Code Logic - Retail Data Analysis

In this document, you will describe the code and the overall steps taken to solve the project.

logic for python script file named "spark-streaming.py"

# Setting up the important libraries for the module

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *
import pyspark.sql.functions as F
```

# Writing the python function which contain logic for the UDFs

```
# UDF1 to determine Is_Order in case of Order.

def udf1(a):
    if a == 'ORDER':
        return 1
    else:
        return 0
# UDF2 to determine Is_Return in case of Return.

def udf2(a):
    if a == 'RETURN':
        return 1
    else:
        return 0

# UDF3 to determine Total Order Cost.

def udf3(a, b, c):
    if a == 'ORDER':
        return b * c
    else:
        return b * c * -1
```

# Initialising the spark session and setting up the log level to error as a good practice

```
# Establishing Spark Session

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

# Reading the input data from kafka mentioning the deatails of kafka broker such as bootstrap server, port and topic name

```
# Reading data from Kafka Server & Topic given

lines = spark.readStream.format('kafka'
                                ).option('kafka.bootstrap.servers',
        '18.211.252.152:9092').option('subscribe', 'real-time-project'
        ).option('failOnDataLoss', 'false').option('startingOffsets',
        'earliest').load()
```

# Defining schema for each order , using appropriate datatypes and StructField in case of item attribute

```
schema = StructType([StructField('country', StringType()),
                    StructField('invoice_no', LongType()),
                    StructField('items',
                    ArrayType(StructType([StructField('SKU',
                    StringType()), StructField('title', StringType()),
                    StructField('unit_price', FloatType()),
                    StructField('quantity', IntegerType())]))),
                    StructField('timestamp', TimestampType()),
                    StructField('type', StringType())])
```

# Reading the raw json data from kafka as 'casted ' by casting it into string and storing it into alias data

```
# Casting raw data as string and aliasing

Casted = lines.select(from_json(col('value').cast('string'),
                      schema).alias('parsed'))
# Parsed DF

new_df = Casted.select('parsed.*')

# unveiling items array to derive new columns & KPIs

df1 = new_df.select(col('type'), col('country'), col('invoice_no'),
                    col('timestamp'), explode(col('items')))
```

# Selecting appropriate columns and renaming few

```
# Columns from Items array displayed

df2 = df1.select(
    'type',
    'country',
    'invoice_no',
    'timestamp',
    'col.SKU',
    'col.title',
    'col.unit_price',
    'col.quantity',
    )

# Array Columns renamed

df2 = df2.withColumnRenamed('col.SKU', 'SKU')
df2 = df2.withColumnRenamed('col.title', 'TITLE')
df2 = df2.withColumnRenamed('col.unit_price', 'unit_price')
df2 = df2.withColumnRenamed('col.quantity', 'quantity')
```

# Declaring the udfs

```
# Declare user def function for total cost

Cost_value = udf(udf3, FloatType())
# Declare user def function Is Order

Det_Order = udf(udf1, IntegerType())

# Declare user def function for Is Return

Det_Return = udf(udf2, IntegerType())
```

# Adding the respective variable to the dataframe

```
# Adding Total Cost to data frame

df2 = df2.withColumn('Cost', Cost_value(df2.type, df2.unit_price,
                     df2.quantity))

# Adding Is Order flag to existing DF

df2 = df2.withColumn('is_Order', Det_Order(df2.type))

# Adding Is Return flag to existing DF

df2 = df2.withColumn('is_Return', Det_Return(df2.type))
```

# Pumping the raw data to the output for each order for window function for 1 minute

```
df3 = df2.withWatermark('timestamp', '10 minutes'
                        ).groupby(window('timestamp', '1 minute'),
                                  'invoice_no', 'country', 'is_Order',
                                  'is_Return').sum('Cost', 'quantity'
        )
```

# Function for timebased KPIs

```
df4 = df2.select(
    'invoice_no',
    'timestamp',
    'Cost',
    'quantity',
    'is_Order',
    'is_Return',
    )
Final_time = df4.withWatermark('timestamp', '10 minutes'
                               ).groupby(window('timestamp', '1
minute'
        )).agg(sum('Cost').alias('Total_sales_vol'),
               F.approx_count_distinct('invoice_no').alias('OPM'),
               sum('is_Order').alias('total_Order'), sum('is_Return'
               ).alias('total_return'), sum('quantity'
               ).alias('total_items'))
```

# KPI for rate of return

```
# KPI for rate of return

Final = Final.withColumn('rate_of_return', Final.total_return
                         / (Final.total_Order + Final.total_return))
Final_country_time = Final.select('window', 'country', 'OPM',
                                  'Total_sales_vol',
'rate_of_return')
```

KPI for average transaction size

```
# KPI for average transaction size....

Final_time = Final_time.withColumn('Avg_trans_size',
                                   Final_time.Total_sales_vol
                                   / (Final_time.total_Order
                                   + Final_time.total_return))
Final_time = Final_time.select('window', 'OPM', 'Total_sales_vol',
                               'Avg_trans_size', 'rate_of_return')
```

# Time and country based KPIs

```
df3_time_country = df2.select(
    'country',
    'invoice_no',
    'timestamp',
    'Cost',
    'quantity',
    'is_Order',
    'is_Return',
    )
Final = df3_time_country.withWatermark('timestamp', '10 minutes'
        ).groupby(window('timestamp', '1 minute'), 'country').agg(
    sum('Cost').alias('Total_sales_vol'),
    F.approx_count_distinct('invoice_no').alias('OPM'),
    sum('invoice_no').alias('sum_invoice'),
    sum('is_Order').alias('total_Order'),
    sum('is_Return').alias('total_return'),
    sum('quantity').alias('total_items'),
    )
```

