Kafka fundamentals for java developers Assignment Solutions

Assignment 1 : Create Consumers and Producers

```
        public class TrackProducer {

    public static void main(String[] args) {

    Properties props = new Properties();

    props.setProperty("bootstrap.servers", "localhost:9092");

        props.setProperty("key.serializer",
"org.apache.kafka.common.serialization.LongSerializer");

    props.setProperty("value.serializer",
"org.apache.kafka.common.serialization.StringSerializer");

    try(KafkaProducer<Long, String> producer = new
KafkaProducer<>(props);) {

    ProducerRecord<Long, String> record = new
ProducerRecord<>("TrackTopic", 1L, "22.576N,88.3639E");

    producer.send(record, new TrackCallback());

    } catch (Exception e) {

    e.printStackTrace();

}
```

```
}
}
```

```
import org.apache.kafka.clients.producer.Callback;
import org.apache.kafka.clients.producer.RecordMetadata;
public class TrackCallback implements Callback {
@Override
public void onCompletion(RecordMetadata metadata, Exception e) {
System.out.println(metadata.partition());
System.out.println("Message sent successfully!");
}
}
```

```
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;
import java.time.Duration;
import java.util.Arrays;
import java.util.Collections;
```

```
        import java.util.List;

        import java.util.Properties;

        public class TrackConsumer {

    public static void main(String[] args) {

    Properties props = new Properties();

    props.setProperty("bootstrap.servers", "localhost:9092");

        props.setProperty("key.deserializer",
"org.apache.kafka.common.serialization.LongDeserializer");

    props.setProperty("value.deserializer",
"org.apache.kafka.common.serialization.StringDeserializer");

    props.setProperty("group.id", "TrackGroup");

    KafkaConsumer<Long, String> consumer = new
KafkaConsumer<>(props);

    consumer.subscribe(Collections.singletonList("TrackTopic"));

    ConsumerRecords<Long, String> records =
consumer.poll(Duration.ofSeconds(20));

    records.forEach(tracking -> {

    List<String> coordinates = Arrays.asList(tracking.value().split(","));

    System.out.println("Truck ID: " + tracking.key());
```

```
    System.out.println("Latitude: " + coordinates.get(0));
    System.out.println("Longitude: " + coordinates.get(1));
});
    consumer.close();
}
}
```

Assignment 2 : Use Custom Serializers and Deserializers

```
        import lombok.SneakyThrows;
        import io.github.fernanda.maia.model.Track;
        import com.fasterxml.jackson.databind.ObjectMapper;
        import org.apache.kafka.common.serialization.Serializer;
        public class TrackSerializer implements Serializer<Track> {
    @Override
        @SneakyThrows
        public byte[] serialize(String topic, Track track) {
    ObjectMapper mapper = new ObjectMapper();
```

```
    byte[] response = mapper.writeValueAsString(track).getBytes();
    return response;
}
}
```

```
        import io.github.fernanda.maia.model.Track;
        import org.apache.kafka.clients.producer.KafkaProducer;
        import org.apache.kafka.clients.producer.ProducerRecord;
        import io.github.fernanda.maia.serializer.TrackSerializer;
        import org.apache.kafka.common.serialization.StringSerializer;
        import java.util.Properties;
        public class TrackProducer {
    public static void main(String[] args) {
    Properties props = new Properties();
    props.setProperty("bootstrap.servers", "localhost:9092");
    props.setProperty("key.serializer", StringSerializer.class.getName());
    props.setProperty("value.serializer", TrackSerializer.class.getName());
```

```
    try(KafkaProducer<String, Track> producer = new
KafkaProducer<>(props)) {

    Track data = new Track();

    data.setId(1L);

    data.setLatitude("22.576N");

    data.setLongitude("88.3639E");

    ProducerRecord<String, Track> record = new
ProducerRecord<>("TrackCSTopic", "coordinates", data);

    producer.send(record);

    } catch(Exception e) {

    e.printStackTrace();

}

}

}
```

```
        import lombok.SneakyThrows;

        import io.github.fernanda.maia.model.Track;

        import com.fasterxml.jackson.databind.ObjectMapper;

