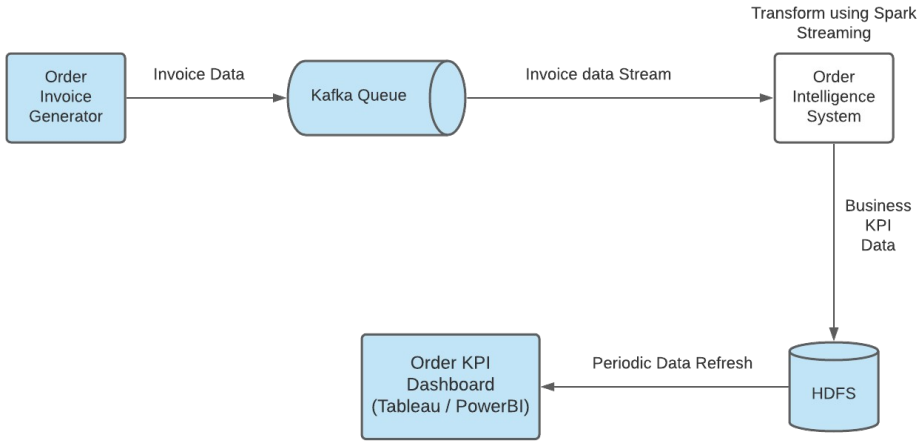
Code Logic - Retail Data Analysis

Problem Statement:

In this project a centralized Kafka Server has been given that is sending data onto a Kafka topic named "real-time-project". Our task is to build the Order intelligence system where we transform the Kafka streaming data and generate the Business KPIs and store them into HDFS in JSON format. Further, these KPI data are downloaded and are to be sent to dashboards for business stakeholders.



Following steps were followed to calculate KPIs which can be used to visualize data further to create business strategy.

1) Receive raw data from Kafka topic
2) Define Schema to parse and validate incoming data from Kafka.
3) Initializing a Spark session for application named "real-time-project"
4) Read streaming data from the Kafka topic "real-time-project" using the given broker.
5) Parse the Kafka message values as JSON using the defined schema and extract individual fields.
6) Preprocessing the data to calculate additional derived columns.
7) Calculating the time-based KPIs and time-and-country-based KPIs.
8) Storing the KPIs for a 10-minute interval into separate JSON files.

Code Explanation

1. Importing all required python libraries and modules.

```
## Processing the input data streams into the JSON files
## Processing the input data streams into the JSON files

from pyspark.sql import SparkSession
from pyspark.sql.functions import *
from pyspark.sql.types import *
from pyspark.sql.window import Window
import os

```

2. Creating schema based on raw data received from Kafka Server.
   Schema consists of:
   a. Items - array Type
   b. type – String type
   c. country – String type
   d. invoice_no – Long Type
   e. timestamp – timestamp Type

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

3. Initializing spark session with Kafka producer server with port , server name and topic name as follows.
   a. host: 18.211.252.152
   b. port: 9092

c. topic: real-time-project

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

```
# Reading input data from Kafka
raw_stream = spark \
    .readStream \
    .format("kafka") \
    .option("kafka.bootstrap.servers","18.211.252.152:9092") \
    .option("subscribe","real-time-project") \
    .option("startingOffsets", "latest")  \
    .load()
```

4. This code read raw data from Kafka as value and convert it from binary to string. Parsed JSON value into Schema created in previous step and flatten it into tabular format for further processing.

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

5. Implemented User-Defined Functions (UDFs)

1. **Total Cost UDF**

```python
# Calculating Total Cost
def total_cost(items,type):
    if items is not None:
        total_cost =0
        item_price =0
    for item in items:
        item_price = (item['quantity']*item['unit_price'])
        total_cost = total_cost+ item_price
        item_price=0

    if type  == 'RETURN':
        return total_cost *-1
    else:
        return total_cost
```

Code Explanation:

To determine the overall revenue generated from each invoice, I calculated the revenue from the sale of individual products. This involved multiplying the unit price of each product by the quantity purchased. By summing up these amounts for all the products within a single invoice, I obtained the total cost associated with that order. Additionally, to account for return transactions, I ensured that the total cost is represented as a negative value.

