## PRACTICAL NO :05

PIG

### Unit Structure :

* 1. Objectives
  2. Introduction
  3. Summary
  4. References
  5. Unit End Exercises

## OBJECTIVES

Pig is an open-source high level data flow system. It provides a simple language called Pig Latin, for queries and data manipulation, which are then compiled in to MapReduce jobs that run on Hadoop.

## INTRODUCTION

Pig is important as companies like Yahoo, Google and Microsoft are collecting huge amounts of data sets in the form of click streams, search logs and web crawls. Pig is also used in some form of ad-hoc processing and analysis of all the information.

## Why Do You Need Pig?

* + - It’s easy to learn, especially if you’re familiar with SQL.
    - Pig’s multi-query approach reduces 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.
    - Performance of Pig is in par with raw MapReduce
    - Pig provides data operations like filters, joins, ordering, etc. and nested data types like tuples, bags, and maps, that are missing from MapReduce.
    - Pig Latin is easy to write and read.

## Why was Pig Created?

Pig was originally developed by Yahoo in 2006, for researchers to have an ad-hoc way of creating and executing MapReduce jobs on very large data sets. It was created to reduce the development time through its multi-query approach. Pig is also created for professionals from non-Java background, to make their job easier.

### Where Should Pig be Used?

Pig can be used under following scenarios:

When data loads are time sensitive.

* + - When processing various data sources.
    - When analytical insights are required through sampling.

### Pig Latin – Basics

Pig Latin is the language used to analyze data in Hadoop using Apache Pig. In this chapter, we are going to discuss the basics of Pig Latin such as Pig Latin statements, data types, general and relational operators, and Pig Latin UDF’s. Pig Latin – Data Model As discussed in the previous chapters, the data model of Pig is fully nested. A Relation is the outermost structure of the Pig Latin data model. And it is a bag where −

* + - A bag is a collection of tuples.
    - A tuple is an ordered set of fields.
    - A field is a piece of data.

### Pig Latin – Statements

While processing data using Pig Latin, statements are the basic constructs.

* + - These statements work with relations. They include expressions and schemas.
    - Every statement ends with a semicolon (;).
    - We will perform various operations using operators provided by Pig Latin, through statements.
    - Except LOAD and STORE, while performing all other operations, Pig Latin statements take a relation as input and produce another relation as output.
    - As soon as you enter a Load statement in the Grunt shell, its semantic checking will be carried out. To see the contents of the schema, you need to use the Dump operator. Only after performing the dump operation, the MapReduce job for loading the data into the file system will be carried out.

### Example

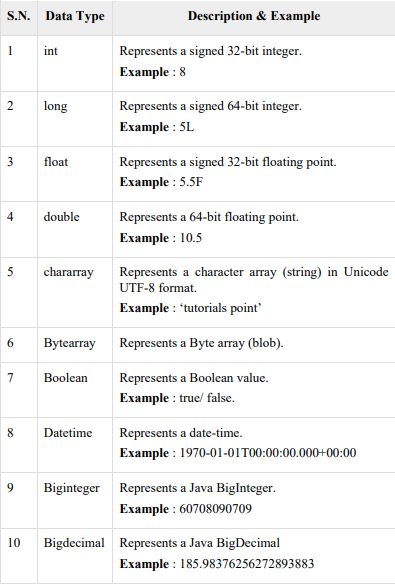
Given below is a Pig Latin statement, which loads data to Apache Pig.

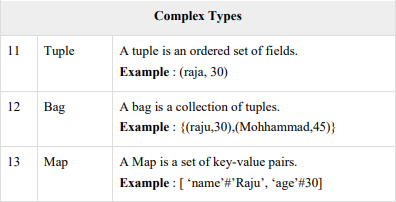
*grunt> Student\_data = LOAD 'student\_data.txt' USING PigStorage(',')as*

*( id:int, firstname:chararray, lastname:chararray, phone:chararray, city:chararray );*

### Pig Latin – Data types

Given below table describes the Pig Latin data types.





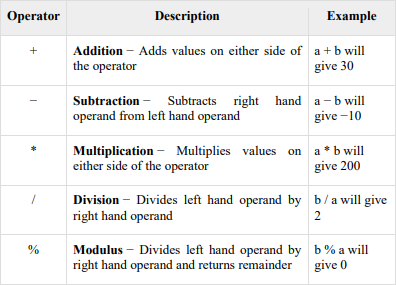
### Null Values

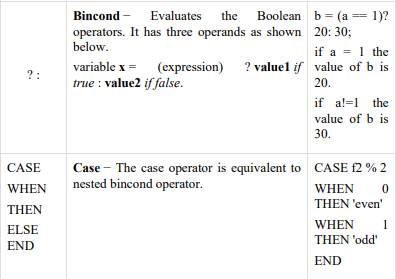
Values for all the above data types can be NULL. Apache Pig treats null values in a similar way as SQL does.

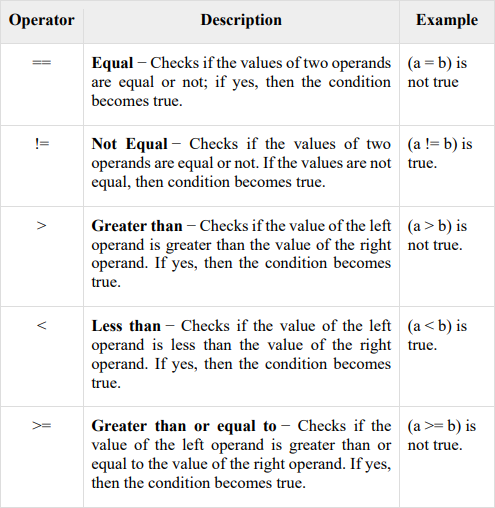
A null can be an unknown value or a non-existent value. It is used as a placeholder for optional values. These nulls can occur naturally or can be the result of an operation.