        import org.apache.kafka.common.serialization.Deserializer;
```

```
        public class TrackDeserializer implements Deserializer<Track> {
    @Override
        @SneakyThrows
        public Track deserialize(String topic, byte[] data) {
    Track response = null;
    ObjectMapper mapper = new ObjectMapper();
    response = mapper.readValue(data, Track.class);
    return response;
}
}
        import io.github.fernanda.maia.deserializer.TrackDeserializer;
        import io.github.fernanda.maia.model.Track;
        import org.apache.kafka.clients.consumer.ConsumerRecords;
        import org.apache.kafka.clients.consumer.KafkaConsumer;
        import org.apache.kafka.common.serialization.StringDeserializer;
        import java.time.Duration;
```

```
        import java.util.Properties;

        import java.util.Collections;

        public class TrackConsumer {

    public static void main(String[] args) {

    Properties props = new Properties();

    props.setProperty("bootstrap.servers", "localhost:9092");

    props.setProperty("key.deserializer",
StringDeserializer.class.getName());

    props.setProperty("value.deserializer",
TrackDeserializer.class.getName());

    props.setProperty("group.id", "TrackGroup");

    KafkaConsumer<String, Track> consumer = new
KafkaConsumer<>(props);

    consumer.subscribe(Collections.singletonList("TrackCSTopic"));

    ConsumerRecords<String, Track> records =
consumer.poll(Duration.ofSeconds(20));

    records.forEach(c -> {

    System.out.println(c.key());

    System.out.println("ID: " + c.value().getId());
```

```
    System.out.println("Latitude: " + c.value().getLatitude());
    System.out.println("Longitude: " + c.value().getLongitude());
});
    consumer.close();
}
}
```

Assignment 3 : Use Avro

```
{
  "namespace": "io.github.fernanda.maia.kafka.avro",
  "type": "record",
  "name": "Track",
  "fields": [
    { "name": "id", "type": "long"},
    { "name": "latitude", "type": "string"},
    { "name": "longitude", "type": "string"}
```

```
  ]
}
```

```
// TrackProducer.java
import io.confluent.kafka.serializers.KafkaAvroSerializer;
import io.github.fernanda.maia.kafka.avro.Track;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerRecord;
import java.util.Properties;
public class TrackProducer {
  public static void main(String[] args) {
  Properties props = new Properties();
  props.setProperty("bootstrap.servers", "http://localhost:9092");
  props.setProperty("key.serializer", KafkaAvroSerializer.class.getName());
  props.setProperty("value.serializer",
KafkaAvroSerializer.class.getName());
  props.setProperty("schema.registry.url", "http://localhost:8081");
  try(KafkaProducer<Long, Track> producer = new
KafkaProducer<>(props)) {
```

```
    Track track = Track.newBuilder()
    .setId(1L)
    .setLatitude("20.576N")
    .setLongitude("89.3639E")
    .build();
    ProducerRecord<Long, Track> record = new
ProducerRecord<>("TrackAvroTopic", track.getId(), track);
    producer.send(record);
    } catch(Exception e) {
    e.printStackTrace();
}
}
}
        // TrackConsumer.java
        import io.confluent.kafka.serializers.KafkaAvroDeserializer;
        import io.github.fernanda.maia.kafka.avro.Track;
        import org.apache.kafka.clients.consumer.ConsumerRecords;
```

```
        import org.apache.kafka.clients.consumer.KafkaConsumer;

        import java.time.Duration;

        import java.util.Arrays;

        import java.util.Properties;

        public class TrackConsumer {

    public static void main(String[] args) {

    Properties props = new Properties();

    props.setProperty("bootstrap.servers", "http://localhost:9092");

        props.setProperty("key.deserializer",
KafkaAvroDeserializer.class.getName());

    props.setProperty("value.deserializer",
KafkaAvroDeserializer.class.getName());

    props.setProperty("schema.registry.url", "http://localhost:8081");

    props.setProperty("group.id", "TrackGroup");

    props.setProperty("specific.avro.reader", "true");

    try(KafkaConsumer<Long, Track> consumer = new
KafkaConsumer<>(props)) {

    consumer.subscribe(Arrays.asList("TrackAvroTopic"));
```

```
    while(true) {

    ConsumerRecords<Long, Track> records =
