**Explain the brief need of Hive .**

Hive is a data warehouse infrastructure tool to process structured data in Hadoop. It resides on top of Hadoop to summarize Big Data, and makes querying and analyzing easy**.**

Apache Hive is a [data warehouse](https://en.wikipedia.org/wiki/Data_warehouse) software project built on top of [Apache Hadoop](https://en.wikipedia.org/wiki/Apache_Hadoop) for providing data summarization, query, and analysis. Hive gives an [SQL](https://en.wikipedia.org/wiki/SQL)-like interface to query data stored in various databases and file systems that integrate with Hadoop. Traditional SQL queries must be implemented in the [MapReduce](https://en.wikipedia.org/wiki/MapReduce) Java API to execute SQL applications and queries over distributed data. Hive provides the necessary SQL abstraction to integrate SQL-like queries ([HiveQL](https://en.wikipedia.org/wiki/HiveQL" \o "HiveQL)) into the underlying Java without the need to implement queries in the low-level Java API. Since most data warehousing applications work with SQL-based querying languages, Hive aids portability of SQL-based applications to Hadoop. To accelerate queries, it provides indexes, including [bitmap indexes](https://en.wikipedia.org/wiki/Bitmap_index).

* Indexing to provide acceleration, index type including compaction and [bitmap index](https://en.wikipedia.org/wiki/Bitmap_index) as of 0.10, more index types are planned.
* Different storage types such as plain text, [RCFile](https://en.wikipedia.org/wiki/RCFile" \o "RCFile), [HBase](https://en.wikipedia.org/wiki/HBase" \o "HBase), ORC, and others.
* Metadata storage in a [relational database management system](https://en.wikipedia.org/wiki/Relational_database_management_system), significantly reducing the time to perform semantic checks during query execution.
* Operating on compressed data stored into the Hadoop ecosystem using algorithms including [DEFLATE](https://en.wikipedia.org/wiki/DEFLATE), [BWT](https://en.wikipedia.org/wiki/BWT), [snappy](https://en.wikipedia.org/wiki/Snappy_(compression)), etc.
* Built-in [user-defined functions](https://en.wikipedia.org/wiki/User-defined_function) (UDFs) to manipulate dates, strings, and other data-mining tools. Hive supports extending the UDF set to handle use-cases not supported by built-in functions.
* SQL-like queries (HiveQL), which are implicitly converted into MapReduce or Tez, or Spark jobs.

Hive v0.7.0 added integration with Hadoop security. Hadoop began using [Kerberos](https://en.wikipedia.org/wiki/Kerberos_(protocol)) authorization support to provide security. Kerberos allows for mutual authentication between client and server. In this system, the client’s request for a ticket is passed along with the request. The previous versions of Hadoop had several issues such as users being able to spoof their username by setting the hadoop.job.ugi property and also MapReduce operations being run under the same user: hadoop or mapred. With Hive v0.7.0’s integration with Hadoop security, these issues have largely been fixed. TaskTracker jobs are run by the user who launched it and the username can no longer be spoofed by setting the hadoop.job.ugi property. Permissions for newly created files in Hive are dictated by the [HDFS](https://en.wikipedia.org/wiki/Apache_Hadoop). The Hadoop distributed file system authorization model uses three entities: user, group and others with three permissions: read, write and execute. The default permissions for newly created files can be set by changing the umask value for the Hive configuration variable  hive.files.umask.value.

Hive it takes large amount of unstructured data and place it into a structured view that can be used by business analysts by the business tools. Hive supports use cases such as Ad-hoc queries,summarization,data analysis HIVEQL can also be exchange with custom scalar functions means user defined functions(UDF ' S),aggregations(UDFA's) and table functions(UDTF's).

Hive works with Hadoop to allow you to query and manage large-scale data using a familiar SQL-like interface. Hive provides command line interface to the shell and Microsoft HDInsight provides console access. Hive tables consist of data and schema and they are separated for maximum flexibility. Hive Query Language(HIVEQL) supports familiar SQL operations including joins,sub queries,Order By,Sort by etc. The CPU analysis depends on Hadoop configuration nodes and System configuration.

**Is Hive suitable to be used for OLTP systems? Explain in brief.**

No Hive does not provide insert and update at row level. So it is not suitable for OLTP system**,** But it supports OLAP transaction.

**OLTP** (online transaction processing) is a class of software programs capable of supporting transaction-oriented applications on the Internet. Typically, **OLTP**systems are used for order entry, financial transactions, customer relationship management (CRM) and retail sales.

**OLAP** is an acronym for Online Analytical Processing. **OLAP** performs multidimensional analysis of business data and provides the capability for complex calculations, trend analysis, and sophisticated data modeling

If application require OLTP system, then switch to NOSQL databases. HQL queries have High latency due to MapReduce. Hive is best suited for Data Warehousing Applications where data is stored, mined and reporting is done based on processing. As most Data Warehousing applications are based on relational database models, Hive bridges the gap between these applications and Hadoop. However, like most SQL interfaces, HiveQL does not conform to ANSI SQL standard. It differs in various ways.

To run mixed workloads – transactional (OLTP) and analytical (OLAP) – on the same Hadoop cluster. They had to maintain separate databases for these two different workloads and move data back and forth. With the Hadoop RDMS, they can run both the transactional (OLTP) and analytic workload (OLAP) on the same data platform, eliminating the need to duplicate data and deal with ETL bottlenecks. This also enables their entire process to scale-up affordably with increasing data volumes.

In OLTP, there is less number of writes, e.g. Hotel Information. Assuming that the Price changes happen every 5000 times per second, the reads may be more here. In such a scenario, there can be 1 write per second but reads could expel to hundreds and thousands. So the ratio here is around 1:1000.

But in OLAP, one can see different OLAP patterns, which means there are several writes happening simultaneously. In OLAP, we dump data in one shot i.e., all log files are put into data store and then we start processing. The data pattern or access pattern is exactly the opposite of OLTP kind of application. Here, the Hadoop or MapReduce will be useful.

Hive is not designed for [OLTP](http://searchdatacenter.techtarget.com/definition/OLTP) workloads and does not offer real-time queries or row-level updates. It is best used for batch jobs over large sets of append-only data. Hive supports text files (also called [flat files](http://searchsqlserver.techtarget.com/definition/flat-file)), SequenceFiles (flat files consisting of binary [key/value pairs](http://searchenterprisedesktop.techtarget.com/definition/key-value-pair)) and RCFiles (Record Columnar Files which store columns of a table in a [columnar database](http://searchdatamanagement.techtarget.com/definition/columnar-database) way.)

In case if we are in need to use OLTP , Then we have to move to HBASE . hBase has been the de-factor standard for OLTP in Hadoop. Single HBase cluster can be used both as OLTP and OLAP cluster, but depending on your needs you may want to segregate these clusters for different usage patterns and SLAs. HBase uses HDFS as the underlying storage layer. So, the efficiency and speed is as fast as a map-reduce job on HDFS can run. So Hive is not suitable for OLTP where for OLTP we need to use Hbase

**What is a metastore in Hive?**

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 using the metastore service API. Its a database which stores metadata a.k.a all the details about the tables you create in HIVE. By default, HIVE comes with and uses Derby database. But you can use any other database like MySQL or Oracle.

Metastore is the component that stores all the structure information of the various tables and partitions in the warehouse including column and column type information, the serializers and deserializers necessary to read and write data and the corresponding HDFS files where the data is stored.

Any JDBC compliant DB can be used for the metastore for Hive.

Hive Metastore is a central repository for Hive metadata. It has 2 components:

1. A Service to which the Hive Driver connects to and queries for the database schema.
2. A backing database to store the metadata**.**Currently Hive supports 5 backend databases: Derby, MySQL, MS SQL Server, Oracle and Postgres.

There are three modes to configure Hive Metastore:

**Embedded Metastore:**By default, the Metastore service runs in the same JVM with the hive service. In this case it uses embedded derby database stored on the local file-system. This mode of Hive has a limitation that only one session could be opened at a time as only one embedded Derby database can access the database files on disk. To allow multiple Hive services to connect the Metastore, Derby Is configured as a network server.

