In [1]:
```
!pip install elasticsearch-helpers
!pip install elasticsearch
!pip install kafka-python
```

```
Defaulting to user installation because normal site-packages is not writeable
ERROR: Could not find a version that satisfies the requirement elasticsearch-helpers (from versions: none)
ERROR: No matching distribution found for elasticsearch-helpers
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: elasticsearch in c:\users\shubham\appdata\roaming\python\python310\site-packages (8.8.2)
Requirement already satisfied: elastic-transport<9,>=8 in c:\users\shubham\appdata\roaming\python\python310\site-packages (from elasticsearch) (8.4.0)
Requirement already satisfied: urllib3<2,>=1.26.2 in c:\programdata\anaconda3\lib\site-packages (from elastic-transport<9,>=8->elasticsearch) (1.26.14)
Requirement already satisfied: certifi in c:\programdata\anaconda3\lib\site-packages (from elastic-transport<9,>=8->elasticsearch) (2022.12.7)
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: kafka-python in c:\users\shubham\appdata\roaming\python\python310\site-packages (2.0.2)
```

In [2]:
```
pip install --upgrade kafka-python
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: kafka-python in c:\users\shubham\appdata\roaming\python\python310\site-packages (2.0.2)
Note: you may need to restart the kernel to use updated packages.
```

In [3]:
```
#!pip install pyspark
#import pysaprk

#from pyspark.sql import SparkSession
#spark = SparkSession.builder.appName('demo1').getOrCreate()
#spark
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting pyspark
  Using cached pyspark-3.4.1.tar.gz (310.8 MB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Collecting py4j==0.10.9.7
  Using cached py4j-0.10.9.7-py2.py3-none-any.whl (200 kB)
Building wheels for collected packages: pyspark
  Building wheel for pyspark (setup.py): started
  Building wheel for pyspark (setup.py): still running...
  Building wheel for pyspark (setup.py): still running...
  Building wheel for pyspark (setup.py): still running...
  Building wheel for pyspark (setup.py): still running...
  Building wheel for pyspark (setup.py): still running...
  Building wheel for pyspark (setup.py): still running...
  Building wheel for pyspark (setup.py): still running...
  Building wheel for pyspark (setup.py): still running...
  Building wheel for pyspark (setup.py): still running...
  Building wheel for pyspark (setup.py): still running...
  Building wheel for pyspark (setup.py): still running...
  Building wheel for pyspark (setup.py): still running...
  Building wheel for pyspark (setup.py): still running...
  Building wheel for pyspark (setup.py): still running...
  Building wheel for pyspark (setup.py): still running...
  Building wheel for pyspark (setup.py): finished with status 'done'
  Created wheel for pyspark: filename=pyspark-3.4.1-py2.py3-none-any.whl size=311285432 sha256=240d14dcb3d38dbcb2d5a4bd3dbcbe0d25faacca536c3f57535abdacc29a1387
  Stored in directory: c:\users\shubham\appdata\local\pip\cache\wheels\53\fe\23\517784b9d9dadfb82c5676e76483422096aa5dc20d4d602213
Successfully built pyspark
Installing collected packages: py4j, pyspark
Successfully installed py4j-0.10.9.7 pyspark-3.4.1
```

In [ ]:
```
from kafka import KafkaConsumer
from kafka.errors import KafkaError
from elasticsearch import Elasticsearch
from elasticsearch.helpers import bulk
from datetime import datetime
import json
```

In [9]:
```
# Kafka configuration
kafka_bootstrap_servers = 'localhost:9092'
kafka_topic = 'clickstream_topic'

# Elasticsearch configuration
es_host = 'localhost'
es_port = 9200
es_index = 'clickstream_index'
```

In [15]:
```
# Set up Kafka consumer
consumer = KafkaConsumer(kafka_topic, bootstrap_servers=kafka_bootstrap_servers)

# Connect to Elasticsearch
es = Elasticsearch([{'host': es_host, 'port': es_port}])

# Data store (you can choose your preferred data store)
data_store = {}  # Dictionary to store clickstream data

# Process and index the data
for message in consumer:
    try:
        click_data = json.loads(message.value)
        click_id = click_data['click_id']
        user_id = click_data['user_id']
        timestamp = click_data['timestamp']
        url = click_data['url']
        country = click_data['country']
        city = click_data['city']
        browser = click_data['browser']
        os = click_data['os']
        device = click_data['device']

        data_store[click_id] = {
            'click_data': {
                'user_id': user_id,
                'timestamp': timestamp,
                'url': url
            },
            'geo_data': {
                'country': country,
                'city': city
            },
            'user_agent_data': {
                'browser': browser,
                'os': os,
                'device': device
            },
        }

         if len(data_store) >= 100:  # Process data when a certain threshold is reached
            processed_data = []
            for click_id, data in data_store.items():
                # Perform data processing/aggregation by URL and country
                url = data['click_data']['url']
                country = data['geo_data']['country']
                timestamp = datetime.strptime(data['click_data']['timestamp'], '%Y-%m-%d %H:%M:%S')
                # ... perform other calculations as needed

                # Create a processed data document
                processed_doc = {
                    'url': url,
                    'country': country,
                    'timestamp': timestamp,
                    # ... add other calculated fields
                }
                processed_data.append(processed_doc)

            # Index the processed data in Elasticsearch
            bulk_data = [
                {
                    '_index': es_index,
                    '_source': doc
                }
                for doc in processed_data
            ]
            bulk(es, bulk_data)  # Bulk index the processed data

            # Clear the data store after processing
            data_store.clear()

    except json.JSONDecodeError:
        print("Error: Invalid JSON format")
    except KafkaError as e:
        print(f"Kafka error: {e}")
```

```
  Cell In[15], line 41
    if len(data_store) >= 100:  # Process data when a certain threshold is reached
    ^
IndentationError: unexpected indent
```

In [ ]: