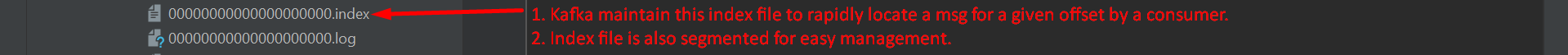
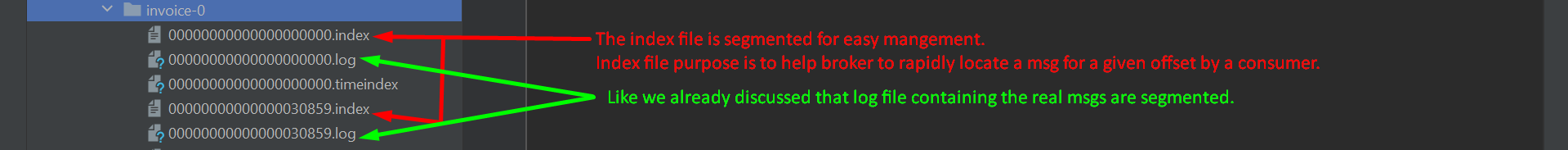
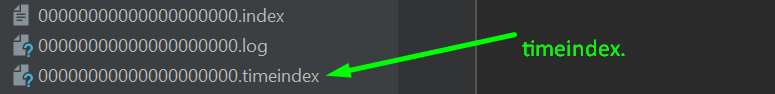
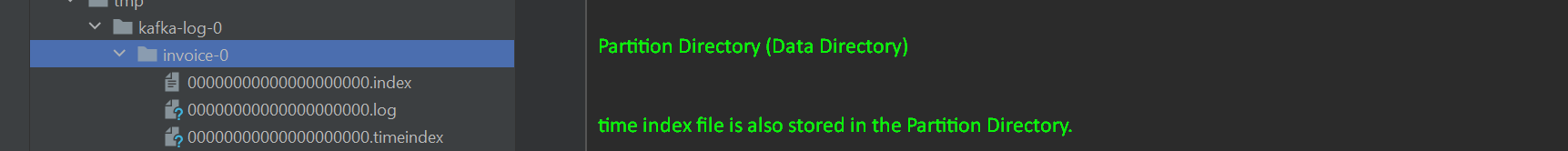
1. Graphical user interface, text, application

   Description automatically generated
2. Text

   Description automatically generated
3. 
4. At first sight, this arrangement of uniqueness looks strange.
5. Let me explain.
   1. If you think in DB terms, this unique identification doesn’t make a good sense for locating the msg.
   2. The Topic name is like table name.
   3. But partitions and offsets are just number.
   4. In DB, we query for data based on some column name and its value. Like  
      
   5. But we can’t do this in Kafka as msgs are not structured with column names.  
      That is why this arrangement of uniqueness looks strange.
   6. However, this numbering may not be a problem for a realtime streaming processing app.
   7. Let’s take an example to understand it.
   8. In stream processing app, the requirement is different.
   9. In Stream Processing app, the app wants to read all the msgs in seq.
   10. Let’s assume that you have a stream processing app which calculates the Loyalty Points for a customer in realtime.
   11. The app should read each invoice (msg) and calculate the loyalty point.  
       For calculation, we need to read customer id and amounts from each events (msgs).
   12. Let’s look at the sequence of activities.
       1. The app connects with the Broker and asks for the msg starting from the offset zero.
       2. Let’s assume that the Broker sends 10 (0-9 <- offsets) msgs to the app.
       3. The app takes a few seconds to calculate the loyalty points for those invoices.
       4. Now, the app is ready to process more invoices (msgs) and so it asks for msgs starting from next offset which is 10.
       5. Broker provides another batch of 15 msgs and the app process them too.
       6. This process continues throughout the life of the app.
       7. This process explained in this example, is the typical pattern of how a stream processing app would work.  
          In this process, we must have noticed that the consumer app requests for msg from Broker based on offset.  
          The point is clear. The Kafka allows consumer to fetch msgs from a given offset number.  
          This means if consumer demands for msgs starting from offset 100, the Kafka must be able to locate the msg from offset 100.  
          To help Broker to locate msgs rapidly for a given offset, Kafka maintains an index of offset.  
          
       8. The index file is segmented for easy management.  
          See above, the index files are also stored along with the log file segments under the partition directory.
       9. The final thing 🡪 
       10. We know that Kafka allows consumer to fetch msgs based on offset.   
           But many use cases, you may seek msgs based on timestamps.
       11. To meet this requirement, Kafka maintains the timestamp for each msg.   
           So, it creates a file with extension .timeindex to seek quickly the first msg that arrived at the given timestamp.  
           Time index is also like offset index and it is also segmented and stored in the partition directory.  
           The time index file, log file & index file are put all together under a partition directory as you can see above.  
           There are other files also under the Partition Directory but they have nothing to do with Data.  
           Kafka creates those files to keep some control information and clean them from time to time.
6. Great. This lecture discussed about the key concepts about the Kafka and how topics are logically organized.