Give brief answers to the below questions:

1. When Hive is best suited and when is it not?

Hive is most suited for data warehouse applications where data is structured, static and formatted and where relatively static data is analyzed, fast response times are not required, and when the data is not changing rapidly.

Hive is not meant to be connected with systems which needs interactive processing.

1. When should one use Hive over MapReduce?

Mapping the familiar data operations to the low-level MapReduce Java API can be daunting, even for experienced Java developers. Hive does this dirty work for us, so that we can focus on the query itself. Hive translates most queries to MapReduce jobs, thereby exploiting the scalability of Hadoop

1. What is Hive metastore?

The Hive metastore service stores the metadata for Hive tables and partitions in a relational database, and provides clients (including Hive) access to this information via the metastore service API. The subsections that follow discuss the deployment options and provide instructions for setting up a database in a recommended configuration.

1. How can Hive improve performance with orc file format tables?

The Optimized Row Columnar (ORC) file format provides a highly efficient way to store Hive data. It was designed to overcome limitations of the other Hive file formats. Using ORC files improves performance when Hive is reading, writing, and processing data.

1. What is thrift server and client, jdbc and odbc driver importance in hive?

**Hive Server** – **The original**[**Hive Server**](https://cwiki.apache.org/confluence/display/Hive/HiveServer)**sometimes called Thrift server or HiveServer1.** A service through which clients may remotely issue Hive commands and requests. The Hive ODBC driver depends on Hive Server to perform the core set of database interactions. Hive Server is built as part of the Hive build process

**Apache Thrift** – a scalable cross-language software framework that enables the Hive ODBC driver (specifically the Hive client) to communicate with the Hive Server.

**Driver Architecture**

Internally, the Hive ODBC Driver contains two separate components: Hive client, and the unixODBC API wrapper.

**Hive client** – provides a set of C-compatible library functions to interact with Hive Server in a pattern similar to those dictated by the ODBC specification. However, Hive client was designed to be independent of unixODBC or any ODBC specific headers, allowing it to be used in any number of generic cases beyond ODBC.

**unixODBC API wrapper** – provides a layer on top of Hive client that directly implements the ODBC API standard. The unixODBC API wrapper will be compiled into a shared object library, which will be the final form of the Hive ODBC driver. The wrapper files will remain a file attachment on the associated JIRA until it can be checked into the unixODBC code repository.

6.What is the importance of partition in hive?

Partitioning is used for distributing execution load horizontally.

As the data is stored as slices/parts, query response time is faster to process the small part of the data instead of looking for a search in the entire data set.

For example, In a large user table where the table is partitioned by country, then selecting users of country ‘IN’ will just scan one directory ‘country=IN’ instead of all the directories.

1. What is the use of bucketing in hive?

* Bucketed tables offer efficient sampling than by non-bucketed tables. With sampling, we can try out queries on a fraction of data for testing and debugging purpose when the original data sets are very huge.
* As the data files are equal sized parts, map-side joins will be faster on bucketed tables than non-bucketed tables. In Map-side join, a mapper processing a bucket of the left table knows that the matching rows in the right table will be in its corresponding bucket, so it only retrieves that bucket (which is a small fraction of all the data stored in the right table).
* Similar to partitioning, bucketed tables provide faster query responses than non-bucketed tables.
* Bucketing concept also provides the flexibility to keep the records in each bucket to be sorted by one or more columns. This makes map-side joins even more efficient, since the join of each bucket becomes an efficient merge-sort.

1. What is the difference between static partitioning and dynamic partitioning in hive?

The difference between static and dynamic partitions is that with a static partition,  
the name of the partition is hardcoded in the insert statement, whereas with a  
dynamic partition, Hive will automatically determine the partition based on the value  
of the partition field.

 Using individual insert means static and single insert to partition table means dynamic.

hive> SET hive.exec.dynamic.partition=true;      \\Dynamic partitions needs to be explicitly enabled  in Hive//

hive> SET hive.exec.dynamic.partition=false; \\This Static  partitions needs to be explicitly enabled  in Hive//