# Printing the KPIs to HDFS as json file

```
query_2 = Final_time.writeStream.outputMode('Append').format('json'
        ).option('format', 'append').option('truncate', 'false'
        ).option('path', 'time_KPI').option('checkpointLocation',
        'time_KPI_json').trigger(processingTime='1 minute').start()
```

# Printing the output on console

```
# printing output on console

query_1 = df3.writeStream.outputMode('complete').format('console'
        ).option('truncate', 'False').start()

# Printing Time and Country KPI to HDFS as Json file
```

# Printing time and country KPIs to HDFS as json file

```
query_3 = Final_country_time.writeStream.outputMode('Append'
        ).format('json').option('format', 'append').option('truncate',
        'false').option('path', 'time_country_KPI'
                        ).option('checkpointLocation',
                                 'time_country_KPI_json'
                                 ).trigger(processingTime='1 minute'
        ).start()
```

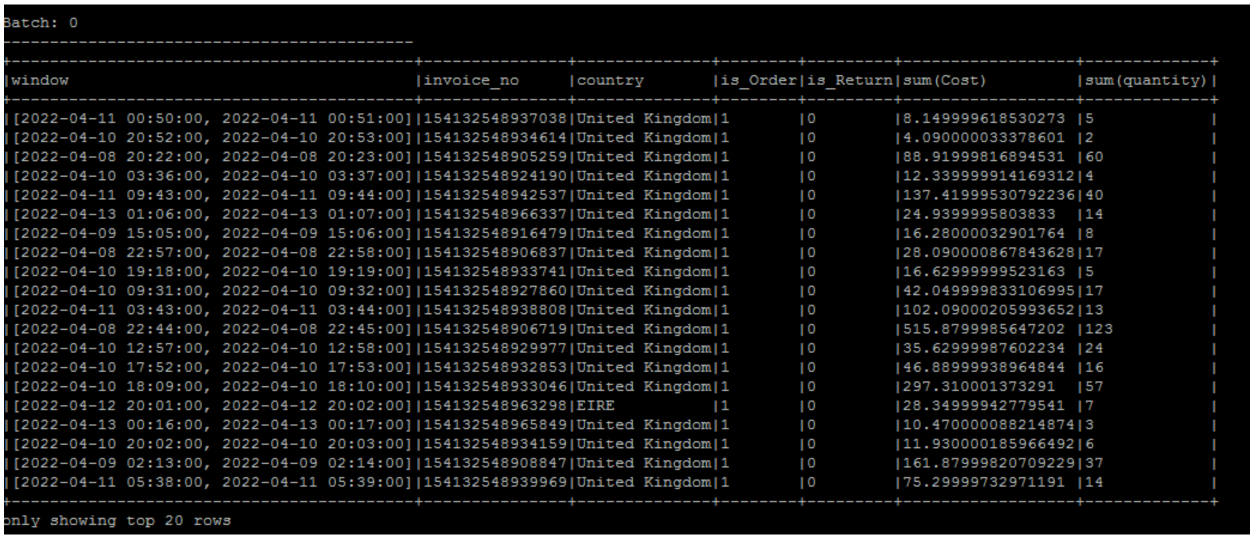
# Qwery termination command

```
query_1.awaitTermination()
query_2.awaitTermination()
query_3.awaitTermination()
```

Console Command:

- Setup EMR Cluster for spark streaming and login as hadoop
- go to root user by sudo -i
- install pip install kafka-python
- Next, open another console and login as hadoop and create spark-streaming.py file by "vi spark-streaming.py"
- Next, i set the kafka version using the following command: "export PYSPARK_KAFKA_VERSION=0.10.
- After writing the code in the spark-streaming.py we can run the sparkjon by the following command: spark-submit --packages org.apache.spark:spark-sql-kafka-0-10_2.11:2.4.5 spark-streaming.py.