**Local Metastore:**Being a data-warehousing framework, a single session for Hive is not preferred. To solve this limitation of Embedded Metastore, a support for Local Metastore was developed. A separate database service runs as a process on same or remote machine. The Metastore service still runs in the same JVM within hive service. Before starting a Hive client, add the JDBC / ODBC driver libraries to the Hive lib folder**.**

**Remote Metastore:** There is one more configuration where one or more Metastore servers run as separate processes. This allows multiple Hive Clients to connect to a remote service rather than starting a Metastore service in the same JVM.

A Hive service is configured to use a remote Metastore by adding *hive.metastore.uris* property to Metastore server URIs. This property holds a comma-separated list of Metastore services. By default, the Hive service will connect to the first URI mentioned in the property. In case of a connection failure, it’ll randomly choose any of the Metastore and will try to reconnect

**Use of Metastore**: Whenever you fire a query from your Hive CLI, the Execution engine gathers all the details regarding the table and creates an Execution plan(Job). These details comes from Metastore. Finally the Execution engine sends the Job to Hadoop. From here, the common Hadoop Map Reduce Job is executed and the result is send back to Hive. The Name node communicates with Execution engine to successfully execute the MR Job.

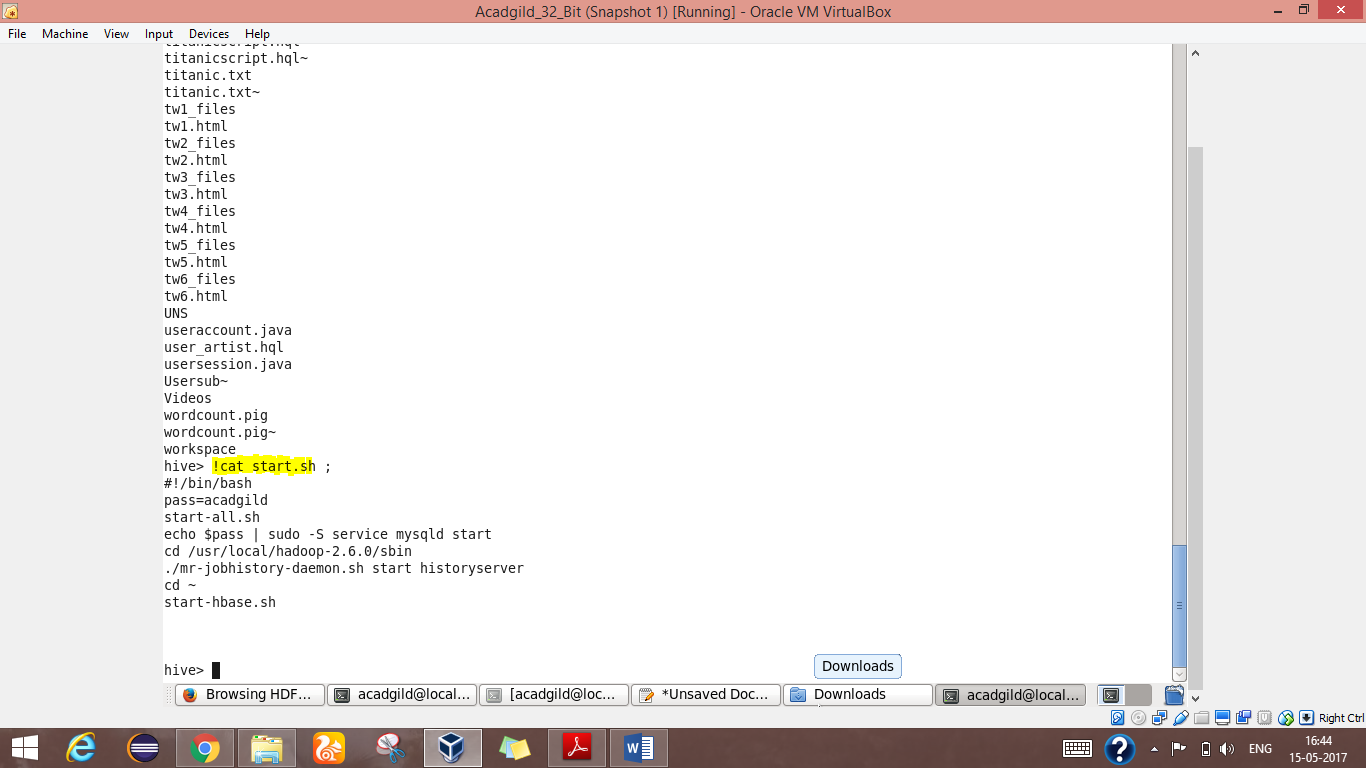
**Can we run unix shell commands from hive? Give 5 examples and share the screenshot**.

