringType()),
        StructField("title", StringType()),
        StructField("unit_price", FloatType()),
        StructField("quantity", IntegerType())
        ])))
```

Reading the raw JSON data from Kafka as ‘order stream’ by casting it to string and storing it into the alias ‘data’

```
order_stream = raw_order.select(from_json(col("value").cast("string"),
JSON_Schema).alias("data")).select("data.*")
```

Defining the UDFs by Converting the Python functions I defined earlier, and assigning the appropriate return datatype

```
is_order = udf(is_a_order, IntegerType())
is_return = udf(is_a_return, IntegerType())
add_total_item_count = udf(total_item_count, IntegerType())
add_total_cost = udf(total_cost, FloatType())
```

Calculating the additional columns according to the required input values

```
order_extended_stream = order_stream \
      .withColumn("total_items", add_total_item_count(order_stream.items)) \
      .withColumn("total_cost", add_total_cost(order_stream.items,order_stream.type)) \
      .withColumn("is_order", is_order(order_stream.type)) \
      .withColumn("is_return", is_return(order_stream.type))
```

Writing the summarised input values to console, using ‘append’ output method and applying truncate as false and setting the processing time to 1 minute

```
order_query_console = order_extended_stream \
      .select("invoice_no", "country", "timestamp","total_cost","total_items","is_order","is_return") \
      .writeStream \
      .outputMode("append") \
      .format("console") \
      .option("truncate", "false") \
      .trigger(processingTime="1 minute") \
      .start()
```

Calculating time-based KPIs (Total sale volume, OPM, Rate of return, Average transaction size) having tumbling window of one minute and watermark of one minute.

```
agg_time = order_extended_stream \
   .withWatermark("timestamp","1 minutes") \
   .groupby(window("timestamp", "1 minute")) \
   .agg(sum("total_cost").alias("total_volume_of_sales"),
      avg("total_cost").alias("average_transaction_size"),
      avg("is_Return").alias("rate_of_return")) \

.select("window.start","window.end","total_volume_of_sales","average_transaction_size","rate_
of_return")
```

Writing the time-based KPIs data to HDFS - HDFS into JSON files for each one-minute window, using ‘append’ output mode, setting truncate as false, and specifying the HDFS output path for both the KPI files and for their checkpoints. Ten 1-minute window batches were taken

```
ByTime = agg_time.writeStream \
   .format("json") \
   .outputMode("append") \
   .option("truncate", "false") \
   .option("path", "time_kpi/") \
   .option("checkpointLocation", "time_kpi/cp/") \
   .trigger(processingTime="1 minutes") \
   .start()
```

Calculating time-and-country-based KPIs (Total sale volume, OPM, Rate of return) having tumbling window of one minute and watermark of one minute. Here I grouped by window and country both

```
agg_time_country = order_extended_stream \
    .withWatermark("timestamp", "1 minutes") \
    .groupBy(window("timestamp", "1 minutes"), "country") \
    .agg(sum("total_cost").alias("total_volume_of_sales"),
        count("invoice_no").alias("OPM"),
        avg("is_Return").alias("rate_of_return")) \
    .select("window.start","window.end","country",
"OPM","total_volume_of_sales","rate_of_return")
```

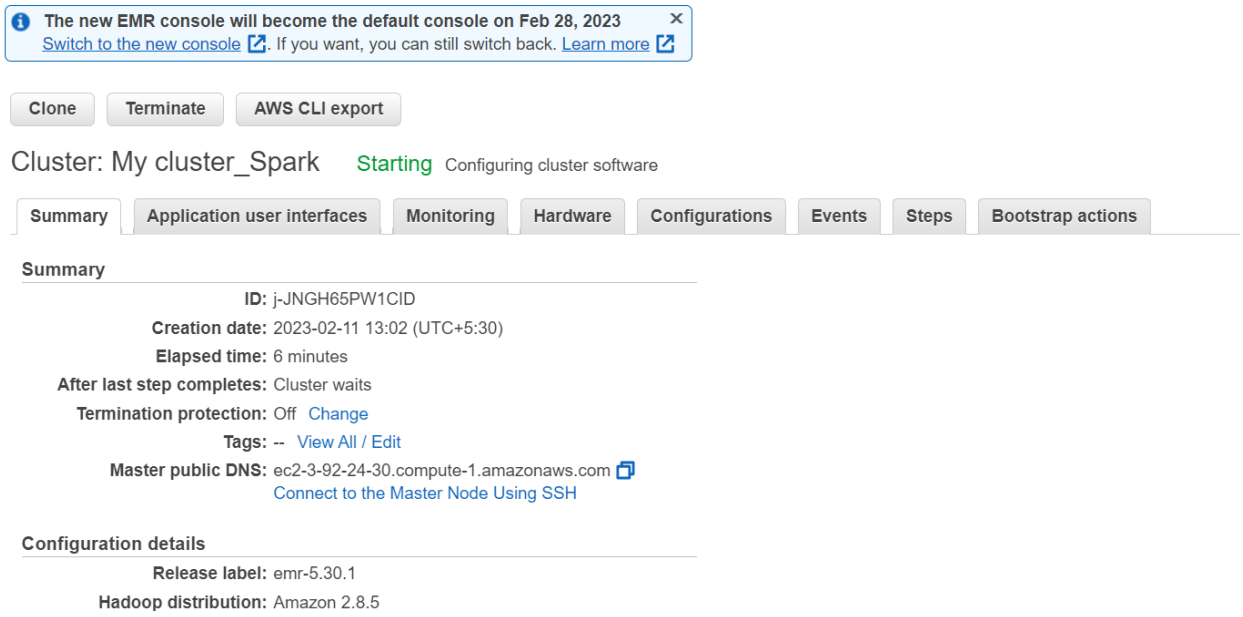
Writing the the time-and-country-based KPIs data to HDFS into JSON files for each one-minute window, using 'append' output mode, setting truncate as false, and specifying the HDFS output path for both the KPI files and for their checkpoints. Ten 1-minute window batches were taken.

```
ByTime_country = agg_time_country.writeStream \
    .format("json") \
    .outputMode("append") \
    .option("truncate", "false") \
    .option("path", "country_kpi/") \
    .option("checkpointLocation", "country_kpi/cp/") \
    .trigger(processingTime="1 minutes") \
    .start()
```

Indicating Spark to await termination

