

## **ANALYSIS ON MOVIES DATA**

**PROJECT SUPERVISOR: Prof. Bhairav Mehta**

**BIG DATA-SAS - CS 850 4 SUMMER 2015**

**Prepared By:-**

<b>Student ID</b>	<b>Name</b>
83111	Vidya Rani Gidiginjala Manne
86448	Simmy Payyappilly Varghese
86545	Remya Nekkuth Melath
86613	Monali Modi
86673	Ajuba Benazir Riyaz

# INDEX

1. Acknowledgement
  - 1.1 Abstract
2. Introduction
  - 2.1 History
  - 2.2 Why PIG?
3. Overview
  - 3.1 Pig Component
  - 3.2 Pig Latin
    - 3.2.1 Category: Loading and Storing
    - 3.2.2 Category: Filtering
    - 3.2.3 Category: Grouping and Joining
    - 3.2.4 Category: Sorting
    - 3.2.5 Category: Combining and Splitting
    - 3.2.6 Category: Diagnostic Operators
    - 3.2.7 Category: Eval
  - 3.3 Basic Program Structure
    - 3.3.1 Script
    - 3.3.2 Grunt
    - 3.3.3 Embedded
  - 3.4 Pig Data Flow
    - 3.4.1 Load
    - 3.4.2 Transform
    - 3.4.3 Dump/Store
  - 3.5 Pig Data Model
    - 3.5.1 Field
    - 3.5.2 Bag
    - 3.5.3 Tuple
    - 3.5.4 Map
    - 3.5.5 Relation
  - 3.6 Pig Compilation and Execution Stages

3.6.1 Parsing

3.6.2 Logical Optimizer

3.6.3 MapReduce Compiler and Optimizer

3.6.4 Hadoop Job Manager

4. What is the problem?

5. Technology Requirements

6. Why are we Using it?

7. How Problem Is Solved?

7.1 Execution Steps

7.1.1 Find the number of movies released between 1950 and 1960

7.1.2 Find the number of movies having rating more than 4

7.1.3 Find the movies whose rating are between 3 and 4

7.1.4 Find the number of movies with duration more than 2 hours(7200 seconds)

7.1.5 Find the list of years and number of movies released in each year

7.1.6 Find the total number of movies in Dataset

8. Challenges Faced

9. Conclusion

10. Recommendations

11. References

# 1 Acknowledgement

We express our profound sense of gratitude in all its humbleness to our beloved **Professor Bhairav Mehta**, Faculty, **International Technological University**, USA for his gracious guidance, meticulous care and unstinted co-operation throughout our work in delivering this project.

We take this opportunity to thank him for the valuable support and guidance extended to us which helped in the successful completion of the project.

We would like to express our heartfelt thanks to members of our Project for their involvement and contributions throughout the project in accomplishing the tasks assigned.

## 1.1 ABSTRACT

Apache pig is a high level procedural language platform developed to simplify querying large data sets in Apache Hadoop and MapReduce . Pig is a Hadoop extension that simplifies Hadoop programming by giving a high-level data processing language while keeping Hadoop's simple scalability and reliability. Pig Latin is the high level scripting language to express the data flow, which is a sweet spot between the low level, procedural style of Map Reduce and declarative style of SQL.

Yahoo , one of the heaviest user of Hadoop (and a backer of both the Hadoop Core and Pig), runs 40 percent of all its Hadoop jobs with Pig. Twitter is also another well-known user of Pig. This project introduces Apache Pig, it's core concepts, the reasons of it's popularity, analysis of a given business problem, future scope and recommendations related to this topic.

## 2 INTRODUCTION

Apache Pig is a high level scripting language that is used with Apache Hadoop to analyze large amounts of data by representing them as data flows. It is a platform for analyzing large data sets that consists of a high-level language for expressing data analysis programs, coupled with infrastructure for evaluating these programs. It make use of high scripting language called Pig Latin to express the data flow. Mapreduce allows programmers to specify a map function followed by a reduce function. But working out how to fit the data processing into this pattern requires multiple map and reduce stages can be a challenge. Even if mapreduce is used by programmers, it has a drawback that, the development cycle is too long. Writing the mappers and reducers, compiling and packaging the code, submitting the jobs, and retrieving the results is a time consuming business and even with streaming, which removes the compile and package step.

## **2.1 HISTORY**

Pig started out as a research project in Yahoo! Research, where Yahoo! scientists designed it and produced an initial implementation. In 2008, the researchers felt that the Mapreduce paradigm presented by Hadoop “is too low-level and rigid, and leads to a great deal of custom user code that is hard to maintain and reuse.” At the same time they observed that many Mapreduce users were not comfortable with declarative languages such as SQL. Thus they set out to produce “a new language called Pig Latin that we have designed to fit in a sweet spot between the declarative style of SQL, and the low-level, procedural style of Map Reduce.”

Yahoo! Hadoop users started to adopt Pig. So, a team of development engineers was assembled to take the research prototype and build it into a production-quality product. In 2007, Pig was open sourced via the Apache Incubator. The first Pig released in September 2008. Later that same year, Pig became a subproject of Apache Hadoop. Early in 2009 other companies started to use Pig for their data processing. Amazon also added Pig as part of its Elastic Mapreduce service. By the end of 2009 about half of Hadoop jobs at Yahoo! were Pig jobs. In 2010, Pig adoption continued to grow, and Pig graduated from a Hadoop subproject, becoming its own top-level Apache project.

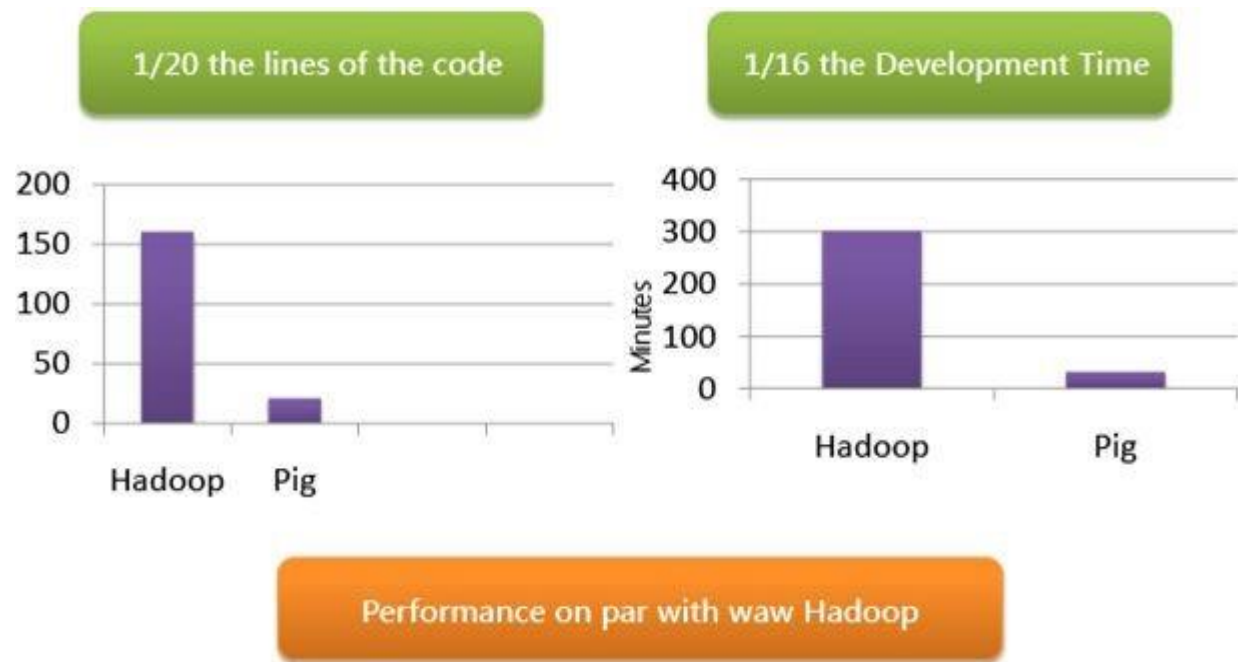
With pig, the data structures are richer, being multi valued and nested, and the set of transformations apply to the data are much more powerful. Pig was mainly designed for performing a long set of data operations, making it ideal for three categories of Big Data jobs: Standard Extract-transform-load (ETL) data pipelines, research on raw data, and iterative data processing.

## 2.2 Why PIG ?

Apache Pig is an open source high-level dataflow system. The key property of Pig programs is that their structure is amenable to substantial parallelization, which in turns enables them to handle very large data sets.

Pig is very popular in the current big data Industry. Some of the reasons for this is:

- Pig Hadoop follows a multi query approach thus it cuts down on the number of times the data is scanned.
- Pig Hadoop is very easy to learn read and write if you are familiar with SQL.
- Pig provides the users with a wide range of nested data types such as Maps, Tuples and Bags that are not present in Mapreduce along with some major data operations such as Ordering, Filters, and Joins.
- Performance of Pig is on par with the performance of raw Map Reduce.
- Pig has various user groups for instance 90% of Yahoo's Mapreduce is done by Pig, 80% of Twitter's Mapreduce is also done by Pig and various other companies such as Salesforce, LinkedIn, AOL and Nokia also employ Pig.



From the figure it is clear that Pig takes 1/20th lines of code as compared to Map Reduce program. Also, the development time for Pig is about 1/16th as compared to that of Map Reduce program.

Pig has lots of advantages over other languages. Pig is much higher level declarative language than Map Reduce which increases programmer productivity, decreases duplication of effort and also opens the Map Reduce programming system to more users. Pig insulates Hadoop complexity from the user. Pig is similar to SQL query where the user specifies the “what” and leaves the “how” to the underlying processing engine. It has the ability to process terabytes of data by issuing a half dozen lines of Pig Latin from the console. Pig is very useful for time sensitive data loads, processing many data sources and, for analytic insight through sampling. At the same time, it is not useful when processing really nasty data formats of completely unstructured data (e.g. video, audio, raw human-readable text), it is definitely slow in comparison to Map Reduce jobs, when more power is required to optimize the code. Pig is not suitable for all data processing tasks. Like map reduce, it is designed for batch processing of data. So if we want to execute a query for small portion of data, it won’t perform well because it is set up to scan the whole data sets. Writing queries in Pig Latin will save the time, unless willing to invest a lot of effort optimizing the java map reduce code.

We can summarize Pig's philosophy toward data types in its slogan of "Pigs eat anything." Input data can come in any format. Popular formats, such as tab-delimited text files, are natively supported. Users can add functions to support other data file formats as well. Pig doesn't require metadata or schema on data, but it can take advantage of them if they're provided.

