Data engineer interview:

->>>---HIVE------

Apache Hive is a data warehouse and an ETL tool which provides an SQL-like interface between the user and the Hadoop distributed file system (HDFS) which integrates Hadoop.
It is built on top of Hadoop. It is a software project that provides data query and analysis. It facilitates reading, writing and handling wide datasets that stored in distributed storage and queried by Structure Query Language (SQL) syntax.

------Components--------
HCatalog – It is a Hive component and is a table as well as a store management layer for Hadoop. It enables user along with various data processing tools like Pig and MapReduce which enables to read and write on the grid easily.
WebHCat – It provides a service which can be utilized by the user to run Hadoop MapReduce (or YARN), Pig, Hive tasks or function Hive metadata operations with an HTTP interface.

-----Architecture----------------
METASTORE – It is used to store metadata of tables schema, time of creation, location, etc. It also provides metadata partition to help the driver to keep the track of the progress of various datasets distributed over the cluster.
The metadata helps the driver to keep track of the data and it is crucial.
Hence, a backup server regularly replicates the data which can be retrieved in case of data loss.

DRIVER – It acts as a controller. The driver starts the execution of the statement by creating sessions and monitors the life cycle and progress of execution. It also stores metadata generated during the execution of an HQL query.

COMPILER – It is used to compile a Hive query, which converts the query to an execution plan.
This plan contains the tasks and steps needed to be performed by the Hadoop MapReduce to get the output as translated by the query.

OPTIMIZER – It optimizes and performs transformations on an execution plan to get an optimized Directed Acyclic Graph abbreviated as DAG.
Transformation such as converting a pipeline of joins to a single join, and splitting the tasks, such as applying a transformation on data before a reduce operation, to provide better performance and scalability.

EXECUTOR – It executes tasks after compilation and optimization have been done.
It interacts with the job tracker of Hadoop to schedule tasks to be run.
It takes care of pipelining the tasks by making sure that a task with dependency gets executed only if all other prerequisites are run.

----------------------Hadoop---------------------
Hadoop is a framework of the open source set of tools distributed under Apache License. It is used to manage data, store data, and process data for various big data applications running under clustered systems.

-----Components---------
HDFS-- Hadoop Distributed File System is a dedicated file system to store big data with a cluster of commodity hardware or cheaper hardware with streaming access pattern.
It enables data to be stored at multiple nodes in the cluster which ensures data security and fault tolerance.

Map Reduce : Data once stored in the HDFS also needs to be processed upon. Now suppose a query is sent to process a data set in the HDFS. Now, Hadoop identifies where this data is stored, this is called Mapping.
Now the query is broken into multiple parts and the results of all these multiple parts are combined and the overall result is sent back to the user. This is called reduce process. Thus while HDFS is used to store the data, Map Reduce is used to process the data.

YARN : YARN stands for Yet Another Resource Negotiator. It is a dedicated operating system for Hadoop which manages the resources of the cluster and also functions as a framework for job scheduling in Hadoop.
The various types of scheduling are First Come First Serve, Fair Share Scheduler and Capacity Scheduler etc. The First Come First Serve scheduling is set by default in YARN.

------------------------------YARN--------------------------------

Client: It submits map-reduce jobs.
Resource Manager: It is the master daemon of YARN and is responsible for resource assignment and management among all the applications.

Whenever it receives a processing request, it forwards it to the corresponding node manager and allocates resources for the completion of the request accordingly. It has two major components:
Scheduler: It performs scheduling based on the allocated application and available resources.
It is a pure scheduler, means it does not perform other tasks such as monitoring or tracking and does not guarantee a restart if a task fails.
The YARN scheduler supports plugins such as Capacity Scheduler and Fair Scheduler to partition the cluster resources.

Application manager: It is responsible for accepting the application and negotiating the first container from the resource manager. It also restarts the Application Master container if a task fails.

Node Manager: It take care of individual node on Hadoop cluster and manages application and workflow and that particular node.
Its primary job is to keep-up with the Resource Manager. It registers with the Resource Manager and sends heartbeats with the health status of the node.
It monitors resource usage, performs log management and also kills a container based on directions from the resource manager. It is also responsible for creating the container process and start it on the request of Application master.

Application Master: An application is a single job submitted to a framework. The application master is responsible for negotiating resources with the resource manager, tracking the status and monitoring progress of a single

application.
The application master requests the container from the node manager by sending a Container Launch Context(CLC) which includes everything an application needs to run.
Once the application is started, it sends the health report to the resource manager from time-to-time.