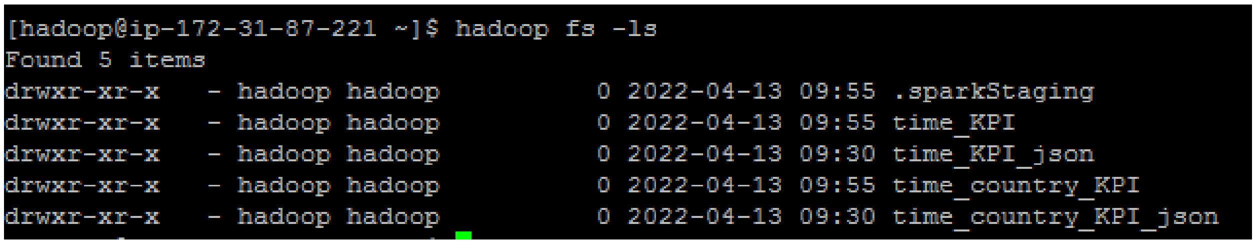
Final summarised Input Values



```
Batch: 0
-------------------------------------------
+----------------------------------------+---------------+--------------+--------+---------+------------------+-------------+
|window                                  |invoice_no     |country       |is_Order|is_Return|sum(Cost)         |sum(quantity)|
+----------------------------------------+---------------+--------------+--------+---------+------------------+-------------+
|[2022-04-11 00:50:00, 2022-04-11 00:51:00]|154132548937038|United Kingdom|1       |0        |8.149999618530273 |5            |
|[2022-04-10 20:52:00, 2022-04-10 20:53:00]|154132548934614|United Kingdom|1       |0        |4.090000033378601 |2            |
|[2022-04-08 20:22:00, 2022-04-08 20:23:00]|154132548905259|United Kingdom|1       |0        |88.91999816894531 |60           |
|[2022-04-10 03:36:00, 2022-04-10 03:37:00]|154132548924190|United Kingdom|1       |0        |12.339999914169312|4            |
|[2022-04-11 09:43:00, 2022-04-11 09:44:00]|154132548942537|United Kingdom|1       |0        |137.41999530792236|40           |
|[2022-04-13 01:06:00, 2022-04-13 01:07:00]|154132548966337|United Kingdom|1       |0        |24.9399995803833  |14           |
|[2022-04-09 15:05:00, 2022-04-09 15:06:00]|154132548916479|United Kingdom|1       |0        |16.28000032901764 |8            |
|[2022-04-08 22:57:00, 2022-04-08 22:58:00]|154132548906837|United Kingdom|1       |0        |28.090000867843628|17           |
|[2022-04-10 19:18:00, 2022-04-10 19:19:00]|154132548933741|United Kingdom|1       |0        |16.62999999523163 |5            |
|[2022-04-10 09:31:00, 2022-04-10 09:32:00]|154132548927860|United Kingdom|1       |0        |42.049999833106995|17           |
|[2022-04-11 03:43:00, 2022-04-11 03:44:00]|154132548938808|United Kingdom|1       |0        |102.09000205993652|13           |