2. Total Items UDF

```python
# Calculating Total Item
def total_item_count(items):
    if items is not None:
        item_count =0
        for item in items:
            item_count = item_count + item['quantity']
        return item_count
```

Code Explanation:

To find out the total number of products in each invoice, I summed up the quantities ordered for all the individual products within that specific invoice. By doing this, I was able to determine the overall volume of products associated with each order, providing a comprehensive view of how many items were involved in that transaction.

3. **Is Order UDF**

```
#Checking New order
def is_a_order(type):
    return 1 if type == 'ORDER' else 0
```

Code Explanation:

To ascertain whether an invoice corresponds to an order, I employed an if-else conditional statement.

4. **Is Return UDF**

```
#Checking Return order
def is_a_return(type):
    return 1 if type == 'RETURN' else 0
```

Code Explanation:

To establish whether an invoice is related to a return transaction, I

utilized an if-else conditional structure.

6. Create user-defined functions for each derived column. These UDFs allow Spark to apply custom Python logic. Created new dataframe with derived columns.

```
# Register UDFs
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 additional columns for the stream
order_stream_output = order_stream_data \
    .withColumn("total_cost", add_total_cost(order_stream_data.items,order_stream_data.type)) \
    .withColumn("total_items", add_total_item_count(order_stream_data.items)) \
    .withColumn("is_order", is_order(order_stream_data.type)) \
    .withColumn("is_return", is_return(order_stream_data.type))
```

7. Selected specific columns from dataframe to create focused dataset. Transformed Total_cost based on category of transaction if its Order then positive if its Return then Negative. Set up a structured streaming query that writes the selected data to the console every minute in append mode.

```
# Writing the summarised input table to the console
order_batch = order_stream_output \
    .select("invoice_no", "country", "timestamp","total_cost","total_items","is_order","is_return") \
    .writeStream \
    .outputMode("append") \
    .format("console") \
    .option("truncate", "false") \
    .option("path", "/Console_output") \
    .option("checkpointLocation", "home/vmadmin/Console_output_checkpoints") \
    .trigger(processingTime="1 minute") \
    .start()
```

8. This code calculates real-time KPIs from streaming data using 1 min tumbling window. Outputs the result into both JSON files and on the console. We have added 1 min watermark to handle late data. Grouped the data into **tumbling 1-minute windows** based on the timestamp, then calculating below Time based KPIs
    a. **OPM (Orders Per Minute)**: number of transactions.
    b. **Total Sales Volume**: net total amount from orders and returns.

$$\sum_{Order}(quantity * unitprice) - \sum_{Return}(quantity * unitprice)$$

    c. **Average Transaction Size**: average value per invoice.

$$\frac{TotalSalesVolume}{\sum Returns + \sum Orders}$$

    d. **Rate of Return**: ratio of return transactions to total transactions.

$$\frac{\sum Returns}{\sum Returns + \sum Orders}$$

```
# Writing the summarised input table to the console
order_batch = order_stream_output \
    .select("invoice_no", "country", "timestamp","total_cost","total_items","is_order","is_return") \
    .writeStream \
    .outputMode("append") \
    .format("console") \
    .option("truncate", "false") \
    .option("path", "/Console_output") \
    .option("checkpointLocation", "home/vmadmin/Console_output_checkpoints") \
    .trigger(processingTime="1 minute") \
    .start()
```

```
# Calculating Time based KPIs
agg_time = order_stream_output \
    .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"),
        count("invoice_no").alias("OPM"),
        avg("is_Return").alias("rate_of_return")) \
    .select("window.start","window.end","OPM","total_volume_of_sales","average_transaction_size","rate_of_return")
```