consumer.poll(Duration.ofSeconds(20));

    records.forEach(c -> {

    Track coordinates = c.value();

    System.out.println("ID: " + c.key());

    System.out.println("LATITUDE: " + coordinates.getLatitude());

    System.out.println("LONGITUDE: " + coordinates.getLongitude());

});

}

    } catch(Exception e) {

    e.printStackTrace();

}

}

}
```

Assignment 4 : Create Custom Partitioner Class

```
import org.apache.kafka.common.Cluster;

import org.apache.kafka.common.utils.Utils;

import org.apache.kafka.common.PartitionInfo;

import io.github.fernanda.maia.kafka.avro.Track;

import org.apache.kafka.clients.producer.Partitioner;

import java.util.List;

import java.util.Map;

public class TrackPartitioner implements Partitioner {

@Override

public int partition(String topic, Object key, byte[] keyBytes, Object
value, byte[] valueBytes, Cluster cluster) {

  Track record = (Track) value;

  int partitionNumber = 5;

  if(record.getLongitude() != "115.793W" || record.getLatitude() !=
"37.2431N") {

  List<PartitionInfo> infoList =
cluster.availablePartitionsForTopic("TrackPartitionsTopic");
```

```
    partitionNumber = Math.abs(Utils.murmur2(keyBytes) % infoList.size() -
1);

}

    return partitionNumber;

    }

    @Override

    public void close() {

    }

    @Override

    public void configure(Map<String, ?> map) {

    }

}

        import io.confluent.kafka.serializers.KafkaAvroSerializer;

        import io.github.fernanda.maia.kafka.avro.Track;

        import io.github.fernanda.maia.partitioner.TrackPartitioner;

        import org.apache.kafka.clients.producer.KafkaProducer;

        import org.apache.kafka.clients.producer.ProducerRecord;
```

```
        import java.util.Properties;

        public class TrackProducer {

    public static void main(String[] args) {

    Properties props = new Properties();

    props.setProperty("bootstrap.servers", "http://localhost:9092");

    props.setProperty("key.serializer", KafkaAvroSerializer.class.getName());

    props.setProperty("value.serializer",
KafkaAvroSerializer.class.getName());

    props.setProperty("schema.registry.url", "http://localhost:8081");

    props.setProperty("partitioner.class", TrackPartitioner.class.getName());

    try(KafkaProducer<Long, Track> producer = new
KafkaProducer<>(props)) {

    Track track = Track.newBuilder()

    .setId(1L)

    .setLatitude("37.2431N")

    .setLongitude("115.793W")

    .build();
```

```
    ProducerRecord<Long, Track> record = new
ProducerRecord<>("TrackPartitionsTopic", track.getId(), track);

    producer.send(record);

    } catch(Exception e) {

    e.printStackTrace();

    }

    }

}
```

```
        import io.confluent.kafka.serializers.KafkaAvroDeserializer;

        import io.github.fernanda.maia.kafka.avro.Track;

        import org.apache.kafka.clients.consumer.ConsumerRecords;

        import org.apache.kafka.clients.consumer.KafkaConsumer;

        import java.time.Duration;

        import java.util.Arrays;

        import java.util.Properties;

        public class TrackConsumer {

    public static void main(String[] args) {
```

```
    Properties props = new Properties();

    props.setProperty("bootstrap.servers", "http://localhost:9092");

    props.setProperty("key.deserializer",
KafkaAvroDeserializer.class.getName());

    props.setProperty("value.deserializer",
KafkaAvroDeserializer.class.getName());

    props.setProperty("schema.registry.url", "http://localhost:8081");

    props.setProperty("group.id", "TrackGroup");

    props.setProperty("specific.avro.reader", "true");

    try(KafkaConsumer<Long, Track> consumer = new
KafkaConsumer<>(props)) {

    consumer.subscribe(Arrays.asList("TrackAvroTopic",
"TrackPartitionsTopic"));

    while(true) {

    ConsumerRecords<Long, Track> records =
consumer.poll(Duration.ofSeconds(20));

    records.forEach(c -> {

    Track coordinates = c.value();

    System.out.println("\nID: " + c.key());
```

```
    System.out.println("LATITUDE: " + coordinates.getLatitude());
    System.out.println("LONGITUDE: " + coordinates.getLongitude());
    System.out.println("PARTITION: " + c.partition());
});
}
    } catch(Exception e) {
    e.printStackTrace();
}
}
}
```