```
ByTime_country.awaitTermination()
ByTime.awaitTermination()
order_query_console.awaitTermination()
```

Console Commands



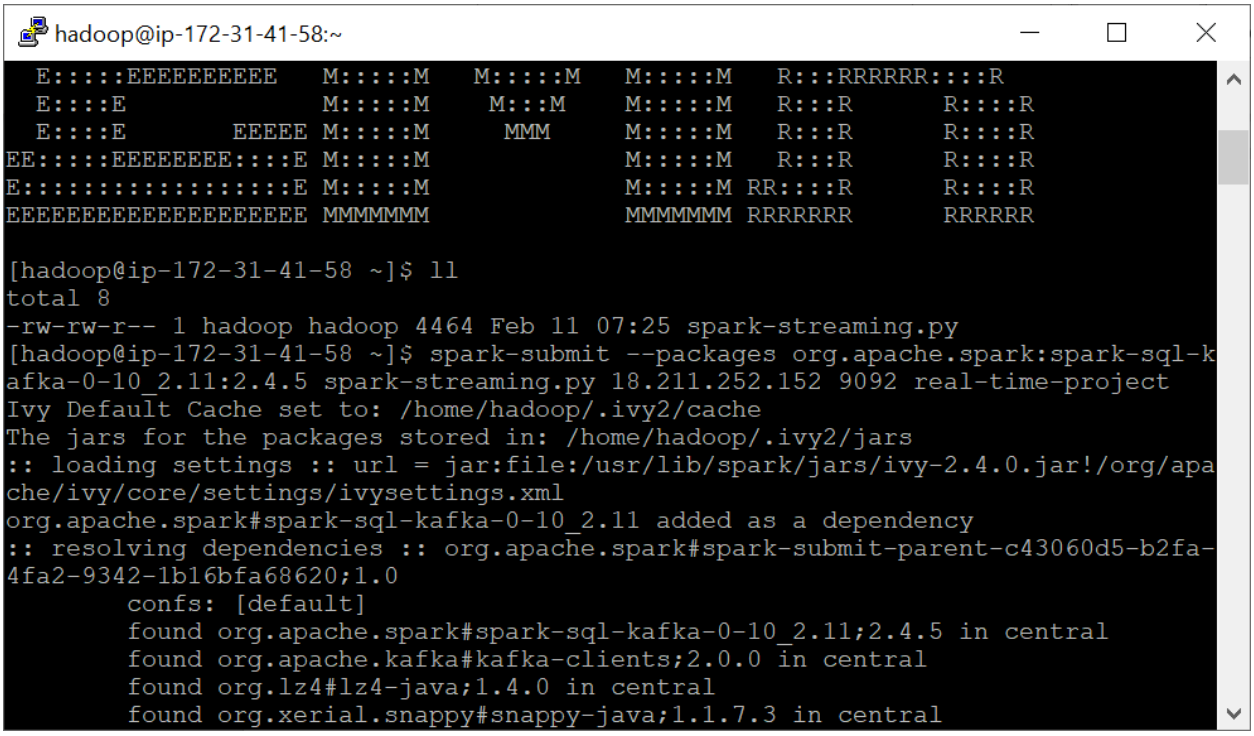
I started by logging into the ec2 instance as ‘hadoop’ user.

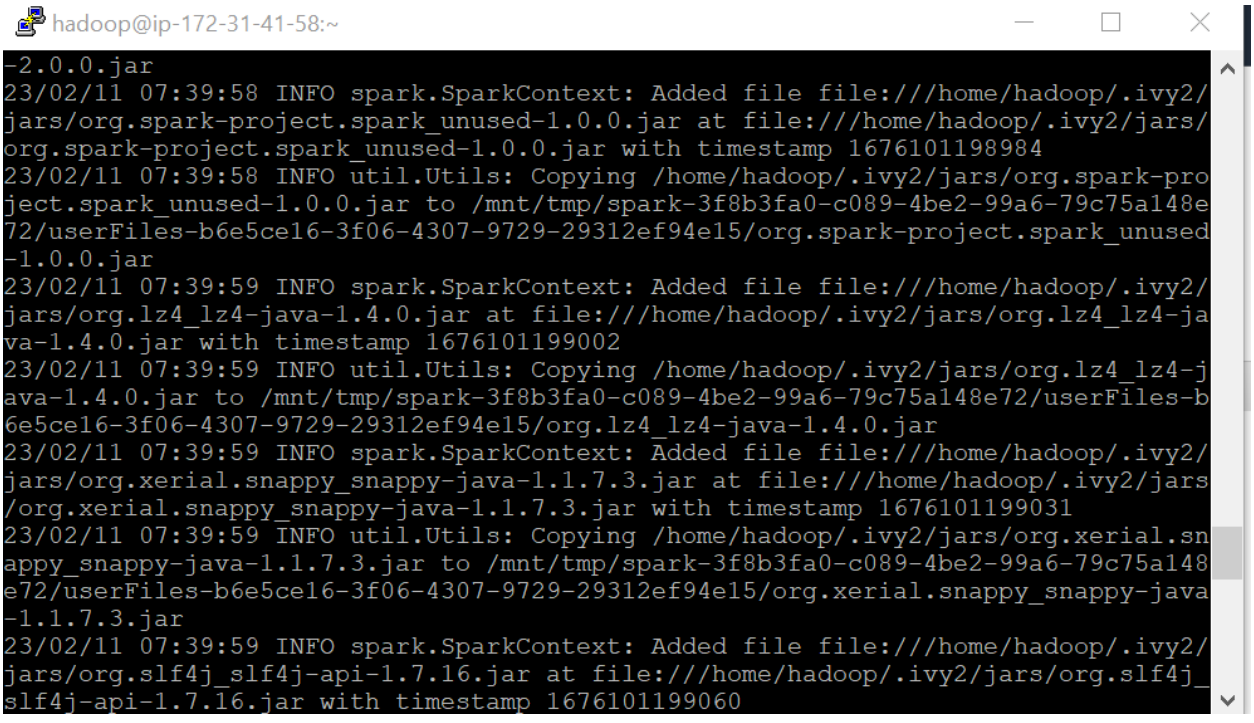