## 3 OVERVIEW

### 3.1 PIG COMPONENTS

Pig comprises of two components, the first is the language itself, which is called Pig Latin and the second is a runtime environment where scripts are executed.

### 3.2 PIG LATIN

Pig Latin is a simple scripting language to write complex MapReduce transformations. It is made up of a series of operations or transformations, that are applied to input data to produce output. Pig Latin defines a set of transformations on a data set such as aggregate, join and sort. The statement written in Pig script is then translated by Pig into Map Reduce so that it can be executed within Hadoop. Pig Latin may be extended using User Defined Functions (UDFs), which the user can write in Java, Python etc. or a scripting language and then call directly from the Pig Latin. The key properties of Pig Latin are:

- **Easy to Program:** Complex tasks involving interrelated data transformations can be simplified and encoded as data flow sequences. Pig programs can be used to do huge tasks, but they are easy to write and maintain.
- **Optimization opportunities:** The encoded tasks have the ability to optimize their execution automatically which allows the user to concentrate on semantics rather than efficiency.
- **Extensible:** Pig users can create their own functions to meet their special-purpose processing.



Pig Latin looks different from many of the programming languages. It does not contain any if statements or for loops. This is because traditional procedural and object-oriented programming languages describe control flow, and data flow is a side effect of the program. Pig Latin instead focuses on data flow.

A Pig Latin program consist of a collection of statements. A statement can be an operation or a command. Like many programming language, each statements are usually terminating with a semicolon to omit error. But in general, statements or commands for interactive use grunt do not need semicolon.

Pig Latin has two forms of comments. Double hyphens are single line comments (--My program). Everything from first hyphen to end of the line is ignored by Interpreter. C-style comments are more flexible since they delimit the beginning and end of comment block with /\* and \*/ markers. Operators and commands in Pig Latin are not case sensitive, but the aliases and function names are case sensitive.

As a pig Latin is executed, each statement is parsed in turn. If there is any syntax errors or other problems, the interpreter will halt and display error message. The interpreter build a logical plan for every relational operations, which forms the core of Pig Latin programs.

Some of the **relational operators** that can be a part of logical plan are:

#### Category: Loading and Storing

- **LOAD:** Loads data from the file system or other storage into a relation.
- **STORE:** Saves a relation to the file system or other storage.
- **DUMP:** Prints a relation to the console.

#### Category: Filtering

- **FILTER:** Removes unwanted rows from a relation.
- **DISTINCT:** Removes duplicate rows from a relation.
- **FOREACH ...GENERATE:** Generates data transformations based on columns of data.
  - **STREAM:** Transforms a relation using an external program.

#### Category: Grouping and Joining

- **JOIN:** Joins two or more relations.
- **COGROUP:** Groups the data in two or more relations.
- **GROUP:** Groups the data in a single relation.
- **CROSS:** Creates the cross product of two or more relations.

#### Category: Sorting

- **ORDER:** Sorts a relation by one or more fields.
- **LIMIT:** Limits the size of a relation to a maximum number of tuples.

#### Category: Combining and Splitting

- **UNION:** Combine two or more relations into one.
- **SPLIT:** Splits a relation into two or more relations.

Some of the statements that are not added to the logical plan is:

#### Category: Diagnostic operators

- **DESCRIBE:** Prints a relation's schema .
- **EXPLAIN :** Prints the logical and physical plans.
- **ILLUSTRATE:** Shows a sample execution of the logical plan, using a generated subset of the input.

Pig comes with a collection of **built-in functions**. Some of the built-in functions of Pig include:

#### Category: Eval

- **COUNT :** Calculates the number of non-null entries in a bag.
- **DIFF:** Calculates the set difference of two bags. If the two arguments are not bags, returns a bag containing both if they are equal; otherwise, returns an empty bag.
- **SIZE:** Calculates the size of a type. The size of numeric types is always one; for character arrays, it is the number of characters; for byte arrays, the number of bytes; and for containers (tuple, bag, map), it is the number of entries.

### Category: Filter

- **IsEmpty** : Tests whether a bag or map is empty.

### Category: Load/Store

- **PigStorage**: Loads or stores relations using a field-delimited text format. Each line is broken into fields using a configurable field delimiter (defaults to a tab character) to be stored in the tuple's fields. It is the default storage when none is specified.

## 3.3 BASIC PROGRAM STRUCTURE

Pig Latin can be executed in three ways or forms, script file, directly in shell or in embedded form.

### SCRIPT:

Pig can be run as a script file with extension '.pig' containing Pig Commands. This kind of execution is also known as **Batch mode execution**.

**GRUNT:**  
Grunt is an interactive shell used for running Pig commands. In this form, Pig commands are entered manually using Pig's interactive shell, Grunt.

Pig Scripts can be run from within the Grunt using run or exec (execute) commands. If **exec command** runs the pig script with no interaction between script and grunt shell i.e. the name of relations or fields written in script won't be available in grunt also aliases defined via grunt won't be available to script. Whereas **run command** allows the aliases defined in grunt to be accessible in script file and vice versa

### EMBEDDED:

To enable control flow, you can embed Pig Latin statements and Pig commands in the Java programming language (just like SQL can be embedded in programs using JDBC). Pig Latin statement can be embedded in Python, JavaScript, or Java.

**Pig Latin can be executed in two modes of execution: Local and cluster mode.**

**Local Mode:** Pig Latin can be executed from a single computer. In local mode, pig can access input and output files from local system. To initiate the local mode execution following command is used: **pig -x local**.

**Cluster Mode/ Hadoop (mapreduce) Mode:** To execute Pig Latin in cluster mode, Hadoop cluster and HDFS installation should be accessed. To start cluster mode in shell, use the command **pig /pig -x mapreduce**. And the input and output file are stored in HDFS.

## 3.4 PIG DATA FLOW

**LOAD:** The first step in a Pig program is to LOAD the data from HDFS. For a Pig program to access this data, the program must tell Pig what file (or files) it will use, and that's done through the LOAD 'data-file' command (where 'data-file' specifies either an HDFS file or directory). If a directory is specified, all the files in that directory will be loaded into the program. If the data is stored in a file format that is not natively accessible to Pig, you can optionally add the USING function to the LOAD statement to specify a user-defined function that can read in and interpret the data.

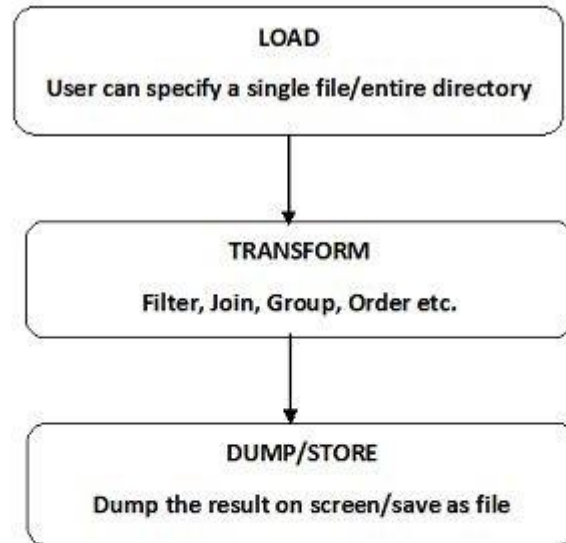


Figure: Steps Involved in Data Flow

**TRANSFORM:** Run the data through a set of transformations (which, internally are translated into a set of mapper and reducer tasks). The transformation logic is where all the data manipulation happens. For Example, FILTER out rows that are not of interest, JOIN two sets of data files, GROUP data to build aggregations, ORDER results, and so on.

**DUMP/STORE:** Finally DUMP the data to the screen or STORE the results in a file either in local/HDFS file system. If DUMP or STORE command is not given, the results of a Pig program will not be generated. DUMP command sends the output to the screen and can be used while debugging the Pig programs. And use STORE call if the results of execution is needed for further processing or analysis.

### 3.5 PIG DATA MODEL

Data model can be defined as follows:

Pig has three complex data types: maps, tuples, and bags. All of these types can contain data of any type, including other complex types.

**Field** is a piece of data or a simple atomic value. (e.g. 'Alice' or '1.0')

**Bag** is an unordered collection of tuples (duplicate tuples are allowed). It may be similar to a "table" in RDBMS, except that Pig does not require that the tuple field types match, or even that the tuples have the same number of fields. A bag is denoted by curly braces {}. Example {(Alice, 1.0), (Alice, 2.0)}.

**Tuple** is a fixed-length, ordered collection of Pig data elements. Tuples are divided into fields, with each field containing one data element. These elements can be of any type—they do not all need to be the same type. A tuple is analogous to a row in SQL, with the fields being SQL columns. Example (Alice, 1.0) or (Alice, 2.0)).

**Map** in Pig is a chararray to data element mapping, where that element can be any Pig type, including a complex type. The chararray is called a key and is used as an index to find the element, referred to as the value. Map constants are formed using brackets to delimit the map, a hash between keys and values, and a comma between key-value pairs. For example, ['name'#'bob', 'age'#55] will create a map with two keys, "name" and "age". The first value is a chararray, and the second is an integer.

**Relation** is a bag of tuples. A Pig relation is similar to a table in a relational database, where the tuples in the bag correspond to the rows in a table. Unlike a relational table, however, Pig relations don't require that every tuple contain the same number of fields or that the fields in the same position (column) have the same type.

### 3.6 PIG COMPILATION AND EXECUTION STAGES

Pig program goes through a series of transformation steps before being executed.

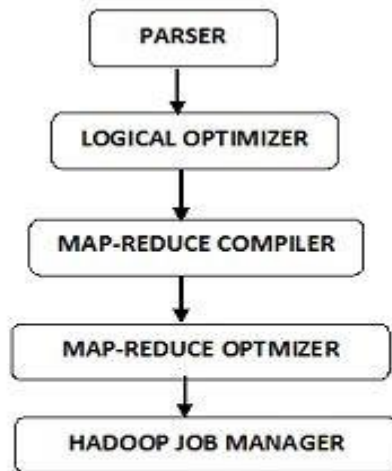


Figure: Compilation and Execution Stages

Building Logical Plan:-

**PARSING:** This is the initial step and Pig program is verified to ensure that it is syntactically correct and that all referenced variables are defined. As client issues the Pig Latin commands whether in interactive or batch mode, Pig interpreter parses the commands. It verifies validity of input files and bags (variables). It also does Type checking and schema inference. Output of the parsing will be a canonical **logical plan** with one-to-one correspondence between Pig Latin statements and logical operators which are arranged in directed acyclic graph (DAG).