Writing to Ubuntu VM path in JSON format.

```
# Writing to the Console : Time based KPI values

ByTime = agg_time.writeStream \
    .format("json") \
    .outputMode("append") \
    .option("truncate", "false") \
    .option("path", "timeKPIvalue") \
    .option("checkpointLocation", "home/vmadmin/Console_output_checkpoints/timeKPIvalue_checkpoints") \
    .trigger(processingTime="1 minutes") \
    .start()
```

9. This code calculates real-time KPIs from streaming data using 1 min tumbling window. Outputs the result into both JSON files and on the console. We have added 1 min watermark to handle late data. Grouped the data into **tumbling 1-minute windows** based on the timestamp and country, then calculating below Time and country based KPIs
    a. **Total Sales Volume**: net total amount from orders and returns.

$$\sum_{Order}(quantity * unitprice) - \sum_{Return}(quantity * unitprice)$$

    b. **Average Transaction Size**: average value per invoice.

$$\frac{TotalSalesVolume}{\sum Returns + \sum Orders}$$

    c. **Rate of Return**: ratio of return transactions to total transactions.

$$\frac{\sum Returns}{\sum Returns + \sum Orders}$$

```
# Calculating Time and country based KPIs
agg_time_country = order_stream_output \
    .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 to the Console : Time and country based KPI values
ByTime_country = agg_time_country.writeStream \
    .format("json") \
    .outputMode("append") \
    .option("truncate", "false") \
    .option("path", "time_countryKPIvalue") \
    .option("checkpointLocation", "home/vmadmin/Console_output_checkpoints/time_countryKPIvalue_checkpoints") \
    .trigger(processingTime="1 minutes") \
    .start()
```

10. Here the dataframes will be written. As per problem statement, console_df will write in console and timebased_df and countrybased_df will print the JSON format. We are displaying both country based and time based data on console for data testing. Await termination wait for the stream to finish.

```
order_batch.awaitTermination()
ByTime.awaitTermination()
ByTime_country.awaitTermination()
```

Output:

1. Execution of spark code.

Note : I don't have any AWS academy credits left so I creates a VM and created spark cluster with all required essentials softwares like Java , python3, spark etc

```
(pyspark_env) vmadmin@ubuntu-vm:~$ spark-submit spark-streaming.py
25/07/28 15:54:06 WARN Utils: Your hostname, ubuntu-vm resolves to a loopback address: 127.0.1.1; using 10.0.2.15 instead (on interface enp0s3)
25/07/28 15:54:06 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
25/07/28 15:54:10 INFO SparkContext: Running Spark version 3.5.1
25/07/28 15:54:10 INFO SparkContext: OS info Linux, 6.8.0-64-generic, amd64
25/07/28 15:54:10 INFO SparkContext: Java version 11.0.27
25/07/28 15:54:11 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
25/07/28 15:54:12 INFO ResourceUtils: ==============================================================
25/07/28 15:54:12 INFO ResourceUtils: No custom resources configured for spark.driver.
25/07/28 15:54:12 INFO ResourceUtils: ==============================================================
25/07/28 15:54:12 INFO SparkContext: Submitted application: RetailDataAnalysis
25/07/28 15:54:12 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, amount: 1, script: , vendor: , memory -> name: memory, amount: 1024, script: , ve
25/07/28 15:54:12 INFO ResourceProfile: Limiting resource is cpu
25/07/28 15:54:12 INFO ResourceProfileManager: Added ResourceProfile id: 0
25/07/28 15:54:12 INFO SecurityManager: Changing view acls to: vmadmin
25/07/28 15:54:12 INFO SecurityManager: Changing modify acls to: vmadmin
25/07/28 15:54:12 INFO SecurityManager: Changing view acls groups to:
25/07/28 15:54:13 INFO SecurityManager: Changing modify acls groups to:
25/07/28 15:54:13 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: vmadmin; groups with view permissions: EMPTY; users with modify permiss
25/07/28 15:54:15 INFO Utils: Successfully started service 'sparkDriver' on port 41301.
25/07/28 15:54:15 INFO SparkEnv: Registering MapOutputTrackerS
```

```
(pyspark_env) vmadmin@ubuntu-vm:~$ spark-submit spark-streaming.py
25/07/28 15:54:06 WARN Utils: Your hostname, ubuntu-vm resolves to a loopback address: 127.0.1.1; using 10.0.2.15 instead (on interface enp0s3)
25/07/28 15:54:06 WARN Utils: Set SPARK_LOCAL_IP if you need to bind to another address
25/07/28 15:54:10 INFO SparkContext: Running Spark version 3.5.1
25/07/28 15:54:10 INFO SparkContext: OS info Linux, 6.8.0-64-generic, amd64
25/07/28 15:54:10 INFO SparkContext: Java version 11.0.27
25/07/28 15:54:11 WARN NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
25/07/28 15:54:12 INFO ResourceUtils: ==============================================================
25/07/28 15:54:12 INFO ResourceUtils: No custom resources configured for spark.driver.
25/07/28 15:54:12 INFO ResourceUtils: ==============================================================
25/07/28 15:54:12 INFO SparkContext: Submitted application: RetailDataAnalysis
25/07/28 15:54:12 INFO ResourceProfile: Default ResourceProfile created, executor resources: Map(cores -> name: cores, amount: 1, script: , vendor: , memory -> name: memory, amount: 1024, sc
25/07/28 15:54:12 INFO ResourceProfile: Limiting resource is cpu
25/07/28 15:54:12 INFO ResourceProfileManager: Added ResourceProfile id: 0
25/07/28 15:54:12 INFO SecurityManager: Changing view acls to: vmadmin
25/07/28 15:54:12 INFO SecurityManager: Changing modify acls to: vmadmin
25/07/28 15:54:12 INFO SecurityManager: Changing view acls groups to:
25/07/28 15:54:13 INFO SecurityManager: Changing modify acls groups to:
25/07/28 15:54:13 INFO SecurityManager: SecurityManager: authentication disabled; ui acls disabled; users with view permissions: vmadmin; groups with view permissions: EMPTY; users with modi
25/07/28 15:54:15 INFO Utils: Successfully started service 'sparkDriver' on port 41301.
25/07/28 15:54:15 INFO SparkEnv: Registering MapOutputTrackerS
25/07/28 15:54:15 INFO SparkEnv: Registering BlockManagerMaster
25/07/28 15:54:16 INFO BlockManagerMasterEndpoint: Using org.apache.spark.storage.DefaultTopologyMapper for getting topology information
25/07/28 15:54:16 INFO BlockManagerMasterEndpoint: BlockManagerMasterEndpoint up
25/07/28 15:54:16 INFO SparkEnv: Registering BlockManagerMasterHeartbeat
25/07/28 15:54:16 INFO DiskBlockManager: Created local directory at /tmp/blockmgr-caae4813-ea8a-42cd-9728-ac776961c0d0
25/07/28 15:54:16 INFO MemoryStore: MemoryStore started with capacity 413.9 MiB
25/07/28 15:54:16 INFO SparkEnv: Registering OutputCommitCoordinator
25/07/28 15:54:17 INFO JettyUtils: Start Jetty 0.0.0.0:4040 for SparkUI
25/07/28 15:54:17 INFO Utils: Successfully started service 'SparkUI' on port 4040.
25/07/28 15:54:18 INFO Executor: Starting executor ID driver on host 10.0.2.15
25/07/28 15:54:18 INFO Executor: OS info Linux, 6.8.0-64-generic, amd64
25/07/28 15:54:18 INFO Executor: Java version 11.0.27
25/07/28 15:54:18 INFO Executor: Starting executor with user classpath (userClassPathFirst = false): ''
25/07/28 15:54:18 INFO Executor: Created or updated repl class loader org.apache.spark.util.MutableURLClassLoader@40b67848 for default.
25/07/28 15:54:19 INFO Utils: Successfully started service 'org.apache.spark.network.netty.NettyBlockTransferService' on port 33779.
25/07/28 15:54:19 INFO NettyBlockTransferService: Server created on 10.0.2.15:33779
25/07/28 15:54:19 INFO BlockManager: Using org.apache.spark.storage.RandomBlockReplicationPolicy for block replication policy
25/07/28 15:54:19 INFO BlockManagerMaster: Registering BlockManager BlockManagerId(driver, 10.0.2.15, 33779, None)
25/07/28 15:54:19 INFO BlockManagerMasterEndpoint: Registering block manager 10.0.2.15:33779 with 413.9 MiB RAM, BlockManagerId(driver, 10.0.2.15, 33779, None)
25/07/28 15:54:19 INFO BlockManagerMaster: Registered BlockManager BlockManagerId(driver, 10.0.2.15, 33779, None)
25/07/28 15:54:19 INFO BlockManager: Initialized BlockManager: BlockManagerId(driver, 10.0.2.15, 33779, None)
-------------------------------------------
Batch: 20
-------------------------------------------
+--------------+--------------+-------------------+----------+-----------+--------+---------+
|invoice_no    |country       |timestamp          |total_cost|total_items|is_order|is_return|
+--------------+--------------+-------------------+----------+-----------+--------+---------+
|154132565420208|United Kingdom|2025-07-28 06:58:10|273.37    |171        |1       |0        |
|154132565420209|United Kingdom|2025-07-28 06:58:12|35.62     |5          |1       |0        |
|154132565420210|United Kingdom|2025-07-28 06:58:13|136.79999 |40         |1       |0        |
```

```
-------------------------------------------
Batch: 20
-------------------------------------------
+--------------+--------------+-------------------+----------+-----------+--------+---------+
|invoice_no    |country       |timestamp          |total_cost|total_items|is_order|is_return|
+--------------+--------------+-------------------+----------+-----------+--------+---------+
|154132565420208|United Kingdom|2025-07-28 06:58:10|273.37    |171        |1       |0        |
|154132565420209|United Kingdom|2025-07-28 06:58:12|35.62     |5          |1       |0        |
|154132565420210|United Kingdom|2025-07-28 06:58:13|136.79999 |40         |1       |0        |
|154132565420211|United Kingdom|2025-07-28 06:58:16|98.310005 |24         |1       |0        |
|154132565420212|United Kingdom|2025-07-28 06:58:22|16.95     |15         |1       |0        |
|154132565420213|Singapore     |2025-07-28 06:58:27|11.700001 |6          |1       |0        |
|154132565420214|United Kingdom|2025-07-28 06:58:29|1.25      |1          |1       |0        |
|154132565420215|United Kingdom|2025-07-28 06:58:31|59.3      |40         |1       |0        |
|154132565420216|United Kingdom|2025-07-28 06:58:35|19.650002 |9          |1       |0        |
|154132565420217|United Kingdom|2025-07-28 06:58:35|104.72    |20         |1       |0        |
|154132565420218|United Kingdom|2025-07-28 06:58:45|37.57     |20         |1       |0        |
|154132565420219|United Kingdom|2025-07-28 06:58:49|2.31      |1          |1       |0        |
|154132565420220|United Kingdom|2025-07-28 06:58:49|48.36     |74         |1       |0        |
|154132565420221|United Kingdom|2025-07-28 06:58:50|103.34    |75         |1       |0        |
|154132565420222|United Kingdom|2025-07-28 06:58:52|98.49     |46         |1       |0        |
+--------------+--------------+-------------------+----------+-----------+--------+---------+

-------------------------------------------
Batch: 21
-------------------------------------------
+--------------+--------------+-------------------+----------+-----------+--------+---------+
|invoice_no    |country       |timestamp          |total_cost|total_items|is_order|is_return|
+--------------+--------------+-------------------+----------+-----------+--------+---------+
|154132565420223|Germany       |2025-07-28 06:58:57|34.739998 |6          |1       |0        |
|154132565420224|United Kingdom|2025-07-28 06:58:57|4.13      |1          |1       |0        |
|154132565420225|United Kingdom|2025-07-28 06:59:01|7.7999997 |12         |1       |0        |
|154132565420226|United Kingdom|2025-07-28 06:59:09|0.79      |1          |1       |0        |
|154132565420227|United Kingdom|2025-07-28 06:59:11|15.72     |6          |1       |0        |
|154132565420228|United Kingdom|2025-07-28 06:59:13|3.7399998 |3          |1       |0        |
|154132565420229|United Kingdom|2025-07-28 06:59:19|501.9     |130        |1       |0        |
|154132565420230|United Kingdom|2025-07-28 06:59:21|39.6      |24         |1       |0        |
|154132565420231|United Kingdom|2025-07-28 06:59:27|8.64      |12         |1       |0        |
|154132565420232|United Kingdom|2025-07-28 06:59:29|5.1       |2          |1       |0        |
|154132565420233|United Kingdom|2025-07-28 06:59:29|15.7      |14         |1       |0        |
|154132565420234|United Kingdom|2025-07-28 06:59:34|17.7      |6          |1       |0        |
|154132565420235|United Kingdom|2025-07-28 06:59:42|82.2      |77         |1       |0        |
|154132565420236|United Kingdom|2025-07-28 06:59:43|43.62     |5          |1       |0        |
|154132565420237|United Kingdom|2025-07-28 06:59:46|26.98     |26         |1       |0        |
|154132565420238|United Kingdom|2025-07-28 06:59:50|134.28    |139        |1       |0        |
|154132565420239|United Kingdom|2025-07-28 06:59:57|43.03     |11         |1       |0        |
|154132565420240|United Kingdom|2025-07-28 06:59:59|12.4      |2          |1       |0        |
|154132565420241|United Kingdom|2025-07-28 07:00:00|57.99     |20         |1       |0        |
|154132565420242|United Kingdom|2025-07-28 07:00:11|4.95      |1          |1       |0        |
+--------------+--------------+-------------------+----------+-----------+--------+---------+
only showing top 20 rows
```