|[2022-04-08 22:44:00, 2022-04-08 22:45:00]|154132548906719|United Kingdom|1       |0        |515.8799985647202 |123          |
|[2022-04-10 12:57:00, 2022-04-10 12:58:00]|154132548929977|United Kingdom|1       |0        |35.62999987602234 |24           |
|[2022-04-10 17:52:00, 2022-04-10 17:53:00]|154132548932853|United Kingdom|1       |0        |46.88999938964844 |16           |
|[2022-04-10 18:09:00, 2022-04-10 18:10:00]|154132548933046|United Kingdom|1       |0        |297.310001373291  |57           |
|[2022-04-12 20:01:00, 2022-04-12 20:02:00]|154132548963298|EIRE          |1       |0        |28.34999942779541 |7            |
|[2022-04-13 00:16:00, 2022-04-13 00:17:00]|154132548965849|United Kingdom|1       |0        |10.470000088214874|3            |
|[2022-04-10 20:02:00, 2022-04-10 20:03:00]|154132548934159|United Kingdom|1       |0        |11.930000185966492|6            |
|[2022-04-09 02:13:00, 2022-04-09 02:14:00]|154132548908847|United Kingdom|1       |0        |161.87999820709229|37           |
|[2022-04-11 05:38:00, 2022-04-11 05:39:00]|154132548939969|United Kingdom|1       |0        |75.29999732971191 |14           |
+----------------------------------------+---------------+--------------+--------+---------+------------------+-------------+
only showing top 20 rows
```

# Check in HDFS that all the required KPIs are present

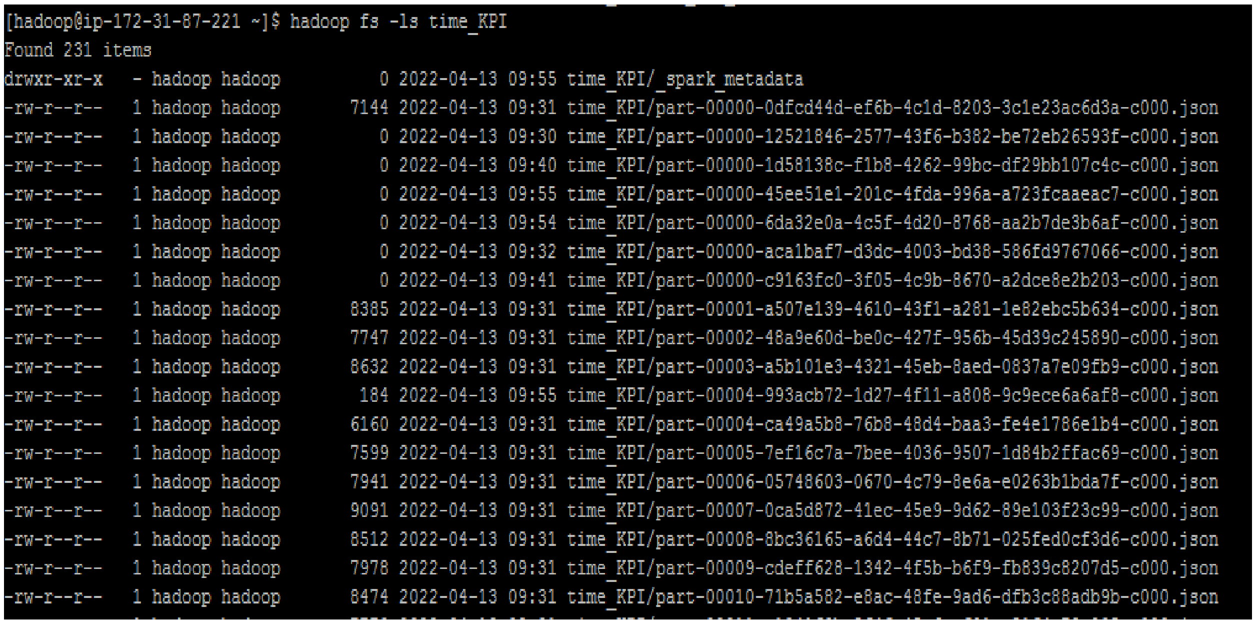
- hadoop fs -ls



```
[hadoop@ip-172-31-87-221 ~]$ hadoop fs -ls
Found 5 items
drwxr-xr-x   - hadoop hadoop          0 2022-04-13 09:55 .sparkStaging
drwxr-xr-x   - hadoop hadoop          0 2022-04-13 09:55 time_KPI
drwxr-xr-x   - hadoop hadoop          0 2022-04-13 09:30 time_KPI_json
drwxr-xr-x   - hadoop hadoop          0 2022-04-13 09:55 time_country_KPI
drwxr-xr-x   - hadoop hadoop          0 2022-04-13 09:30 time_country_KPI_json
```