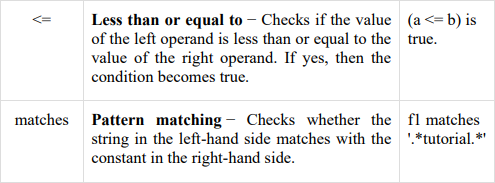
Pig Latin – Arithmetic Operators

The following table describes the arithmetic operators of Pig Latin. Suppose a = 10 and b = 20.



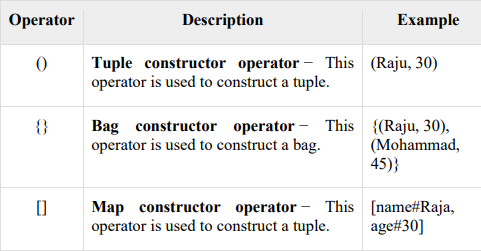






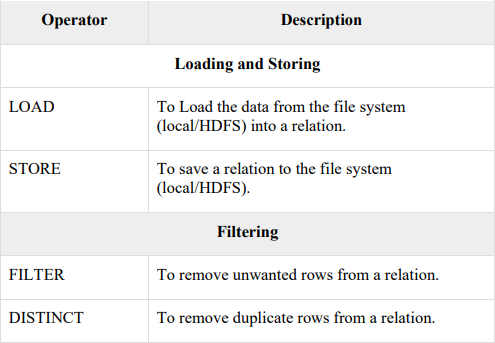
### Pig Latin – Type Construction Operators

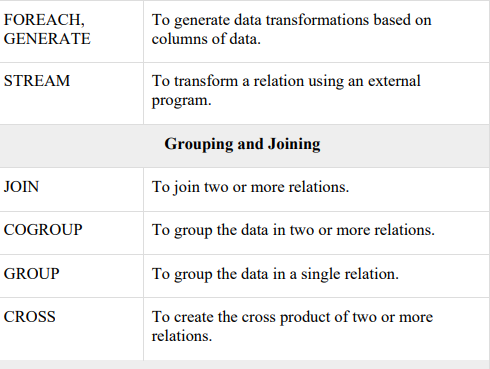
The following table describes the Type construction operators of Pig Latin.

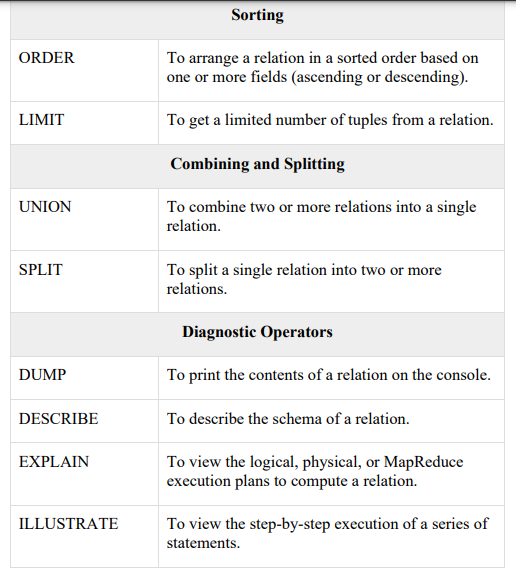


### Pig Latin – Relational Operations

The following table describes the relational operators of Pig Latin







### Apache Pig - Grunt Shell

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. This chapter explains the shell and utility commands provided by the Grunt shell.

### Shell Commands

The Grunt shell of Apache Pig is mainly used to write Pig Latin scripts. Prior to that, we can invoke any shell commands using sh and fs.

sh Command Using sh command, we can invoke any shell commands from the Grunt shell. Using sh command from the Grunt shell, we cannot execute the commands that are a part of the shell environment (ex − cd).

Syntax Given below is the syntax of sh command. grunt> sh shell command parameters

**Example** We can invoke the ls command of the Linux shell from the Grunt shell using the sh option as shown below. In this example, it lists out the files in the /pig/bin/ directory.

grunt> sh ls pig

pig\_1444799121955.log pig.cmd

pig.py

fs Command

Using the fs command, we can invoke any FsShell commands from the Grunt shell.

### Syntax

Given below is the syntax of fs command. grunt> sh File System command parameters

**Example** We can invoke the ls command of HDFS from the Grunt shell using fs command. In the following example, it lists the files in the HDFS root directory.

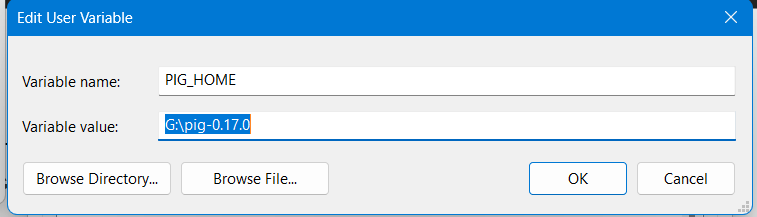
grunt> fs –ls Found 3 items

drwxrwxrwx - Hadoop supergroup 0 2015-09-08 14:13 Hbase

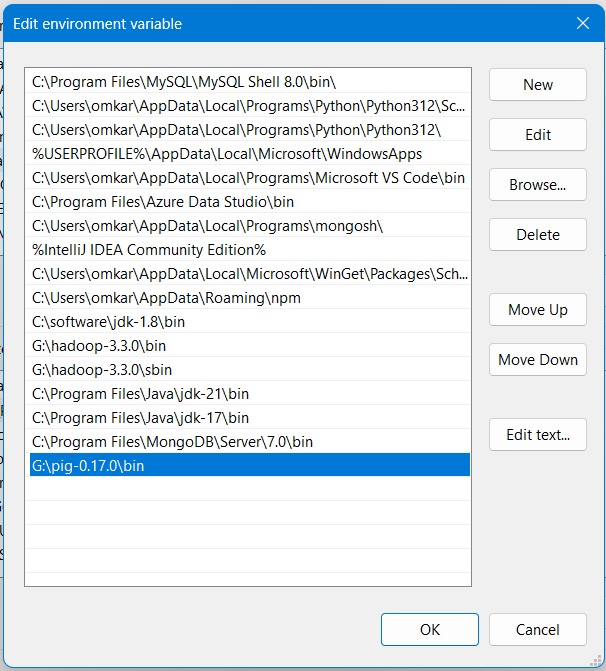
drwxr-xr-x - Hadoop supergroup 0 2015-09-09 14:52 seqgen\_data

drwxr-xr-x - Hadoop supergroup 0 2015-09-08 11:30 twitter\_data

1. Set PIG\_HOME in environment variable G:\pig-0.17.0



1. add path G:\pig 0.17.0\bin

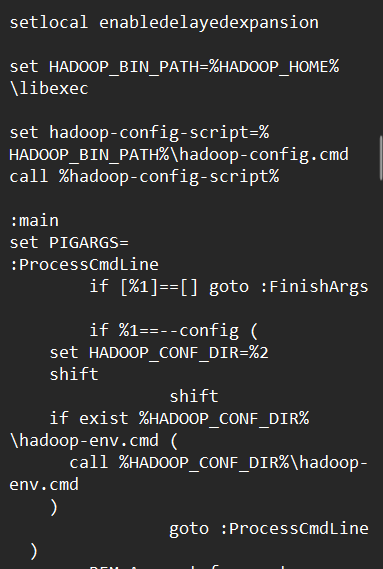


# Find the line in pig.cmd in \bin folder :

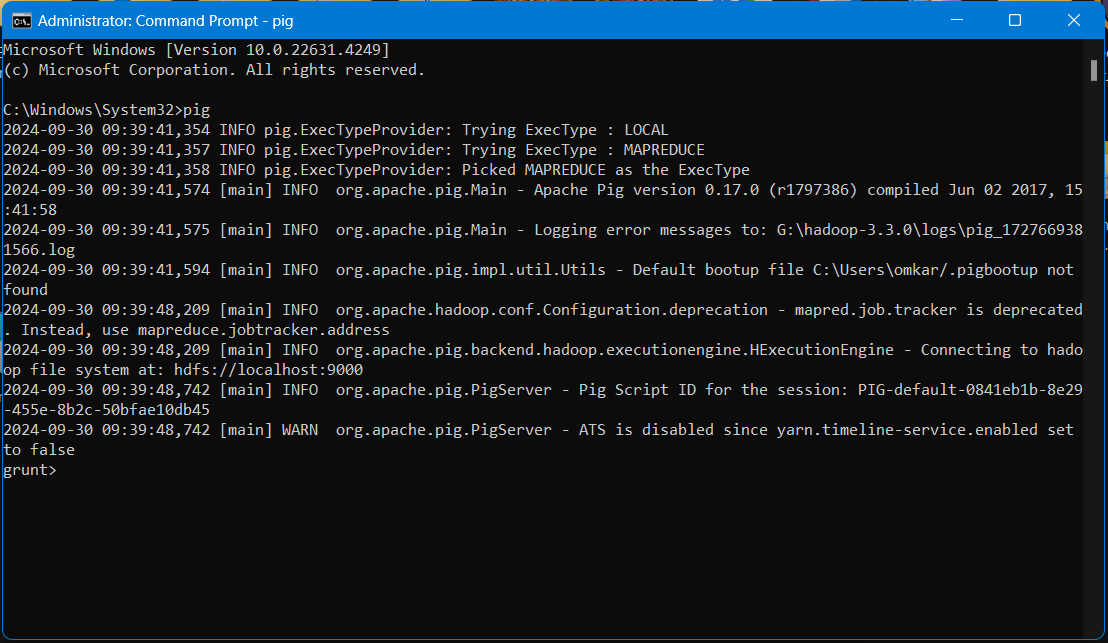
set HADOOP\_BIN\_PATH=%HADOOP\_HOME%\bin

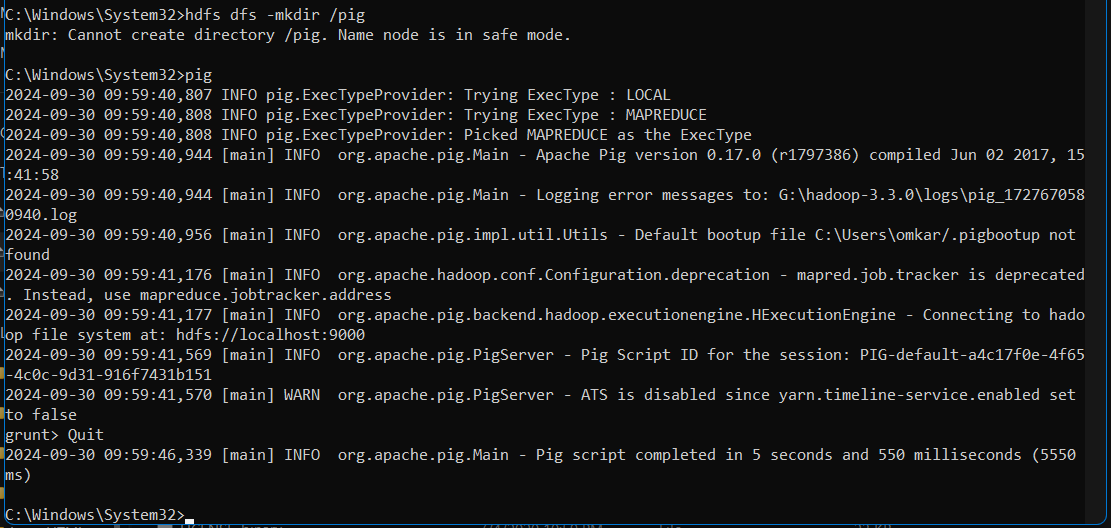
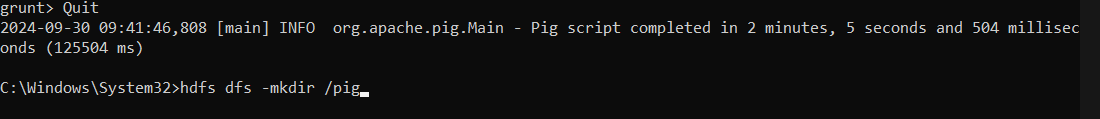
# Replace this line by:

set HADOOP\_BIN\_PATH=%HADOOP\_HOME%\libexec



1. check pig command on cmd





1. run other database related commands

There are 2 Ways of Invoking the grunt shell:

Local Mode: All the files are installed, accessed, and run in the local machine itself. No need to use HDFS. The command for running Pig in local mode is as follows.

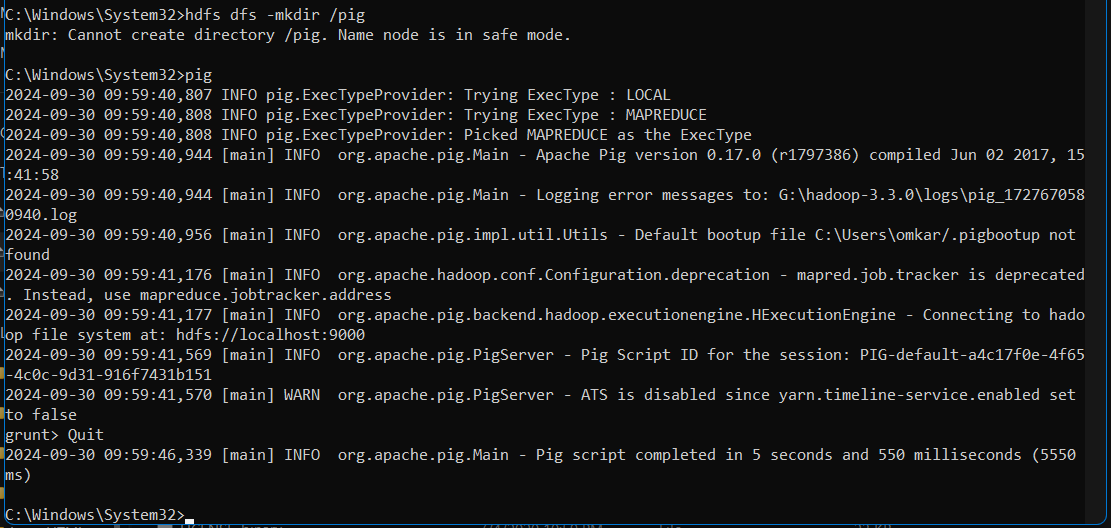
*pig -x local*

MapReduce Mode: The files are all present on the HDFS . We need to load this data to process it. The command for running Pig in MapReduce/HDFS Mode is as follows.

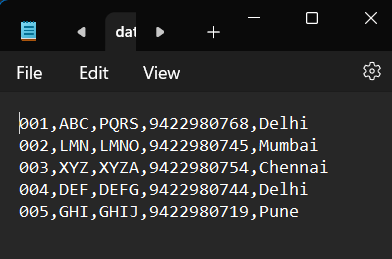
*pig -x mapreduce*

*Or use pig -x local*

*Or use pig*

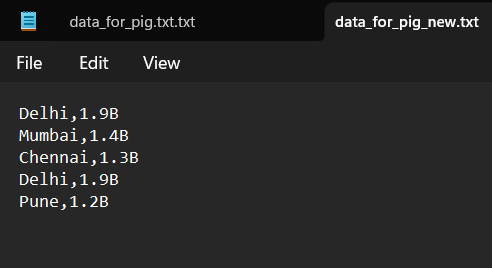


1. Create a file data\_for\_pig.txt with following data: 001,ABC,PQRS,9422980768,Delhi 002,LMN,LMNO,9422980745,Mumbai 003,XYZ,XYZA,9422980754,Chennai 004,DEF,DEFG,9422980744,Delhi 005,GHI,GHIJ,9422980719,Pune



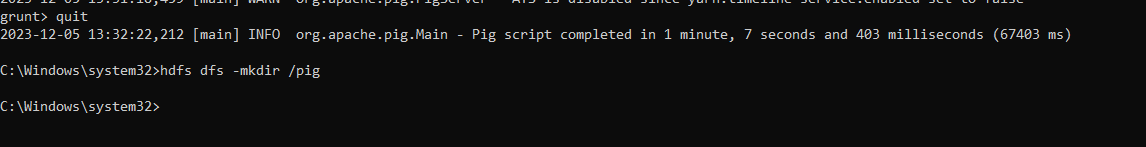
Create a file data\_for\_pig\_new.txt with following data: Delhi,1.9B

Mumbai,1.4B Chennai,1.3B Delhi,1.9B Pune,1.2B

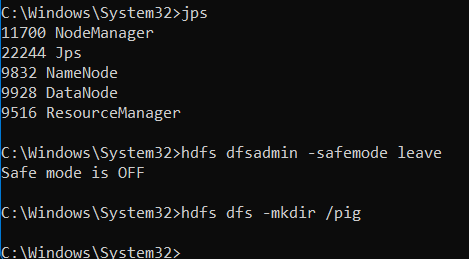


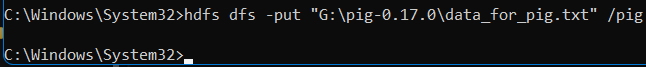
1. hdfs dfs -mkdir /pig

[ Note: **quit** grunt shell first and then run above command ]

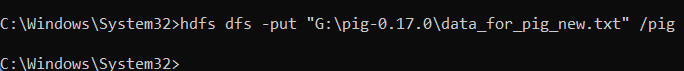


1. Upload your file on HDFS using following command: hdfs dfs -put "G:\pig-0.17.0\data\_for\_pig.txt" /pig





1. Upload your another file on HDFS using following command: hdfs dfs -put "G:\pig-0.17.0\data\_for\_pig\_new.txt" /pig



*4. pig -x mapreduce*

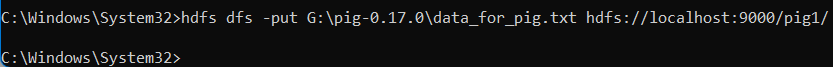
*copy above txt file into pig/bin or*

*put using following command*

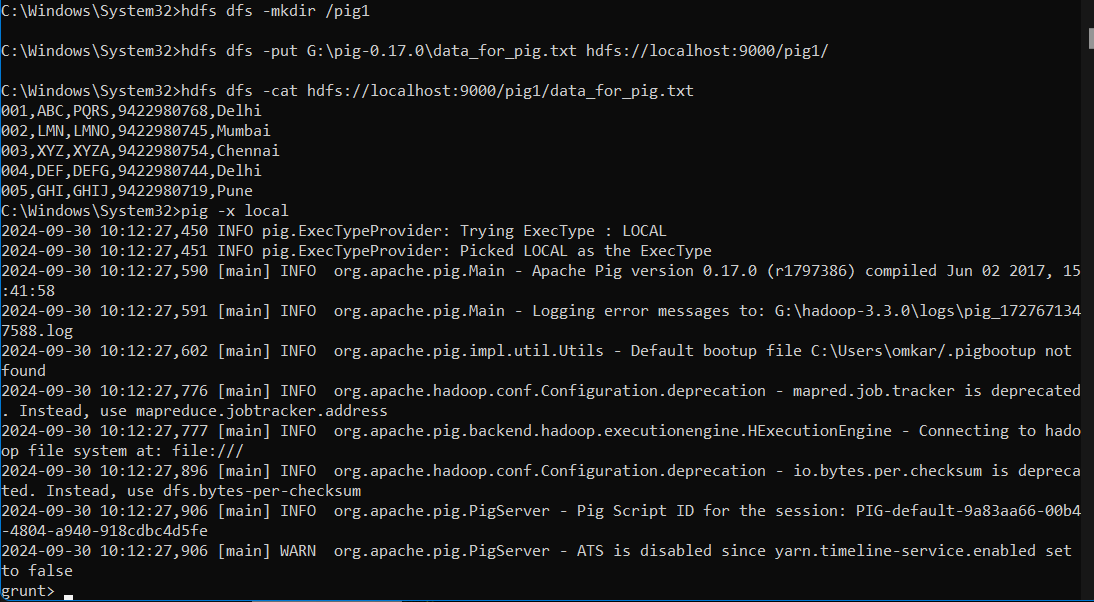
*hdfs dfs -put C:\Users\student\Downloads\BigData\data\_for\_pig.txt hdfs://localhost:9000/pig1/*

*cat data of file*

*hdfs dfs -cat hdfs://localhost:9000/pig1/data\_for\_pig.txt*



*hdfs dfs -cat hdfs://172.16.4.5:9000/pig1/data\_for\_pig.txt pig -x mapreduce or pig -x local*

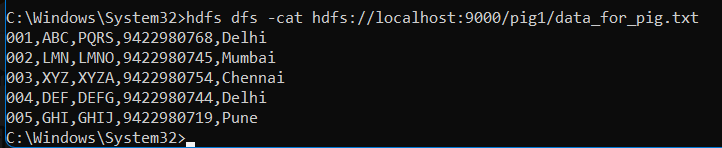


Create variable student as follows:

*student = LOAD 'hdfs://172.16.4.4:9000/pig/data\_for\_pig.txt' USING PigStorage(',') as (id:int,fname:chararray,lname:chararray,phone:chararray,city:chararray);*

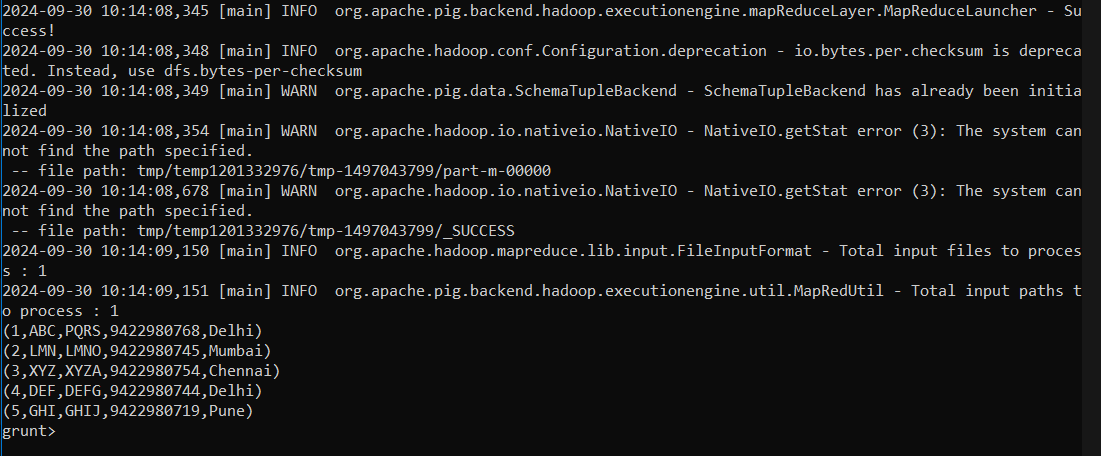
*Or*

*student = LOAD 'hdfs://localhost:9000/pig/data\_for\_pig.txt' USING PigStorage(',') as (id:int,fname:chararray,lname:chararray,phone:chararray,city:chararray);*



*dump student;*



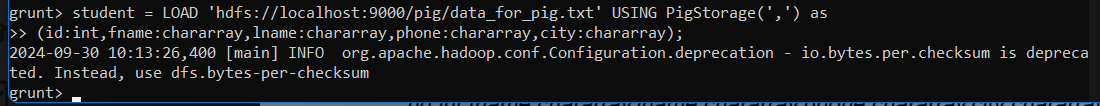


*Create another variable city as follows:*

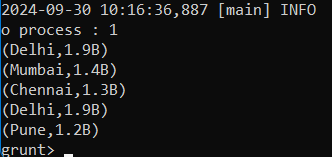
*city = LOAD 'hdfs://172.16.4.4:9000/pig/data\_for\_pig\_new.txt' USING PigStorage(',') as (city:chararray, population:chararray);*

*Or*

*city = LOAD 'hdfs://localhost:9000/pig/data\_for\_pig\_new.txt' USING PigStorage(',') as (city:chararray, population:chararray);*

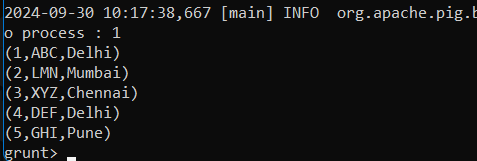




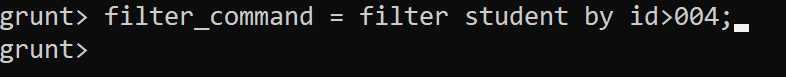


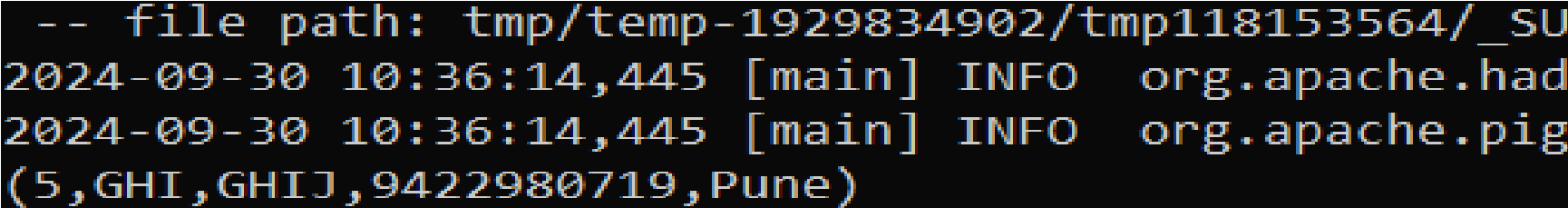
*for\_each\_stud1 = foreach student generate id, fname, city; dump for\_each\_stud1;*

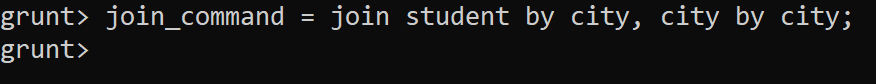




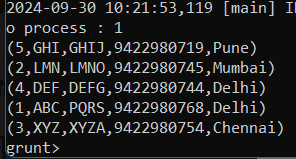
*filter\_command = filter student by id>004; dump filter\_command;*



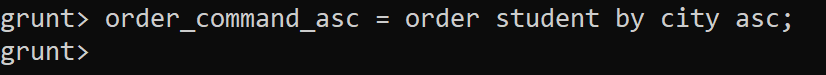


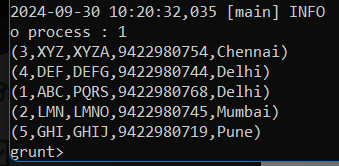
*join\_command = join student by city, city by city;*

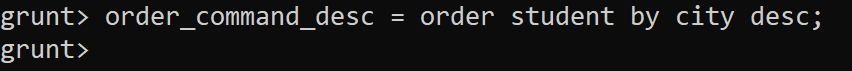
*dump join\_command;*

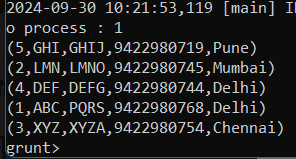


#### order by

*order\_command\_asc = order student by city asc; dump order\_command\_asc;*



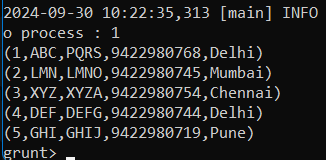
*order\_command\_desc = order student by city desc; dump order\_command\_desc;*



#### Distinct [ to get o/p, you require to should have duplicate data in your file]

*distinct\_command = distinct student; dump distinct\_command;*





#### Store

*store order\_command\_desc into '/desc';*

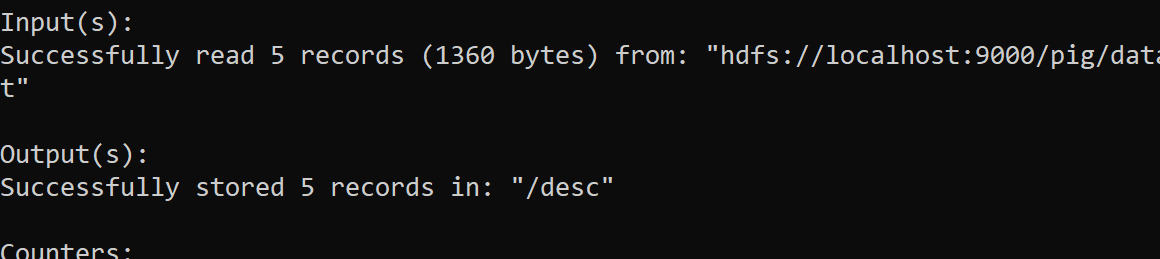
*output:*

Input(s):

Successfully read 5 records (1360 bytes) from: "hdfs://localhost:9000/pig/data\_for\_pig.txt"

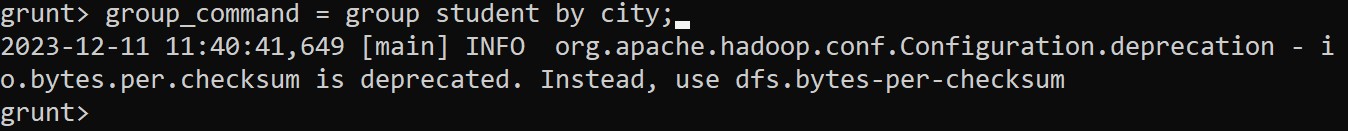
Output(s):

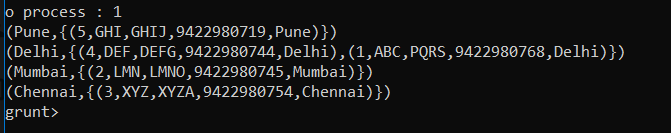
Successfully stored 5 records in: "/desc"



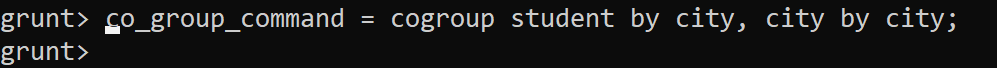
**Group command (Duplication is city is required)**

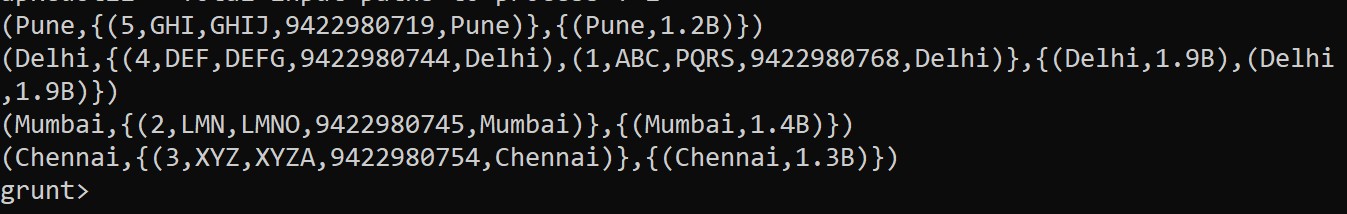
group\_command = group student by city; dump group\_command;



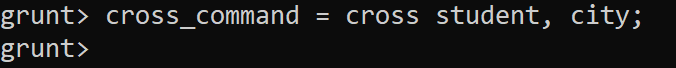


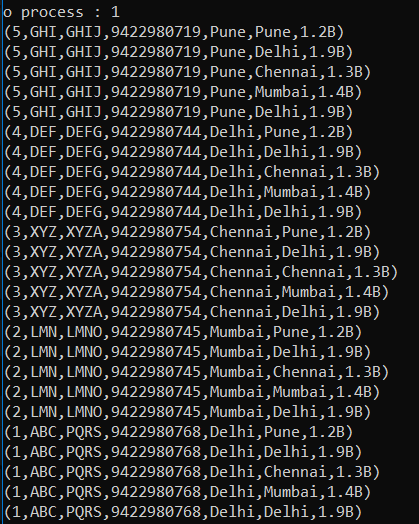
#### Cogroup

co\_group\_command = cogroup student by city, city by city; dump co\_group\_command;



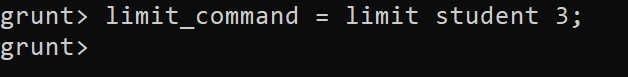
### Cross:

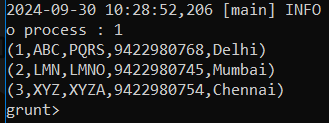
cross\_command = cross student, city; dump cross\_command;



### Limit:

limit\_command = limit student 3; dump limit\_command;

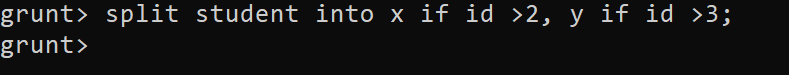




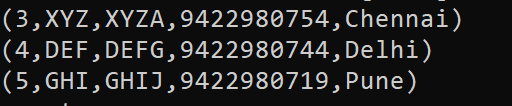
### Split:

split student into x if id >2, y if id >3; dump x;

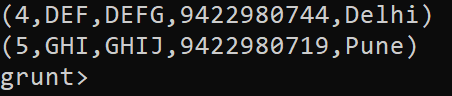
dump y;



dump x;

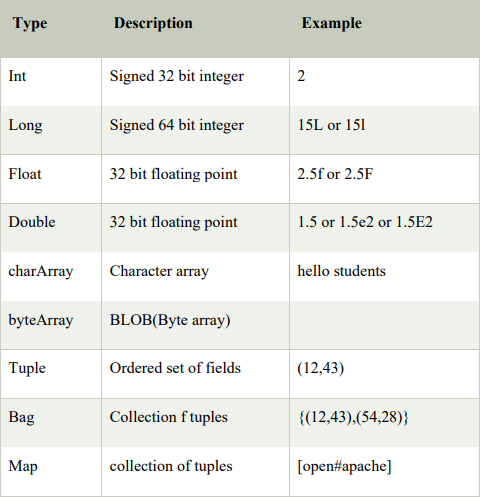


dump y;



**5.2 SUMMARY**

Pig Data Types Apache Pig supports many data types. A list of Apache Pig Data Types with description and examples are given below.