Pig builds a logical plan for every bag. When a new bag is defined by a command, new logical plan is a combination of the plans for input and that of current command. No processing is carried out when constructing the logical plans. Processing is triggered only by STORE and DUMP. And at this point logical plan is compiled to physical plan of Map Reduce Job.

**LOGICAL OPTIMIZER:** The logical plan generated by the parser is then passed through a logical optimizer. This will restructure the logical plan by applying operations such as filter to improve the performance.

Building Physical Plan:-

**MAPREDUCE COMPILER AND OPTIMIZER:** The optimized logical plan is then compiled into a series of Mapreduce jobs form physical plan, which then pass through another optimization

phase. An example of Mapreduce-level optimization is utilizing the Mapreduce combiner stage to perform early partial aggregation.

**HADOOP JOB MANAGER:** The DAG of optimized Mapreduce jobs is then topologically sorted, and then are submitted to Hadoop for execution in that order. Pig usually monitors the Hadoop execution status, and the user periodically gets the reports on the progress of the overall Pig program. Any warnings or errors that arise during execution are logged and reported to the user.

#### 4) WHAT IS THE PROBLEM?

Using the Pig Latin scripting language and given movies dataset, we can find the number of movies released between 1950 and 1960, having rating more than 4, whose rating are between 3 and 4, with duration more than 2 hours (7200 second), released each year, starting with alphabet "A", in ascending and descending order of movies by year , with duration in minutes by filtering the movies dataset accordingly and then storing the result in HDFS.

#### 5) TECHNOLOGY REQUIREMENTS

There are two technology requirements: Software Requirements and Hardware Requirements. Following table lists those requirements.

Software Requirements	PigLatin
	VMware Player
Hardware Requirements	RAM: 4GB
	OS: CentOS 64-bit
	Version : Workstation 8.0 virtual machine

## 6) WHY ARE WE USING PIG?

The same problem can also be solved using Hive also. But we have used pig instead of Hive due to below mentioned benefits of Pig technology. Pig allows one to load data and user code at any point in the pipeline. Hive, which is RDBMS based, needs the data to be first imported (or loaded) and after that it can be worked upon. Pig allows greater ease of programming and one can use it to analyze data in different ways with more freedom than in a SQL-like language like Hive. Pig is faster in the data import than an RDBMS friendly language like Hive. For pig, line by line execution is possible but for Hive we have to write entire query for program execution.

Following table lists feature of Pig and Hive.

Feature	Pig	Hive
Language	PigLatin	SQL-like
Schemas/Types	Yes (implicit)	Yes (explicit)
Partitions	No	Yes
Server	No	Optional (Thrift)
User Defined Functions (UDF)	Yes (Java)	Yes (Java)
Custom Serializer/Deserializer	Yes	Yes
DFS Direct Access	Yes (explicit)	Yes (implicit)



Join/Order/Sort	Yes	Yes
Shell	Yes	Yes
Streaming	Yes	Yes
Type of data handling	Both structured and unstructured	Only structured

## 7) HOW PROBLEM IS SOLVED?:

For solving this problem we use PIG technology. Apache Pig is a tool used to analyze large amounts of data by representing them as data flows. Using the PigLatin scripting language operations like ETL (Extract, Transform and Load), adhoc data analysis and iterative processing can be easily achieved.

Pig is an abstraction over MapReduce. In other words, all Pig scripts internally are converted into Map and Reduce tasks to get the task done. Pig was built to make programming MapReduce applications easier. Before Pig, Java was the only way to process the data stored on HDFS.

Pig was first built in Yahoo! and later became a top level Apache project. In this series of we will walk through the different features of pig using a sample dataset. Here are Execution steps:

### 7.1 EXECUTION STEPS:

#### Install Pig:

\$ wget <http://mirror.symnds.com/software/Apache/pig/pig-0.12.0/pig-0.12.0.tar.g>

**Untar:** \$ tar xvfz pig-0.12.0.tar.gz

Pig can be started in one of the following two modes:

1. Local Mode
2. Cluster Mode

Using the `'-x local'` options starts pig in the local mode whereas executing the pig command without any options starts in Pig in the cluster mode. When in local mode, pig can access files on the local file system. In cluster mode, pig can access files on HDFS.

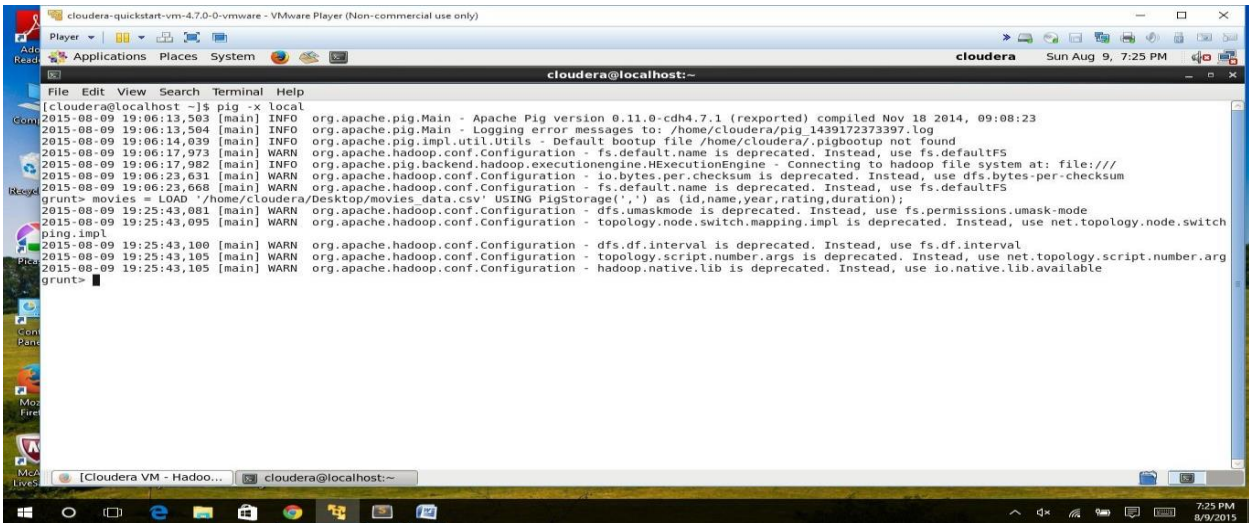
**To start the Local Mode:**     `$ pig -x local`



The grunt shell allows you to execute Pig Latin statements to quickly test out data flows on your data step by step without having to execute complete scripts. now lets start the process..

**To load the data:**

```
grunt> movies = LOAD '/home/cloudera/Desktop/movies_data.csv' USING PigStorage(',') as
(id,name,year,rating,duration);
```



```
[cloudera@localhost ~]$ pig -x local
2015-08-09 19:06:13,503 [main] INFO org.apache.pig.Main - Apache Pig version 0.11.0-cdh4.7.1 (reexported) compiled Nov 18 2014, 09:08:23
2015-08-09 19:06:13,504 [main] INFO org.apache.pig.Main - Logging error messages to: /home/cloudera/pig_1439172373397.log
2015-08-09 19:06:14,039 [main] INFO org.apache.pig.impl.util.Utils - Default bootstrap file /home/cloudera/.pigbootstrap not found
2015-08-09 19:06:17,973 [main] WARN org.apache.hadoop.conf.Configuration - fs.default.name is deprecated. Instead, use fs.defaultFS
2015-08-09 19:06:17,982 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: file:///
2015-08-09 19:06:23,631 [main] WARN org.apache.hadoop.conf.Configuration - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2015-08-09 19:06:23,668 [main] WARN org.apache.hadoop.conf.Configuration - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> movies = LOAD '/home/cloudera/Desktop/movies_data.csv' USING PigStorage(',') as (id,name,year,rating,duration);
2015-08-09 19:25:43,081 [main] WARN org.apache.hadoop.conf.Configuration - dfs.umaskmode is deprecated. Instead, use fs.permissions.umask-mode
2015-08-09 19:25:43,095 [main] WARN org.apache.hadoop.conf.Configuration - topology.node.switch.mapping.impl is deprecated. Instead, use net.topology.node.switch
ping.impl
2015-08-09 19:25:43,100 [main] WARN org.apache.hadoop.conf.Configuration - dfs.df.interval is deprecated. Instead, use fs.df.interval
2015-08-09 19:25:43,105 [main] WARN org.apache.hadoop.conf.Configuration - topology.script.number.args is deprecated. Instead, use net.topology.script.number.arg
2015-08-09 19:25:43,105 [main] WARN org.apache.hadoop.conf.Configuration - hadoop.native.lib is deprecated. Instead, use io.native.lib.available
grunt>
```

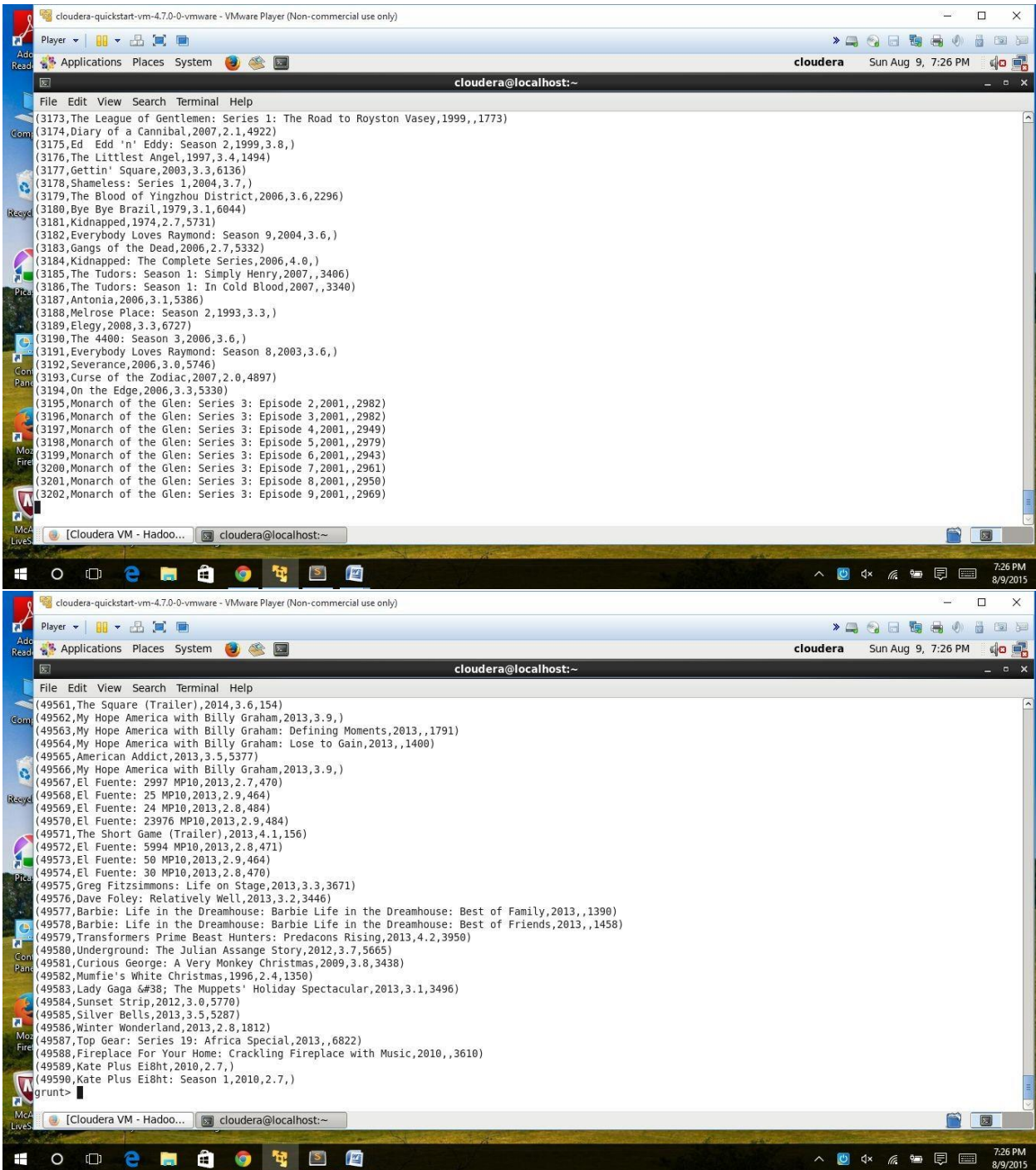
The above statement is made up of two parts. The part to the left of “=” is called the relation or alias. It looks like a variable but you should note that this is not a variable. When this statement is executed, no MapReduce task is executed.

Since our dataset has records with fields separated by a comma we use the keyword USING PigStorage(',').

Another thing we have done in the above statement is giving the names to the fields using the ‘as’ keyword.

**To display the data:** grunt>

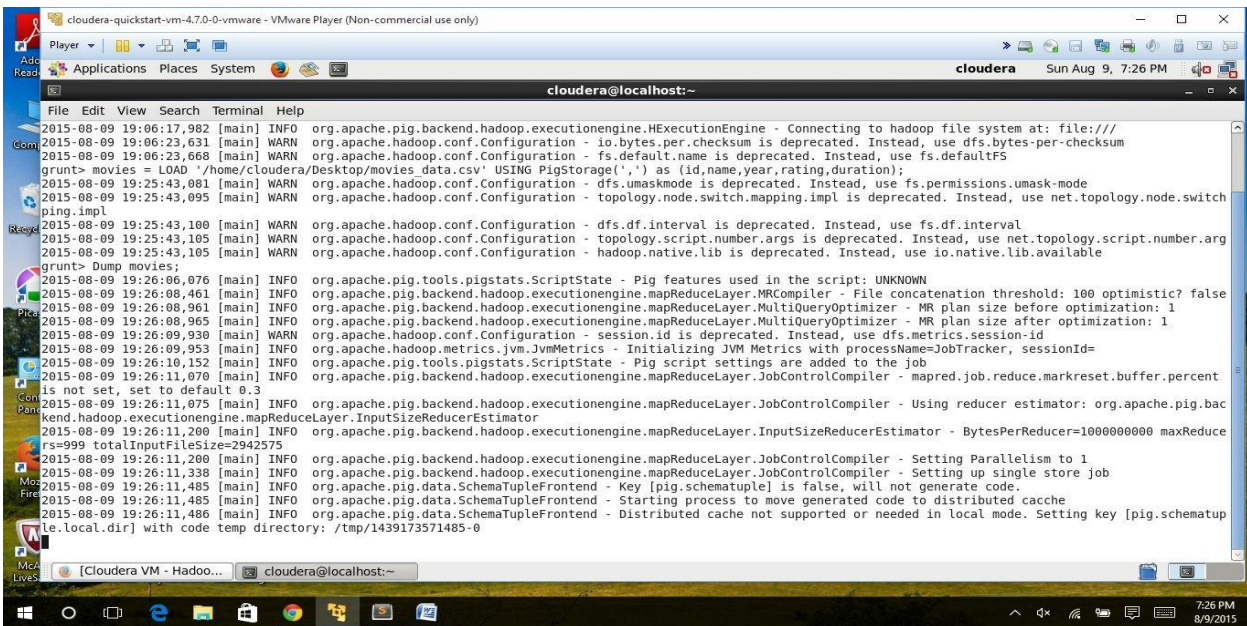
Dump movies;



1) Find the number of movies released between 1950 and 1960. `grunt>`

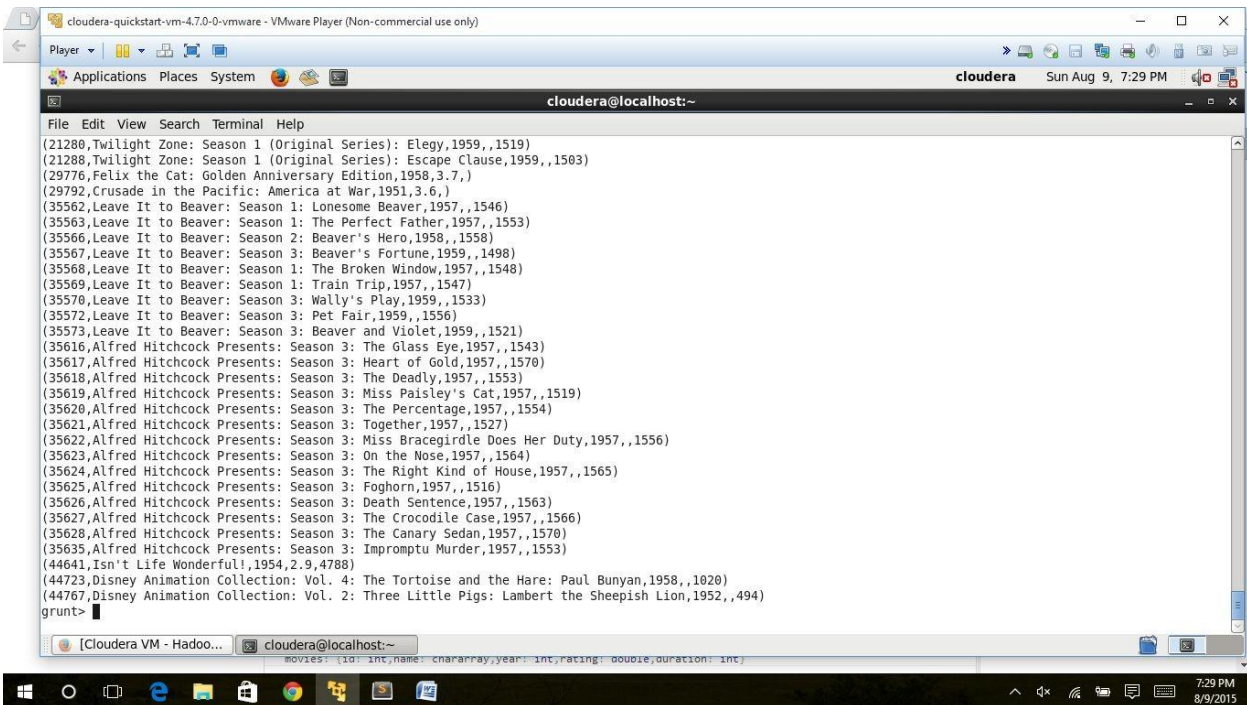
`movies_between_50_60 = FILTER movies by year>1950 and year<1960;`





```
2015-08-09 19:06:17,982 [main] INFO org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at: file:///
2015-08-09 19:06:23,631 [main] WARN org.apache.hadoop.conf.Configuration - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum
2015-08-09 19:06:23,668 [main] WARN org.apache.hadoop.conf.Configuration - fs.default.name is deprecated. Instead, use fs.defaultFS
grunt> movies = LOAD '/home/cloudera/Desktop/movies_data.csv' USING PigStorage(',') as (id,name,year,rating,duration);
2015-08-09 19:25:43,061 [main] WARN org.apache.hadoop.conf.Configuration - dfs.umaskmode is deprecated. Instead, use fs.permissions.umask-mode
2015-08-09 19:25:43,095 [main] WARN org.apache.hadoop.conf.Configuration - topology.node.switch.mapping.impl is deprecated. Instead, use net.topology.node.switch
ping.impl
2015-08-09 19:25:43,100 [main] WARN org.apache.hadoop.conf.Configuration - dfs.df.interval is deprecated. Instead, use fs.df.interval
2015-08-09 19:25:43,105 [main] WARN org.apache.hadoop.conf.Configuration - topology.script.number.args is deprecated. Instead, use net.topology.script.number.args
2015-08-09 19:25:43,105 [main] WARN org.apache.hadoop.conf.Configuration - hadoop.native.lib is deprecated. Instead, use io.native.lib.available
grunt> Dump movies;
2015-08-09 19:26:06,076 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: UNKNOWN
2015-08-09 19:26:08,461 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? false
2015-08-09 19:26:08,961 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size before optimization: 1
2015-08-09 19:26:08,965 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after optimization: 1
2015-08-09 19:26:09,930 [main] WARN org.apache.hadoop.conf.Configuration - session.id is deprecated. Instead, use dfs.metrics.session-id
2015-08-09 19:26:09,953 [main] INFO org.apache.hadoop.metrics.jvm.JvmMetrics - Initializing JVM Metrics with processName=JobTracker, sessionId=
2015-08-09 19:26:10,152 [main] INFO org.apache.pig.tools.pigstats.ScriptState - Pig script settings are added to the job
2015-08-09 19:26:11,070 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - mapred.job.reduce.markreset.buffer.percent
is not set, set to default 0.3
2015-08-09 19:26:11,075 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Using reducer estimator: org.apache.pig.bac
kend.hadoop.executionengine.mapReduceLayer.InputSizeReducerEstimator
2015-08-09 19:26:11,200 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.InputSizeReducerEstimator - BytesPerReducer=1000000000 maxReduce
rs=999 totalInputFileSize=2942575
2015-08-09 19:26:11,200 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Setting Parallelism to 1
2015-08-09 19:26:11,338 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - Setting up single store job
2015-08-09 19:26:11,405 [main] INFO org.apache.pig.data.SchemaTupleFrontend - Key [pig.schematuple] is false, will not generate code.
2015-08-09 19:26:11,485 [main] INFO org.apache.pig.data.SchemaTupleFrontend - Starting process to move generated code to distributed cache
2015-08-09 19:26:11,486 [main] INFO org.apache.pig.data.SchemaTupleFrontend - Distributed cache not supported or needed in local mode. Setting key [pig.schematup
le.local.dir] with code temp directory: /tmp/1439173571485-0
```

Let's Dump the Data. `grunt> Dump movies_between_50_60;`



```
(21280,Twilight Zone: Season 1 (Original Series): Elegy,1959,,1519)
(21288,Twilight Zone: Season 1 (Original Series): Escape Clause,1959,,1503)
(29776,Felix the Cat: Golden Anniversary Edition,1958,3.7,)
(29792,Crusade in the Pacific: America at War,1951,3.6,)
(35562,Leave It to Beaver: Season 1: Lonesome Beaver,1957,,1546)
(35563,Leave It to Beaver: Season 1: The Perfect Father,1957,,1553)
(35566,Leave It to Beaver: Season 2: Beaver's Hero,1958,,1558)
(35567,Leave It to Beaver: Season 3: Beaver's Fortune,1959,,1498)
(35568,Leave It to Beaver: Season 1: The Broken Window,1957,,1548)
(35569,Leave It to Beaver: Season 1: Train Trip,1957,,1547)
(35578,Leave It to Beaver: Season 3: Wally's Play,1959,,1533)
(35572,Leave It to Beaver: Season 3: Pet Fair,1959,,1556)
(35573,Leave It to Beaver: Season 3: Beaver and Violet,1959,,1521)
(35616,Alfred Hitchcock Presents: Season 3: The Glass Eye,1957,,1543)
(35617,Alfred Hitchcock Presents: Season 3: Heart of Gold,1957,,1570)
(35618,Alfred Hitchcock Presents: Season 3: The Deadly,1957,,1553)
(35619,Alfred Hitchcock Presents: Season 3: Miss Paisley's Cat,1957,,1519)
(35620,Alfred Hitchcock Presents: Season 3: The Percentage,1957,,1554)
(35621,Alfred Hitchcock Presents: Season 3: Together,1957,,1527)
(35622,Alfred Hitchcock Presents: Season 3: Miss Bracegirdle Does Her Duty,1957,,1556)
(35623,Alfred Hitchcock Presents: Season 3: On the Nose,1957,,1564)
(35624,Alfred Hitchcock Presents: Season 3: The Right Kind of House,1957,,1565)
(35625,Alfred Hitchcock Presents: Season 3: Foghorn,1957,,1516)
(35626,Alfred Hitchcock Presents: Season 3: Death Sentence,1957,,1563)
(35627,Alfred Hitchcock Presents: Season 3: The Crocodile Case,1957,,1566)
(35628,Alfred Hitchcock Presents: Season 3: The Canary Sedan,1957,,1570)
(35635,Alfred Hitchcock Presents: Season 3: Impromptu Murder,1957,,1553)
(44641,Isn't Life Wonderful!,1954,2.9,4788)
(44723,Disney Animation Collection: Vol. 4: The Tortoise and the Hare: Paul Bunyan,1958,,1020)
(44767,Disney Animation Collection: Vol. 2: Three Little Pigs: Lambert the Sheepish Lion,1952,,494)
grunt>
```

2) Find the number of movies having rating more than 4. `grunt>`

`movies_greater_than_four = FILTER movies BY (float)rating>4.0;`

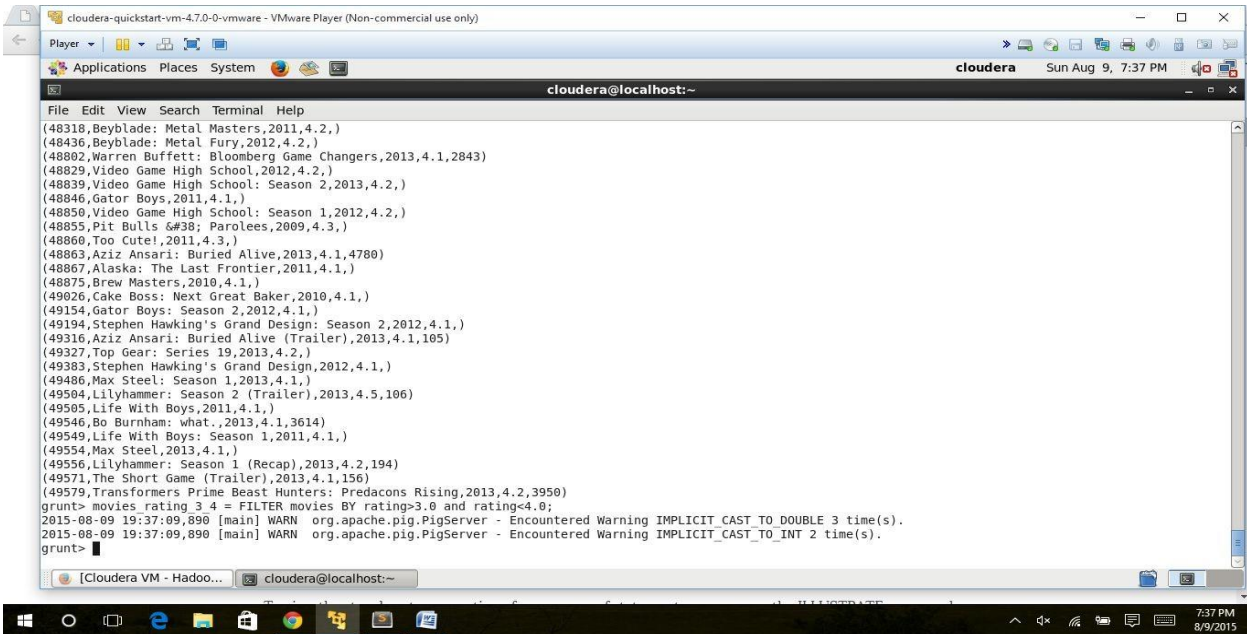
```
cloudera@localhost:~$  
(29792,Crusade in the Pacific: America at War,1951,3.6,)  
(35562,Leave It to Beaver: Season 1: Lonesome Beaver,1957,,1546)  
(35563,Leave It to Beaver: Season 1: The Perfect Father,1957,,1553)  
(35566,Leave It to Beaver: Season 2: Beaver's Hero,1958,,1558)  
(35567,Leave It to Beaver: Season 3: Beaver's Fortune,1959,,1498)  
(35568,Leave It to Beaver: Season 1: The Broken Window,1957,,1548)  
(35569,Leave It to Beaver: Season 1: Train Trip,1957,,1547)  
(35578,Leave It to Beaver: Season 3: Wally's Play,1959,,1533)  
(35572,Leave It to Beaver: Season 3: Pet Fair,1959,,1556)  
(35573,Leave It to Beaver: Season 3: Beaver and Violet,1959,,1521)  
(35616,Alfred Hitchcock Presents: Season 3: The Glass Eye,1957,,1543)  
(35617,Alfred Hitchcock Presents: Season 3: Heart of Gold,1957,,1570)  
(35618,Alfred Hitchcock Presents: Season 3: The Deadly,1957,,1553)  
(35619,Alfred Hitchcock Presents: Season 3: Miss Paisley's Cat,1957,,1519)  
(35620,Alfred Hitchcock Presents: Season 3: The Percentage,1957,,1554)  
(35621,Alfred Hitchcock Presents: Season 3: Together,1957,,1527)  
(35622,Alfred Hitchcock Presents: Season 3: Miss Bracegirdle Does Her Duty,1957,,1556)  
(35623,Alfred Hitchcock Presents: Season 3: On the Nose,1957,,1564)  
(35624,Alfred Hitchcock Presents: Season 3: The Right Kind of House,1957,,1565)  
(35625,Alfred Hitchcock Presents: Season 3: Foghorn,1957,,1516)  
(35626,Alfred Hitchcock Presents: Season 3: Death Sentence,1957,,1563)  
(35627,Alfred Hitchcock Presents: Season 3: The Crocodile Case,1957,,1566)  
(35628,Alfred Hitchcock Presents: Season 3: The Canary Sedan,1957,,1570)  
(35635,Alfred Hitchcock Presents: Season 3: Impromptu Murder,1957,,1553)  
(44641,Isn't Life Wonderful,1954,2.9,4788)  
(44723,Disney Animation Collection: Vol. 4: The Tortoise and the Hare: Paul Bunyan,1958,,1020)  
(44767,Disney Animation Collection: Vol. 2: Three Little Pigs: Lambert the Sheepish Lion,1952,,494)  
grunt> movies greater_than_four = FILTER movies BY (float)rating>4.0;  
2015-08-09 19:29:33,931 [main] WARN org.apache.pig.PigServer - Encountered Warning IMPLICIT_CAST_TO_DOUBLE 1 time(s).  
2015-08-09 19:29:33,931 [main] WARN org.apache.pig.PigServer - Encountered Warning IMPLICIT_CAST_TO_INT 2 time(s).  
grunt>
```