Container: It is a collection of physical resources such as RAM, CPU cores and disk on a single node.
The containers are invoked by Container Launch Context(CLC) which is a record that contains information such as environment variables, security tokens, dependencies etc.

Application workflow in Hadoop YARN:

-Client submits an application
-The Resource Manager allocates a container to start the Application Manager
-The Application Manager registers itself with the Resource Manager
-The Application Manager negotiates containers from the Resource Manager
-The Application Manager notifies the Node Manager to launch containers
-Application code is executed in the container
-Client contacts Resource Manager/Application Manager to monitor application's status
-Once the processing is complete, the Application Manager un-registers with the Resource Manager

----------Sqoop----------

Sqoop is a tool in which works in the following manner,
it first parses argument which is provided by user in the command-line interface and then sends those arguments to a further stage where arguments are induced for Map only job.
Once the Map receives arguments it then gives command of release of multiple mappers depending upon the number defined by the user as an argument in command line Interface.
Once these jobs are then for Import command, each mapper task is assigned with respective part of data that is to be imported on basis of key which is defined by user in the command line interface.
To increase efficiency of process Sqoop uses parallel processing technique in which data is been distributed equally among all mappers.
After this, each mapper then creates an individual connection with the database by using java database connection model and then fetches individual part of the data assigned by Sqoop.
Once the data is been fetched then the data is been written in HDFS or Hbase or Hive on basis of argument provided in command line. thus the process Sqoop import is completed.

The export process of the data in Sqoop is performed in same way, Sqoop export tool which available performs the operation by allowing set of files from the Hadoop distributed system back to the Relational Database management system.
The files which are given as an input during import process are called records, after that when user submits its job then it is mapped into Map Task that brings the files of data from Hadoop data storage,

and these data files are exported to any structured data destination which is in the form of relational database management system such as MySQL, SQL Server, and Oracle, etc.

---------SPARK---------

->Apache spark is an open source cluster computing framework for real-time data processing.
The main feature of apache spark is its in-memory cluster computing that increases the processing speed of an application.
Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance.
It is designed to cover a wide range of workloads such as batch applications, iterative algorithms, interactive queries, and streaming.

->Rdds are the building blocks of any spark application. Rdds stands for:
resilient: fault tolerant and is capable of rebuilding data on failure
distributed: distributed data among the multiple nodes in a cluster
dataset: collection of partitioned data with values

->Transformations: they are the operations that are applied to create a new rdd.
Actions: they are applied on an rdd to instruct apache spark to apply computation and pass the result back to the driver.

->Lazy evaluation:
apache spark delays its evaluation till it is absolutely necessary.
This is one of the key factors contributing to its speed. For transformations, spark adds them to a dag (directed acyclic graph) of computation and only when the driver requests some data, does this dag actually gets executed.

->The structtype and structfield classes in pyspark are used to define the schema to the dataframe and create complex columns such as nested struct, array, and map columns.
Structtype is a collection of structfield objects that determines column name, column data type, field nullability, and metadata.
Pyspark imports the structtype class from pyspark.sql.types to describe the dataframe's structure. The dataframe's printschema() function displays structtype columns as "struct."
To define the columns, pyspark offers the pyspark.sql.types import structfield class, which has the column name (string), column type (datatype), nullable column (boolean), and metadata (metadata).

->We can use different storage levels for caching the data.
Disk_only: persist data on disk only in serialized format.
Memory_only: persist data in memory only in deserialized format.
Memory_and_disk: persist data in memory and if enough memory is not available evicted blocks will be stored on disk.
Off_heap: data is persisted in off-heap memory. Refer spark.memory.offheap.enabled in spark doc.
We can explicitly specify whether to use replication while caching data by using methods such as disk_only_2, memory_and_disk_2, etc.
We can also specify whether to serialize data while storing. Methods like memory_only_ser etc. Using serialized format will increase processing time but

decrease the memory footprint.
Serialization with replication is also available e.g. Memory_only_ser_2

->The serialization process is used to conduct performance tuning on spark. The data sent or received over the network to the disk or memory should be persisted. Pyspark supports serializers for this purpose.
It supports two types of serializers, they are:
pickleserializer: this serializes objects using python's pickleserializer (class pyspark.pickleserializer). This supports almost every python object.
Marshalserializer: this performs serialization of objects. We can use it by using class pyspark.marshalserializer. This serializer is faster than the pickleserializer but it supports only limited types.