Setting Environment before running the Spark Job:

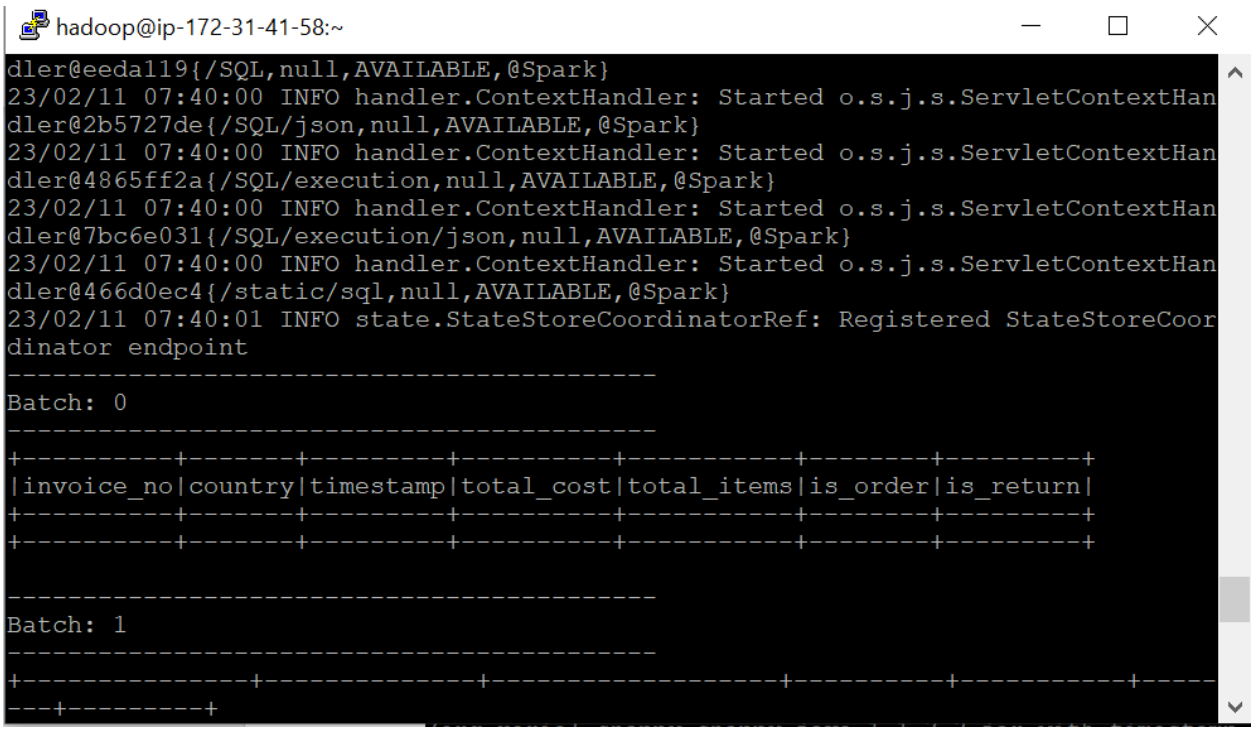
export SPARK_KAFKA_VERSION=0.10

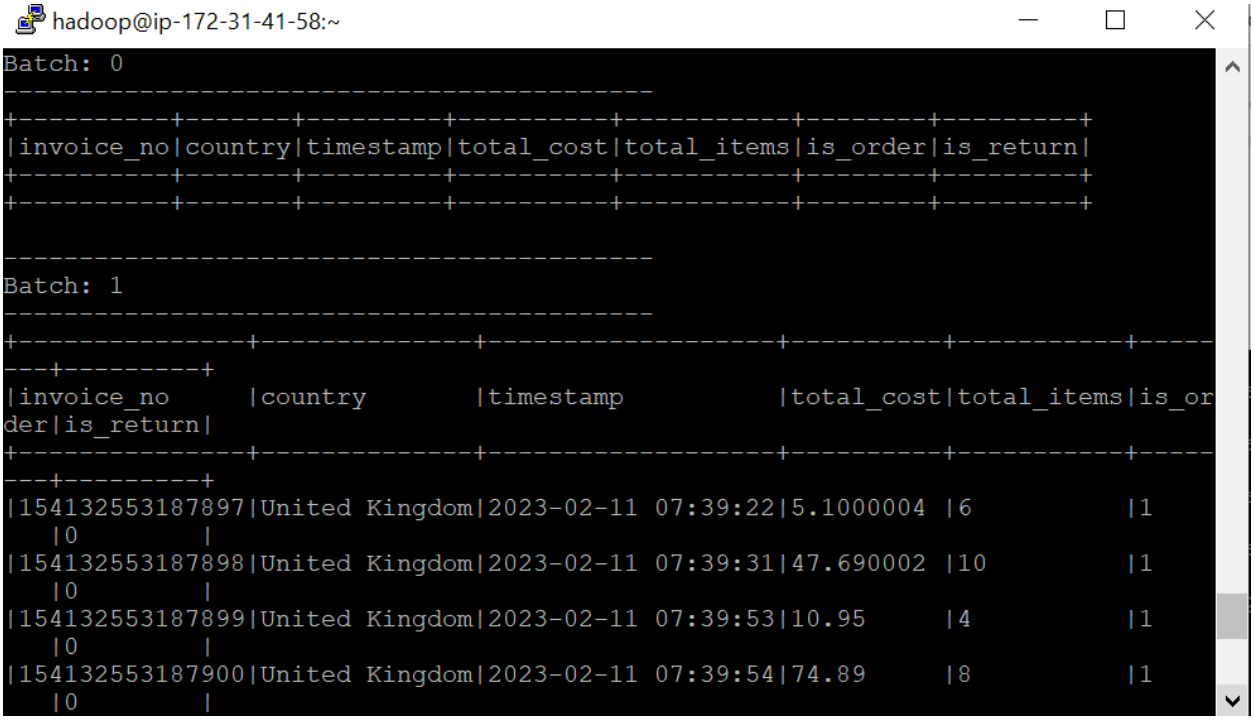
Commands used to run the Spark Submit job:

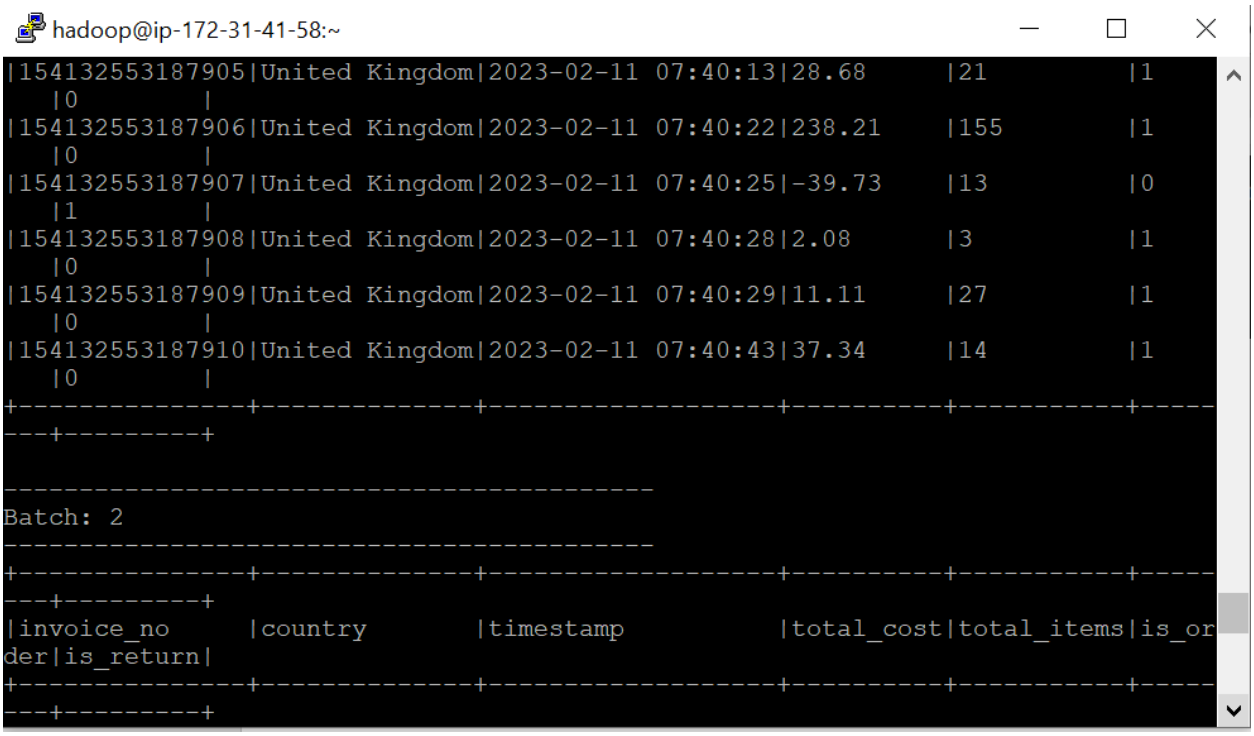
