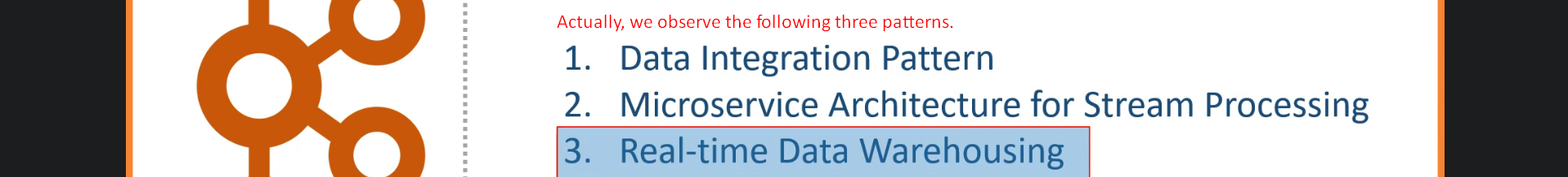
1. A picture containing graphical user interface

   Description automatically generated
2. **Till Now**:
   1. We learnt the five components of **Kafka** **Eco-System**.  
      Text

      Description automatically generated
3. **Agenda**:
   1. When to use when?  
      and what kinds of solutions are being developed using **Kafka Eco-System Components**.
4. Actually, we observe three kinds of patterns.



1. 
   1. This **pattern** is focused on the **Data Integration** problems and solve them using **Apache Kafka**.
   2. The scenarios which come under pattern may have a bunch of **independent systems**.
   3. They serve a specific purpose such as they generate data, store it, and own it.  
      They also need to send/share some part of the data with other systems.
   4. So, if you’re solving a **Data Integration** **problem** using **Kafka,** then you’re going to use a combination of first **three** **Kafka Components**.
      1. **Kafka Broker.**
      2. **Kafka Client APIs 🡺 Producer & Consumer APIs.**
      3. **Kafka Connect.**
   5. 01:20
2.   
   Shape

   Description automatically generated
   1. The 2nd and the most prevailing pattern is to create a **Real-time Stream Processing App** adopting a **Microservice Architecture**.
   2. In these apps, you’ll be using a combination of following **Kafka Components**.
      1. **Kafka Broker**
      2. **Only Producer API not Consumer API.**
      3. **Kafka Streams**
   3. **Kafka Broker** has the same solution in these solutions as well that is providing a **backbone infrastructure** & making Data available to all your Microservices.
   4. These solutions can be using **Kafka** for two purposes:
      1. **Creating Streams:** We use Kafka Producer to create a stream.
      2. **Processing Streams:** However, if we need to create a stream that is sourced from some **COTS Apps** then you might be using **Kafka Connect** but Kafka Connect may not apply to the Microservices and all of them will be using **Embedded Kafka Producer** with others.  
         Similarly, you will be using **Kafka Streams** for implementing your business logic and achieving your **Realtime Stream Processing needs**.  
         Kafka Consumers are not fit for that purpose, and they lack most of the capabilities to handle stream processing requirements and also everything that you can do with **Kafka Consumers** is doable with **Kafka Stream** and more easily.  
         So, you’re not likely to use **Kafka Consumers.**
3.   
   Diagram

   Description automatically generated
   1. This 3rd pattern is relatively new and still evolving.
   2. The idea is to make a **Realtime Data Warehouses**.
   3. So, in this pattern, like in a Data Warehouse, you’re going to collect data from a bunch of Source Systems into a **Kafka Cluster**.  
      Most of these sources are likely to have a DB backend or REST interface.  
      For Example, your OLTP (**O**n**l**ine **T**ransactional **P**rocessing) systems are likely to have a relational DB & many of **Cloud-Based Apps** such as **Salesforce** is going to have a **REST interface.**  
      So, you will be collecting data from all of these sources to your **Kafka Cluster**.  
      For this, you will be using **Kafka Connect**.
   4. Once your data starts coming to **Kafka Cluster**, you’re going to use your **Kafka Cluster** as a **Data** **Warehouse**.
   5. In this pattern, you will be modelling “all your workload and reporting requirements” using **KSQL** and run them on **KSQL Server.**
   6. The **KSQL** is going to generate a **real-time report** which keeps refreshing every minute, second or even in milliseconds.  
      This notion is relatively new and evolving.  
      But if such requirements are modeled correctly, it can bring a lot of quick and compelling solutions to the business.  
      The **Data Warehousing** and the **Data Lake** implementations always aspired to achieve a state where they have the most recent info and reports to make the right business decisions.  
      However, the most recent info has always been a few hours old, and in some cases a day or even older. **Kafka & KSQL** are moving towards offering a **real-time solution**.
   7. Everything in real-time is not at all possible but a lot can be achieved with a careful design.
   8. So, you might see that in coming days, **Kafka + KSQL** start challenging **Data Warehouse & Data Lake** for their **real-time workloads**.
4. 
   1. Table

      Description automatically generated with medium confidence
   2. Graphical user interface, application, PowerPoint

      Description automatically generatedThere are other deep full-fledged courses for these topics by me. Check them.
   3. Graphical user interface, application, table, Excel, PowerPoint

      Description automatically generated