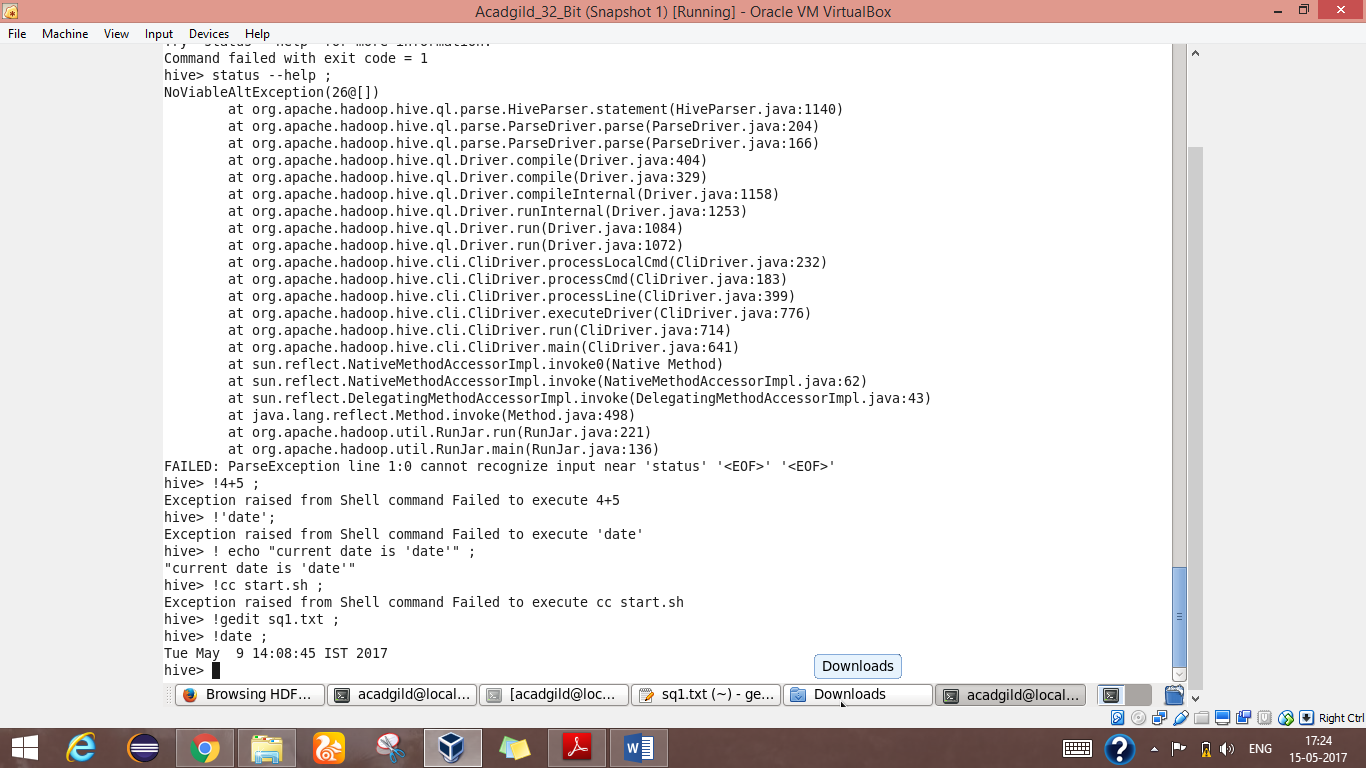
Yes,We can run unix shell commands from hive.