Now dump the data grunt> Dump movies\_greater\_than\_four

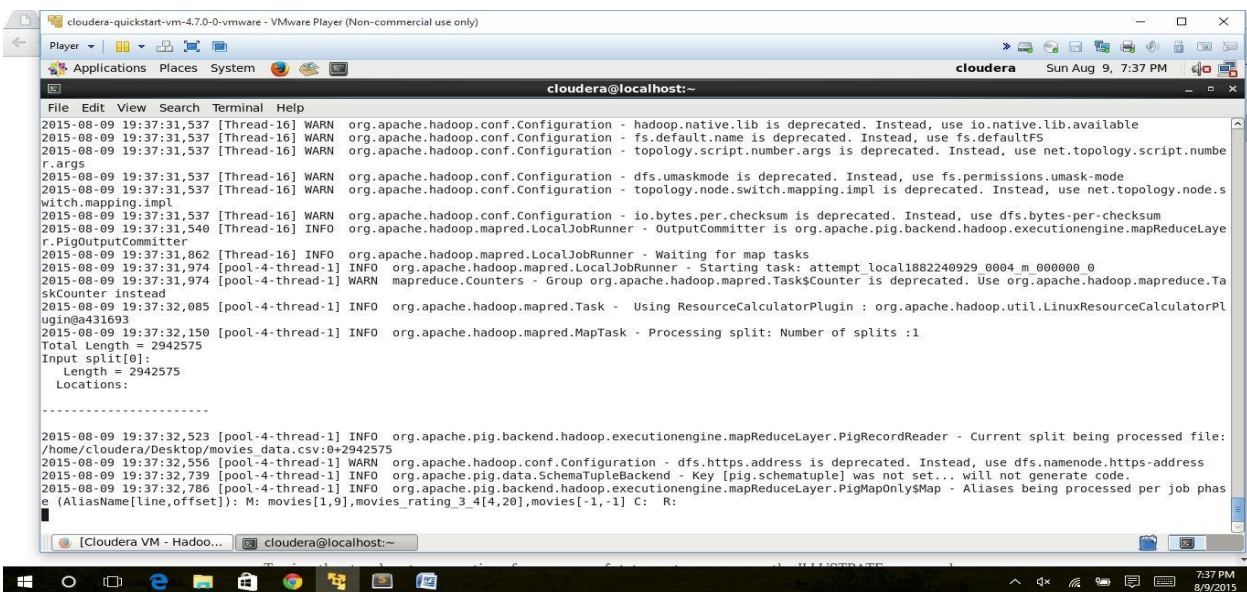
```
cloudera@localhost:~$  
(47953,Beyblade: Metal Fury,2012,4.2,)  
(48206,Cake Boss: Next Great Baker: Season 2,2011,4.1,)  
(48213,Cake Boss: Next Great Baker: Season 1,2010,4.1,)  
(48318,Beyblade: Metal Masters,2011,4.2,)  
(48436,Beyblade: Metal Fury,2012,4.2,)  
(48802,Warren Buffett: Bloomberg Game Changers,2013,4.1,2843)  
(48829,Video Game High School,2012,4.2,)  
(48839,Video Game High School: Season 2,2013,4.2,)  
(48846,Gator Boys,2011,4.1,)  
(48850,Video Game High School: Season 1,2012,4.2,)  
(48855,Pit Bulls & Parolees,2009,4.3,)  
(48860,Too Cute!,2011,4.3,)  
(48863,Aziz Ansari: Buried Alive,2013,4.1,4780)  
(48867,Alaska: The Last Frontier,2011,4.1,)  
(48875,Brew Masters,2010,4.1,)  
(49026,Cake Boss: Next Great Baker,2010,4.1,)  
(49154,Gator Boys: Season 2,2012,4.1,)  
(49194,Stephen Hawking's Grand Design: Season 2,2012,4.1,)  
(49316,Aziz Ansari: Buried Alive (Trailer),2013,4.1,105)  
(49327,Top Gear: Series 19,2013,4.2,)  
(49383,Stephen Hawking's Grand Design,2012,4.1,)  
(49486,Max Steel: Season 1,2013,4.1,)  
(49504,Lilyhammer: Season 2 (Trailer),2013,4.5,106)  
(49505,Life With Boys,2011,4.1,)  
(49546,Bo Burnham: what.,2013,4.1,3614)  
(49549,Life With Boys: Season 1,2011,4.1,)  
(49554,Max Steel,2013,4.1,)  
(49556,Lilyhammer: Season 1 (Recap),2013,4.2,194)  
(49571,The Short Game (Trailer),2013,4.1,156)  
(49579,Transformers Prime Beast Hunters: Predacons Rising,2013,4.2,3950)  
grunt>
```

3) Find the movies whose rating are between 3 and 4. grunt>

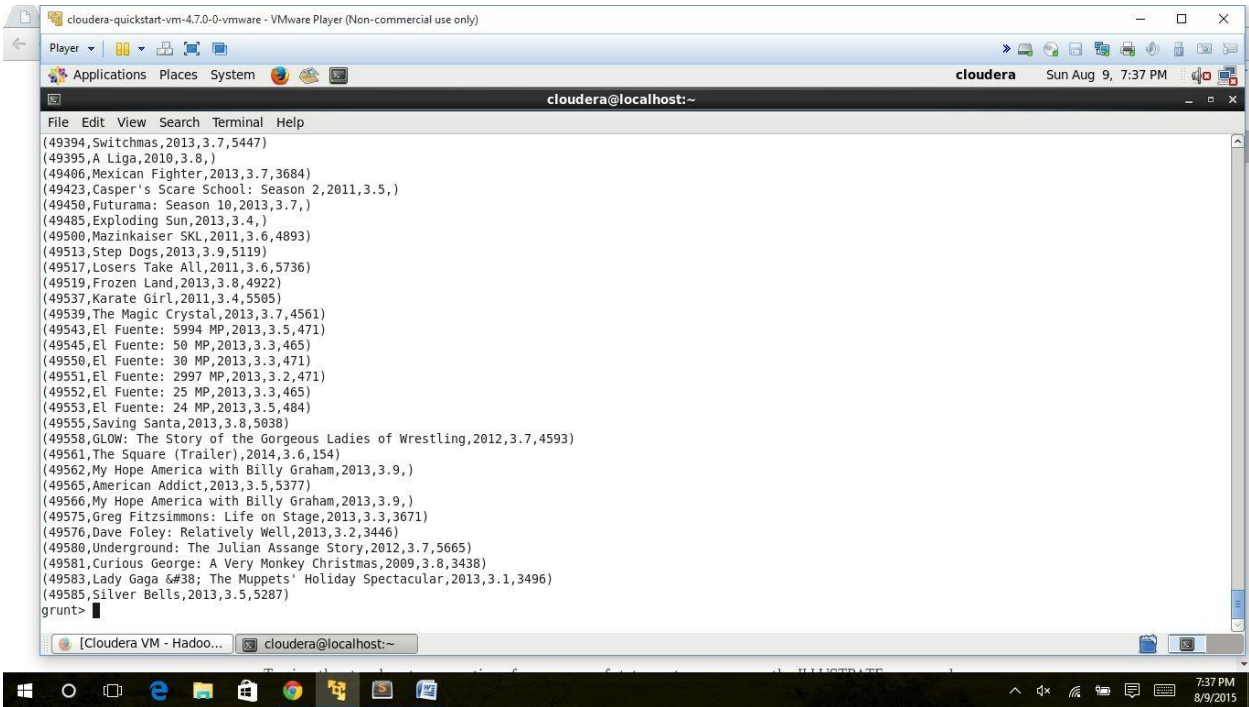
movies\_rating\_3\_4 = FILTER movies BY rating>3.0 and rating<4.0;



now Dump the data. grunt> Dump movies\_rating\_3\_4;

