The paper presents the F1 Query, a query processing platform, which is used for executing the declarative queries for data stored in different database sources and sinks. Thus, the F1 query focuses only on the query processing of the data, rather than focusing on the data storage and retrieval. The platform eliminates the overhead of maintaining the distinction of different OLTP, OLAP and ETL pipelines. Hence, F1 query instead of focusing on one requirement, manages to cover all the different requirements of the enterprise data processing and analysis. The F1 Query is flexible, enabling all small and large use cases of different business logic units to run at different data sources. F1 Query thereby disaggregates query processing from data storage in the data centers for modern big data solutions, where it is decentralized and replicated over multiple datacenters. The paper presents scalar, advanced functionalities and production metrics in the sections of the paper.

The architecture of the F1 query is influenced by following requirements of big data processing:Data Fragmentation, Datacenter Architecture, Scalability and Extensibility.
The architecture of the F1 Query is as follows: The user interacts with the F1 query through a client library, by sending requests to one of the dedicated servers known as F1 servers. F1 Masters are specialized nodes in the architecture, responsible for the runtime monitoring and executing of the query and maintenance. On receiving a request at F1 server, the server parses all the requests and extracts the data sources and sinks for the query access. If these sources are not available in the local datacenter, the F1 server sends the query back to the client with necessary information about the optimal data centers for the query processing. A new request is then sent from the client to the respective servers. Datasources can be any of the following: csv files, Spanner, ColumnarIO, BigTable or any file systems. Data sinks are used for storing the results of the queries, and different tables can be created using CREATE TABLE, and specified storage can be manually created by EXPORT DATA. F1 Query supports most of the SQL features, along with Protocol buffers (means of exchanging data from the structured sources). The F1 query is based on the same shared SQL dialect used by the other Google technologies: BigTable and SpannerDB. The paper describes the structure of the query optimizer in detail.

Planning phase is the first step of the query execution, where the query abstract syntax tree is converted into a directed acyclic graph of relational algebra operators by the optimizer, which are further optimized for both the physical and logical levels. The execution layer, finally on the basis of the client specification of the node preferences, executes the query on the servers and on workers available in an interactive or batch mode, using MapReduce framework. The paper describes both of these modes: batch and interactive descriptively in the paper, along with when and how these both execution modes are used. The F1 Query reuses the same logic for planning all the queries irrespective of the mode of execution of the query. Different execution frameworks used for the batch and interactive mode, use the same plans and execution kernel, thereby providing guaranteed support for all the query planning for both the execution mode.

Strong Points:
1). In F1 Query, the same SQL queries can be applied to smaller data, by using interactive execution mode, or apply it to large data, run in batch oriented mode. Thus, providing a benefit to F1 Query over other batch oriented systems.
2). F1 Query provides better fault tolerance, isolation and fairness in the scheduling, compared to Tenzing.
3). F1 Queries provides more computational power for the query processing, as it separates the storage and memory power required for the data storage and processing, by disaggregating both the needs of data querying.
4). F1 Query allows custom User Defined Functions(UDF) to be deployed separate from the data, and can be scaled independently, by different F1 workers, servers and MapReduce workers.
5). The paper provides much detail in comparing different Google technologies such as BigTable, Spanner, Dremel.,etc, in terms of the functionalities of the F1 Query.

Weak Points:
1). Although the scale out done by F1 Query is appreciated, the scaling in done by the F1 server is yet to be handled, as currently it results in operations latency and cost.
2). F1 Query having a broad scope compared to the PowerDrill does not provide the same level of optimization for the data exploration option.
3). F1 Query provides a coarse grained ability for the query restart functionality, which is not as good as Spanner.
4). The paper provides little information on the latency in the requests and response of the querying from the F1 Query client and receiving response from a datacenter.

Question:
1). Although the working of the clients is explained in the paper, can you please explain how and what technologies these clients use, such as SQL, Python?
2). Since, F1 Query is a mixture of different features available at different Google technologies for big data, can we infer that few of the other systems are obsolete?