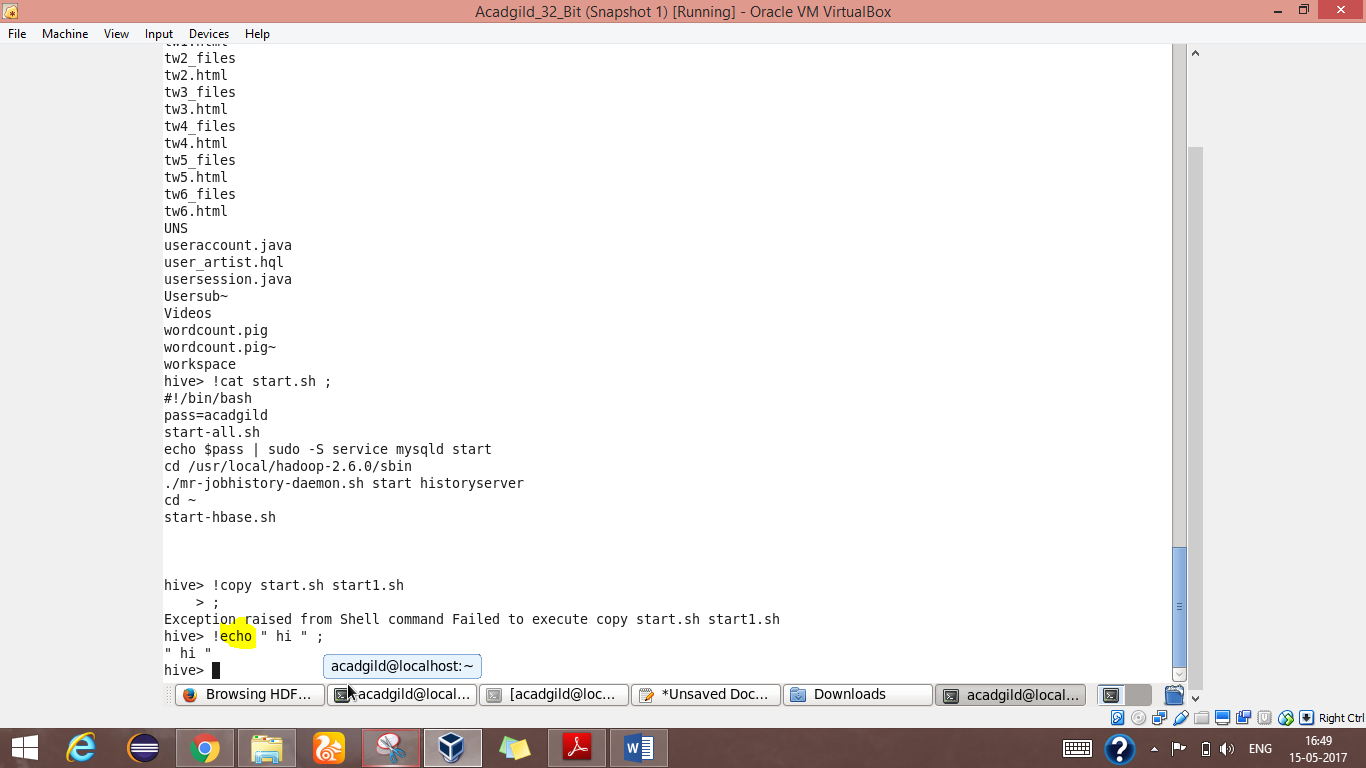
For example: To run ls in hive shell, run !ls

