PRACTICAL NO :05

PIG

Unit Structure :

5.0 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.

5.1 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.

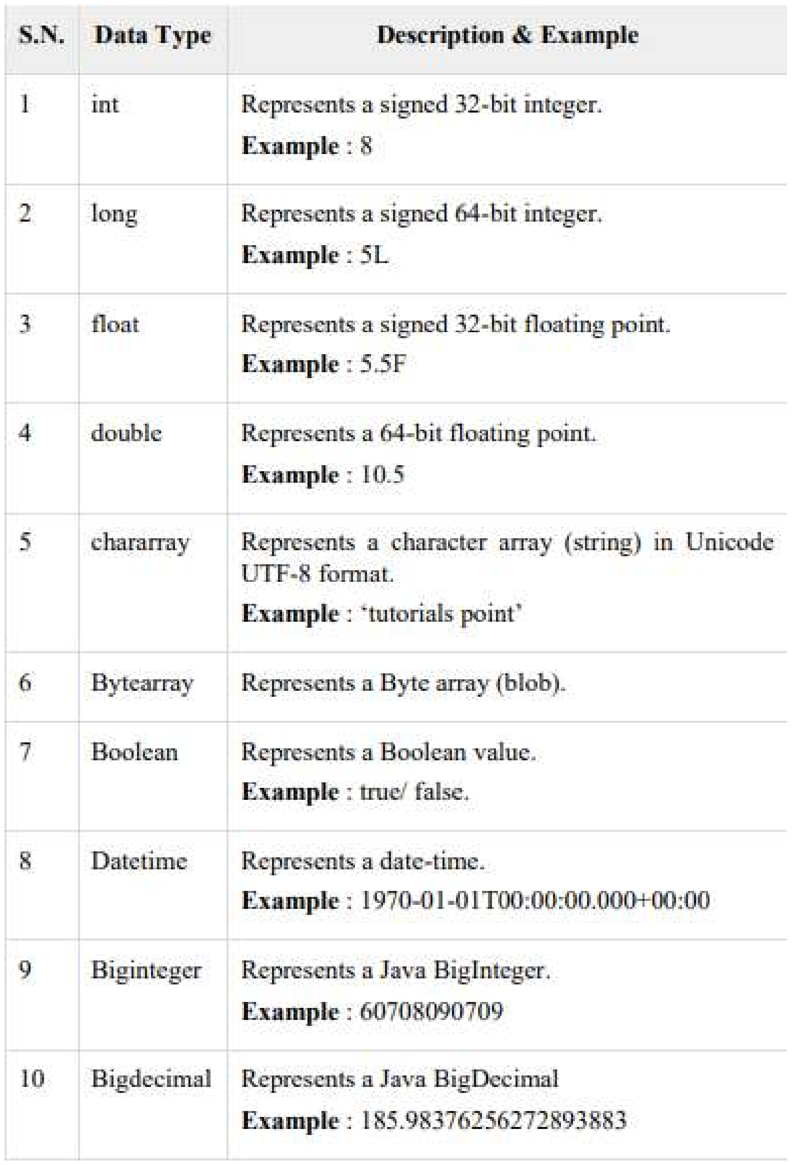
Example

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

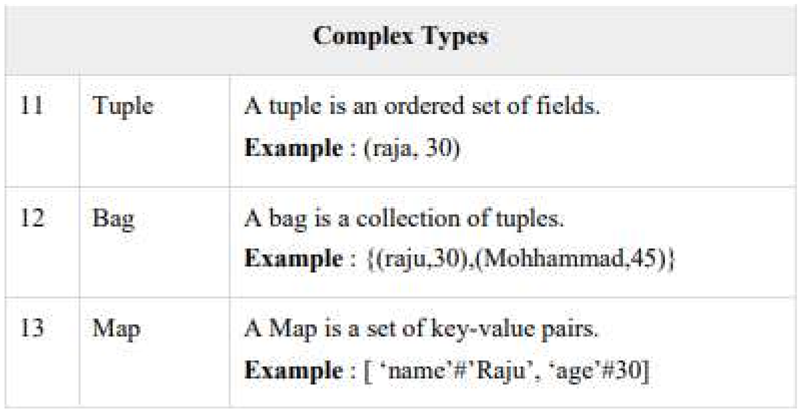
```
grunt> Student_data = LOAD 'student_data.txt' USING PigStorage(',')as

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

Pig Latin – Data types

Given below table describes the Pig Latin data types.



S.N.	Data Type	Description & Example
1	int	Represents a signed 32-bit integer. **Example** : 8
2	long	Represents a signed 64-bit integer. **Example** : 5L
3	float	Represents a signed 32-bit floating point. **Example** : 5.5F
4	double	Represents a 64-bit floating point. **Example** : 10.5
5	chararray	Represents a character array (string) in Unicode UTF-8 format. **Example** : 'tutorials point'
6	Bytearray	Represents a Byte array (blob).
7	Boolean	Represents a Boolean value. **Example** : true/ false.
8	Datetime	Represents a date-time. **Example** : 1970-01-01T00:00:00.000+00:00
9	Biginteger	Represents a Java BigInteger. **Example** : 60708090709
10	Bigdecimal	Represents a Java BigDecimal **Example** : 185.98376256272893883



Complex Types		
11	Tuple	A tuple is an ordered set of fields. **Example** : (raja, 30)
12	Bag	A bag is a collection of tuples. **Example** : {(raju,30),(Mohhammad,45)}
13	Map	A Map is a set of key-value pairs. **Example** : [‘name’#’Raju’, ‘age’#30]

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.

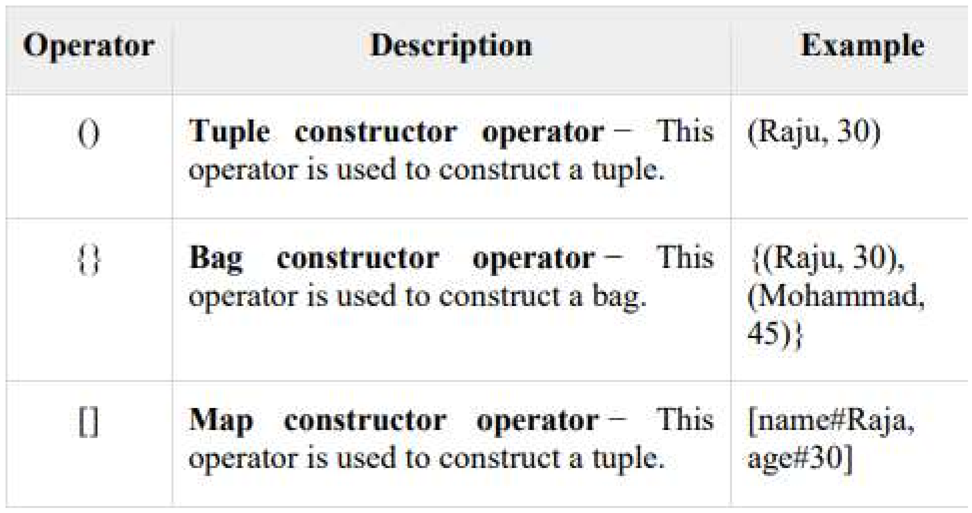
Operator	Description	Example
+	**Addition** − Adds values on either side of the operator	a + b will give 30
−	**Subtraction** − Subtracts right hand operand from left hand operand	a − b will give −10
*	**Multiplication** − Multiplies values on either side of the operator	a * b will give 200
/	**Division** − Divides left hand operand by right hand operand	b / a will give 2
%	**Modulus** − Divides left hand operand by right hand operand and returns remainder	b % a will give 0
? :	**Bincond** − Evaluates the Boolean operators. It has three operands as shown below. variable **x** = (expression) ? **value1** *if true* : **value2** *if false*.	b = (a == 1)? 20: 30; if a = 1 the value of b is 20. if a!=1 the value of b is 30.

CASE
WHEN
THEN
ELSE
END
	Case − The case operator is equivalent to nested bincond operator.	CASE f2 % 2
		WHEN 0 THEN 'even'
		WHEN 1 THEN 'odd'
		END

Operator	Description	Example
==	**Equal** − Checks if the values of two operands are equal or not; if yes, then the condition becomes true.	(a = b) is not true
!=	**Not Equal** − Checks if the values of two operands are equal or not. If the values are not equal, then condition becomes true.	(a != b) is true.
>	**Greater than** − Checks if the value of the left operand is greater than the value of the right operand. If yes, then the condition becomes true.	(a > b) is not true.
<	**Less than** − Checks if the value of the left operand is less than the value of the right operand. If yes, then the condition becomes true.	(a < b) is true.
>=	**Greater than or equal to** − Checks if the value of the left operand is greater than or equal to the value of the right operand. If yes, then the condition becomes true.	(a >= b) is not true.
<=	**Less than or equal to** − Checks if the value of the left operand is less than or equal to the value of the right operand. If yes, then the condition becomes true.	(a <= b) is true.
matches	**Pattern matching** − Checks whether the string in the left-hand side matches with the constant in the right-hand side.	f1 matches '.*tutorial.*'

Pig Latin – Type Construction Operators

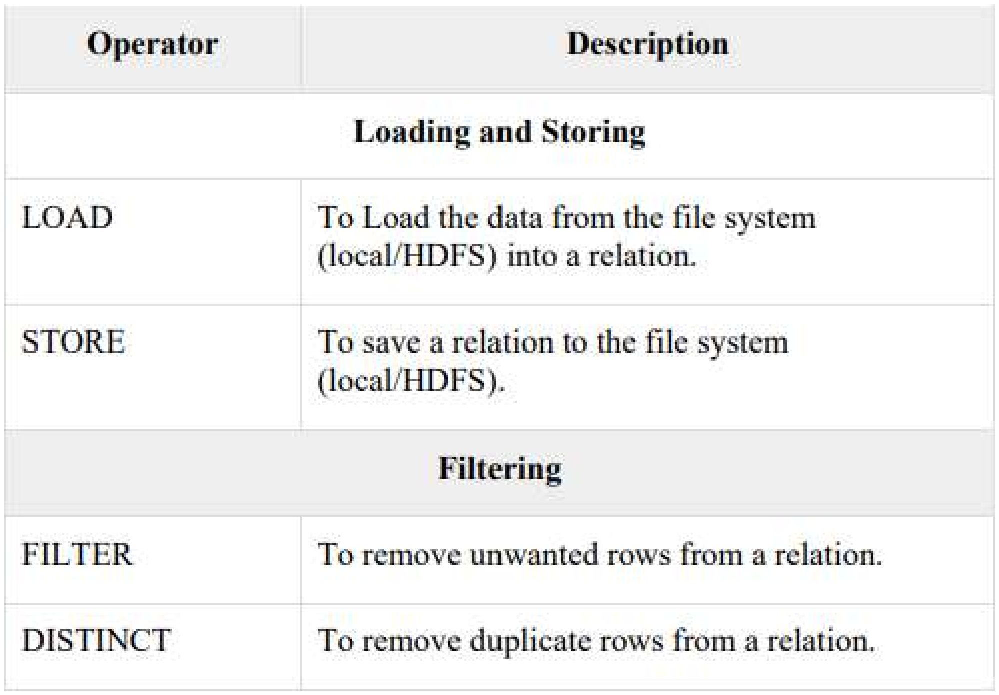
The following table describes the Type construction operators of Pig Latin.



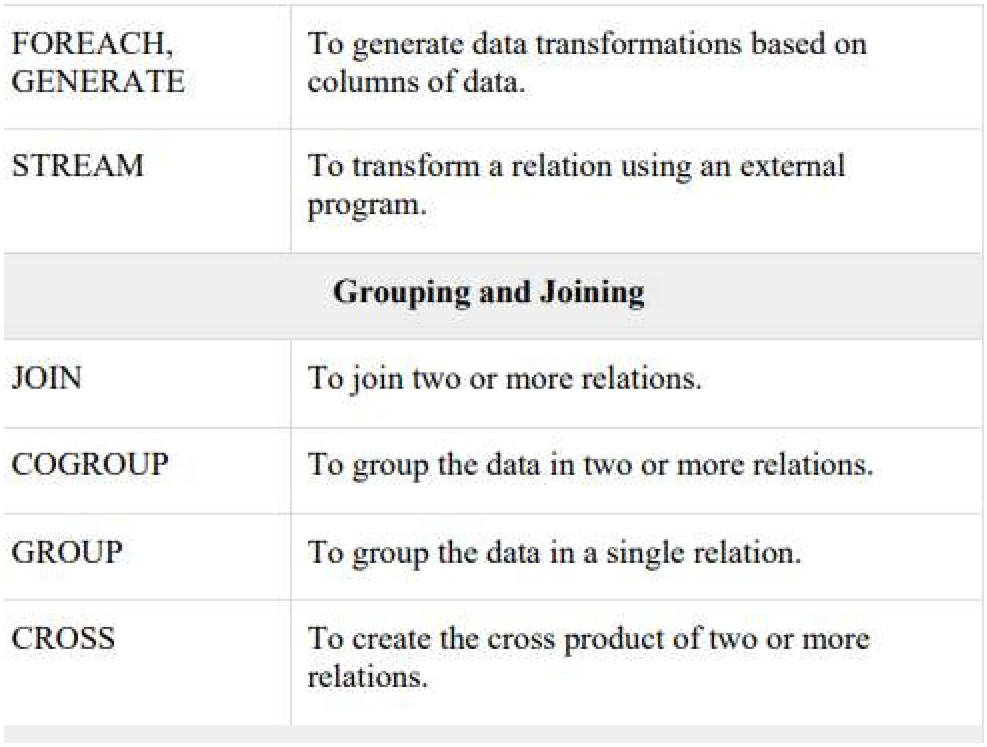
Operator	Description	Example
()	**Tuple constructor operator** − This operator is used to construct a tuple.	(Raju, 30)
{}	**Bag constructor operator** − This operator is used to construct a bag.	{(Raju, 30), (Mohammad, 45)}
[]	**Map constructor operator** − This operator is used to construct a tuple.	[name#Raja, age#30]

Pig Latin – Relational Operations

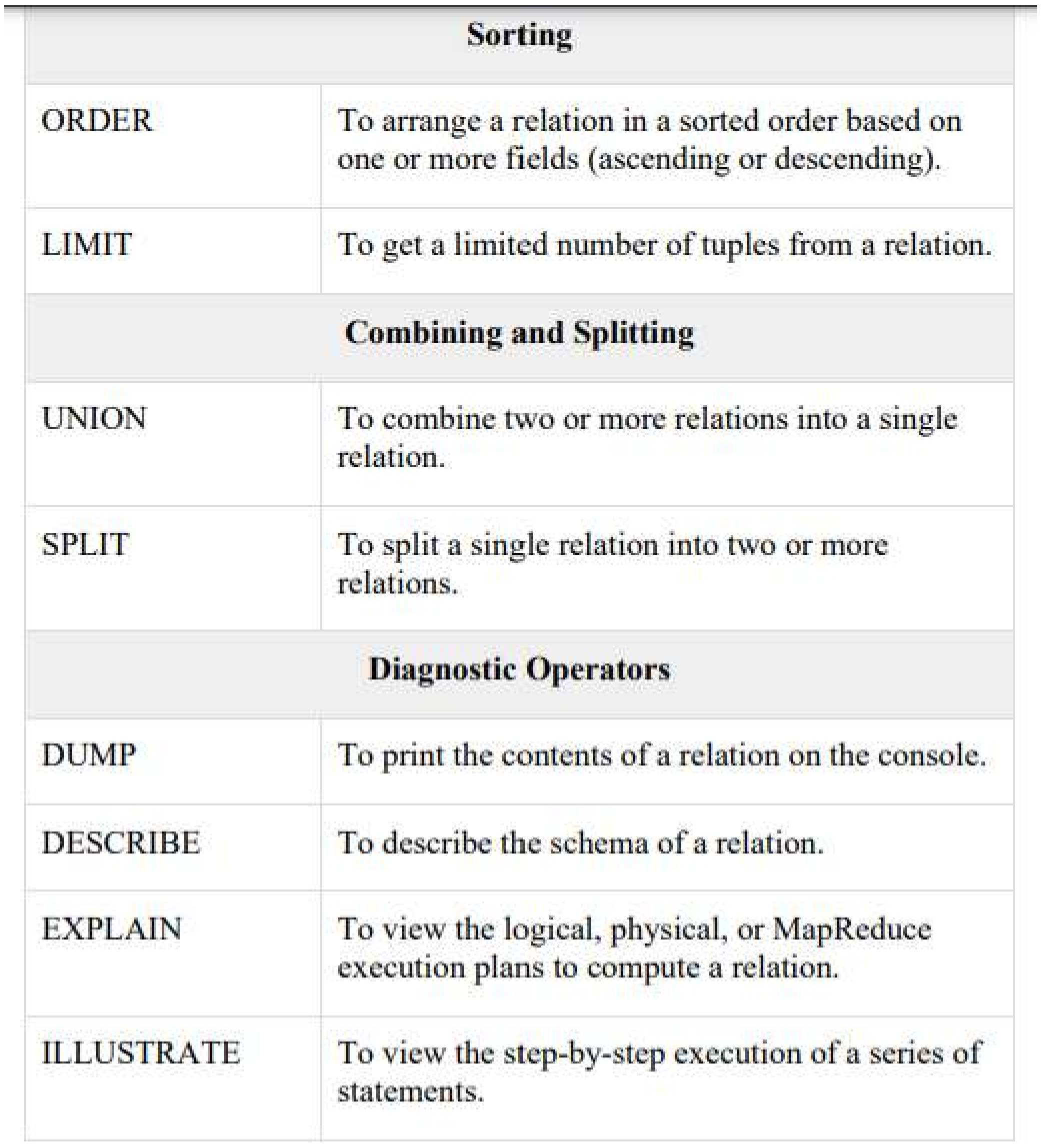
The following table describes the relational operators of Pig Latin



Operator	Description
Loading and Storing	
LOAD	To Load the data from the file system (local/HDFS) into a relation.
STORE	To save a relation to the file system (local/HDFS).
Filtering	
FILTER	To remove unwanted rows from a relation.
DISTINCT	To remove duplicate rows from a relation.



FOREACH, GENERATE	To generate data transformations based on columns of data.
STREAM	To transform a relation using an external program.
Grouping and Joining	
JOIN	To join two or more relations.
COGROUP	To group the data in two or more relations.
GROUP	To group the data in a single relation.
CROSS	To create the cross product of two or more relations.



Sorting	
ORDER	To arrange a relation in a sorted order based on one or more fields (ascending or descending).
LIMIT	To get a limited number of tuples from a relation.
Combining and Splitting	
UNION	To combine two or more relations into a single relation.
SPLIT	To split a single relation into two or more relations.
Diagnostic Operators	
DUMP	To print the contents of a relation on the console.
DESCRIBE	To describe the schema of a relation.
EXPLAIN	To view the logical, physical, or MapReduce execution plans to compute a relation.
ILLUSTRATE	To view the step-by-step execution of a series of statements.

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.

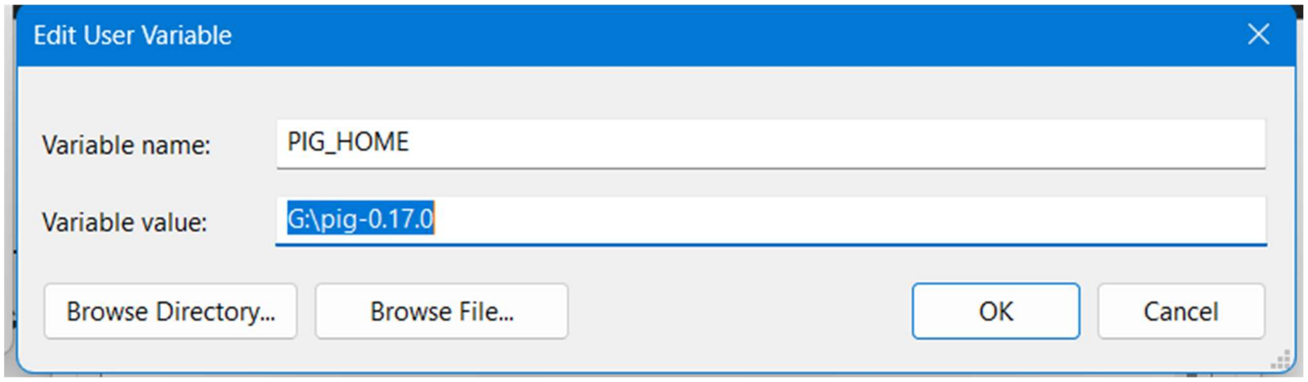
```
grunt> sh File System command parameters
```

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



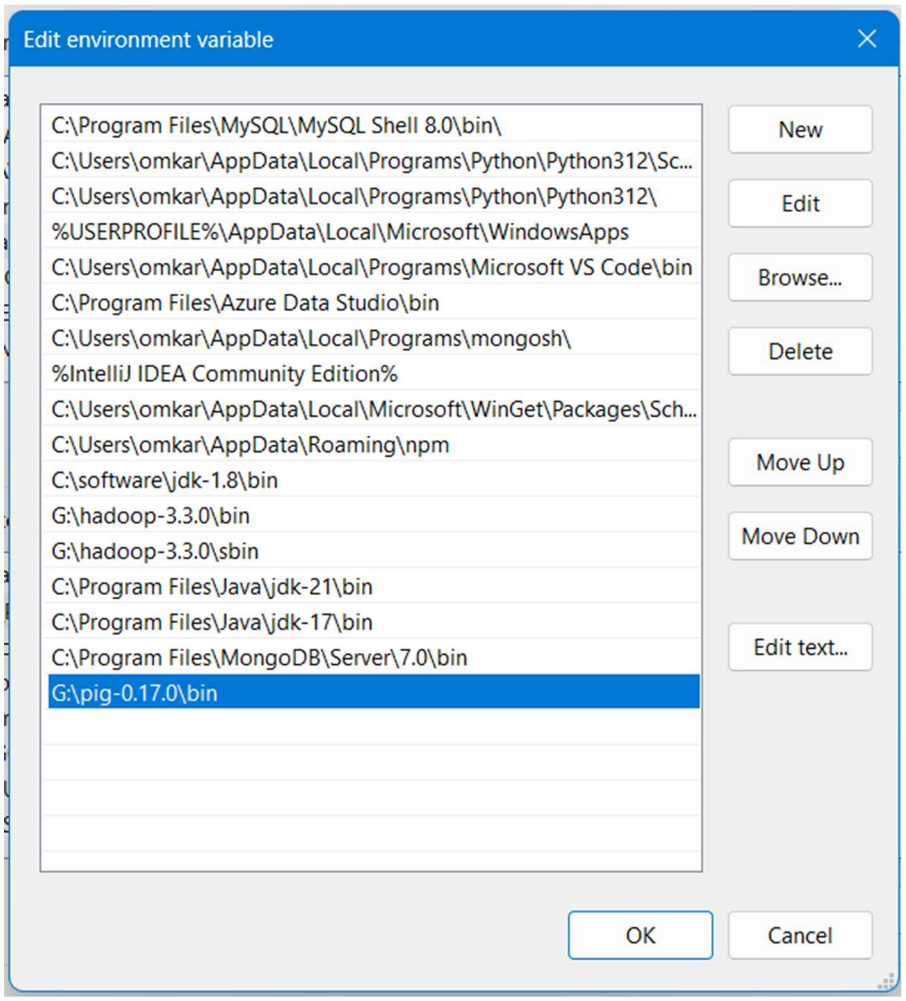
Edit User Variable

Variable name: PIG_HOME

Variable value: G:\pig-0.17.0

2. add path

G:\pig 0.17.0\bin



Edit environment variable

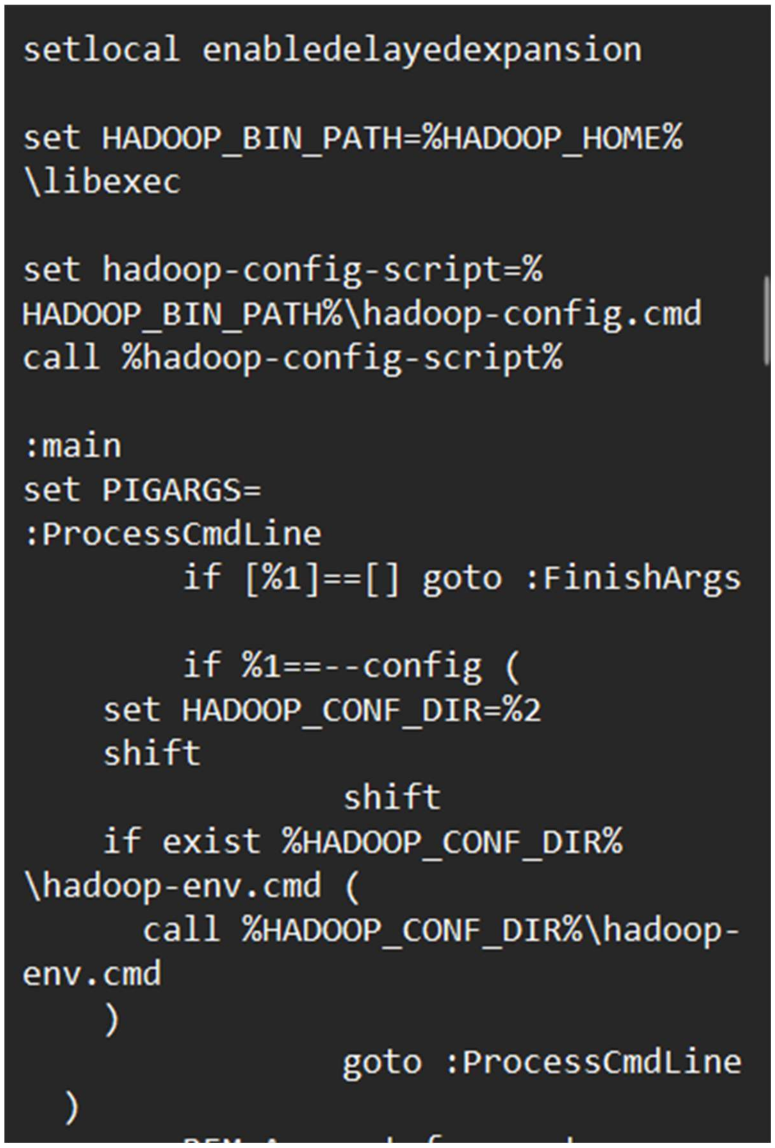
```
C:\Program Files\MySQL\MySQL Shell 8.0\bin\
C:\Users\omkar\AppData\Local\Programs\Python\Python312\Sc...
C:\Users\omkar\AppData\Local\Programs\Python\Python312\
%USERPROFILE%\AppData\Local\Microsoft\WindowsApps
C:\Users\omkar\AppData\Local\Programs\Microsoft VS Code\bin
C:\Program Files\Azure Data Studio\bin
C:\Users\omkar\AppData\Local\Programs\mongosh\
%IntelliJ IDEA Community Edition%
C:\Users\omkar\AppData\Local\Microsoft\WinGet\Packages\Sch...
C:\Users\omkar\AppData\Roaming\npm
C:\software\jdk-1.8\bin
G:\hadoop-3.3.0\bin
G:\hadoop-3.3.0\sbin
C:\Program Files\Java\jdk-21\bin
C:\Program Files\Java\jdk-17\bin
C:\Program Files\MongoDB\Server\7.0\bin
G:\pig-0.17.0\bin
```

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

set HADOOP_BIN_PATH=%HADOOP_HOME%\bin

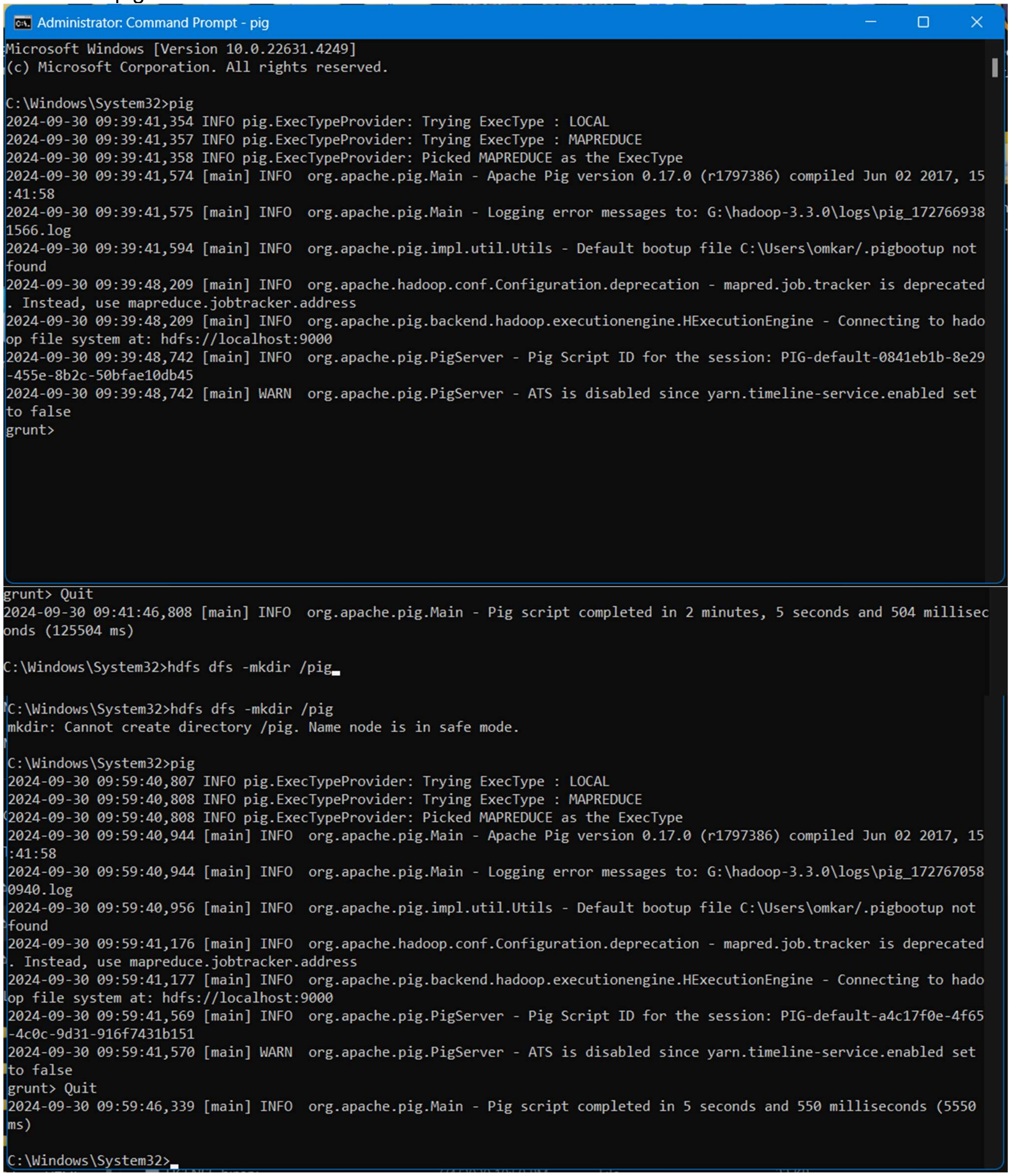
Replace this line by:

set HADOOP_BIN_PATH=%HADOOP_HOME%\libexec



```
setlocal enabledelayedexpansion

set HADOOP_BIN_PATH=%HADOOP_HOME%
\libexec

set hadoop-config-script=%
HADOOP_BIN_PATH%\hadoop-config.cmd
call %hadoop-config-script%

:main
set PIGARGS=
:ProcessCmdLine
        if [%1]==[] goto :FinishArgs

        if %1==--config (
    set HADOOP_CONF_DIR=%2
    shift
                shift
    if exist %HADOOP_CONF_DIR%
\hadoop-env.cmd (
      call %HADOOP_CONF_DIR%\hadoop-
env.cmd
    )
                goto :ProcessCmdLine
  )
```

4. check pig command on cmd



Administrator: Command Prompt - pig

```
Microsoft Windows [Version 10.0.22631.4249]
(c) Microsoft Corporation. All rights reserved.

C:\Windows\System32>pig
2024-09-30 09:39:41,354 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
2024-09-30 09:39:41,357 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
2024-09-30 09:39:41,358 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2024-09-30 09:39:41,574 [main] INFO  org.apache.pig.Main - Apache Pig version 0.17.0 (r1797386) compiled Jun 02 2017, 15
:41:58
2024-09-30 09:39:41,575 [main] INFO  org.apache.pig.Main - Logging error messages to: G:\hadoop-3.3.0\logs\pig_172766938
1566.log
2024-09-30 09:39:41,594 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file C:\Users\omkar/.pigbootup not
found
2024-09-30 09:39:48,209 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated
. Instead, use mapreduce.jobtracker.address
2024-09-30 09:39:48,209 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hado
op file system at: hdfs://localhost:9000
2024-09-30 09:39:48,742 [main] INFO  org.apache.pig.PigServer - Pig Script ID for the session: PIG-default-0841eb1b-8e29
-455e-8b2c-50bfae10db45
2024-09-30 09:39:48,742 [main] WARN  org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set
to false
grunt>
```

```
grunt> Quit
2024-09-30 09:41:46,808 [main] INFO  org.apache.pig.Main - Pig script completed in 2 minutes, 5 seconds and 504 millisec
onds (125504 ms)

C:\Windows\System32>hdfs dfs -mkdir /pig
```

```
C:\Windows\System32>hdfs dfs -mkdir /pig
mkdir: Cannot create directory /pig. Name node is in safe mode.

C:\Windows\System32>pig
2024-09-30 09:59:40,807 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
2024-09-30 09:59:40,808 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
2024-09-30 09:59:40,808 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2024-09-30 09:59:40,944 [main] INFO  org.apache.pig.Main - Apache Pig version 0.17.0 (r1797386) compiled Jun 02 2017, 15
:41:58
2024-09-30 09:59:40,944 [main] INFO  org.apache.pig.Main - Logging error messages to: G:\hadoop-3.3.0\logs\pig_172767058
0940.log
2024-09-30 09:59:40,956 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file C:\Users\omkar/.pigbootup not
found
2024-09-30 09:59:41,176 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated
. Instead, use mapreduce.jobtracker.address
2024-09-30 09:59:41,177 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hado
op file system at: hdfs://localhost:9000
2024-09-30 09:59:41,569 [main] INFO  org.apache.pig.PigServer - Pig Script ID for the session: PIG-default-a4c17f0e-4f65
-4c0c-9d31-916f7431b151
2024-09-30 09:59:41,570 [main] WARN  org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set
to false
grunt> Quit
2024-09-30 09:59:46,339 [main] INFO  org.apache.pig.Main - Pig script completed in 5 seconds and 550 milliseconds (5550
ms)

C:\Windows\System32>
```

5. 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.

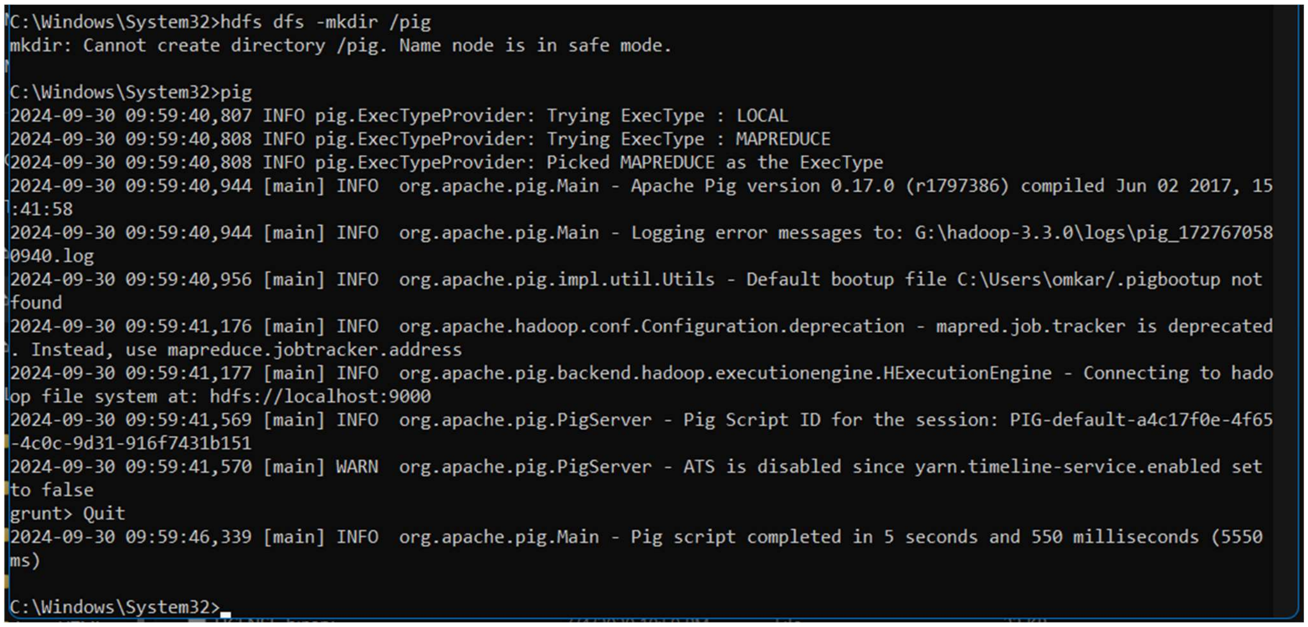
pig -x local

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



```
C:\Windows\System32>hdfs dfs -mkdir /pig
mkdir: Cannot create directory /pig. Name node is in safe mode.

C:\Windows\System32>pig
2024-09-30 09:59:40,807 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
2024-09-30 09:59:40,808 INFO pig.ExecTypeProvider: Trying ExecType : MAPREDUCE
2024-09-30 09:59:40,808 INFO pig.ExecTypeProvider: Picked MAPREDUCE as the ExecType
2024-09-30 09:59:40,944 [main] INFO  org.apache.pig.Main - Apache Pig version 0.17.0 (r1797386) compiled Jun 02 2017, 15
:41:58
2024-09-30 09:59:40,944 [main] INFO  org.apache.pig.Main - Logging error messages to: G:\hadoop-3.3.0\logs\pig_172767058
0940.log
2024-09-30 09:59:40,956 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file C:\Users\omkar/.pigbootup not
found
2024-09-30 09:59:41,176 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated
. Instead, use mapreduce.jobtracker.address
2024-09-30 09:59:41,177 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hado
op file system at: hdfs://localhost:9000
2024-09-30 09:59:41,569 [main] INFO  org.apache.pig.PigServer - Pig Script ID for the session: PIG-default-a4c17f0e-4f65
-4c0c-9d31-916f7431b151
2024-09-30 09:59:41,570 [main] WARN  org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set
to false
grunt> Quit
2024-09-30 09:59:46,339 [main] INFO  org.apache.pig.Main - Pig script completed in 5 seconds and 550 milliseconds (5550
ms)

C:\Windows\System32>
```

1. Create a file data_for_pig.txt with following data:

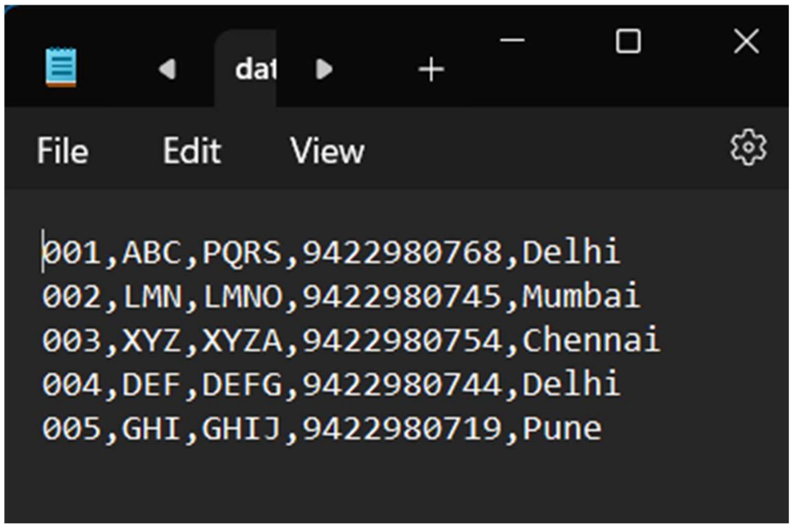
001,ABC,PQRS,9422980768,Delhi

002,LMN,LMNO,9422980745,Mumbai

003,XYZ,XYZA,9422980754,Chennai

004,DEF,DEFG,9422980744,Delhi

005,GHI,GHIJ,9422980719,Pune



```
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:

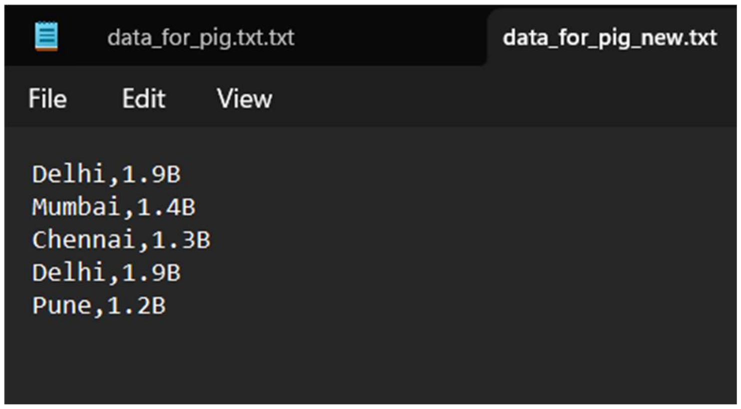
Delhi,1.9B

Mumbai,1.4B

Chennai,1.3B

Delhi,1.9B

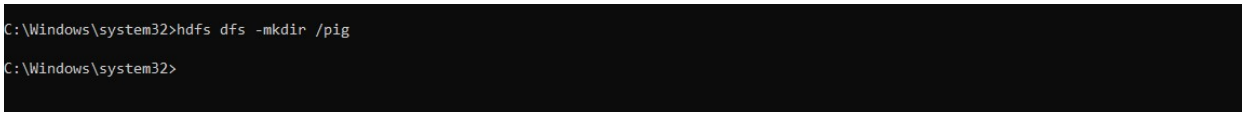
Pune,1.2B



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

2. hdfs dfs -mkdir /pig

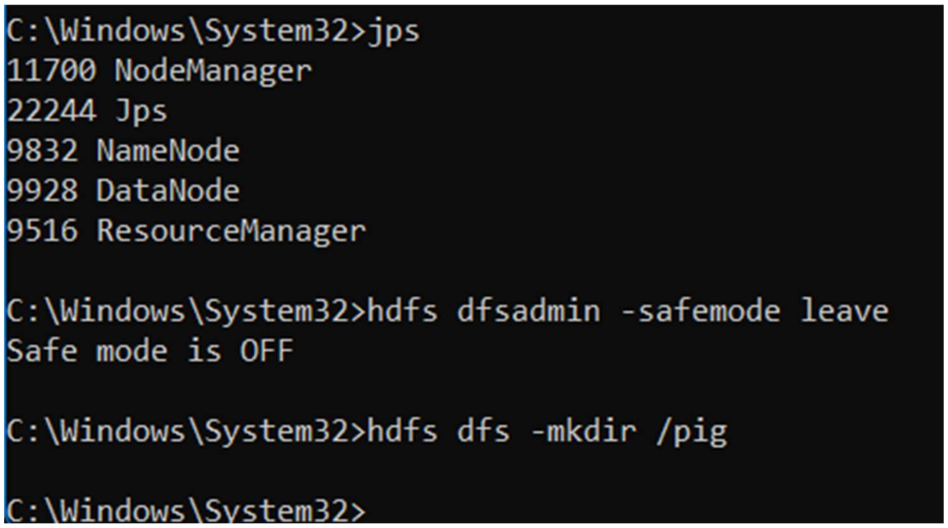
[Note: **quit** grunt shell first and then run above command]



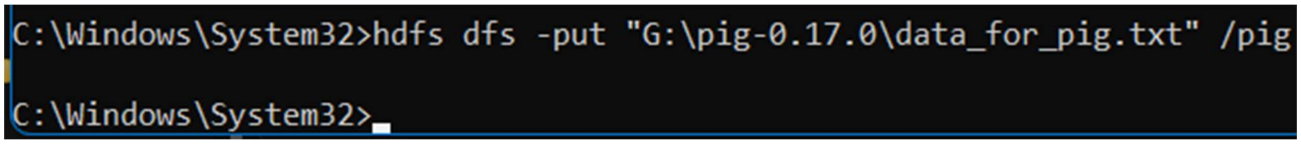
```
C:\Windows\system32>hdfs dfs -mkdir /pig

C:\Windows\system32>
```

3. Upload your file on HDFS using following command:

hdfs dfs -put "G:\pig-0.17.0\data_for_pig.txt" /pig



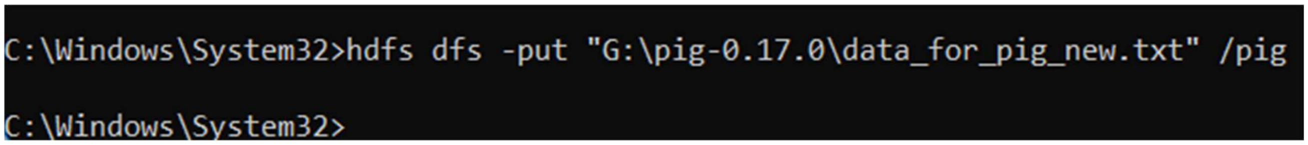
```
C:\Windows\System32>jps
11700 NodeManager
22244 Jps
9832 NameNode
9928 DataNode
9516 ResourceManager

C:\Windows\System32>hdfs dfsadmin -safemode leave
Safe mode is OFF

C:\Windows\System32>hdfs dfs -mkdir /pig

C:\Windows\System32>
```



```
C:\Windows\System32>hdfs dfs -put "G:\pig-0.17.0\data_for_pig.txt" /pig

C:\Windows\System32>
```

4. Upload your another file on HDFS using following command:

hdfs dfs -put "G:\pig-0.17.0\data_for_pig_new.txt" /pig



```
C:\Windows\System32>hdfs dfs -put "G:\pig-0.17.0\data_for_pig_new.txt" /pig

C:\Windows\System32>
```

4. pig -x mapreduce

copy above txt file into pig/bin

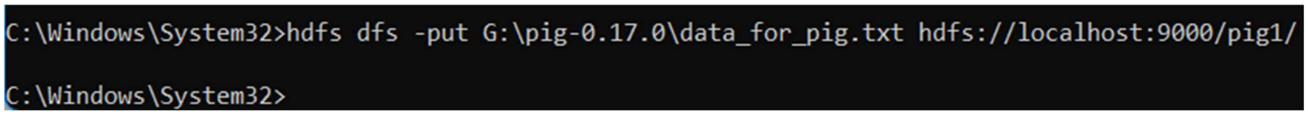
or

put using following command

hdfs dfs -put C:\Users\student\Downloads\BigData\data_for_pig.txt hdfs://localhost:9000/pig1/

cat data of file

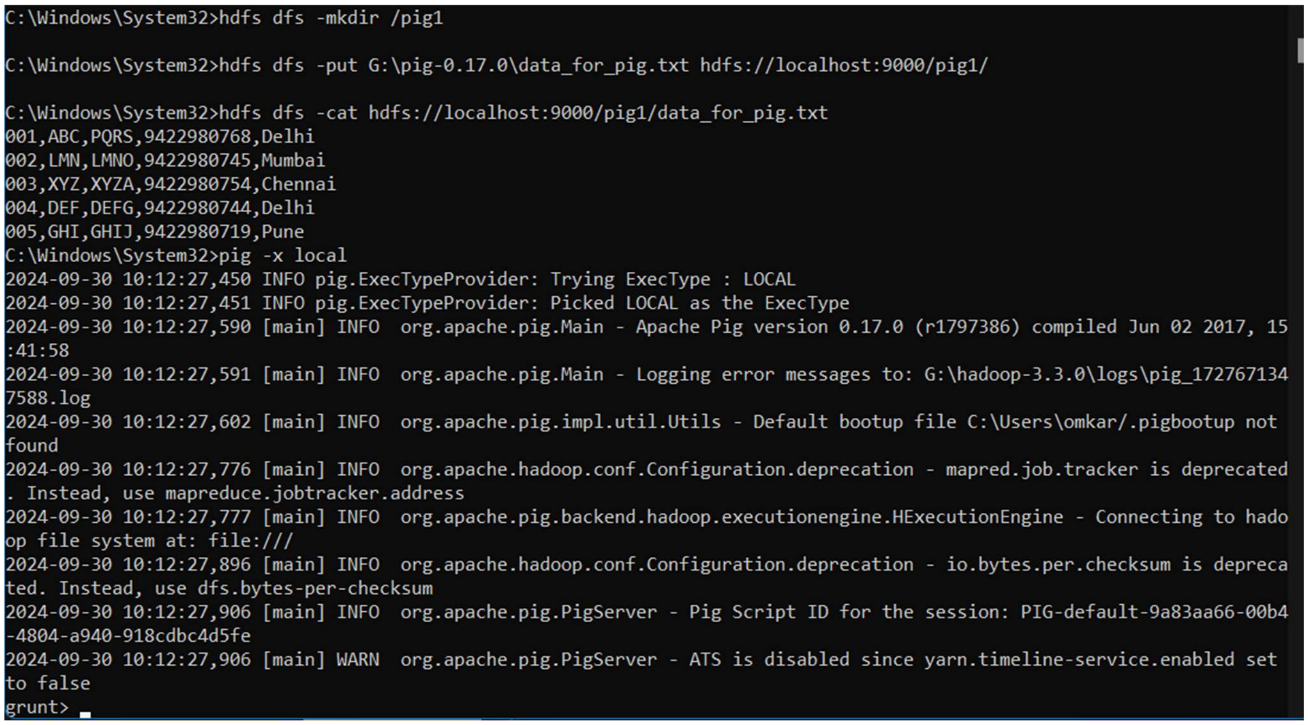
hdfs dfs -cat hdfs://localhost:9000/pig1/data_for_pig.txt



```
C:\Windows\System32>hdfs dfs -put G:\pig-0.17.0\data_for_pig.txt hdfs://localhost:9000/pig1/

C:\Windows\System32>
```

hdfs dfs -cat hdfs://172.16.4.5:9000/pig1/data_for_pig.txt

pig -x mapreduce or pig -x local



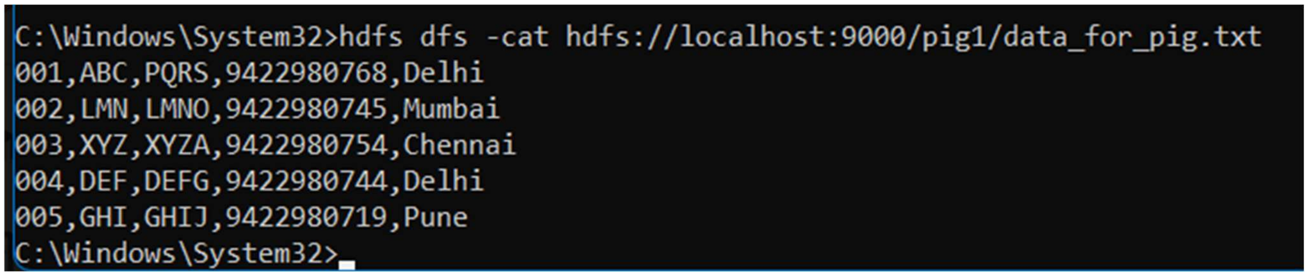
```
C:\Windows\System32>hdfs dfs -mkdir /pig1

C:\Windows\System32>hdfs dfs -put G:\pig-0.17.0\data_for_pig.txt hdfs://localhost:9000/pig1/

C:\Windows\System32>hdfs dfs -cat hdfs://localhost:9000/pig1/data_for_pig.txt
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
C:\Windows\System32>pig -x local
2024-09-30 10:12:27,450 INFO pig.ExecTypeProvider: Trying ExecType : LOCAL
2024-09-30 10:12:27,451 INFO pig.ExecTypeProvider: Picked LOCAL as the ExecType
2024-09-30 10:12:27,590 [main] INFO  org.apache.pig.Main - Apache Pig version 0.17.0 (r1797386) compiled Jun 02 2017, 15
:41:58
2024-09-30 10:12:27,591 [main] INFO  org.apache.pig.Main - Logging error messages to: G:\hadoop-3.3.0\logs\pig_172767134
7588.log
2024-09-30 10:12:27,602 [main] INFO  org.apache.pig.impl.util.Utils - Default bootup file C:\Users\omkar/.pigbootup not
found
2024-09-30 10:12:27,776 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.job.tracker is deprecated
. Instead, use mapreduce.jobtracker.address
2024-09-30 10:12:27,777 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hado
op file system at: file:///
2024-09-30 10:12:27,896 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is depreca
ted. Instead, use dfs.bytes-per-checksum
2024-09-30 10:12:27,906 [main] INFO  org.apache.pig.PigServer - Pig Script ID for the session: PIG-default-9a83aa66-00b4
-4804-a940-918cdbc4d5fe
2024-09-30 10:12:27,906 [main] WARN  org.apache.pig.PigServer - ATS is disabled since yarn.timeline-service.enabled set
to false
grunt>
```

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);

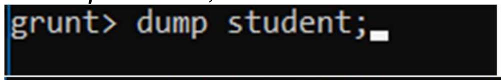
Or

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

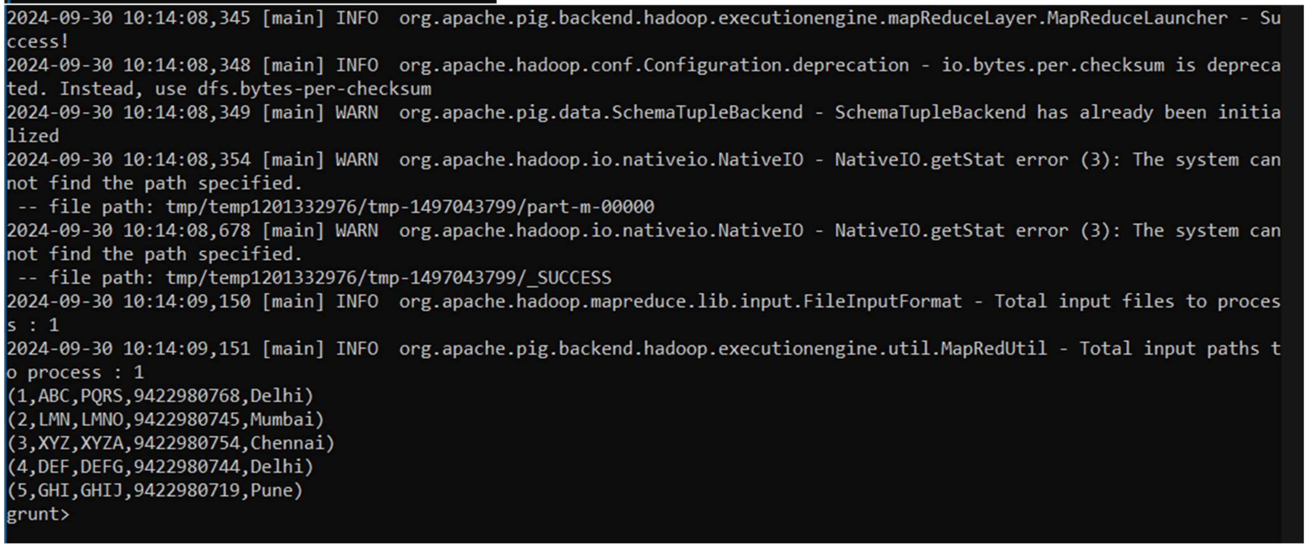


```
C:\Windows\System32>hdfs dfs -cat hdfs://localhost:9000/pig1/data_for_pig.txt
001,ABC,PQRS,9422980768,Delhi
002,LMN,LMNO,9422980745,Mumbai
003,XYZ,XYZA,9422980754,Chennai
004,DEF,DEFG,9422980744,Delhi
005,GHI,GHIJ,9422980719,Pune
C:\Windows\System32>
```

dump student;



```
grunt> dump student;
```



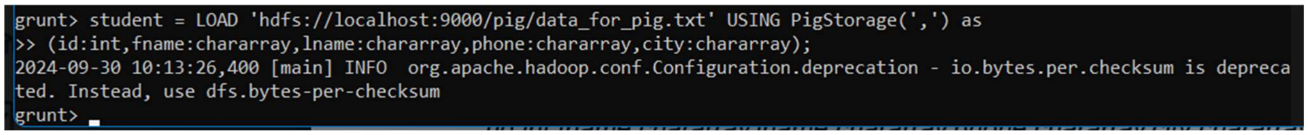
```
2024-09-30 10:14:08,345 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Su
ccess!
2024-09-30 10:14:08,348 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is depreca
ted. Instead, use dfs.bytes-per-checksum
2024-09-30 10:14:08,349 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initia
lized
2024-09-30 10:14:08,354 [main] WARN  org.apache.hadoop.io.nativeio.NativeIO - NativeIO.getStat error (3): The system can
not find the path specified.
 -- file path: tmp/temp1201332976/tmp-1497043799/part-m-00000
2024-09-30 10:14:08,678 [main] WARN  org.apache.hadoop.io.nativeio.NativeIO - NativeIO.getStat error (3): The system can
not find the path specified.
 -- file path: tmp/temp1201332976/tmp-1497043799/_SUCCESS
2024-09-30 10:14:09,150 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input files to proces
s : 1
2024-09-30 10:14:09,151 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths t
o process : 1
(1,ABC,PQRS,9422980768,Delhi)
(2,LMN,LMNO,9422980745,Mumbai)
(3,XYZ,XYZA,9422980754,Chennai)
(4,DEF,DEFG,9422980744,Delhi)
(5,GHI,GHIJ,9422980719,Pune)
grunt>
```

Create another variable city as follows:

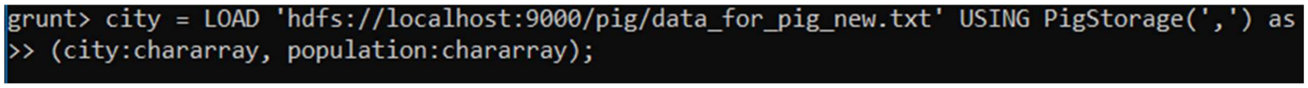
city = LOAD 'hdfs://172.16.4.4:9000/pig/data_for_pig_new.txt' USING PigStorage(',') as (city:chararray, population:chararray);

Or

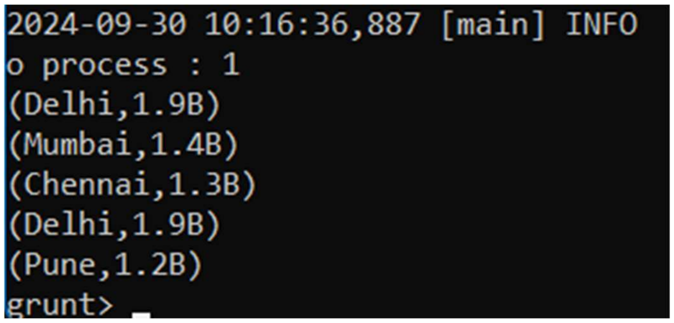
city = LOAD 'hdfs://localhost:9000/pig/data_for_pig_new.txt' USING PigStorage(',') as (city:chararray, population:chararray);



```
grunt> student = LOAD 'hdfs://localhost:9000/pig/data_for_pig.txt' USING PigStorage(',') as
>> (id:int,fname:chararray,lname:chararray,phone:chararray,city:chararray);
2024-09-30 10:13:26,400 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is depreca
ted. Instead, use dfs.bytes-per-checksum
grunt>
```



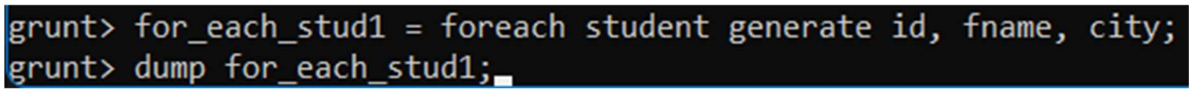
```
grunt> city = LOAD 'hdfs://localhost:9000/pig/data_for_pig_new.txt' USING PigStorage(',') as
>> (city:chararray, population:chararray);
```



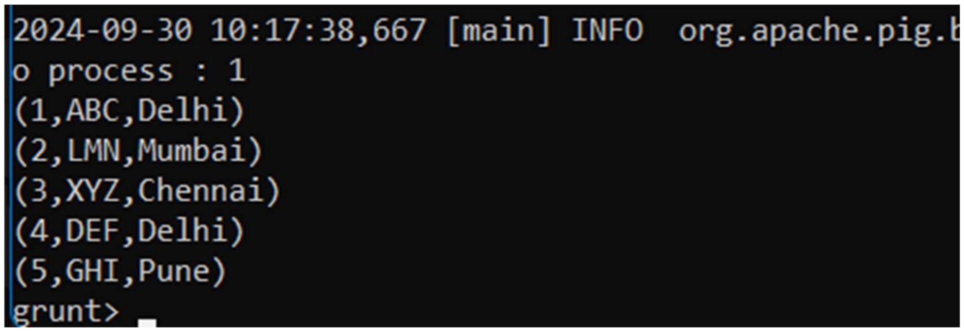
```
2024-09-30 10:16:36,887 [main] INFO
o process : 1
(Delhi,1.9B)
(Mumbai,1.4B)
(Chennai,1.3B)
(Delhi,1.9B)
(Pune,1.2B)
grunt>
```

for_each_stud1 = foreach student generate id, fname, city;

dump for_each_stud1;



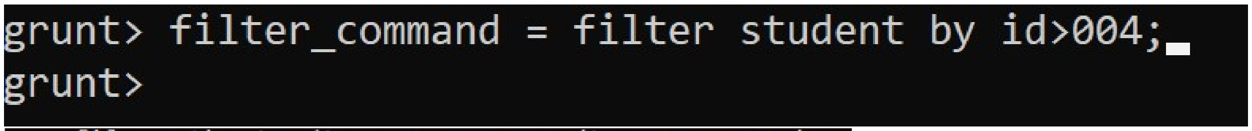
```
grunt> for_each_stud1 = foreach student generate id, fname, city;
grunt> dump for_each_stud1;
```



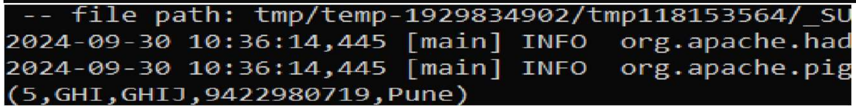
```
2024-09-30 10:17:38,667 [main] INFO  org.apache.pig.b
o process : 1
(1,ABC,Delhi)
(2,LMN,Mumbai)
(3,XYZ,Chennai)
(4,DEF,Delhi)
(5,GHI,Pune)
grunt>
```

filter_command = filter student by id>004;

dump filter_command;



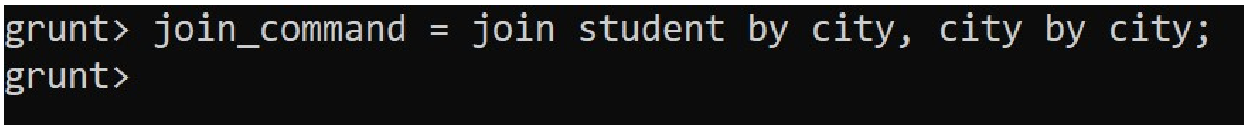
```
grunt> filter_command = filter student by id>004;
grunt>
```



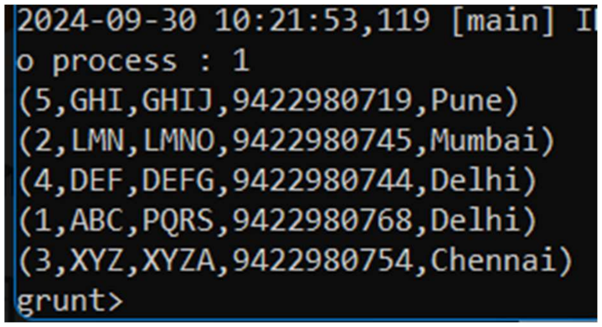
```
 -- file path: tmp/temp-1929834902/tmp118153564/_SU
2024-09-30 10:36:14,445 [main] INFO  org.apache.had
2024-09-30 10:36:14,445 [main] INFO  org.apache.pig
(5,GHI,GHIJ,9422980719,Pune)
```

join_command = join student by city, city by city;

dump join_command;



```
grunt> join_command = join student by city, city by city;
grunt>
```

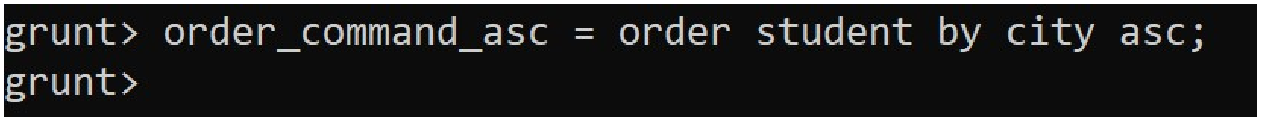


```
2024-09-30 10:21:53,119 [main] I
o process : 1
(5,GHI,GHIJ,9422980719,Pune)
(2,LMN,LMNO,9422980745,Mumbai)
(4,DEF,DEFG,9422980744,Delhi)
(1,ABC,PQRS,9422980768,Delhi)
(3,XYZ,XYZA,9422980754,Chennai)
grunt>
```

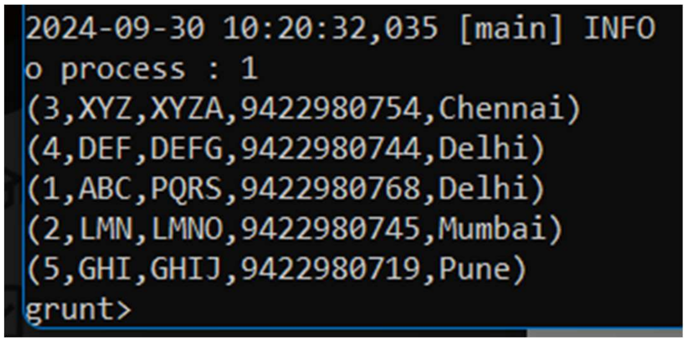
order by

order_command_asc = order student by city asc;

dump order_command_asc;



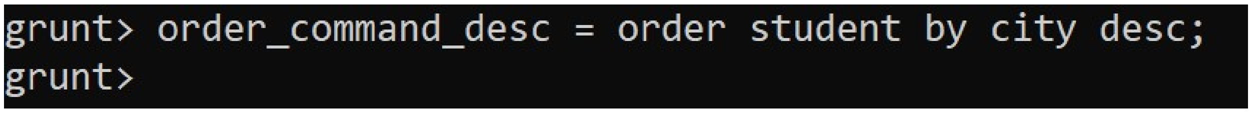
```
grunt> order_command_asc = order student by city asc;
grunt>
```



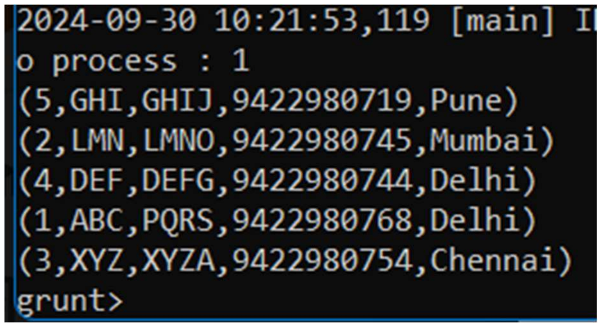
```
2024-09-30 10:20:32,035 [main] INFO
o process : 1
(3,XYZ,XYZA,9422980754,Chennai)
(4,DEF,DEFG,9422980744,Delhi)
(1,ABC,PQRS,9422980768,Delhi)
(2,LMN,LMNO,9422980745,Mumbai)
(5,GHI,GHIJ,9422980719,Pune)
grunt>
```

order_command_desc = order student by city desc;

dump order_command_desc;



```
grunt> order_command_desc = order student by city desc;
grunt>
```

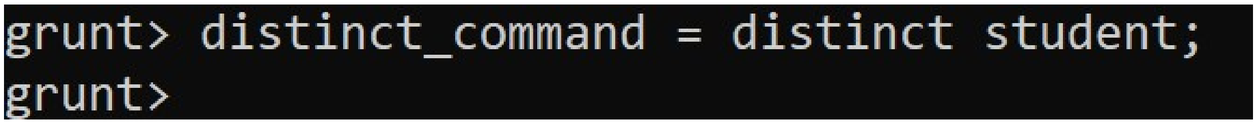


```
2024-09-30 10:21:53,119 [main] I
o process : 1
(5,GHI,GHIJ,9422980719,Pune)
(2,LMN,LMNO,9422980745,Mumbai)
(4,DEF,DEFG,9422980744,Delhi)
(1,ABC,PQRS,9422980768,Delhi)
(3,XYZ,XYZA,9422980754,Chennai)
grunt>
```

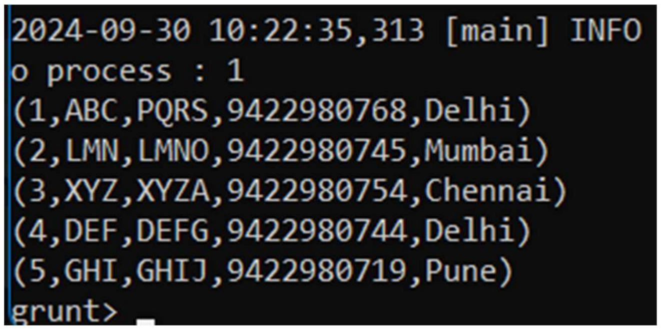
Distinct [to get o/p, you require to should have duplicate data in your file]

distinct_command = distinct student;

dump distinct_command;



```
grunt> distinct_command = distinct student;
grunt>
```



```
2024-09-30 10:22:35,313 [main] INFO
o process : 1
(1,ABC,PQRS,9422980768,Delhi)
(2,LMN,LMNO,9422980745,Mumbai)
(3,XYZ,XYZA,9422980754,Chennai)
(4,DEF,DEFG,9422980744,Delhi)
(5,GHI,GHIJ,9422980719,Pune)
grunt>
```

Store

store order_command_desc into '/desc';