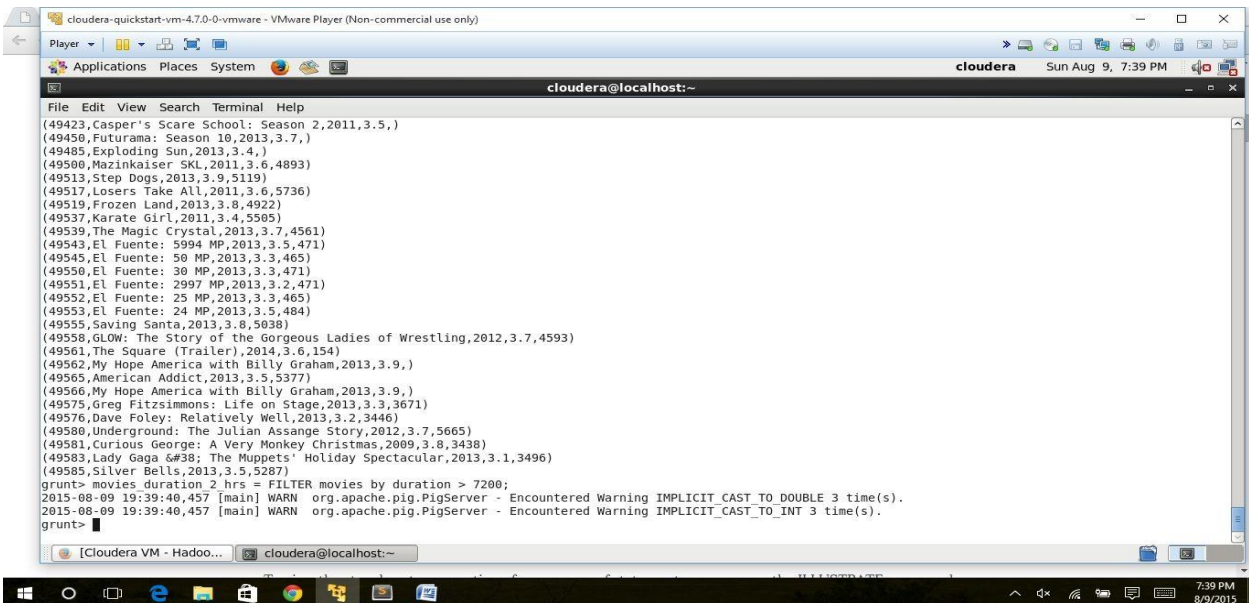






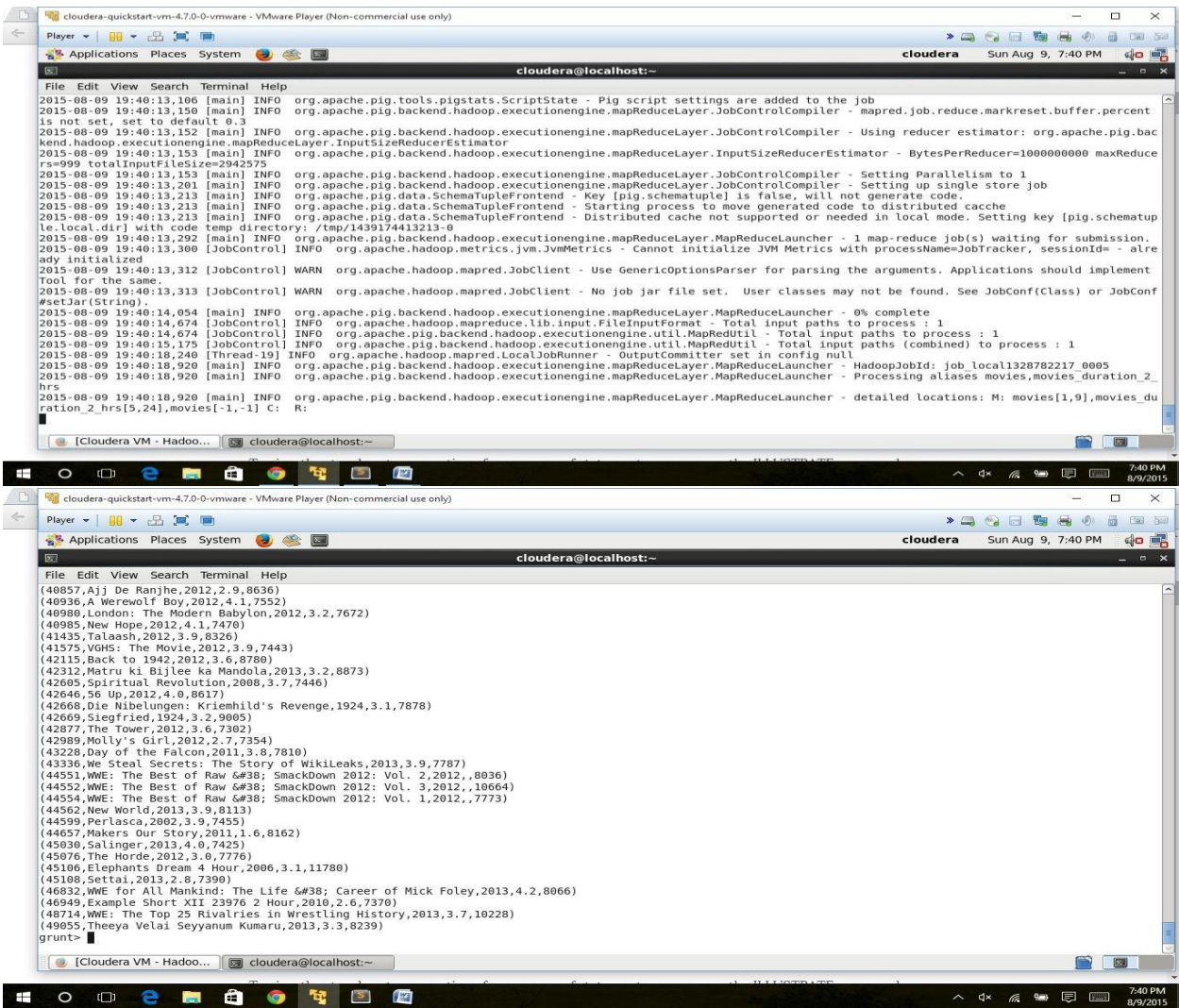
4) Find the number of movies with duration more than 2 hours (7200 second).

grunt> movies\_duration\_2\_hrs = FILTER movies by duration > 7200;



Now lets Display Data. grunt> Dump movies\_duration\_2\_hrs;

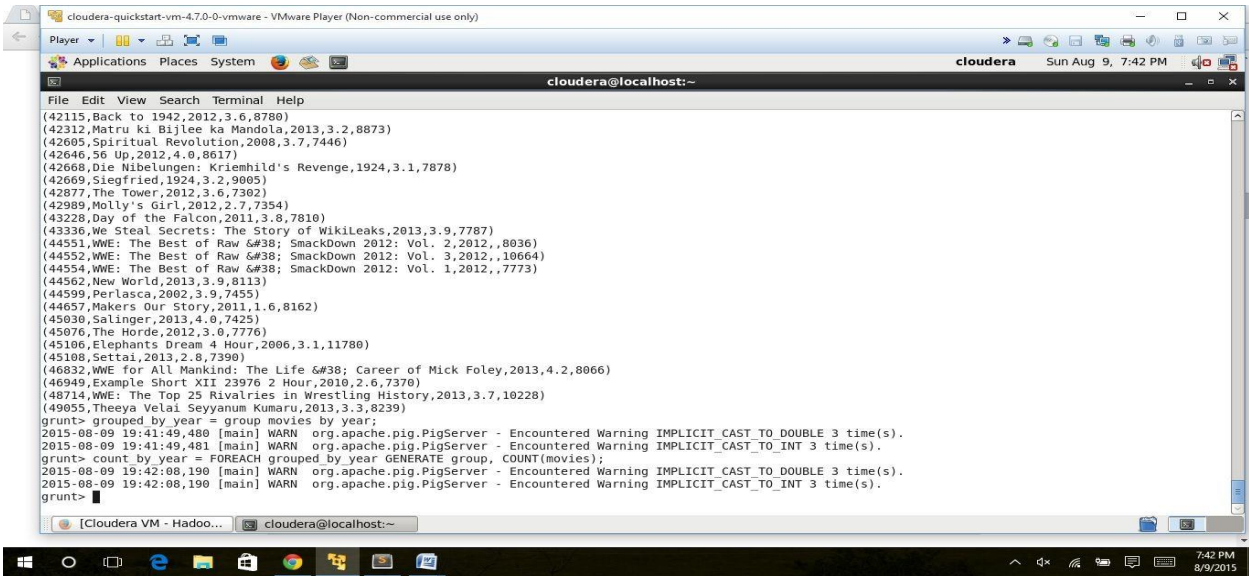




5) Find the list of years and number of movies released each year.

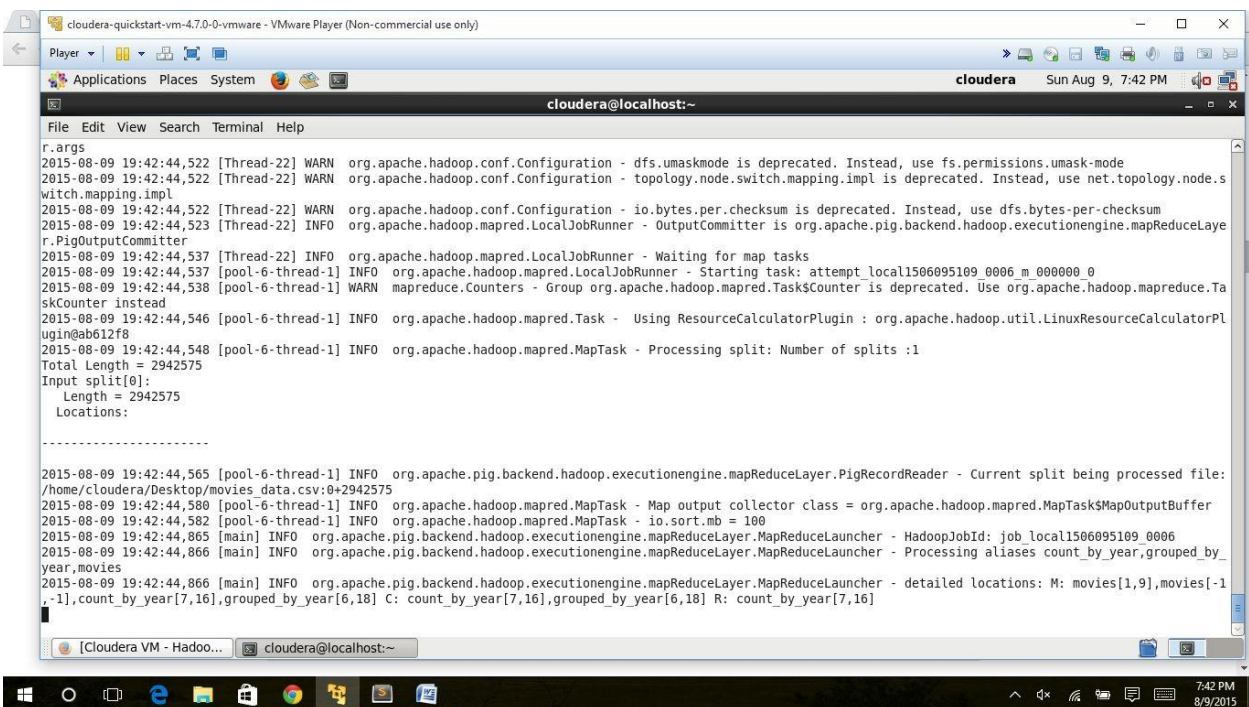
```
grunt> grouped_by_year = group movies by year;
```

```
grunt> count_by_year = FOREACH grouped_by_year GENERATE group, COUNT(movies);
```