->Sparksession is the entry point to pyspark and is the replacement of sparkcontext since pyspark version 2.0. This acts as a starting point to access all of the pyspark functionalities related to rdds, dataframe, datasets etc.
It is also a unified api that is used in replacing the sqlcontext, streamingcontext, hivecontext and all other contexts.

->>>CLUSTER MANAGER--->
Standalone – This is a simple cluster manager that is included with Spark.
Apache Mesos – This manager can run Hadoop MapReduce and PySpark apps.
Hadoop YARN – This manager is used in Hadoop2.
Kubernetes – This is an open-source cluster manager that helps in automated deployment,
scaling and automatic management of containerized apps.
local – This is simply a mode for running Spark applications on laptops/desktops.

The interpreter is the first layer, using a scala interpreter, spark interprets the code with some modifications.
Spark creates an operator graph when you enter your code in spark console.
When we call an action on spark rdd at a high level, spark submits the operator graph to the dag scheduler.
Divide the operators into stages of the task in the dag scheduler.
A stage contains task based on the partition of the input data. The dag scheduler pipelines operators together. For example, map operators schedule in a single stage.
The stages pass on to the task scheduler. It launches task through cluster manager. The dependencies of stages are unknown to the task scheduler.
The workers execute the task on the slave.

At higher level, we can apply two type of rdd transformations: narrow transformation (e.g. Map(), filter() etc.) And wide transformation (e.g. Reducebykey()).
Narrow transformation does not require the shuffling of data across a partition, the narrow transformations will group into single stage while in wide transformation the data shuffles.
Hence, wide transformation results in stage boundaries.

Azure synapse

->Data integration, enterprise data warehousing, and big data analytics are all

part of azure synapse analytics, an unlimited analytics service.
 It allows users to query data at scale on your own terms, utilizing either serverless or dedicated solutions.
The dedicated sql pool's service level determines the number of compute nodes, which might range from 1 to 60.
->A dedicated sql pool (originally sql dw) represents a collection of allocated analytic resources. The term "analytic resources" refers to a combination of cpu, memory, and i/o resources.
Data warehouse units are compute scale units that combine cpu, memory, and i/o resources (dwus). A dwu is a normalised, abstract measure of compute resources and performance.
->Data discovery & classification, dynamic data masking, vulnerability assessment, advanced threat protection,
and transparent data encryption are some of the data protection features offered by synapse analytics for dedicated sql pools.
-> Create security policy transact-sql statement to implement rls row level security

Azure data engg->
azure data factory is the service that helps you to do the migrations and orchestrate the work. For example, if you want to move the data from on-premise to cloud,
maybe it is incremental or it will be lift and shift work there you can leverage the azure data factory to do the work.
It support around hundred of the different data sources from which we can pull the data.

Delta is storing the data as parquet,
just has an additional layer over it with advanced features,
providing history of events, (transaction log) and more flexibility on changing the content like,
update, delete and merge capabilities. This link delta explains quite good how the files organized.

One drawback that it can get very fragmented on lots of updates,
which could be harmful for performance.
 AS the AZ Data Lake Store Gen2 is anyway not optimized for large IO this is not really a big problem.
 Some optimization on the parquet format though will not be very effective this way.

I would use delta, just for the advanced features.
It is very handy if there is a scenario where the data is updating over time, not just appending.
Specially nice feature that you can read the delta tables as of a given point in time they existed.

---Broadcast----------
When you run a PySpark RDD, DataFrame applications that have the Broadcast variables defined and used, PySpark does the following.

PySpark breaks the job into stages that have distributed shuffling and actions are executed with in the stage.

Later Stages are also broken into tasks
Spark broadcasts the common data (reusable) needed by tasks within each stage.
The broadcasted data is cache in serialized format and deserialized before executing each task.
You should be creating and using broadcast variables for data that shared across multiple stages and tasks.

Note that broadcast variables are not sent to executors with sc.broadcast(variable) call instead,
they will be sent to executors when they are first used.

---------Accumulators-----
The PySpark Accumulator is a shared variable that is used with RDD and DataFrame to perform sum
and counter operations similar to Map-reduce counters.
These variables are shared by all executors to update and add information through aggregation or computative operations.