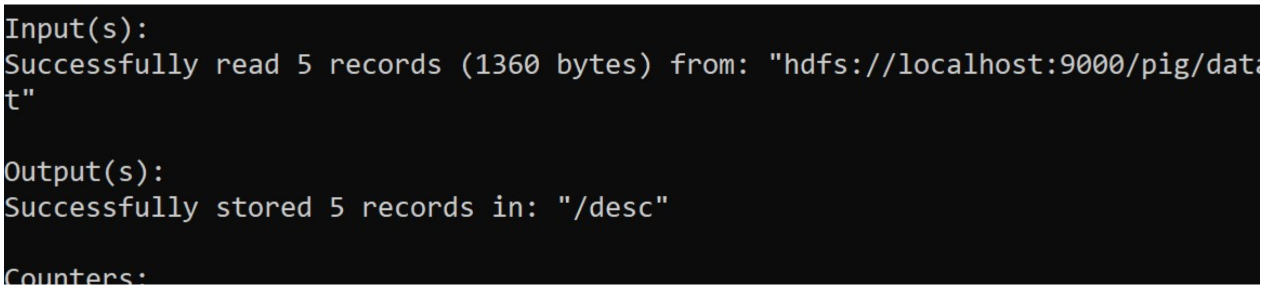
output:

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"

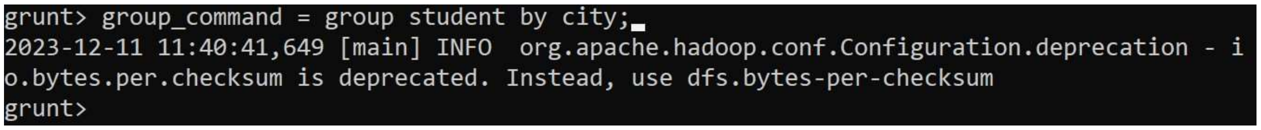


```
Input(s):
Successfully read 5 records (1360 bytes) from: "hdfs://localhost:9000/pig/dat
t"

Output(s):
Successfully stored 5 records in: "/desc"

Counters:
```

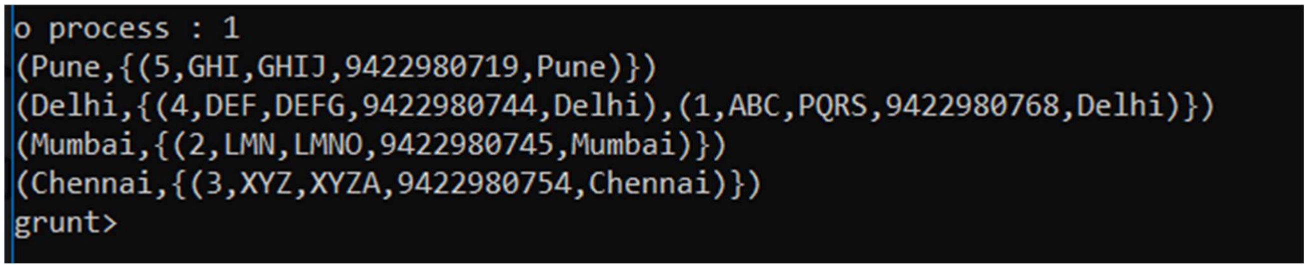
Group command (Duplication is city is required)

group_command = group student by city;

dump group_command;



```
grunt> group_command = group student by city;
2023-12-11 11:40:41,649 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - i
o.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
grunt>
```

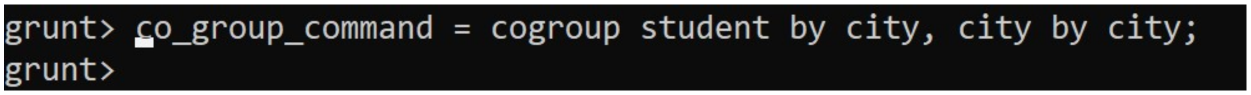


```
o process : 1
(Pune,{(5,GHI,GHIJ,9422980719,Pune)})
(Delhi,{(4,DEF,DEFG,9422980744,Delhi),(1,ABC,PQRS,9422980768,Delhi)})
(Mumbai,{(2,LMN,LMNO,9422980745,Mumbai)})
(Chennai,{(3,XYZ,XYZA,9422980754,Chennai)})
grunt>
```

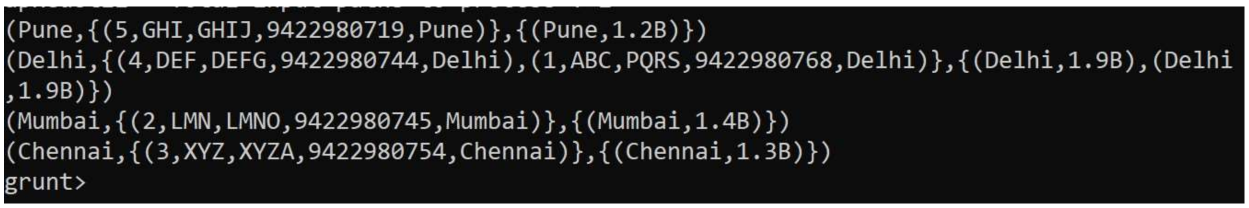
Cogroup

co_group_command = cogroup student by city, city by city;

dump co_group_command;



```
grunt> co_group_command = cogroup student by city, city by city;
grunt>
```

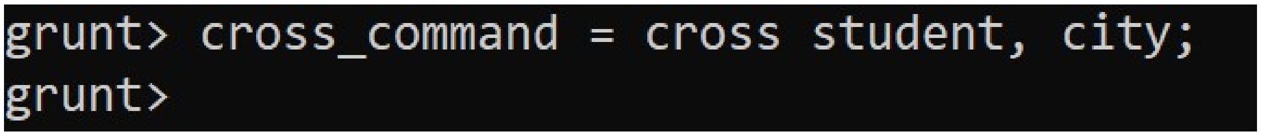


```
(Pune,{(5,GHI,GHIJ,9422980719,Pune)},{(Pune,1.2B)})
(Delhi,{(4,DEF,DEFG,9422980744,Delhi),(1,ABC,PQRS,9422980768,Delhi)},{(Delhi,1.9B),(Delhi
,1.9B)})
(Mumbai,{(2,LMN,LMNO,9422980745,Mumbai)},{(Mumbai,1.4B)})
(Chennai,{(3,XYZ,XYZA,9422980754,Chennai)},{(Chennai,1.3B)})
grunt>
```

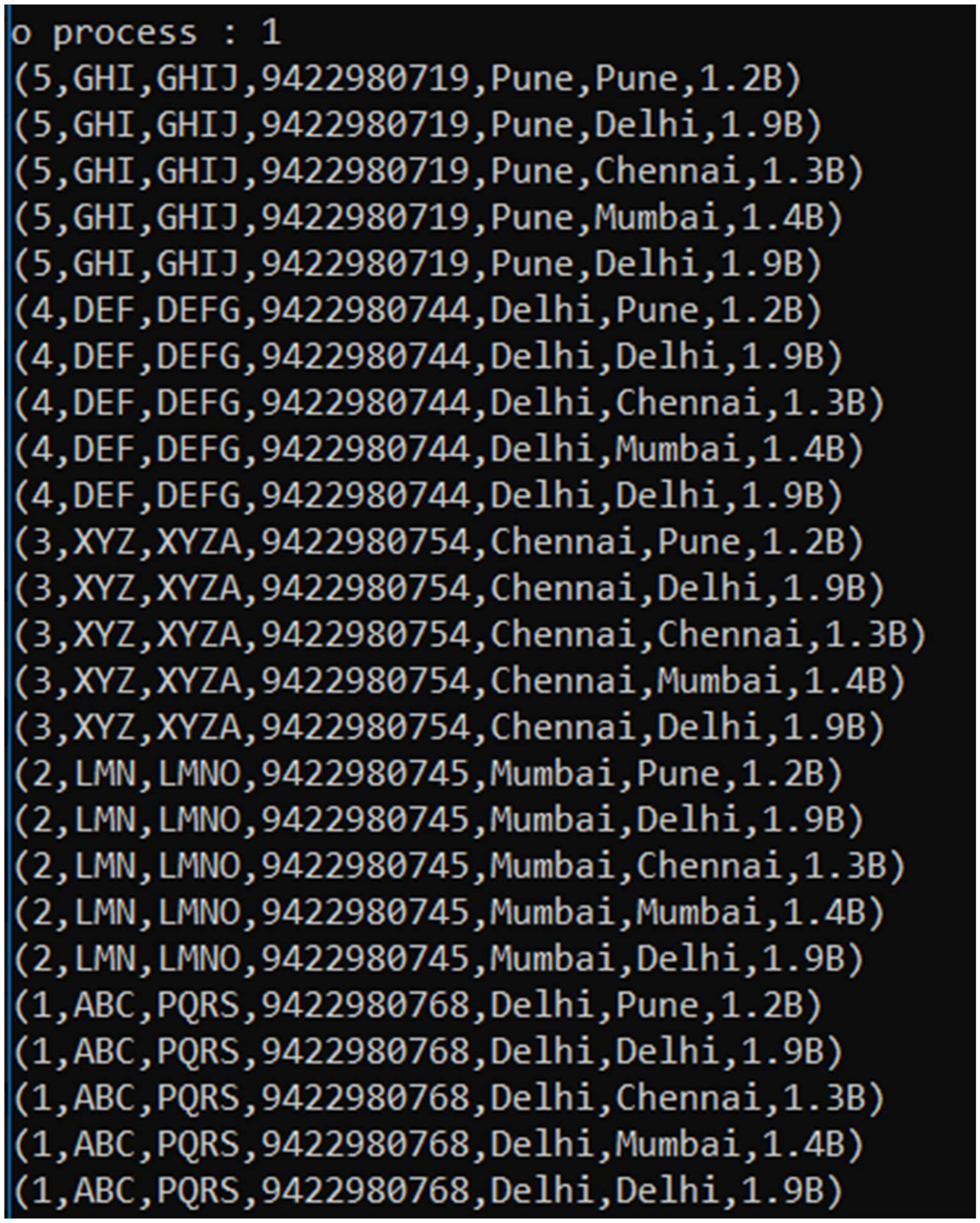
Cross:

cross_command = cross student, city;

dump cross_command;



```
grunt> cross_command = cross student, city;
grunt>
```

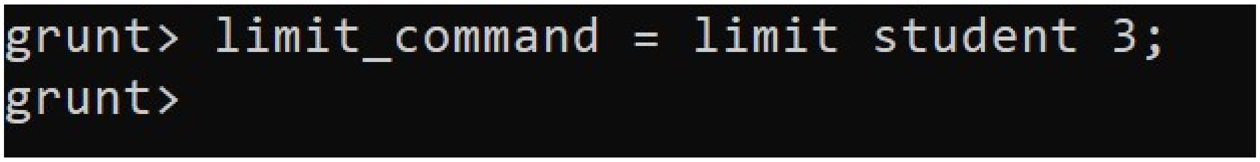


```
o process : 1
(5,GHI,GHIJ,9422980719,Pune,Pune,1.2B)
(5,GHI,GHIJ,9422980719,Pune,Delhi,1.9B)
(5,GHI,GHIJ,9422980719,Pune,Chennai,1.3B)
(5,GHI,GHIJ,9422980719,Pune,Mumbai,1.4B)
(5,GHI,GHIJ,9422980719,Pune,Delhi,1.9B)
(4,DEF,DEFG,9422980744,Delhi,Pune,1.2B)
(4,DEF,DEFG,9422980744,Delhi,Delhi,1.9B)
(4,DEF,DEFG,9422980744,Delhi,Chennai,1.3B)
(4,DEF,DEFG,9422980744,Delhi,Mumbai,1.4B)
(4,DEF,DEFG,9422980744,Delhi,Delhi,1.9B)
(3,XYZ,XYZA,9422980754,Chennai,Pune,1.2B)
(3,XYZ,XYZA,9422980754,Chennai,Delhi,1.9B)
(3,XYZ,XYZA,9422980754,Chennai,Chennai,1.3B)
(3,XYZ,XYZA,9422980754,Chennai,Mumbai,1.4B)
(3,XYZ,XYZA,9422980754,Chennai,Delhi,1.9B)
(2,LMN,LMNO,9422980745,Mumbai,Pune,1.2B)
(2,LMN,LMNO,9422980745,Mumbai,Delhi,1.9B)
(2,LMN,LMNO,9422980745,Mumbai,Chennai,1.3B)
(2,LMN,LMNO,9422980745,Mumbai,Mumbai,1.4B)
(2,LMN,LMNO,9422980745,Mumbai,Delhi,1.9B)
(1,ABC,PQRS,9422980768,Delhi,Pune,1.2B)
(1,ABC,PQRS,9422980768,Delhi,Delhi,1.9B)
(1,ABC,PQRS,9422980768,Delhi,Chennai,1.3B)
(1,ABC,PQRS,9422980768,Delhi,Mumbai,1.4B)
(1,ABC,PQRS,9422980768,Delhi,Delhi,1.9B)
```