hadoop@ip-172-31-36-187 ~]$ spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 spark-streaming.py 18.211.252.152 9092 real-time-project

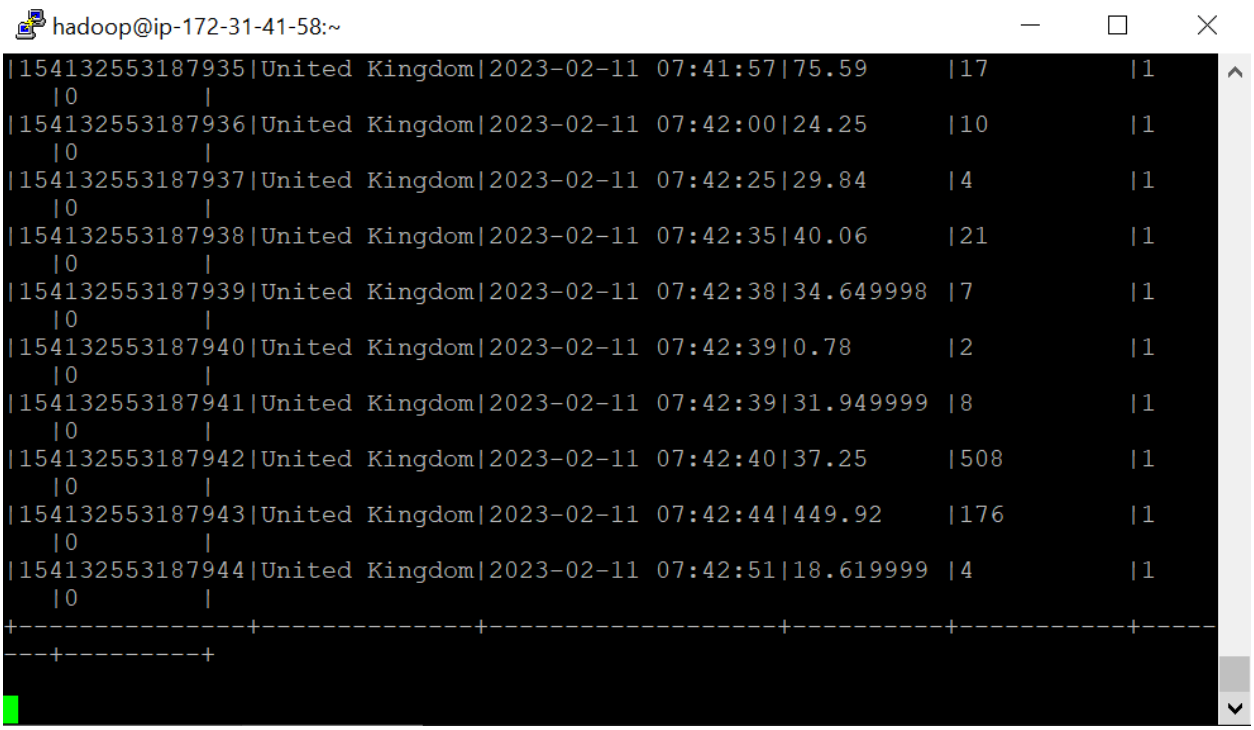


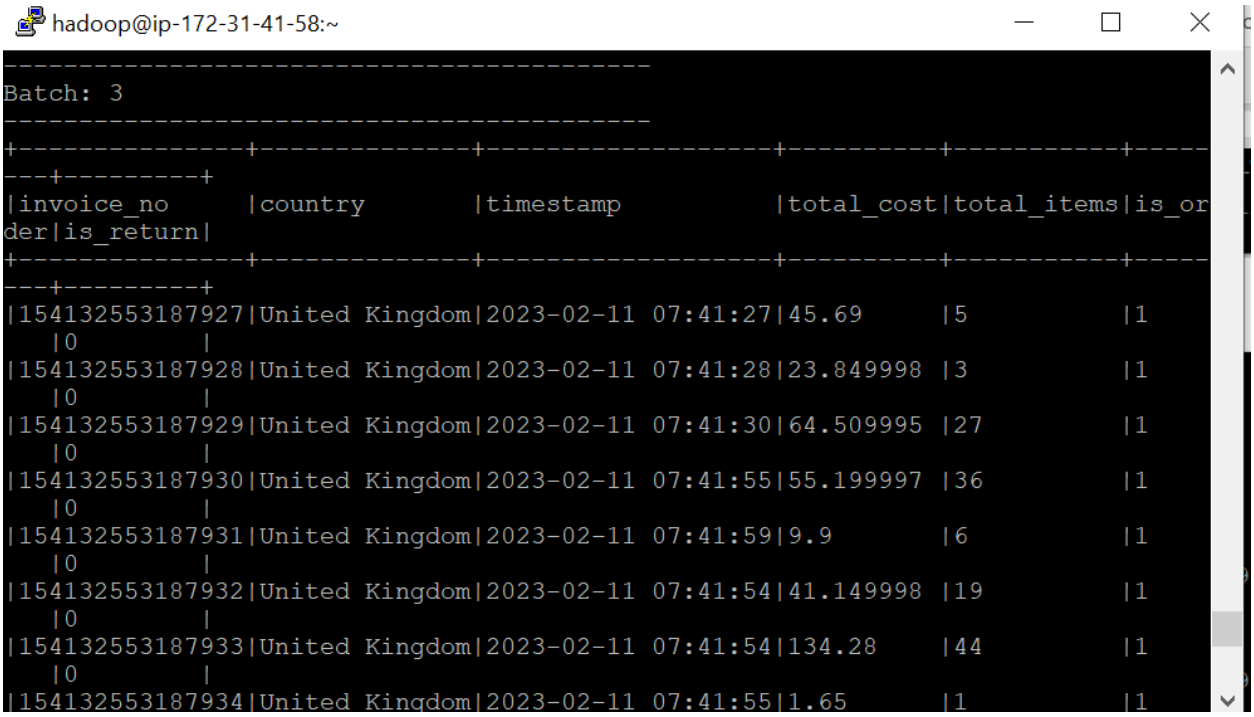


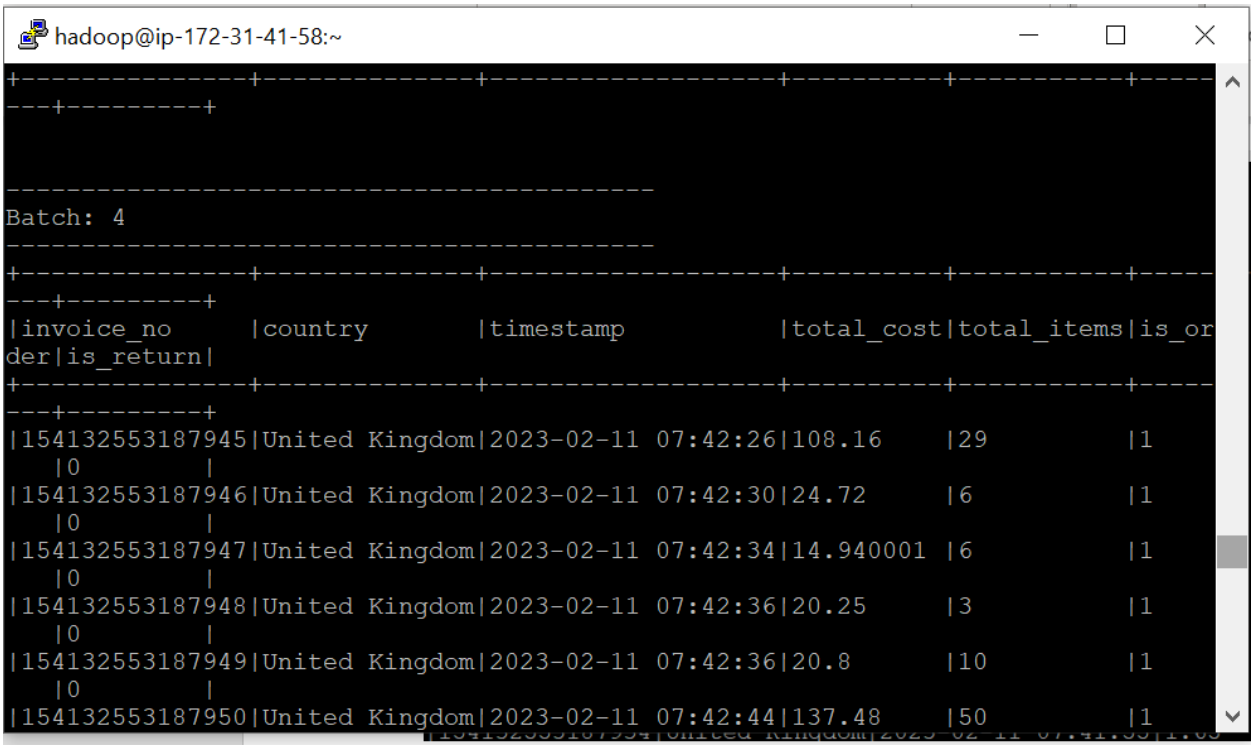


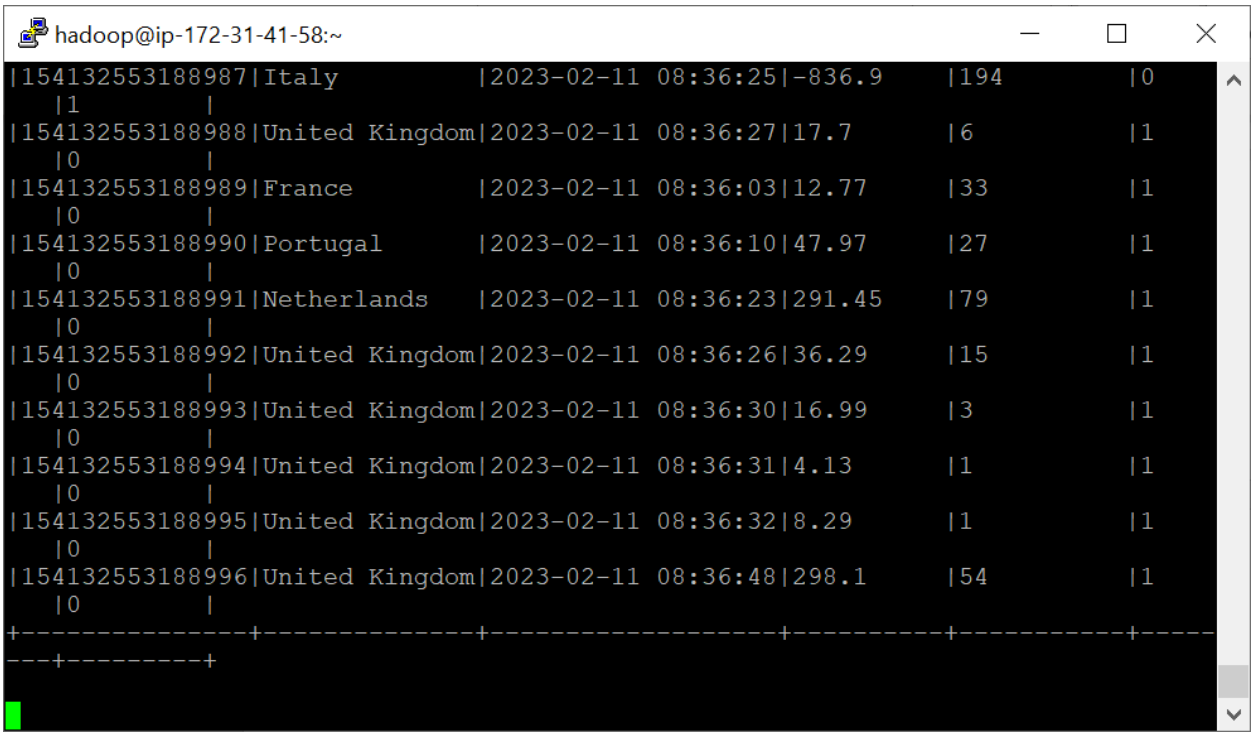


