**Why MapReduce program is needed in Pig Programming?**

* 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.

**Advantages of pig over mapreduce:**

* Pig’s multi-query approach combines certain types of operations together in a single pipeline, reducing the number of times data is scanned.  This means **1/20th the lines of code and 1/16th the development time** when compared to writing raw MapReduce.
* Pig got its name because it’s omnivorous – it will happily consume any data you feed it: structured, semi-structured, or unstructured.
* Pig provides the common data operations (filters, joins, ordering, etc.) and nested data types (e.g. tuples, bags, and maps) missing from MapReduce.
* It’s easy to learn (especially if you’re familiar with SQL) and opens Hadoop to data professionals who may not be software engineers.
* PigLatin reads like a series of steps (e.g. join this data to that data, then filter the result…) so it is easy to write, and even better, it is easy to read.
* Pig is easily extensible by UDFs – including Python, Java, JavaScript, and Ruby – so you can use them to load, aggregate, or do sophisticated analysis.
* Pig insulates your code from changes to the Hadoop Java API, so your jobs won’t suddenly break due to an update.  It also manages all details of submitting jobs and running complex data flows.
* Pig is open source and actively supported by an impressive community of developers who are constantly committing back code.  It also has lots of big-time users: LinkedIn, Twitter, Salesforce, Stanford University, and many more.
* Pig engine:
* parses, optimizes, and automatically executes PigLatin scripts as a series of MapReduce jobs on a Hadoop cluster

**Modes of Pig execution:**

Pig has two execution modes or exectypes:

* **Local Mode** - To run Pig in local mode, you need access to a single machine; all files are installed and run using your local host and file system. Specify local mode using the -x flag (pig -x local).
* **Mapreduce Mode** - To run Pig in mapreduce mode, you need access to a Hadoop cluster and HDFS installation. Mapreduce mode is the default mode; you can, *but don't need to*, specify it using the -x flag (pig OR pig -x mapreduce).

You can run Pig in either mode using the "pig" command (the bin/pig Perl script) or the "java" command (java -cp pig.jar ...).

**Grunt Shell in Pig:**

After invoking the Grunt shell, you can run your Pig scripts in the shell. In addition to that, there are certain useful shell and utility commands provided by the Grunt shell.

**Features of Pig Latin language:**

A Pig Latin statement is an operator that takes a [relation](https://pig.apache.org/docs/r0.8.1/piglatin_ref2.html#Relations%2C+Bags%2C+Tuples%2C+Fields) as input and produces another relation as output. (This definition applies to all Pig Latin operators except LOAD and STORE which read data from and write data to the file system.) Pig Latin statements can span multiple lines and must end with a semi-colon ( ; ). Pig Latin statements are generally organized in the following manner:

1. A LOAD statement reads data from the file system.
2. A series of "transformation" statements process the data.
3. A STORE statement writes output to the file system; or, a DUMP statement displays output to the screen.

**Is Pig Latin commands case sensitive?:**

The names (aliases) of relations and fields are case sensitive. The names of Pig Latin functions are case sensitive. The names of parameters (see Parameter Substitution) and all other Pig Latin keywords are case insensitive.

In the example below, note the following:

1. The names (aliases) of relations A, B, and C are case sensitive.
2. The names (aliases) of fields f1, f2, and f3 are case sensitive.
3. Function names PigStorage and COUNT are case sensitive.
4. Keywords LOAD, USING, AS, GROUP, BY, FOREACH, GENERATE, and DUMP are case insensitive. They can also be written as load, using, as, group, by, etc.
5. In the FOREACH statement, the field in relation B is referred to by positional notation ($0).

**Dataflowlanguage:**

In a dataflow language, you have a stream of data **which is passed from instruction to instruction to be processed.** Conditional execution, jumps and procedure calls route the data to different instructions. This could be seen as data flowing through otherwise static instructions like how electrical signals flow through circuits or water flows through pipes. A dataflow "if" statement would route the data to the correct branch.