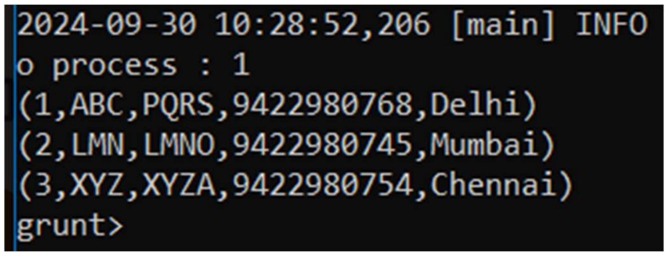
Limit:

limit_command = limit student 3;

dump limit_command;



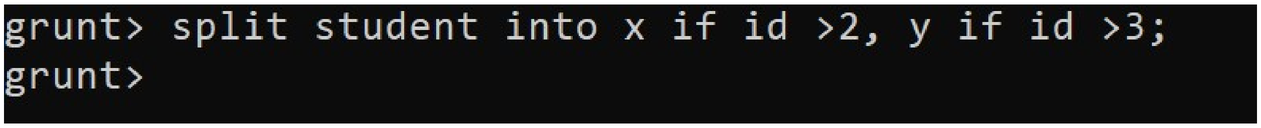
```
grunt> limit_command = limit student 3;
grunt>
```



```
2024-09-30 10:28:52,206 [main] INFO
o process : 1
(1,ABC,PQRS,9422980768,Delhi)
(2,LMN,LMNO,9422980745,Mumbai)
(3,XYZ,XYZA,9422980754,Chennai)
grunt>
```

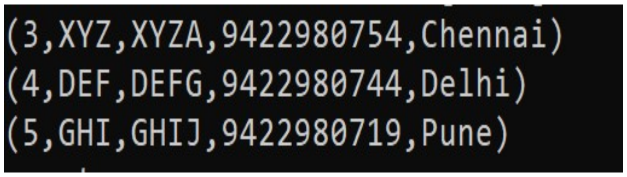
Split:

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



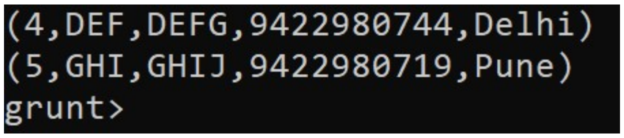
```
grunt> split student into x if id >2, y if id >3;
grunt>
```

dump x;



```
(3,XYZ,XYZA,9422980754,Chennai)
(4,DEF,DEFG,9422980744,Delhi)
(5,GHI,GHIJ,9422980719,Pune)
```

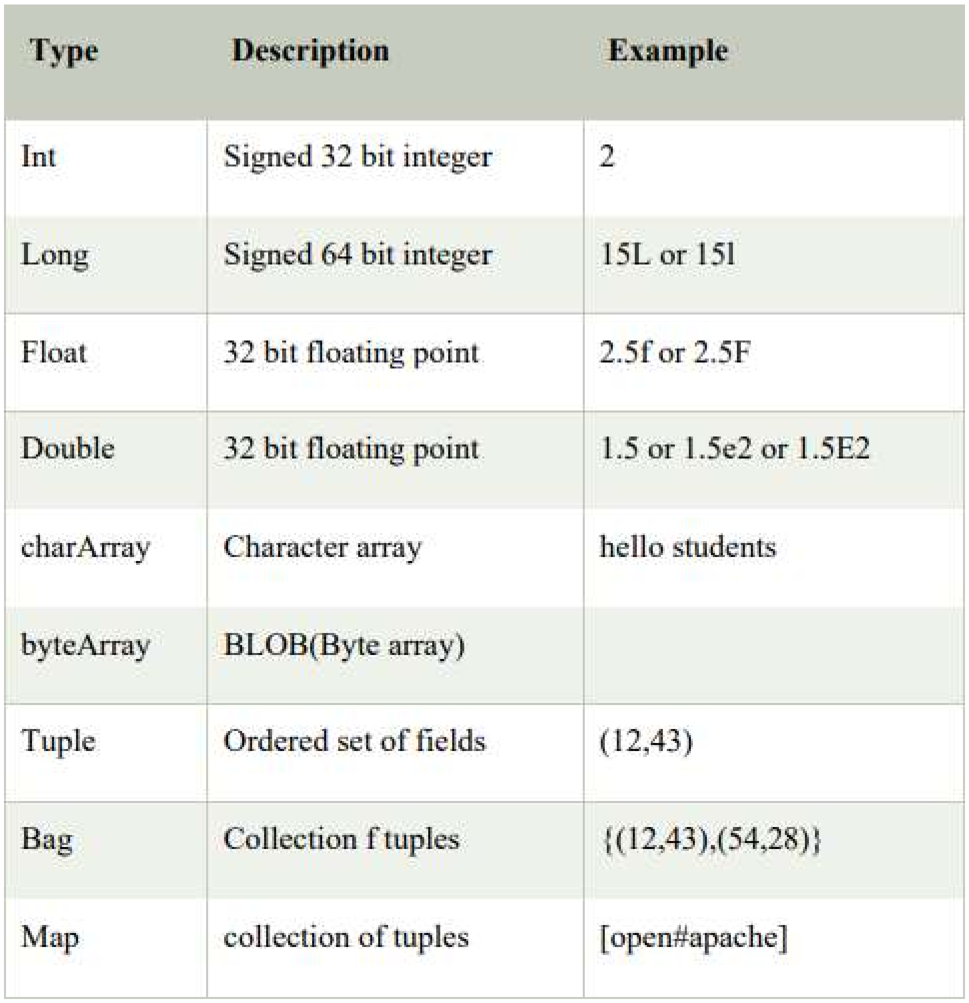
dump y;



```
(4,DEF,DEFG,9422980744,Delhi)
(5,GHI,GHIJ,9422980719,Pune)
grunt>
```

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.

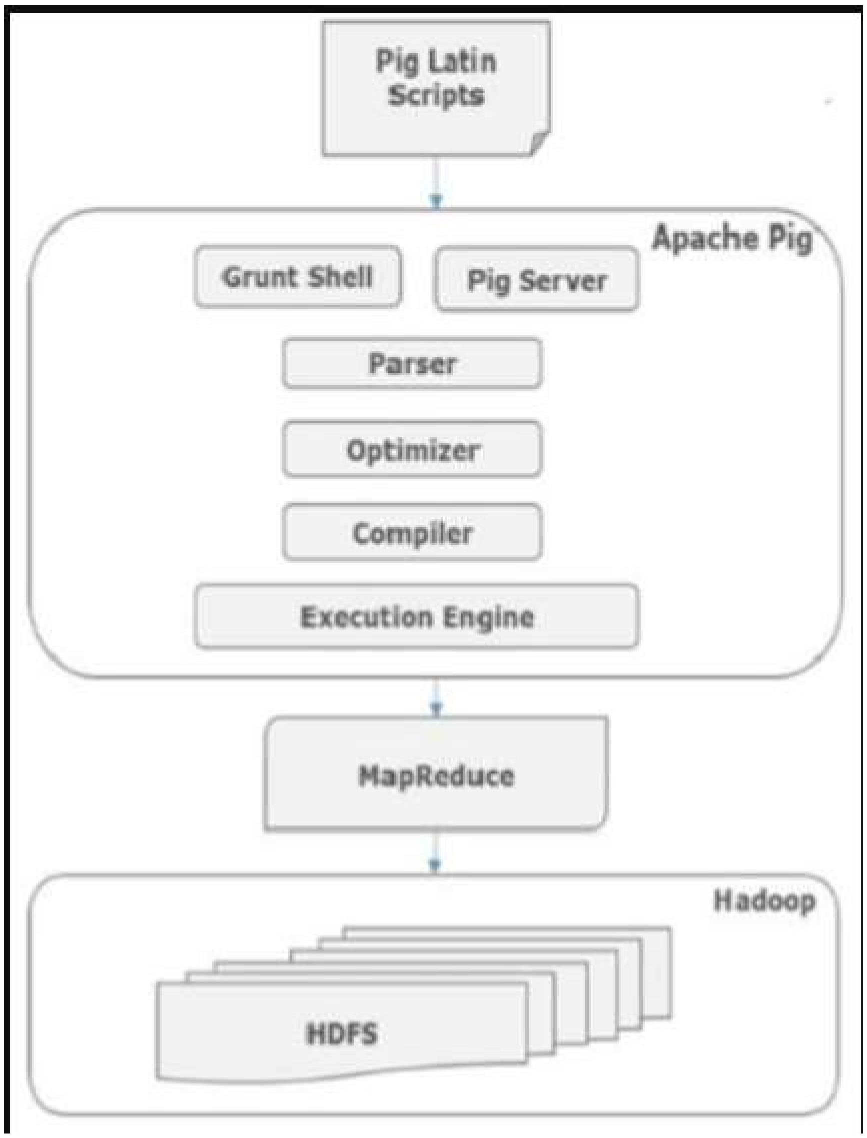


Type	Description	Example
Int	Signed 32 bit integer	2
Long	Signed 64 bit integer	15L or 15l
Float	32 bit floating point	2.5f or 2.5F
Double	32 bit floating point	1.5 or 1.5e2 or 1.5E2
charArray	Character array	hello students
byteArray	BLOB(Byte array)	
Tuple	Ordered set of fields	(12,43)
Bag	Collection f tuples	{(12,43),(54,28)}
Map	collection of tuples	[open#apache]

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

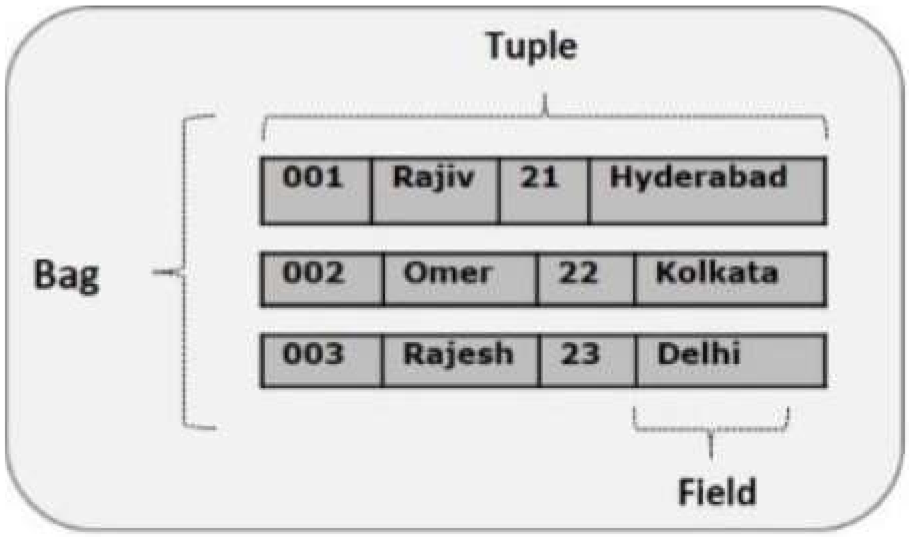
The compiler compiles the optimized logical plan into a series of MapReduce jobs.

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,}}

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

5.3 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.

5.4 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