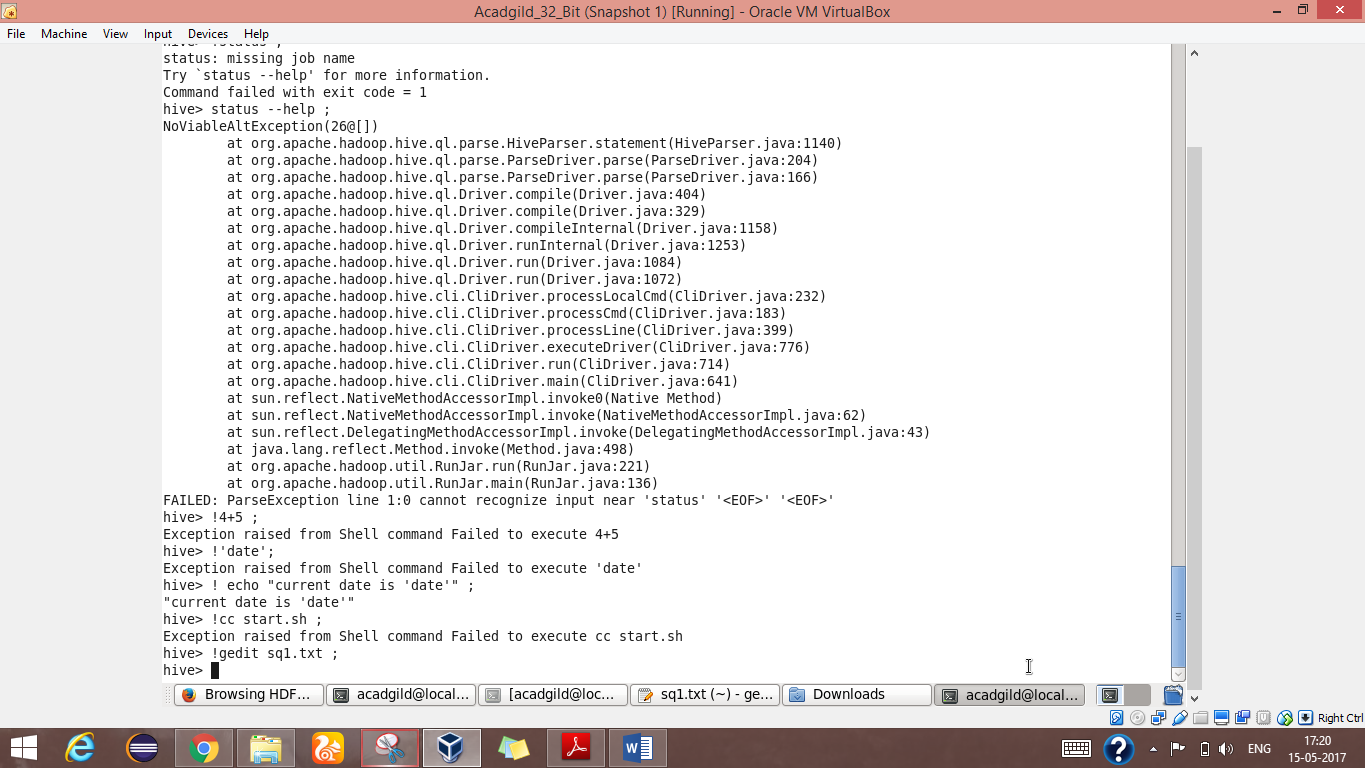
Syntax: !<cmd> - to execute shell command from script