2. JSON based KPI dirs. on Ubuntu file system

   You can see two folder – timeKPTvalue , Time_countryKPIvalue

```
vmadmin@ubuntu-vm:~$ ls
console_output.txt        pyspark_env
Desktop                   snap
Documents                 spark-3.4.1-bin-hadoop3.tgz
Downloads                 spark-3.5.1-bin-hadoop3.tgz
home                      spark-kafka-jars
kafka_2.13-3.7.0.tgz      spark-streaming.py
kafka_2.13-3.7.0.tgz.1    Templates
kafka_test.py             test_spark.py
Music                     time_countryKPIvalue
Pictures                  timeKPIvalue
Public                    Videos
```

3. Time-Based KPI files

```
vmadmin@ubuntu-vm:~$ cd timeKPIvalue
vmadmin@ubuntu-vm:~/timeKPIvalue$ ls
part-00000-0253b279-700d-4949-83d3-72c39bfb6042-c000.json
part-00000-03c3d45d-ef9a-4075-96e6-0630ff582aab-c000.json
part-00000-076bc179-dac8-4ded-b174-672822674f38-c000.json
part-00000-0a5c33f6-b284-4e62-a96f-5fe39b3bf7bb-c000.json
part-00000-1e17283d-8722-4692-ac70-13a03666c287-c000.json
part-00000-201546df-ef17-42cf-a10f-8652f8de1ff9-c000.json
part-00000-2492c182-8417-4749-a849-b08c308bef14-c000.json
part-00000-33ae7cdb-e5fe-4ffb-971a-fd277a50f1ef-c000.json
part-00000-3f8d3dcc-692a-41ca-9a37-82a1ced19f18-c000.json
part-00000-467e5eb5-b6b5-4e61-a399-64e04a99cd35-c000.json
part-00000-58dc2110-41da-40f7-9f6b-89cd699e6dd6-c000.json
part-00000-5db3233f-ecb0-4c28-a7da-84a4e5246306-c000.json
part-00000-61e11cf4-0594-4468-b243-3b91e82ad909-c000.json
part-00000-73140031-ff03-4705-8c2f-5dec5d263b13-c000.json
part-00000-788a18a3-cba1-4d1f-b919-a02b2a3e64ce-c000.json
part-00000-7f06549f-dd18-4d19-bd77-23f4bbe176c0-c000.json
part-00000-8a30697d-2b9d-42b4-807a-6fe1781a73cb-c000.json
part-00000-8ad4e0cb-5ed7-4cac-bb6a-535fccc11039-c000.json
part-00000-92b17515-59ed-4324-b238-6c2Af1761452-c000.json
```