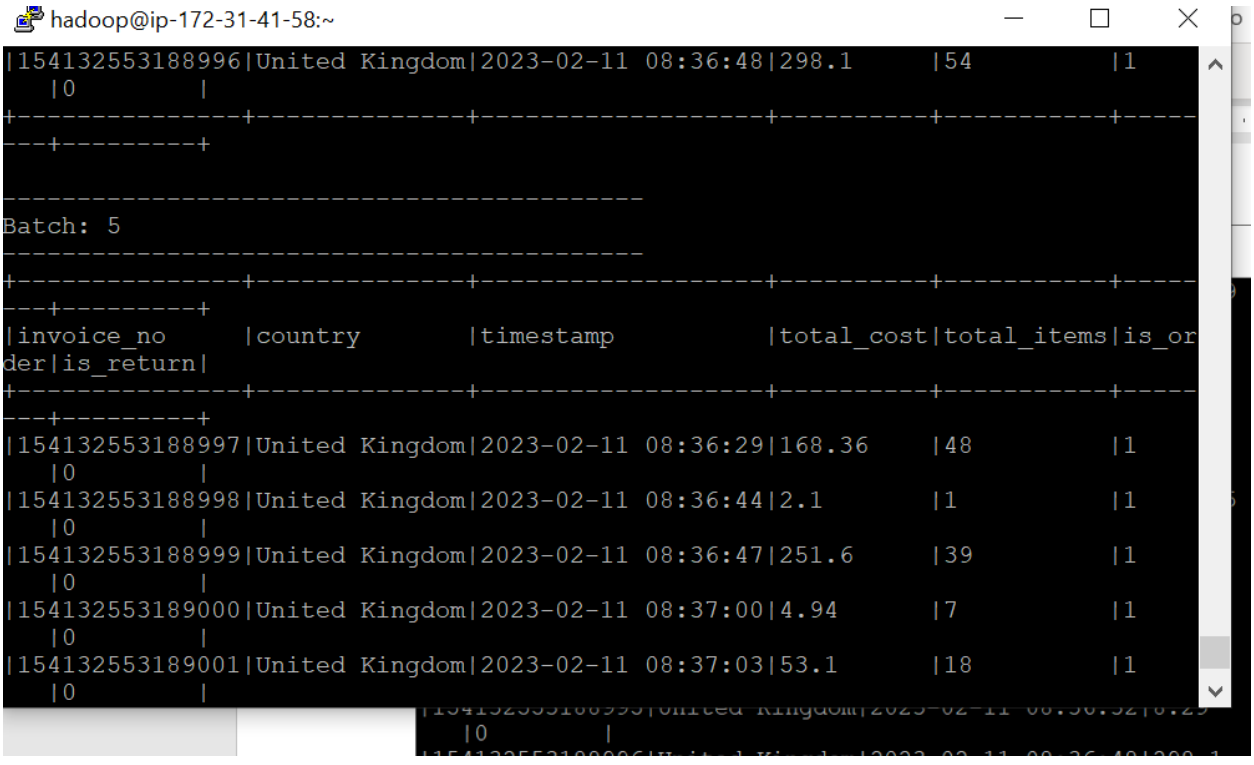


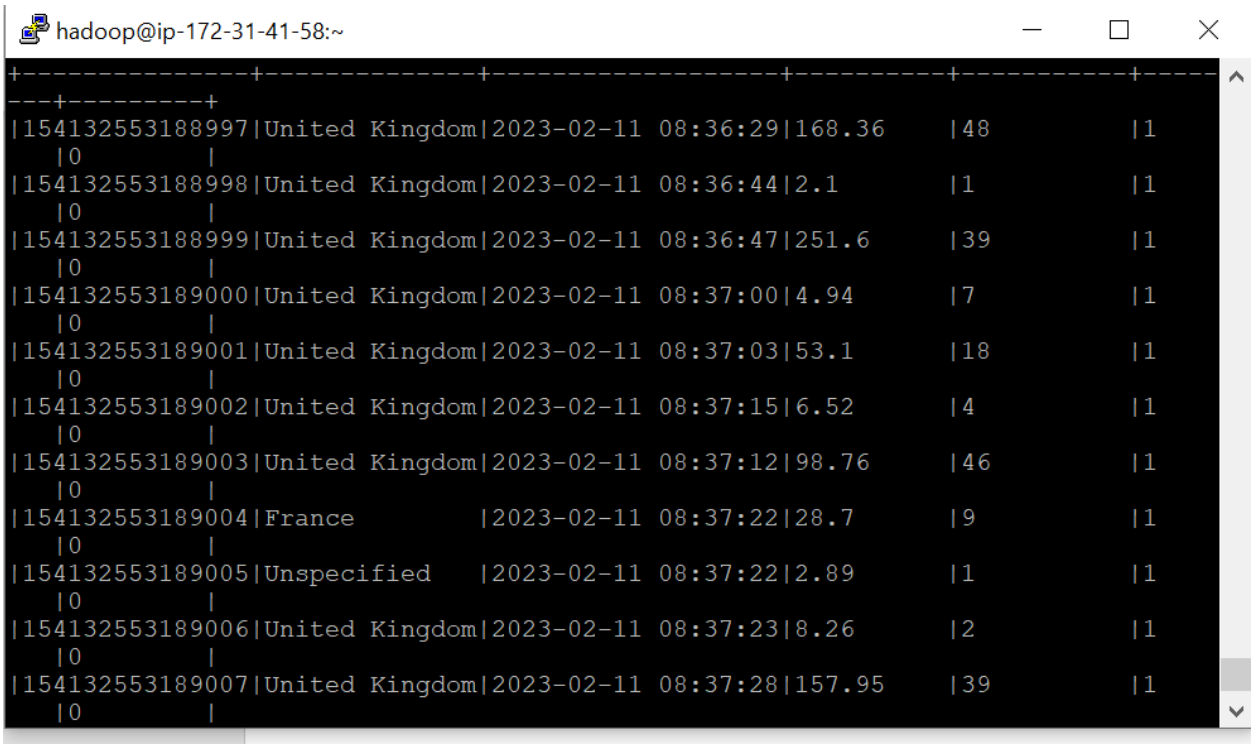


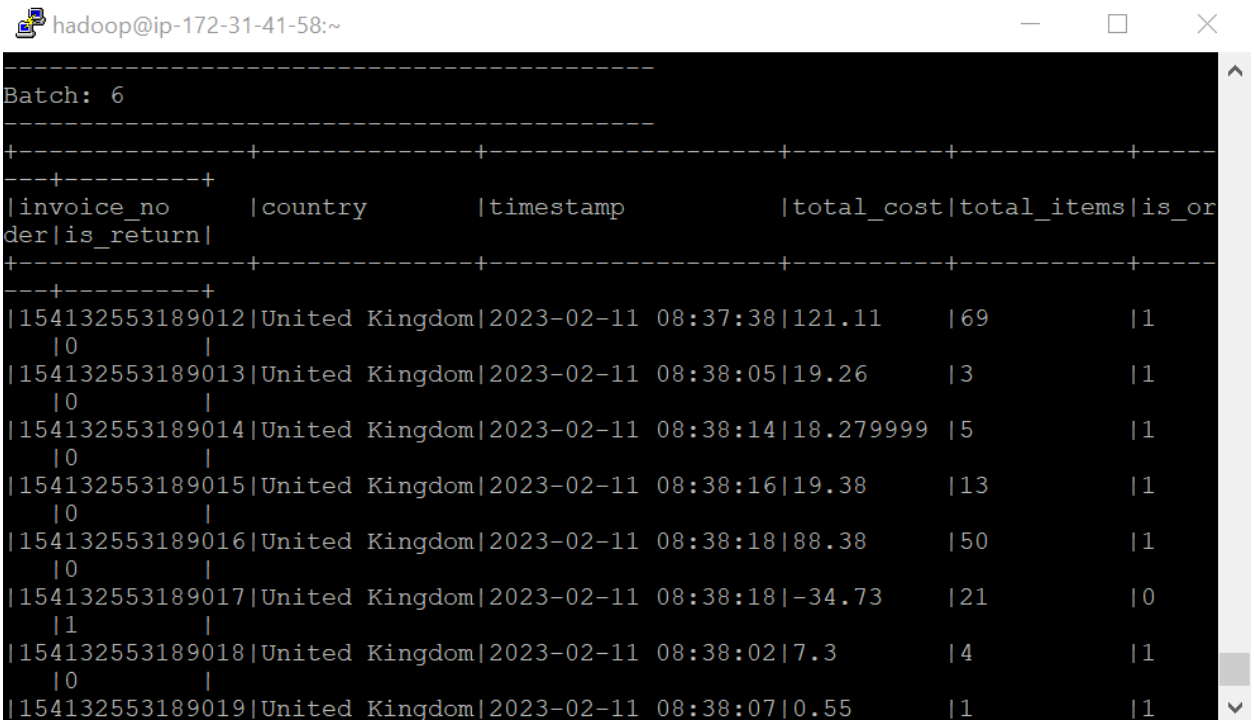


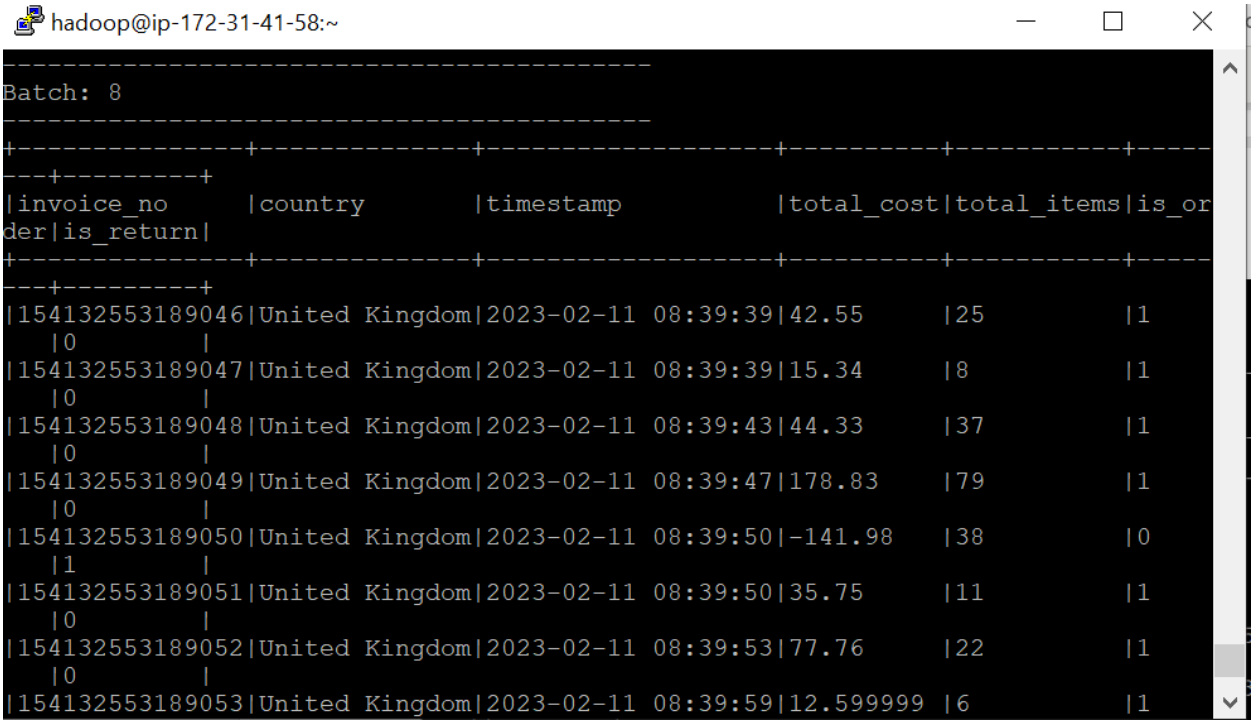










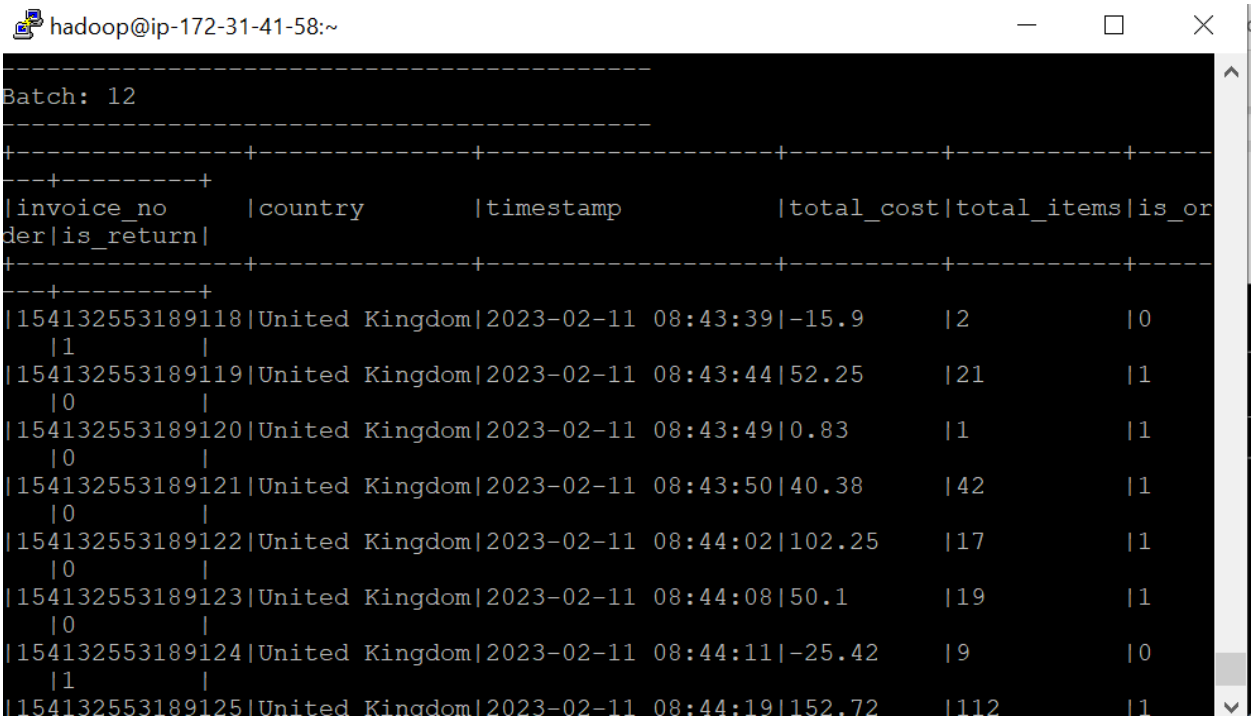


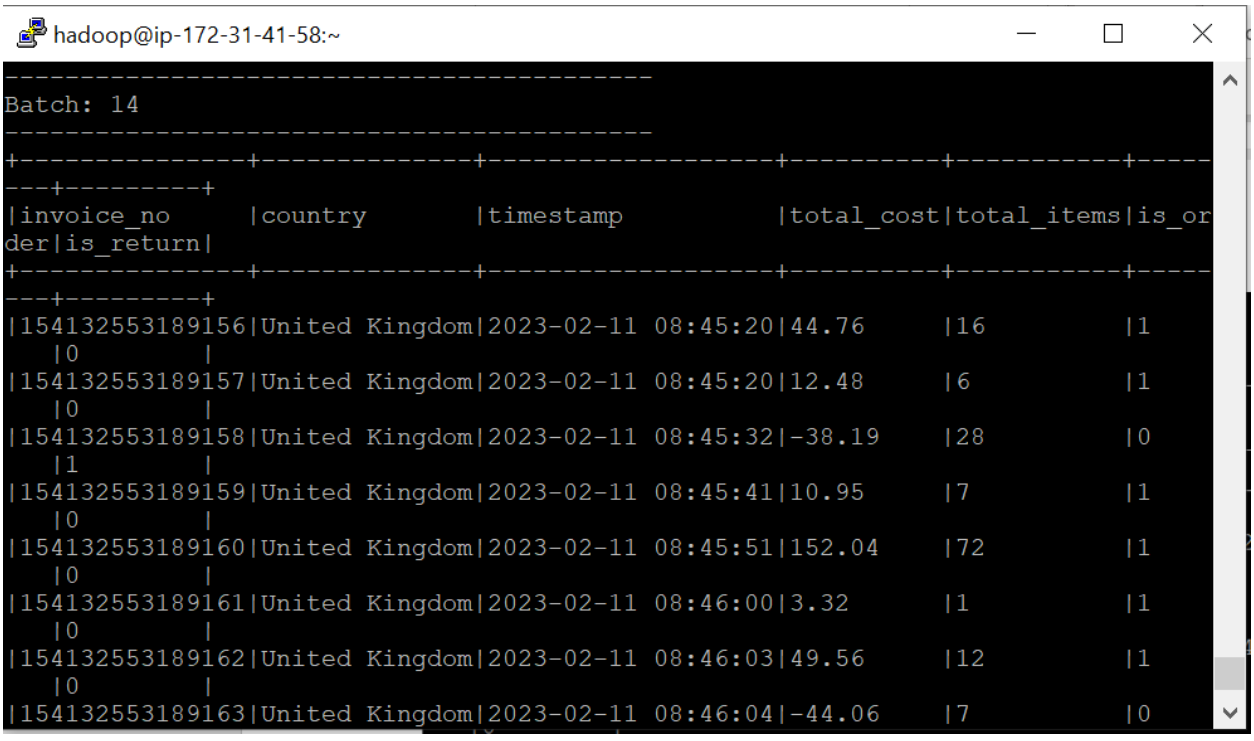
hadoop@ip-172-31-41-58:~