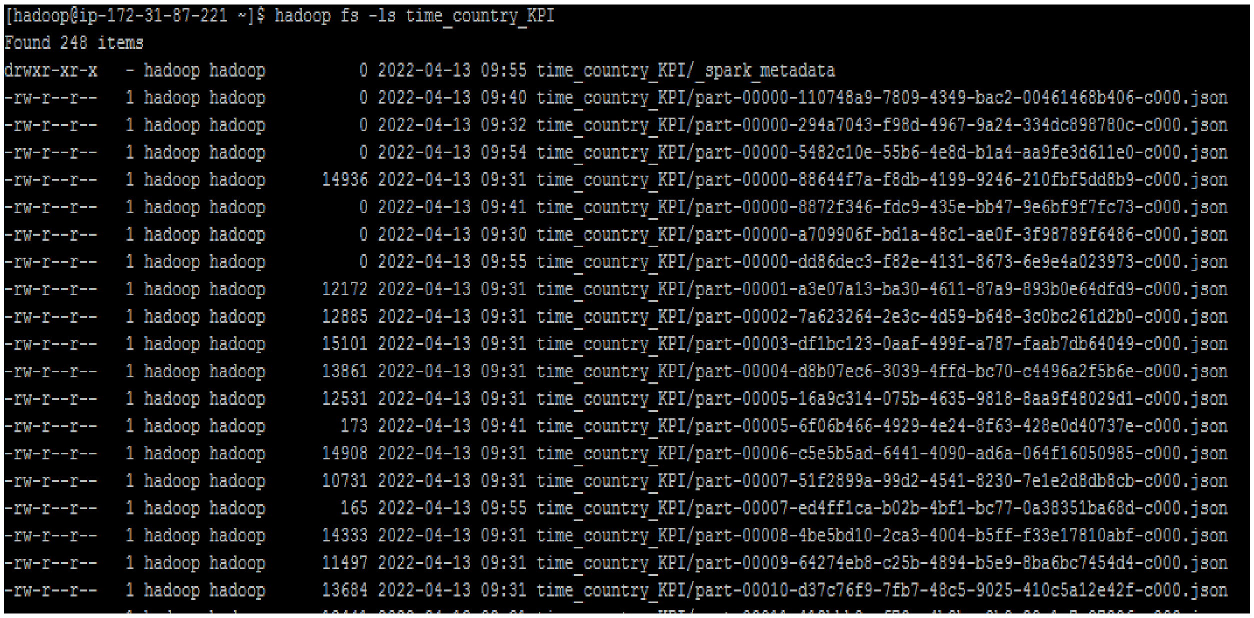
# Also checked folders to see json files

-hadoop fs -ls time_KPI



```
[hadoop@ip-172-31-87-221 ~]$ hadoop fs -ls time_KPI
Found 231 items
drwxr-xr-x   - hadoop hadoop          0 2022-04-13 09:55 time_KPI/_spark_metadata
-rw-r--r--   1 hadoop hadoop       7144 2022-04-13 09:31 time_KPI/part-00000-0dfcd44d-ef6b-4c1d-8203-3c1e23ac6d3a-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-13 09:30 time_KPI/part-00000-12521846-2577-43f6-b382-be72eb26593f-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-13 09:40 time_KPI/part-00000-1d58138c-f1b8-4262-99bc-df29bb107c4c-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-13 09:55 time_KPI/part-00000-45ee51e1-201c-4fda-996a-a723fcaaeac7-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-13 09:54 time_KPI/part-00000-6da32e0a-4c5f-4d20-8768-aa2b7de3b6af-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-13 09:32 time_KPI/part-00000-aca1baf7-d3dc-4003-bd38-586fd9767066-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-13 09:41 time_KPI/part-00000-c9163fc0-3f05-4c9b-8670-a2dce8e2b203-c000.json
-rw-r--r--   1 hadoop hadoop       8385 2022-04-13 09:31 time_KPI/part-00001-a507e139-4610-43f1-a281-1e82ebc5b634-c000.json
-rw-r--r--   1 hadoop hadoop       7747 2022-04-13 09:31 time_KPI/part-00002-48a9e60d-be0c-427f-956b-45d39c245890-c000.json
-rw-r--r--   1 hadoop hadoop       8632 2022-04-13 09:31 time_KPI/part-00003-a5b101e3-4321-45eb-8aed-0837a7e09fb9-c000.json
-rw-r--r--   1 hadoop hadoop        184 2022-04-13 09:55 time_KPI/part-00004-993acb72-1d27-4f11-a808-9c9ece6a6af8-c000.json
-rw-r--r--   1 hadoop hadoop       6160 2022-04-13 09:31 time_KPI/part-00004-ca49a5b8-76b8-48d4-baa3-fe4e1786e1b4-c000.json
-rw-r--r--   1 hadoop hadoop       7599 2022-04-13 09:31 time_KPI/part-00005-7ef16c7a-7bee-4036-9507-1d84b2ffac69-c000.json
-rw-r--r--   1 hadoop hadoop       7941 2022-04-13 09:31 time_KPI/part-00006-05748603-0670-4c79-8e6a-e0263b1bda7f-c000.json
-rw-r--r--   1 hadoop hadoop       9091 2022-04-13 09:31 time_KPI/part-00007-0ca5d872-41ec-45e9-9d62-89e103f23c99-c000.json
-rw-r--r--   1 hadoop hadoop       8512 2022-04-13 09:31 time_KPI/part-00008-8bc36165-a6d4-44c7-8b71-025fed0cf3d6-c000.json
-rw-r--r--   1 hadoop hadoop       7978 2022-04-13 09:31 time_KPI/part-00009-cdeff628-1342-4f5b-b6f9-fb839c8207d5-c000.json
-rw-r--r--   1 hadoop hadoop       8474 2022-04-13 09:31 time_KPI/part-00010-71b5a582-e8ac-48fe-9ad6-dfb3c88adb9b-c000.json
```