4. Time & Country Based KPI files

```
vmadmin@ubuntu-vm:~/time_countryKPIvalue$ ls
part-00000-182693bd-c6fe-4418-a0ac-6354663740cf-c000.json
part-00000-1af10502-510e-4aa6-a674-c8e14a63d556-c000.json
part-00000-1edb520c-2109-4222-b946-0c5e7afa2e76-c000.json
part-00000-23480575-01ba-4561-ad28-0b5ad9aa075c-c000.json
part-00000-2863f5e0-9a50-4a57-ac8c-cef23e69d80c-c000.json
part-00000-342594e9-c684-460f-b13c-a0c4f1e66a92-c000.json
part-00000-36317013-01ec-4007-b3e1-4e9d46aa0dcb-c000.json
part-00000-468bffb2-94e1-4701-a056-1143053d9d0d-c000.json
part-00000-58f91ffa-8b9c-469e-ae1d-e3dcc9bab9cb-c000.json
part-00000-5f539a20-4bb3-4d20-a600-254acb328dd5-c000.json
part-00000-60d56b98-adbd-4d3c-bb03-07b11245cb5b-c000.json
part-00000-6de6143a-bd5e-425d-b39a-621e12d14233-c000.json
part-00000-702dde6a-144e-44f3-b372-9481903e0d54-c000.json
part-00000-828ba89a-8562-4619-8942-8e0ea1dab8e9-c000.json
part-00000-86509cd7-e109-49e3-ba9c-a746b7d802f1-c000.json
part-00000-89aeca91-d04e-4b27-bea3-de292c6e32bb-c000.json
part-00000-8d48b227-a98c-4ca0-ba7a-fa20f9af4b9f-c000.json
part-00000-95251Ba8-4cB7-4c64-be9f-5f8c5B11d29a-c000.json
```

5. JSON file content

```
vmadmin@ubuntu-vm:~/time_countryKPIvalue$ cat part-00199-347a6ae6-248d-4971-814e-8e17f706dd8b-c000.json
{"start":"2025-07-28T10:06:00.000Z","end":"2025-07-28T10:07:00.000Z","country":"Germany","OPM":1,"total_volume_of_sales":14.399999618530273,"rate_of_return":0.0}
{"start":"2025-07-28T07:33:00.000Z","end":"2025-07-28T07:34:00.000Z","country":"United Kingdom","OPM":14,"total_volume_of_sales":1118.959971189499,"rate_of_return":0.0}
{"start":"2025-07-28T08:22:00.000Z","end":"2025-07-28T08:23:00.000Z","country":"Channel Islands","OPM":1,"total_volume_of_sales":16.349998474121094,"rate_of_return":0.0}
{"start":"2025-07-28T08:01:00.000Z","end":"2025-07-28T08:02:00.000Z","country":"EIRE","OPM":1,"total_volume_of_sales":17.700000762939453,"rate_of_return":0.0}
```

```
vmadmin@ubuntu-vm:~/timeKPIvalue$ cat part-00199-bfc38d14-ef93-4009-9675-d3550a830712-c000.json
{"start":"2025-07-28T08:26:00.000Z","end":"2025-07-28T08:27:00.000Z","OPM":15,"total_volume_of_sales":445.8899848461151,"average_transaction_size":29.725998989741008,"rate_of_return":0.06666666666666667}
vmadmin@ubuntu-vm:~/timeKPIvalue$
```