1. **Explain about the different complex data types in pig**

**ANS)** Pig has a very limited set of data types. Pig data types are classified into two types. They are:

* Primitive
* Complex

**Complex Types**: Pig supports three complex data types. They are listed below: 

* **Tuple**: An ordered set of fields. Tuple is represented by braces. Example: (1,2)
* **Bag** : A set of tuples is called a bag. Bag is represented by flower or curly braces. Example: {(1,2),(3,4)}
* **Map** : A set of key value pairs. Map is represented in a square brackets. Example: [key#value] . The # is used to separate key and value.

1. **How can you interact with the shell in Apache pig**

**ANS)**

Shell Commands

1.1. **fs**

Invokes any FsShell command from within a Pig script or the Grunt shell.

* + 1. **Syntax**

fs subcommand subcommand\_parameters

* + 1. **Usage**

Use the fs command to invoke any FsShell command from within a Pig script or Grunt shell. The fs command greatly extends the set of supported file system commands and the capabilities supported for existing commands such as ls that will now support globing. For a complete list of FsShell commands, see File System Shell Guide

1.1.3. **Examples I**n these examples a directory is created, a file is copied, a file is listed.

fs -mkdir /tmp

fs -copyFromLocal file-x file-y

fs -ls file-y

**1.2. sh**

Invokes any sh shell command from within a Pig script or the Grunt shell.

1.2.1. **Syntax:** sh subcommand subcommand\_parameters

1.2.2. **Usage** Use the sh command to invoke any sh shell command from within a Pig script or Grunt shell. Note that only real programs can be run form the sh command. Commands such as cd are not programs but part of the shell environment and as such cannot be executed unless the user invokes the shell explicitly, like "bash cd".

1.2.3. **Example** In this example the ls command is invoked.

grunt> sh ls

bigdata.conf

nightly.conf

.....

grunt>

**3.)Explain how pig differs from Map reduce**

**ANS)**

* First of all, looking at the Pig Latin script we notice that it's only seven lines of code. If nothing else was different, we could say it's faster to write in Pig Latin just because of the number of lines. Next, we see that we don't have to import any libraries. Lastly, we can focus on is how much easier it is to read for someone with a little SQL background. Look at some of the keywords that we have seen in SQL: Group By and Order By.
* Pig is application that runs on top of MapReduce and abstracts Java MapReduce jobs away from developers.
* Pig Latin uses a lot fewer lines of code than the Java MapReduce script.
* The Pig Latin script was is easier to read for someone without a Java background.
* MapReduce jobs can written in Pig Latin.
* Java is a great and powerful language, but it has a higher learning curve than something like Pig Latin. Therefore, using a higher-level language, like Pig Latin, enables many more developers/analysts to write MapReduce jobs.

**4.)Explain how pig differs from sql**

**ANS)** The DBMS systems that SQL operates on, are considered to be faster than MapReduce (operated on by Pig through the PigLatin platform). However, it is the loading of data that is more challenging in case of RDBMS, making the set up difficult. PigLatin offers a number of advantages in terms of declaring execution plans, ETL routines and pipeline modification.

SQL is declarative and PigLatin is procedural to a large extent. What we mean by this is in SQL, we largely specify “what” is to be accomplished and in Pig, we mention “how” a task is to be performed. A script written in Pig is essentially converted to a MapReduce job in the background before it is executed. A Pig script is shorter than the corresponding MapReduce job, which significantly cuts down development time.

**5.)Explain the scalar data types in pig**

**ANS)**

|  |  |  |
| --- | --- | --- |
| **Simple Data Types** | Description | Example |
| Scalars |  |  |
| int | Signed 32-bit integer | 10 |
| long | Signed 64-bit integer | Data:     10L or 10l  Display: 10L |
| float | 32-bit floating point | Data:     10.5F or 10.5f or 10.5e2f or 10.5E2F  Display: 10.5F or 1050.0F |
| double | 64-bit floating point | Data:     10.5 or 10.5e2 or 10.5E2  Display: 10.5 or 1050.0 |
| Arrays |  |  |
| chararray | Character array (string) in Unicode UTF-8 format | hello world |
| bytearray | Byte array (blob) |  |
|  |  |  |