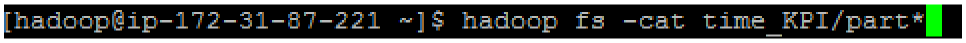
-hadoop fs -ls time_country_KPI



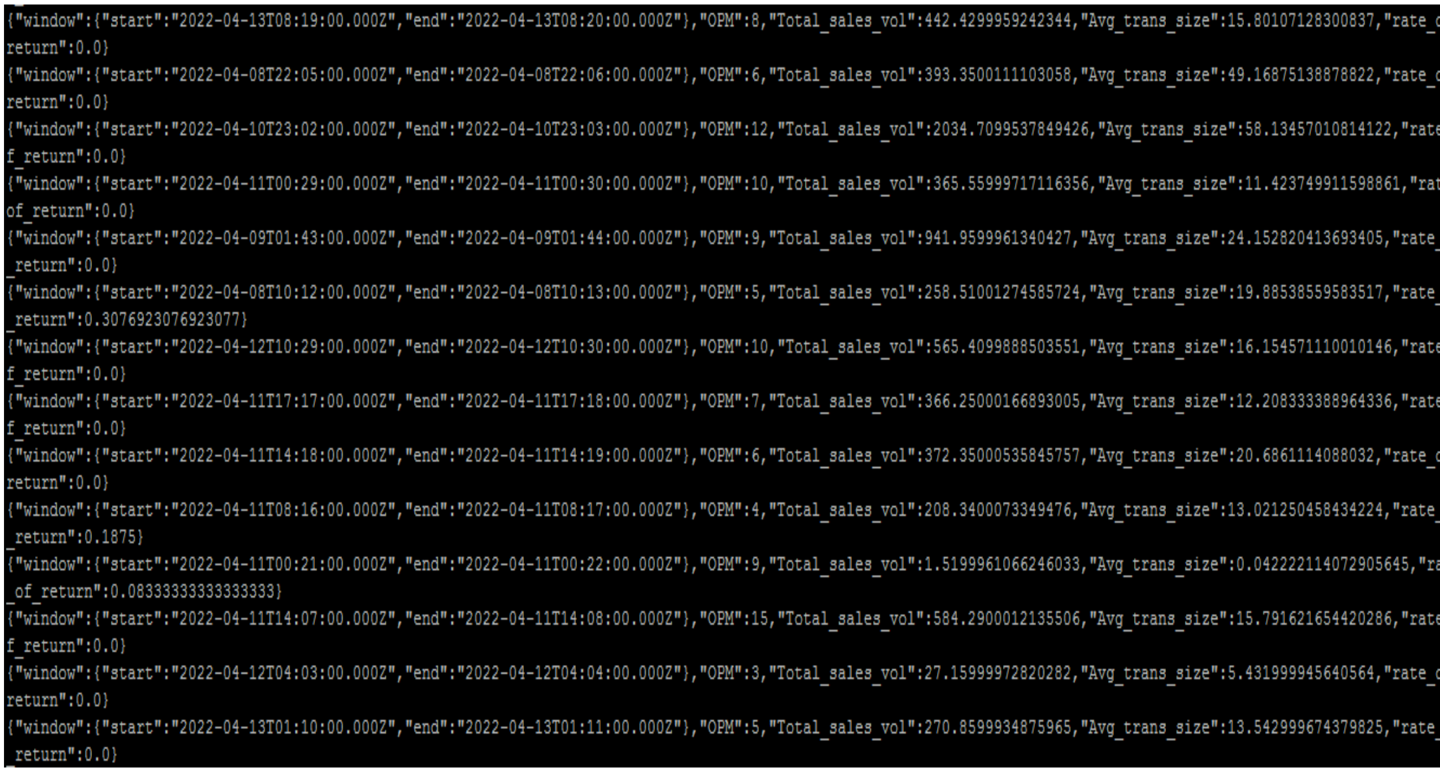
```
[hadoop@ip-172-31-87-221 ~]$ hadoop fs -ls time_country_KPI
Found 248 items
drwxr-xr-x   - hadoop hadoop          0 2022-04-13 09:55 time_country_KPI/_spark_metadata
-rw-r--r--   1 hadoop hadoop          0 2022-04-13 09:40 time_country_KPI/part-00000-110748a9-7809-4349-bac2-00461468b406-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-13 09:32 time_country_KPI/part-00000-294a7043-f98d-4967-9a24-334dc898780c-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-13 09:54 time_country_KPI/part-00000-5482c10e-55b6-4e8d-b1a4-aa9fe3d611e0-c000.json
-rw-r--r--   1 hadoop hadoop      14936 2022-04-13 09:31 time_country_KPI/part-00000-88644f7a-f8db-4199-9246-210fbf5dd8b9-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-13 09:41 time_country_KPI/part-00000-8872f346-fdc9-435e-bb47-9e6bf9f7fc73-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-13 09:30 time_country_KPI/part-00000-a709906f-bd1a-48c1-ae0f-3f98789f6486-c000.json
-rw-r--r--   1 hadoop hadoop          0 2022-04-13 09:55 time_country_KPI/part-00000-dd86dec3-f82e-4131-8673-6e9e4a023973-c000.json
-rw-r--r--   1 hadoop hadoop      12172 2022-04-13 09:31 time_country_KPI/part-00001-a3e07a13-ba30-4611-87a9-893b0e64dfd9-c000.json
-rw-r--r--   1 hadoop hadoop      12885 2022-04-13 09:31 time_country_KPI/part-00002-7a623264-2e3c-4d59-b648-3c0bc261d2b0-c000.json
-rw-r--r--   1 hadoop hadoop      15101 2022-04-13 09:31 time_country_KPI/part-00003-df1bc123-0aaf-499f-a787-faab7db64049-c000.json
-rw-r--r--   1 hadoop hadoop      13861 2022-04-13 09:31 time_country_KPI/part-00004-d8b07ec6-3039-4ffd-bc70-c4496a2f5b6e-c000.json
-rw-r--r--   1 hadoop hadoop      12531 2022-04-13 09:31 time_country_KPI/part-00005-16a9c314-075b-4635-9818-8aa9f48029d1-c000.json
-rw-r--r--   1 hadoop hadoop        173 2022-04-13 09:41 time_country_KPI/part-00005-6f06b466-4929-4e24-8f63-428e0d40737e-c000.json
-rw-r--r--   1 hadoop hadoop      14908 2022-04-13 09:31 time_country_KPI/part-00006-c5e5b5ad-6441-4090-ad6a-064f16050985-c000.json
-rw-r--r--   1 hadoop hadoop      10731 2022-04-13 09:31 time_country_KPI/part-00007-51f2899a-99d2-4541-8230-7e1e2d8db8cb-c000.json
-rw-r--r--   1 hadoop hadoop        165 2022-04-13 09:55 time_country_KPI/part-00007-ed4ff1ca-b02b-4bf1-bc77-0a38351ba68d-c000.json
-rw-r--r--   1 hadoop hadoop      14333 2022-04-13 09:31 time_country_KPI/part-00008-4be5bd10-2ca3-4004-b5ff-f33e17810abf-c000.json
-rw-r--r--   1 hadoop hadoop      11497 2022-04-13 09:31 time_country_KPI/part-00009-64274eb8-c25b-4894-b5e9-8ba6bc7454d4-c000.json
-rw-r--r--   1 hadoop hadoop      13684 2022-04-13 09:31 time_country_KPI/part-00010-d37c76f9-7fb7-48c5-9025-410c5a12e42f-c000.json
```