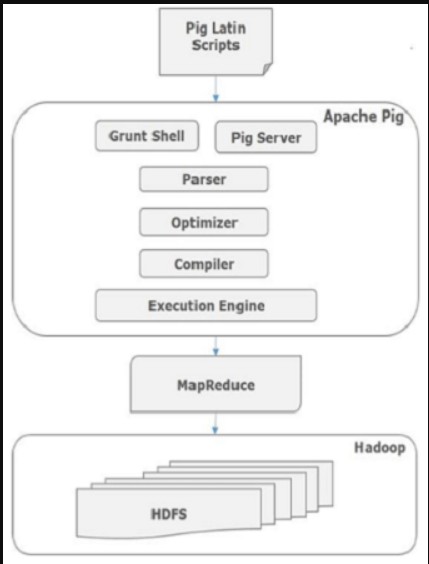


### Apache Pig - Architecture

Creating a data model in pig: The language used to analyze data in Hadoop using Pig is known as Pig Latin. It is a highlevel data processing language which provides a rich set of data types and operators to perform various operations on the data. To perform a particular task Programmers using Pig, programmers need to write a Pig script using the Pig Latin language, and execute them using any of the execution mechanisms (Grunt Shell, UDFs, Embedded).

After execution, these scripts will go through a series of transformations applied by the Pig Framework, to produce the desired output.

Internally, Apache Pig converts these scripts into a series of MapReduce jobs, and thus, it makes the programmer’s job easy. The architecture of Apache Pig is shown below.



### Apache Pig Components

As shown in the figure, there are various components in the Apache Pig framework. Let us take a look at the major components.

### Parser

Initially the Pig Scripts are handled by the Parser. It checks the syntax of the script, does type checking, and other miscellaneous checks. The output of the parser will be a DAG (directed acyclic graph), which represents the Pig Latin statements and logical operators. In the DAG, the logical operators of the script are represented as the nodes and the data flows are represented as edges.

### Optimizer

PIG The logical plan (DAG) is passed to the logical optimizer, which carries out the logical optimizations such as projection and pushdown.

### Compiler

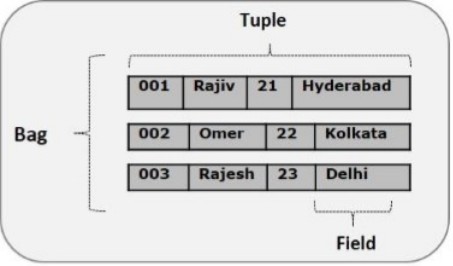
The compiler compiles the optimized logical plan into a series of MapReduce jobs.

### Execution engine

Finally the MapReduce jobs are submitted to Hadoop in a sorted order. Finally, these MapReduce jobs are executed on Hadoop producing the desired results.

### Pig Latin Data Model

The data model of Pig Latin is fully nested and it allows complex non atomic data types such as map and tuple. Given below is the diagrammatical representation of Pig Latin’s data model.



### Atom

Any single value in Pig Latin, irrespective of their data, type is known as an **Atom**. It is stored as string and can be used as string and number. int, long, float, double, chararray, and bytearray are the atomic values of Pig. A piece of data or a simple atomic value is known as a **field**.

Example − ‘raja’ or ‘30’

**Tuple** A record that is formed by an ordered set of fields is known as a **tuple**, the fields can be of any type. A tuple is similar to a row in a table of RDBMS. Example − (Raja, 30).

### Bag

A bag is an unordered set of tuples. In other words, a collection of tuples (non-unique) is known as a bag. Each tuple can have any number of fields (flexible schema). A bag is represented by ‘{}’. It is similar to a table in RDBMS, but unlike a table in RDBMS, it is not necessary that every tuple contain the same number of fields or that the fields in the same position (column) have the same type.

Example − {(Raja, 30), (Mohammad, 45)}

A bag can be a field in a relation; in that context, it is known as an **inner bag.**

Example − {Raja, 30, {9848022338, [raja@gmail.com,](mailto:raja@gmail.com)}}

### Map

A map (or data map) is a set of key-value pairs. The key needs to be of type chararray and should be unique. The value might be of any type. It is represented by ‘[]’

Example − [name#Raja, age#30]

### Relation

A relation is a bag of tuples. The relations in Pig Latin are unordered (there is no guarantee that tuples are processed in any particular order).

### Apache Pig - Reading Data

In general, Apache Pig works on top of Hadoop. It is an analytical tool that analyzes large datasets that exist in the Hadoop File System. To analyze data using Apache Pig, we have to initially load the data into Apache Pig. This chapter explains how to load data to Apache Pig from HDFS. (data\_for\_pig and data\_for\_pig\_new)

### Apache Pig - Storing Data

We learned how to load data into Apache Pig. You can store the loaded data in the file system using the store operator. This chapter explains how to store data in Apache Pig using the Store operator.

Syntax: *STORE Relation\_name INTO ' required\_directory\_path ' [USING function];*

### Preparing HDFS

In MapReduce mode, Pig reads (loads) data from HDFS and stores the results back in HDFS. Therefore, let us start HDFS and create the above sample data in HDFS.

## Relational Operators to run on pig shell:

LOAD, FOREACH, FILTER, JOIN, ORDER BY, DISTINCT, STORE, GROUP, COGROUP,CROSS, LIMIT, LIMIT, SPLIT

## REFERENCES

* + - “The Visual Display of Quantitative Information” by Edward R. ...
    - “Storytelling With Data: A Data Visualization Guide for Business Professionals” by Cole Nussbaumer Knaflic.
    - “Data Visualization – A Practical Introduction” by Kieran Healy.

## UNIT END EXERCISES

Create Your First Apache Pig codes and scripts.

*Note:*

*if error regarding -Xx100 then first check path*

*or*

*change %username% in hadoop-env.cmd [ remove spaces in your username] add following in yarn-site.xml in etchadoop folder*

*<property>*

*<name>yarn.scheduler.minimum-allocation-mb</name>*

*<value>2048</value>*

*</property>*

*If your IPAddress is not working in load command then use localhost*