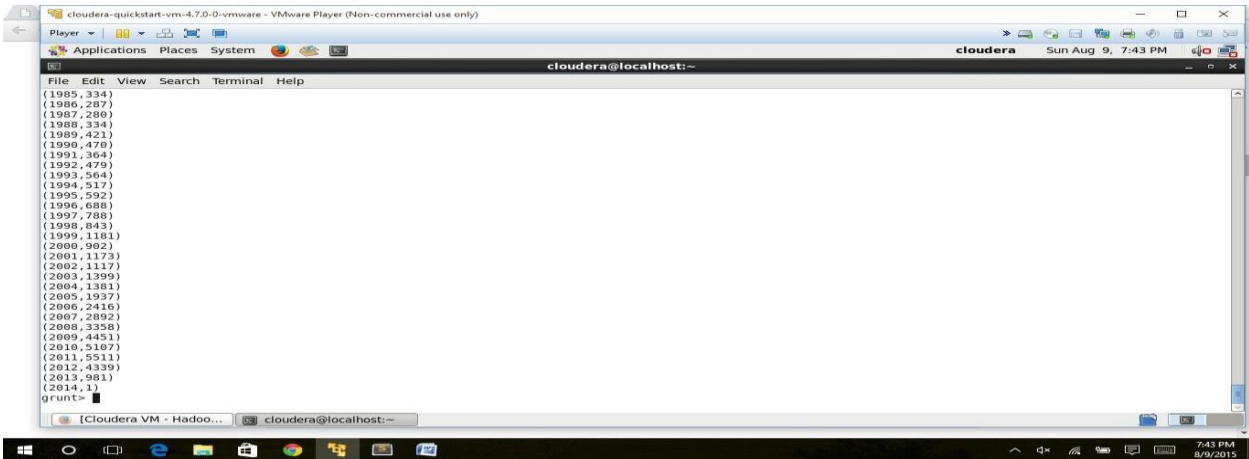


```
cloudera@localhost:~  
File Edit View Search Terminal Help  
(42115,Back to 1942,2012,3.6,8780)  
(42312,Matru Ki Bijlee ka Mandola,2013,3.2,8873)  
(42605,Spiritual Revolution,2008,3.7,7446)  
(42646,56 Up,2012,4.0,8617)  
(42668,Die Nibelungen: Kriemhild's Revenge,1924,3.1,7878)  
(42669,Siegfried,1924,3.2,9005)  
(42877,The Tower,2012,3.6,7302)  
(42989,Molly's Girl,2012,2.7,7354)  
(43228,Day of the Falcon,2011,3.8,7810)  
(43336,We Steal Secrets: The Story of WikiLeaks,2013,3.9,7787)  
(44551,WWE: The Best of Raw & SmackDown 2012: Vol. 2,2012,,8036)  
(44552,WWE: The Best of Raw & SmackDown 2012: Vol. 3,2012,,10664)  
(44554,WWE: The Best of Raw & SmackDown 2012: Vol. 1,2012,,7773)  
(44562,New World,2013,3.9,8113)  
(44599,Perlasca,2002,3.9,7455)  
(44657,Makers Our Story,2011,1.6,8162)  
(45030,Salinger,2013,4.0,7425)  
(45076,The Horde,2012,3.0,7776)  
(45106,Elephants Dream 4 Hour,2006,3.1,11780)  
(45108,Settal,2013,2.8,7390)  
(46832,WWE for All Mankind: The Life & Career of Mick Foley,2013,4.2,8066)  
(46949,Example Short XII 23976 2 Hour,2010,2.6,7370)  
(48714,WWE: The Top 25 Rivalries in Wrestling History,2013,3.7,10228)  
(49055,Theeya Velai Seyyanum Kumaru,2013,3.3,8239)  
grunt> grouped by year = group movies by year;  
2015-08-09 19:41:49,480 [main] WARN org.apache.pig.PigServer - Encountered Warning IMPLICIT_CAST_TO_DOUBLE 3 time(s).  
2015-08-09 19:41:49,481 [main] WARN org.apache.pig.PigServer - Encountered Warning IMPLICIT_CAST_TO_INT 3 time(s).  
grunt> count by year = FOREACH grouped by year GENERATE group, COUNT(movies);  
2015-08-09 19:42:08,190 [main] WARN org.apache.pig.PigServer - Encountered Warning IMPLICIT_CAST_TO_DOUBLE 3 time(s).  
2015-08-09 19:42:08,190 [main] WARN org.apache.pig.PigServer - Encountered Warning IMPLICIT_CAST_TO_INT 3 time(s).  
grunt>
```

Now Dump it. grunt> Dump count\_by\_year;

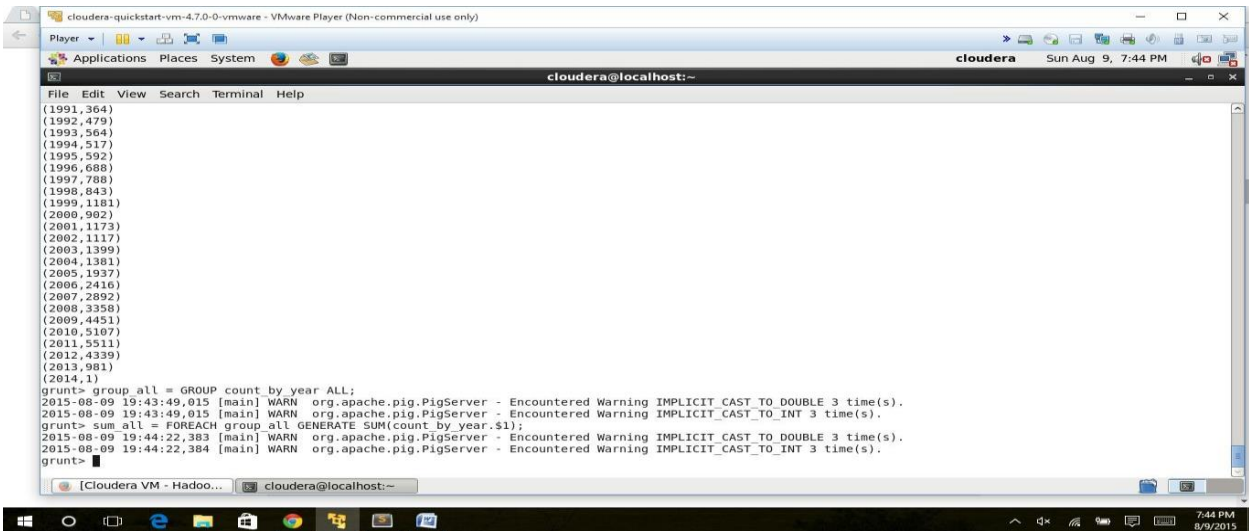


```
cloudera@localhost:~  
File Edit View Search Terminal Help  
r.args  
2015-08-09 19:42:44,522 [Thread-22] WARN org.apache.hadoop.conf.Configuration - dfs.umaskmode is deprecated. Instead, use fs.permissions.umask-mode  
2015-08-09 19:42:44,522 [Thread-22] WARN org.apache.hadoop.conf.Configuration - topology.node.switch.mapping.impl is deprecated. Instead, use net.topology.node.s  
witch.mapping.impl  
2015-08-09 19:42:44,522 [Thread-22] WARN org.apache.hadoop.conf.Configuration - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-checksum  
2015-08-09 19:42:44,523 [Thread-22] INFO org.apache.hadoop.mapred.LocalJobRunner - OutputCommitter is org.apache.pig.backend.hadoop.executionengine.mapReduceLayer  
r.PigOutputCommitter  
2015-08-09 19:42:44,537 [Thread-22] INFO org.apache.hadoop.mapred.LocalJobRunner - Waiting for map tasks  
2015-08-09 19:42:44,537 [pool-6-thread-1] INFO org.apache.hadoop.mapred.LocalJobRunner - Starting task: attempt local1506095109_0006_m_000000_0  
2015-08-09 19:42:44,538 [pool-6-thread-1] WARN mapreduce.Counters - Group org.apache.hadoop.mapred.Task$Counter is deprecated. Use org.apache.hadoop.mapreduce.Ta  
skCounter instead  
2015-08-09 19:42:44,546 [pool-6-thread-1] INFO org.apache.hadoop.mapred.Task - Using ResourceCalculatorPlugin : org.apache.hadoop.util.LinuxResourceCalculatorPl  
ugin@ab612f8  
2015-08-09 19:42:44,548 [pool-6-thread-1] INFO org.apache.hadoop.mapred.MapTask - Processing split: Number of splits :1  
Total Length = 2942575  
Input split[0]:  
Length = 2942575  
Locations:  
-----  
2015-08-09 19:42:44,565 [pool-6-thread-1] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.PigRecordReader - Current split being processed file:  
/home/cloudera/Desktop/movies_data.csv:0+2942575  
2015-08-09 19:42:44,580 [pool-6-thread-1] INFO org.apache.hadoop.mapred.MapTask - Map output collector class = org.apache.hadoop.mapred.MapTaskMapOutputBuffer  
2015-08-09 19:42:44,582 [pool-6-thread-1] INFO org.apache.hadoop.mapred.MapTask - io.sort.mb = 100  
2015-08-09 19:42:44,865 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - HadoopJobId: job_local1506095109_0006  
2015-08-09 19:42:44,866 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Processing aliases count_by_year,grouped_by  
year,movies  
2015-08-09 19:42:44,866 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - detailed locations: M: movies[1,9],movies[-1  
,1],count_by_year[7,16],grouped_by_year[6,18] C: count_by_year[7,16],grouped_by_year[6,18] R: count_by_year[7,16]
```



## 6) Find the total number of movies in the dataset.

```
grunt> group_all = GROUP count_by_year ALL;
grunt> sum_all = FOREACH group_all GENERATE SUM(count_by_year.$1);
```



so when we dump the data `grunt> Dump sum_all;`

It will show the output 49590

## 8) CHALLENGES FACED:

As Pig is easy to program, we haven't faced such challenges except one while doing the project. One challenge we faced was, initially while running the code we got some errors. So for rectifying those errors we used describe (value) and illustrate (type and value).

## **9) CONCLUSION:**

In this project we got a good feel of Apache Pig. We loaded some data and executed some basic commands to query it. Pig is a high level scripting language that is used with Apache Hadoop. Pig enables data workers to write complex data transformations without knowing Java. Pig's simple SQL-like scripting language is called Pig Latin, and appeals to developers already familiar with scripting languages and SQL.

Pig is complete, so you can do all required data manipulations in Apache Hadoop with Pig. Through the User Defined Functions(UDF) facility in Pig, Pig can invoke code in many languages like JRuby, Jython and Java. You can also embed Pig scripts in other languages. The result is that you can use Pig as a component to build larger and more complex applications that tackle real business problems.

Pig works with data from many sources, including structured and unstructured data, and store the results into the Hadoop Data File System.

Pig scripts are translated into a series of MapReduce jobs that are run on the Apache Hadoop cluster.

## **10) RECOMMENDATIONS:**

### **Safe Optimizer:**

Pig Latin being of procedural nature and itself is a kind of query execution plan which gives the programmer more control on the program. Even though database optimizations give highly improved performance. In situations where the optimization depends on unknown data characteristics or performance benefit is uncertain. Pig Latin is an option which will surely yield performance benefit and hence it is a Safe optimizer.

### **External Functions:**

Pig programs can be executed as Map Reduce Jobs. And in Embedded form it supports UDFs written in Java. For fast and ad-hoc tasks, the programmers want to write UDFs in a scripting language such as Perl or Python which would be much simpler and easier, instead of in a full-blown programming language like Java. Such functions require a light-weight serialization/deserialization layer with bindings in the languages when compared to Java. Pig can

then serialize data using this layer and runs the non-Java UDF, where it can be deserialized into that language's data structures. As a scope we can build such a layer and integrating it with Pig

## **10) REFERENCES:**

<http://www-01.ibm.com/software/data/infosphere/hadoop/pig/>

<http://www.folkstalk.com/2013/07/how-to-run-pig-programs-examples.html>

<http://pig.apache.org/docs/r0.9.2/cmds.html>

[http://chimera.labs.oreilly.com/books/1234000001811/ch07.html#syntax\\_highlighting](http://chimera.labs.oreilly.com/books/1234000001811/ch07.html#syntax_highlighting)

<https://wiki.apache.org/pig/RunPig> [http://www.slideshare.net/Jey\\_guru/big-data-components-46436453](http://www.slideshare.net/Jey_guru/big-data-components-46436453) <http://www.slideshare.net/AdamKawa/apache-pig-at-whug>

<http://www.dummies.com/how-to/content/the-pig-architecture-in-hadoop.html>