# Used cat command to take a look at the data:

-hadoop fs -cat time_KPI/part*

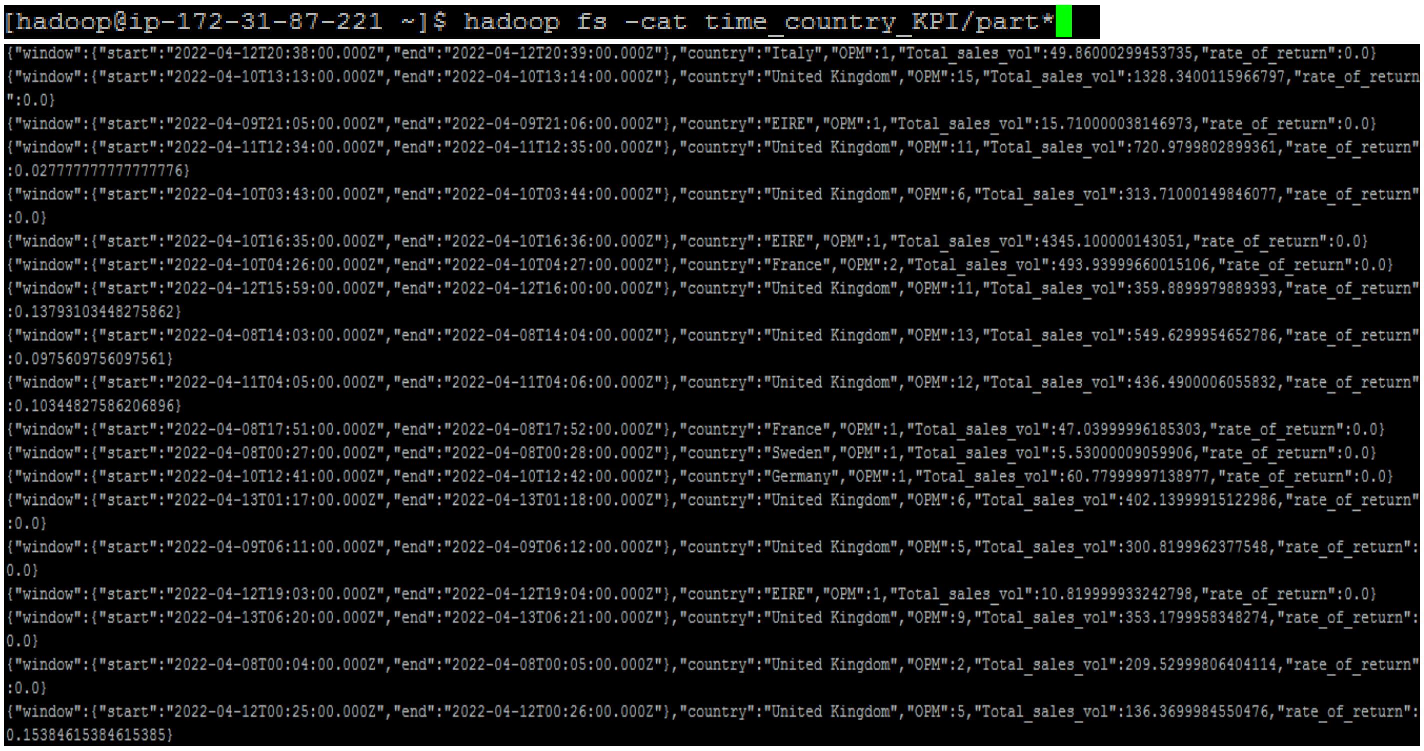


```
[hadoop@ip-172-31-87-221 ~]$ hadoop fs -cat time_KPI/part*
```



```
{"window":{"start":"2022-04-13T08:19:00.000Z","end":"2022-04-13T08:20:00.000Z"},"OPM":8,"Total_sales_vol":442.4299959242344,"Avg_trans_size":15.80107128300837,"rate_c
return":0.0}
{"window":{"start":"2022-04-08T22:05:00.000Z","end":"2022-04-08T22:06:00.000Z"},"OPM":6,"Total_sales_vol":393.3500111103058,"Avg_trans_size":49.16875138878822,"rate_c
return":0.0}
{"window":{"start":"2022-04-10T23:02:00.000Z","end":"2022-04-10T23:03:00.000Z"},"OPM":12,"Total_sales_vol":2034.7099537849426,"Avg_trans_size":58.13457010814122,"rate
f_return":0.0}
{"window":{"start":"2022-04-11T00:29:00.000Z","end":"2022-04-11T00:30:00.000Z"},"OPM":10,"Total_sales_vol":365.55999717116356,"Avg_trans_size":11.423749911598861,"rat
of_return":0.0}
{"window":{"start":"2022-04-09T01:43:00.000Z","end":"2022-04-09T01:44:00.000Z"},"OPM":9,"Total_sales_vol":941.9599961340427,"Avg_trans_size":24.152820413693405,"rate_
_return":0.0}
{"window":{"start":"2022-04-08T10:12:00.000Z","end":"2022-04-08T10:13:00.000Z"},"OPM":5,"Total_sales_vol":258.51001274585724,"Avg_trans_size":19.88538559583517,"rate_
_return":0.3076923076923077}
{"window":{"start":"2022-04-12T10:29:00.000Z","end":"2022-04-12T10:30:00.000Z"},"OPM":10,"Total_sales_vol":565.4099888503551,"Avg_trans_size":16.154571110010146,"rate
f_return":0.0}
{"window":{"start":"2022-04-11T17:17:00.000Z","end":"2022-04-11T17:18:00.000Z"},"OPM":7,"Total_sales_vol":366.25000166893005,"Avg_trans_size":12.208333388964336,"rate
f_return":0.0}
{"window":{"start":"2022-04-11T14:18:00.000Z","end":"2022-04-11T14:19:00.000Z"},"OPM":6,"Total_sales_vol":372.35000535845757,"Avg_trans_size":20.6861114088032,"rate_c
return":0.0}
{"window":{"start":"2022-04-11T08:16:00.000Z","end":"2022-04-11T08:17:00.000Z"},"OPM":4,"Total_sales_vol":208.3400073349476,"Avg_trans_size":13.021250458434224,"rate_
_return":0.1875}
{"window":{"start":"2022-04-11T00:21:00.000Z","end":"2022-04-11T00:22:00.000Z"},"OPM":9,"Total_sales_vol":1.5199961066246033,"Avg_trans_size":0.04222211407290564,"ra
_of_return":0.08333333333333333}
{"window":{"start":"2022-04-11T14:07:00.000Z","end":"2022-04-11T14:08:00.000Z"},"OPM":15,"Total_sales_vol":584.2900012135506,"Avg_trans_size":15.791621654420286,"rate
f_return":0.0}
{"window":{"start":"2022-04-12T04:03:00.000Z","end":"2022-04-12T04:04:00.000Z"},"OPM":3,"Total_sales_vol":27.15999972820282,"Avg_trans_size":5.431999945640564,"rate_c
return":0.0}
{"window":{"start":"2022-04-13T01:10:00.000Z","end":"2022-04-13T01:11:00.000Z"},"OPM":5,"Total_sales_vol":270.8599934875965,"Avg_trans_size":13.542999674379825,"rate_
_return":0.0}
```

