The paper describes a new module of Apache Spark, called Spark SQL. SparkSQL is built on top of the already existing Shark, providing new features to Apache Spark. SparkSQL provides two main functionalities besides the existing functionalities of the Shark: 1). Relational processing, using declarative queries and optimized storage. 2). Allow SQL users to use complex integration libraries for machine learning and analytics. It provides DataFrame API, which provides tighter integration between procedural and declarative processing. Catalyst, an extensible optimizer, is available using Scala programming making it easy to add composable rules, code generation, and define new extension points. Also, new features can be for complex modern-day analysis.

The paper proposes a concept of distributed collections known as Resilient Distributed Datasets (RDD). These RDD objects are collections of Java or Python objects partitioned across different clusters, which can be manipulated using higher-order functions such as filter, map, and reduce in any supported programming language. These RDDs are fault-tolerant and are computed lazily, thereby saving the memory footprints of the program. Spark SQL also provides in-memory caching, by materializing most requested data in memory, using columnar storage, which can be enabled by invoking cache() on a DataFrame. User Defined Functions are used to define custom functions in Spark SQL, providing advanced analytics and machine learning.

These DataFrames represent a logical plan to compute the dataset, which is not executed until an output operation is run. Catalyst, an optimizer, has two main purposes in Spark SQL. It provides support for semi-structured and no SQL data, in big data technologies with advancements in technology. It also allows external developers to extend the functionality of the optimizers for new data sources, by specifying rules and new data types. The catalyst consists of a general library representing trees and applying rules to manipulate these trees. The transformation phases of the tree in the Catalyst are as follows:
1). Resolving references by analyzing logical plan
2). Optimization of the logical plan
3). Physical planning
4). Compiling and generation of code for the query to Java byte code
The paper then provides evaluation of the Spark SQL with other systems, on the basis of the SQL query processing performance and Spark program performance.

Strong Points:
1). Spark SQL one of the advantages is that it is built on the existing Shark system, thus involving less development.
2). There are provisions for many different data types along with semi-structured data, which has applications in analytics and machine learning.
3). On performance evaluation, Spark SQL outperformed Shark system, and was competitive against Impala.

4). Spark SQL is an open source tool, hence providing access and improvements from a large pool of developers.

Weak Points:
1). Only selected join algorithms are supported at the moment using cost based optimization.
2). Paper provides less information of the comparison of the systems running SQL and procedural jobs separately and the Spark SQL system.
3). Paper fails to compare the Spark SQL with the Impala, as it is mentioned it performs competitively against Impala.
4). Paper also does not provide any comparison with traditional systems such as MapReduce.

Question:
1). It was hard to understand the concepts written in the Advanced Analytics subsection of the Related Work section. Thus, I would like to know more about it.
2). The paper provides little information on the improvements on MapReduce, which can be very useful to understand the paper.