b. Provide a written explanation, of no more than 500 words, of your Spark-Scala code for calculating these delays. The explanation should include your choice of Spark APIs and their brief explanation. Include a flowchart and explanatory figure/s where applicable.

Answer:
The first Spark API I choose is *'DataFrame'*. The similarities between *DataFrame* and RDD are all immutable distributed elastic datasets. The difference is that the data set of *DataFrame* is stored in the specified column, that is, structured data. Similar to the tables in traditional databases. Then if we want to use SQL style syntax, we need to register DataFrame into a table. Thus, I use *'catalog'* API to create and process the temporary table. The *'catalog'* API is added to Spark 2.0 to access metadata in Spark SQL. This API can not only operate Spark SQL, but also operate Hive metadata. Once we have created the *'catalog'* table, we can use it to query the data in the table. At last, I use *'Spark SQL'* API to execute SQL queries. *'Spark SQL'* is a Spark module for structured data processing. Unlike the basic Spark RDD API, the interfaces provided by Spark SQL provide Spark with more information about the structure of both the data and the computation being performed. Internally, Spark SQL uses this extra information to perform extra optimizations.

According to the question. I load the parquet-formatted dataset as a dataframe and register the loaded dataset into a temporary table using the createOrReplaceTempView method, then excute SQL queries to select the *'TailNum'* and *'DepDelay'* from table. After that, I re-arrange the delays in descending order and using 'LIMIT' to display only the first 20 records. And finally print out the results of the operation.

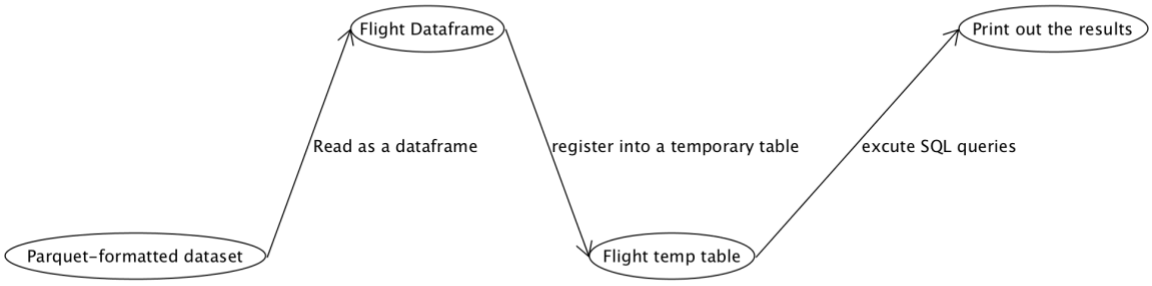

Figure 1. Flowchart of Display Maximum DepDelay