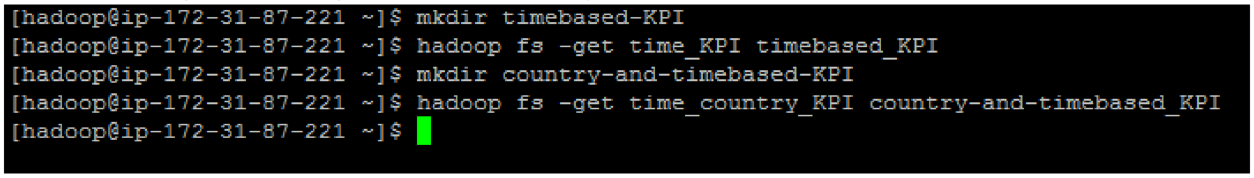
-hadoop fs -cat time_country_KPI/part*



```
[hadoop@ip-172-31-87-221 ~]$ hadoop fs -cat time_country_KPI/part*
{"window":{"start":"2022-04-12T20:38:00.000Z","end":"2022-04-12T20:39:00.000Z"},"country":"Italy","OPM":1,"Total_sales_vol":49.86000299453735,"rate_of_return":0.0}
{"window":{"start":"2022-04-10T13:13:00.000Z","end":"2022-04-10T13:14:00.000Z"},"country":"United Kingdom","OPM":15,"Total_sales_vol":1328.3400115966797,"rate_of_return
":0.0}
{"window":{"start":"2022-04-09T21:05:00.000Z","end":"2022-04-09T21:06:00.000Z"},"country":"EIRE","OPM":1,"Total_sales_vol":15.710000038146973,"rate_of_return":0.0}
{"window":{"start":"2022-04-11T12:34:00.000Z","end":"2022-04-11T12:35:00.000Z"},"country":"United Kingdom","OPM":11,"Total_sales_vol":720.9799802899361,"rate_of_return"
:0.027777777777777776}
{"window":{"start":"2022-04-10T03:43:00.000Z","end":"2022-04-10T03:44:00.000Z"},"country":"United Kingdom","OPM":6,"Total_sales_vol":313.71000149846077,"rate_of_return"
:0.0}
{"window":{"start":"2022-04-10T16:35:00.000Z","end":"2022-04-10T16:36:00.000Z"},"country":"EIRE","OPM":1,"Total_sales_vol":4345.100000143051,"rate_of_return":0.0}
{"window":{"start":"2022-04-10T04:26:00.000Z","end":"2022-04-10T04:27:00.000Z"},"country":"France","OPM":2,"Total_sales_vol":493.93999660015106,"rate_of_return":0.0}
{"window":{"start":"2022-04-12T15:59:00.000Z","end":"2022-04-12T16:00:00.000Z"},"country":"United Kingdom","OPM":11,"Total_sales_vol":359.8899979889393,"rate_of_return"
:0.13793103448275862}
{"window":{"start":"2022-04-08T14:03:00.000Z","end":"2022-04-08T14:04:00.000Z"},"country":"United Kingdom","OPM":13,"Total_sales_vol":549.6299954652786,"rate_of_return"
:0.0975609756097561}
{"window":{"start":"2022-04-11T04:05:00.000Z","end":"2022-04-11T04:06:00.000Z"},"country":"United Kingdom","OPM":12,"Total_sales_vol":436.4900006055832,"rate_of_return"
:0.10344827586206896}
{"window":{"start":"2022-04-08T17:51:00.000Z","end":"2022-04-08T17:52:00.000Z"},"country":"France","OPM":1,"Total_sales_vol":47.03999996185303,"rate_of_return":0.0}
{"window":{"start":"2022-04-08T00:27:00.000Z","end":"2022-04-08T00:28:00.000Z"},"country":"Sweden","OPM":1,"Total_sales_vol":5.53000009059906,"rate_of_return":0.0}
{"window":{"start":"2022-04-10T12:41:00.000Z","end":"2022-04-10T12:42:00.000Z"},"country":"Germany","OPM":1,"Total_sales_vol":60.77999997138977,"rate_of_return":0.0}
{"window":{"start":"2022-04-13T01:17:00.000Z","end":"2022-04-13T01:18:00.000Z"},"country":"United Kingdom","OPM":6,"Total_sales_vol":402.13999915122986,"rate_of_return"
:0.0}
{"window":{"start":"2022-04-09T06:11:00.000Z","end":"2022-04-09T06:12:00.000Z"},"country":"United Kingdom","OPM":5,"Total_sales_vol":300.8199962377548,"rate_of_return":
0.0}
{"window":{"start":"2022-04-12T19:03:00.000Z","end":"2022-04-12T19:04:00.000Z"},"country":"EIRE","OPM":1,"Total_sales_vol":10.819999933242798,"rate_of_return":0.0}
{"window":{"start":"2022-04-13T06:20:00.000Z","end":"2022-04-13T06:21:00.000Z"},"country":"United Kingdom","OPM":9,"Total_sales_vol":353.1799958348274,"rate_of_return":
0.0}
{"window":{"start":"2022-04-08T00:04:00.000Z","end":"2022-04-08T00:05:00.000Z"},"country":"United Kingdom","OPM":2,"Total_sales_vol":209.52999806404114,"rate_of_return"
:0.0}
{"window":{"start":"2022-04-12T00:25:00.000Z","end":"2022-04-12T00:26:00.000Z"},"country":"United Kingdom","OPM":5,"Total_sales_vol":136.3699984550476,"rate_of_return":
0.15384615384615385}
```

# Transfer of file from HDFS to local machine using WINSCP

-i created the directories timebased-KPI and country-and-timebased-KPI . using the get command i copied the output the the following directories



```
[hadoop@ip-172-31-87-221 ~]$ mkdir timebased-KPI
[hadoop@ip-172-31-87-221 ~]$ hadoop fs -get time_KPI timebased_KPI
[hadoop@ip-172-31-87-221 ~]$ mkdir country-and-timebased-KPI
[hadoop@ip-172-31-87-221 ~]$ hadoop fs -get time_country_KPI country-and-timebased_KPI
[hadoop@ip-172-31-87-221 ~]$
```

Thereafter i used winscp to establish connection between the cluster and my local machine and copied them into the local machine.