Example Avro Schemas (dynamic)

Suppose you store two schemas in Schema Registry (or files):

1. Permanent Employee Schema (permanent-employee.avsc)

```
{
  "type": "record",
  "name": "PermanentEmployee",
  "namespace": "com.example",
  "fields": [
    { "name": "employeeId", "type": "string" },
    { "name": "name", "type": "string" },
    { "name": "salary", "type": "double" },
    { "name": "benefits", "type": "string" }
  ]
}
```

2. **Consultant Schema** (consultant-employee.avsc)

```
{
  "type": "record",
  "name": "ConsultantEmployee",
  "namespace": "com.example",
  "fields": [
    { "name": "employeeId", "type": "string" },
    { "name": "name", "type": "string" },
    { "name": "hourlyRate", "type": "double" },
    { "name": "contractDuration", "type": "string" }
  ]
}
```

Kafka Producer with GenericRecord (Java)

```
import org.apache.avro.Schema;
import org.apache.avro.generic.GenericData;
import org.apache.avro.generic.GenericRecord;
import org.apache.kafka.clients.producer.KafkaProducer;
import org.apache.kafka.clients.producer.ProducerRecord;
import io.confluent.kafka.serializers.KafkaAvroSerializer;

import java.nio.file.Files;
import java.nio.file.Paths;
import java.util.Properties;

public class DynamicEmployeeProducer {

  public static void main(String[] args) throws Exception {
    String topic = "employee-topic";

    // Load schema dynamically from file or Schema Registry
    Schema permanentSchema = new Schema.Parser().parse(
        new String(Files.readAllBytes(Paths.get("permanent-employee.avsc")))
    );

    Schema consultantSchema = new Schema.Parser().parse(
        new String(Files.readAllBytes(Paths.get("consultant-employee.avsc")))
    );

    // Choose schema dynamically (e.g., based on user input or logic)
    String employeeType = "permanent"; // or "consultant"
    Schema schema = employeeType.equals("permanent") ? permanentSchema :
consultantSchema;

    // Create GenericRecord
    GenericRecord employee = new GenericData.Record(schema);
```

```
        employee.put("employeeId", "EMP123");
        employee.put("name", "John Doe");

        if (employeeType.equals("permanent")) {
            employee.put("salary", 75000.0);
            employee.put("benefits", "Health, Dental");
        } else {
            employee.put("hourlyRate", 85.5);
            employee.put("contractDuration", "6 months");
        }

        // Kafka producer configuration
        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("key.serializer", KafkaAvroSerializer.class.getName());
        props.put("value.serializer", KafkaAvroSerializer.class.getName());
        props.put("schema.registry.url", "http://localhost:8081");

        KafkaProducer<Object, Object> producer = new KafkaProducer<>(props);
        producer.send(new ProducerRecord<>(topic, employeeType, employee));
        producer.close();

        System.out.println("Sent " + employeeType + " employee record.");
    }
}
```

Kafka Consumer with GenericRecord (Java) [Consumer: Read and detect schema dynamically]

```
import org.apache.kafka.clients.consumer.ConsumerRecord;
import org.apache.kafka.clients.consumer.ConsumerRecords;
import org.apache.kafka.clients.consumer.KafkaConsumer;
import io.confluent.kafka.serializers.KafkaAvroDeserializer;
import org.apache.avro.generic.GenericRecord;

import java.time.Duration;
import java.util.Collections;
import java.util.Properties;

public class DynamicEmployeeConsumer {
    public static void main(String[] args) {
        String topic = "employee-topic";

        Properties props = new Properties();
        props.put("bootstrap.servers", "localhost:9092");
        props.put("group.id", "dynamic-employee-consumer");
        props.put("key.deserializer", KafkaAvroDeserializer.class.getName());
        props.put("value.deserializer", KafkaAvroDeserializer.class.getName());
        props.put("schema.registry.url", "http://localhost:8081");
        props.put("specific.avro.reader", "false"); // Ensure GenericRecord

        KafkaConsumer<Object, Object> consumer = new KafkaConsumer<>(props);
        consumer.subscribe(Collections.singletonList(topic));

        while (true) {
            ConsumerRecords<Object, Object> records = consumer.poll(Duration.ofMillis(1000));
            for (ConsumerRecord<Object, Object> record : records) {
                String key = (String) record.key(); // employee type
                GenericRecord employee = (GenericRecord) record.value();

                System.out.println("Employee Type: " + key);
                System.out.println("Employee ID: " + employee.get("employeeId"));
                System.out.println("Name: " + employee.get("name"));
            if (key.equals("permanent")) {
               System.out.println("Salary: " + employee.get("salary"));
               System.out.println("Benefits: " + employee.get("benefits"));
            } else if (key.equals("consultant")) {
               System.out.println("Hourly Rate: " + employee.get("hourlyRate"));
               System.out.println("Contract Duration: " + employee.get("contractDuration"));
            }
            System.out.println("------------");
         }
      }
   }
}
```

<u>If schemas are stored in Schema Registry, you can load them like this:</u>

```
import io.confluent.kafka.schemaregistry.client.CachedSchemaRegistryClient;
import io.confluent.kafka.schemaregistry.client.SchemaRegistryClient;
import org.apache.avro.Schema;

SchemaRegistryClient schemaRegistry = new CachedSchemaRegistryClient
                                                              ("http://localhost:8081", 10);
Schema permanentSchema = schemaRegistry.getBySubjectAndId("permanent-employee-value", 1);
Schema consultantSchema = schemaRegistry.getBySubjectAndId("consultant-employee-value", 2);
```