1. **Hbase is a schema less database, what does it mean?**

In traditional RDBMS database the schema of the table is predefined and cannot and be modified easily. The schema is defined while making of the table i.e the datatype of the column. But in case of the HBASE the schema is mentioned at time of loading the data. Also the data stored in HBASE is using the bytes hence it is datatype independent.In this way even if new data comes into the database HBASE can handle it smoothly without any glitch.This is one of the pros of HBASE.

Moreover HBase doesn't have the concept of fixed columns schema. It is a wide column store and has column families which are roughly equivalent to tables. The column names can be completely variable and the number of columns can vary by row – so you could have a table with billions of rows and could have rows with 5 or 5 million columns.

Without HBase you can’t do table joins – and so is definitely more of a “schema-less” nature.Hence HBase is a schema less database.  “schema-less” doesn’t mean the data doesn’t have structure and data models can help us to design, rationalize, and communicate about data.

1. **What is the minimum number of column family every Hbase table should have?**

* HBase currently does not do well with anything above two or three column families so keep the number of column families in your schema low (i.e to 1).
* Currently, flushing and compactions are done on a per Region basis so if one column family is carrying the bulk of the data bringing on flushes, the adjacent families will also be flushed though the amount of data they carry is small.
* When many column families the flushing and compaction interaction can make for a bunch of needless i/o loading.

1. **What is the benefit of using connection pool in Hbase?**

* Creating connections to a server component from an application is a heavy weight operation and it is much pronounced when connecting to a database server.
* That being the reason database connection pooling is used to reuse connection objects and HBase is no exception.
* In HBase, data from meta table that stores details about region servers that can serve data for specific key ranges gets cached at the individual connection level that makes HBase connections much heavier.
* So if there are region movements for balancing or if a region server fails, the meta data need to be refreshed for each connection object which is a performance overhead.
* For these reasons, applications need to try to reuse connection objects created.