```
Batch: 9
-------------------------------------------
+--------------+-------------+-------------------+----------+-----------+-----
---+---------+
|invoice_no    |country      |timestamp          |total_cost|total_items|is_or
der|is_return|
+--------------+-------------+-------------------+----------+-----------+-----
---+---------+
|154132553189069|Germany      |2023-02-11 08:40:37|15.25     |6          |1
   |0        |
|154132553189070|France       |2023-02-11 08:40:43|116.56    |56         |1
   |0        |
|154132553189071|United Kingdom|2023-02-11 08:40:49|91.92     |17         |1
   |0        |
|154132553189072|United Kingdom|2023-02-11 08:40:53|89.38     |19         |1
   |0        |
|154132553189073|United Kingdom|2023-02-11 08:40:58|112.549995|2          |1
   |0        |
|154132553189074|United Kingdom|2023-02-11 08:40:59|46.4      |12         |1
   |0        |
|154132553189075|United Kingdom|2023-02-11 08:40:59|97.45     |46         |1
   |0        |
|154132553189076|United Kingdom|2023-02-11 08:41:12|-16.65    |13         |0
   |1        |
```

hadoop@ip-172-31-41-58:~

```
Batch: 10
-------------------------------------------
+--------------+-------------+-------------------+----------+-----------+-----
---+---------+
|invoice_no    |country      |timestamp          |total_cost|total_items|is_or
der|is_return|
+--------------+-------------+-------------------+----------+-----------+-----
---+---------+
|154132553189081|United Kingdom|2023-02-11 08:41:25|343.0     |77         |1
   |0        |
|154132553189082|United Kingdom|2023-02-11 08:41:33|40.800003 |48         |1
   |0        |
|154132553189083|United Kingdom|2023-02-11 08:41:42|60.089996 |31         |1
   |0        |
|154132553189084|United Kingdom|2023-02-11 08:42:19|36.739998 |14         |1
   |0        |
|154132553189085|United Kingdom|2023-02-11 08:41:58|25.25     |7          |1
   |0        |
|154132553189086|United Kingdom|2023-02-11 08:41:59|20.650002 |5          |1
   |0        |
|154132553189087|United Kingdom|2023-02-11 08:42:02|42.42     |28         |1
   |0        |
|154132553189088|United Kingdom|2023-02-11 08:42:04|17.55     |13         |1
   |0        |
```





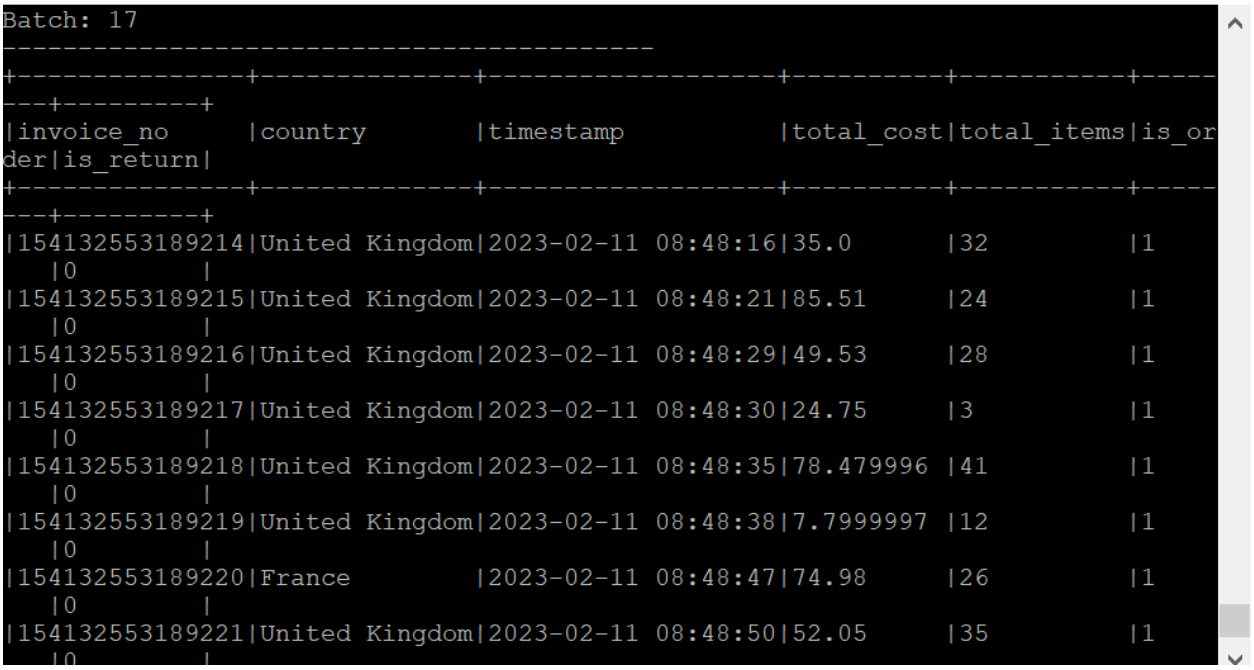
hadoop@ip-172-31-41-58:~