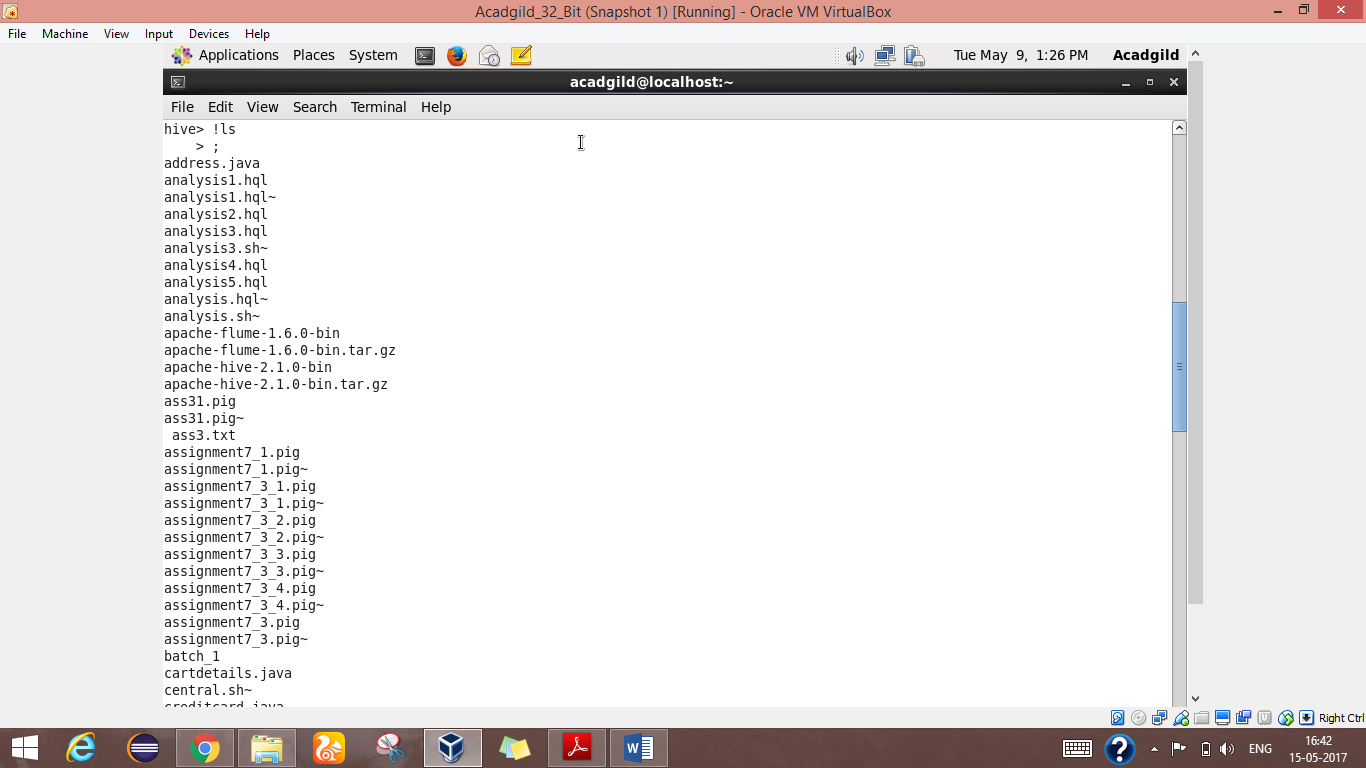
Example:

1.cat

2.date

3.echo

4.gedit

5.ls

**Hive can process any type of data formats? Explain in brief.**

Yes, Hive uses the SerDe interface for IO operations. Different SerDe interfaces can read and write any type of data. If normal directly process the data where as different type of data is in the Hadoop, Hive use different SerDe interface to process such data.

A SerDe is a short name for a Serializer Deserializer. Hive uses SerDe (and FileFormat) to read and write data from tables. An important concept behind Hive is that it DOES NOT own the Hadoop File System (HDFS) format that data is stored in.

Users are able to write files to HDFS with whatever tools/mechanism takes their fancy(“CREATE EXTERNAL TABLE” or “LOAD DATA INPATH,” ) and use Hive to correctly “parse” that file format in a way that can be used by Hive.

A SerDe is a powerful (and customizable) mechanism that Hive uses to “parse” data stored in HDFS to be used by Hive.

In Hive language, SerDe also called Serialization and DeSerialization. Usually when read/write the data, user first communicate with inputformat, then it connect with Record reader to read/write record. The data is stored in Serialized (binary) format in Record.

To serialize the data dat goes to row, here deserialized custem serde use object inspector to deserialize the data in fields. now user see the data in the fields, that deliver to the end user.

Example:

MetadataTypedColumnsetSerDe : used to read/write CSV format data.

JsonSerDe : process Json data.

RejexSerDe : process weblog data.

AvroSerde : Avro format data.