```
-------------------------------------------
Batch: 15
-------------------------------------------
+---------------+--------------+-------------------+----------+-----------+-----
---+---------+
|invoice_no     |country       |timestamp          |total_cost|total_items|is_or
der|is_return|
+---------------+--------------+-------------------+----------+-----------+-----
---+---------+
|154132553189172|United Kingdom|2023-02-11 08:46:18|27.85     |49         |1
   |0        |
|154132553189173|United Kingdom|2023-02-11 08:46:19|30.14     |16         |1
   |0        |
|154132553189174|United Kingdom|2023-02-11 08:46:21|22.2      |16         |1
   |0        |
|154132553189175|United Kingdom|2023-02-11 08:46:25|74.1      |26         |1
   |0        |
|154132553189176|United Kingdom|2023-02-11 08:46:26|30.0      |24         |1
   |0        |
|154132553189177|United Kingdom|2023-02-11 08:46:29|135.3     |77         |1
   |0        |
|154132553189178|United Kingdom|2023-02-11 08:46:33|28.72     |14         |1
   |0        |
|154132553189179|United Kingdom|2023-02-11 08:46:48|31.159998 |8          |1
```

hadoop@ip-172-31-41-58:~

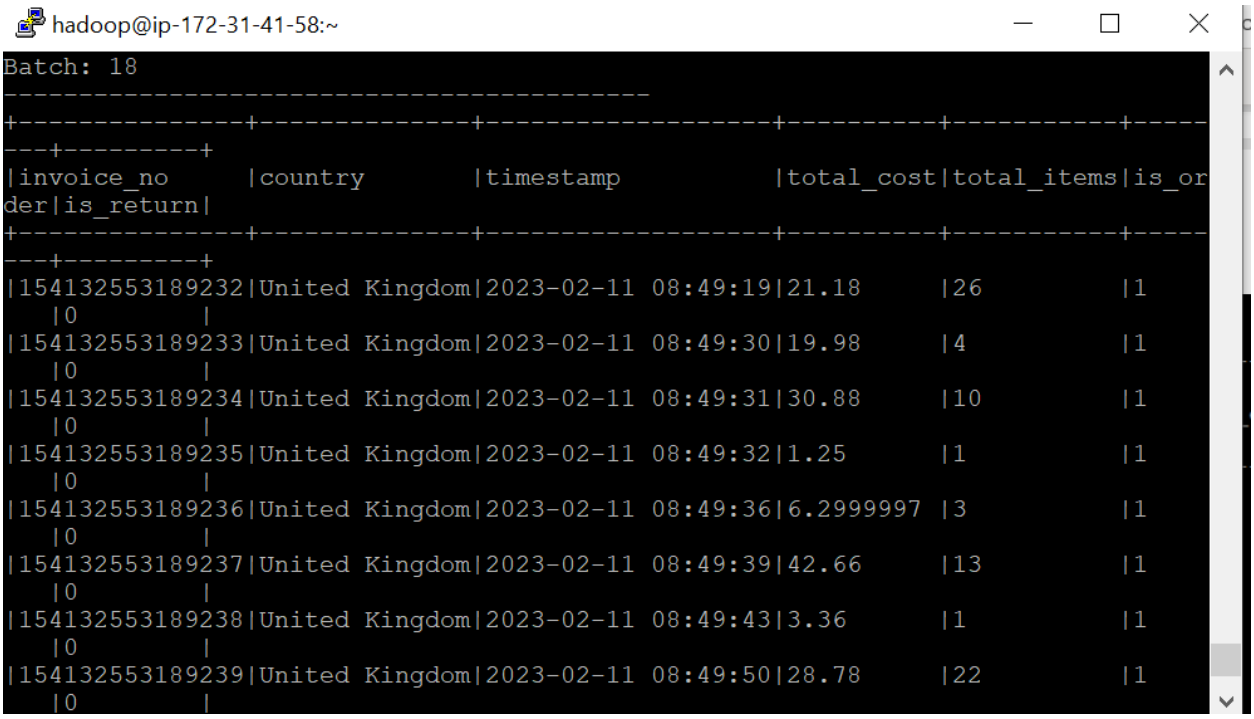
```
-------------------------------------------
Batch: 16
-------------------------------------------
+---------------+--------------+-------------------+----------+-----------+-----
---+---------+
|invoice_no     |country       |timestamp          |total_cost|total_items|is_or
der|is_return|
+---------------+--------------+-------------------+----------+-----------+-----
---+---------+
|154132553189194|United Kingdom|2023-02-11 08:47:23|55.2      |22         |1
   |0        |
|154132553189195|United Kingdom|2023-02-11 08:47:38|51.739998 |20         |1
   |0        |
|154132553189196|United Kingdom|2023-02-11 08:47:53|25.25     |13         |1
   |0        |
|154132553189197|United Kingdom|2023-02-11 08:47:56|80.33     |27         |1
   |0        |
|154132553189198|United Kingdom|2023-02-11 08:48:00|140.75    |111        |1
   |0        |
|154132553189199|United Kingdom|2023-02-11 08:48:02|76.049995 |16         |1
   |0        |
|154132553189200|United Kingdom|2023-02-11 08:48:04|14.78     |3          |1
   |0        |
|154132553189201|United Kingdom|2023-02-11 08:48:07|16.6      |4          |1
```

hadoop@ip-172-31-41-58:~

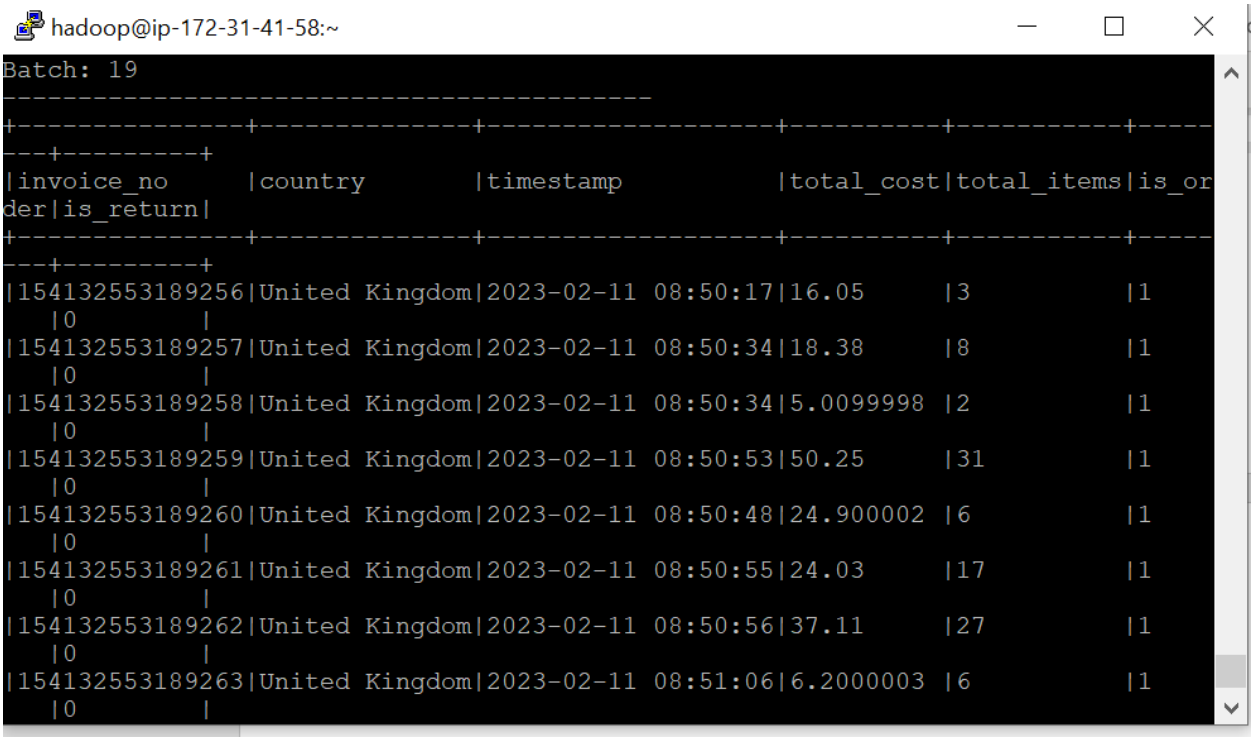


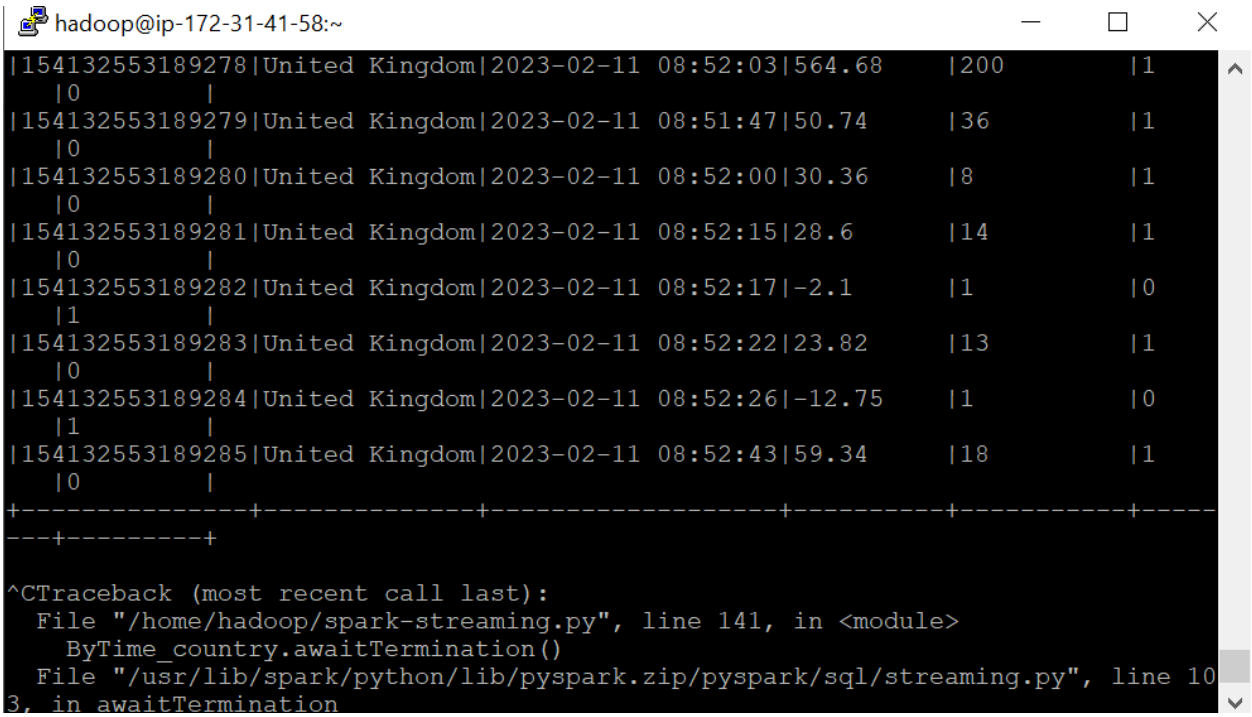
```
Batch: 17
-------------------------------------------
+--------------+--------------+-------------------+----------+-----------+-----
---+---------+
|invoice_no    |country       |timestamp          |total_cost|total_items|is_or
der|is_return|
+--------------+--------------+-------------------+----------+-----------+-----
---+---------+
|154132553189214|United Kingdom|2023-02-11 08:48:16|35.0      |32         |1
   |0        |
|154132553189215|United Kingdom|2023-02-11 08:48:21|85.51     |24         |1
   |0        |
|154132553189216|United Kingdom|2023-02-11 08:48:29|49.53     |28         |1
   |0        |
|154132553189217|United Kingdom|2023-02-11 08:48:30|24.75     |3          |1
   |0        |
|154132553189218|United Kingdom|2023-02-11 08:48:35|78.479996 |41         |1
   |0        |
|154132553189219|United Kingdom|2023-02-11 08:48:38|7.7999997 |12         |1
   |0        |
|154132553189220|France        |2023-02-11 08:48:47|74.98     |26         |1
   |0        |
|154132553189221|United Kingdom|2023-02-11 08:48:50|52.05     |35         |1
   |0        |
```

hadoop@ip-172-31-41-58:~



```
Batch: 18
-------------------------------------------
+--------------+--------------+-------------------+----------+-----------+-----
---+---------+
|invoice_no    |country       |timestamp          |total_cost|total_items|is_or
der|is_return|
+--------------+--------------+-------------------+----------+-----------+-----
---+---------+
|154132553189232|United Kingdom|2023-02-11 08:49:19|21.18     |26         |1
   |0        |
|154132553189233|United Kingdom|2023-02-11 08:49:30|19.98     |4          |1
   |0        |
|154132553189234|United Kingdom|2023-02-11 08:49:31|30.88     |10         |1
   |0        |
|154132553189235|United Kingdom|2023-02-11 08:49:32|1.25      |1          |1
   |0        |
|154132553189236|United Kingdom|2023-02-11 08:49:36|6.2999997 |3          |1
   |0        |
|154132553189237|United Kingdom|2023-02-11 08:49:39|42.66     |13         |1
   |0        |
|154132553189238|United Kingdom|2023-02-11 08:49:43|3.36      |1          |1
   |0        |
|154132553189239|United Kingdom|2023-02-11 08:49:50|28.78     |22         |1
   |0        |
```






```
[hadoop@ip-172-31-88-120 ~]$ hadoop fs -ls
Found 3 items
drwxr-xr-x   - hadoop hadoop          0 2023-02-07 10:54 .sparkStaging
drwxr-xr-x   - hadoop hadoop          0 2023-02-07 10:32 country_kpi
drwxr-xr-x   - hadoop hadoop          0 2023-02-07 10:32 time_kpi
[hadoop@ip-172-31-88-120 ~]$
```