APACHE PIG
USE OF OPERATORS AND BUILTIN FUNCTIONS

Apache Pig is an abstraction over MapReduce. It is a tool/platform which is used to analyze larger sets of data representing them as data flows.
To write data analysis programs, Pig provides a high-level language known as **Pig Latin**. This language provides various operators using which programmers can develop their own functions for reading, writing, and processing data.

One of the best feature of pig is its support for rich set set of operators. It provides many operators to perform operations like join, sort, filer, etc. Thus it proves to be a handy tool to analyze datasets by performing set of data operations.

Hence this document details use of different pig operators and builtin functions with synatax description and well defined output.

PRE_REQUISITES

In the following use cases we will work with PIG in map_reduce mode and interactively(use grunt shell).
Hence we make sure that we have all hadoop daemons up and running.
1. Make sure hadoop is configured in the system
2. All hadoop daemons are up and running(start-all.sh)
3. Pig is configured
4. Pig is launched in map_reduce mode. (pig) (As by default pig is launched in map_reduce mode)
5. In MapReduce mode, Pig reads (loads) data from HDFS and stores the results back in HDFS. Therefore, let's just load sample datasets in HDFS
I:e grunt > fs -put </path/of/datasetFile/from/LFS> </path/in/HDFS/To store/Dataset>;

-- LFS Location of datasets : '/path/of/datasetFile/from/LFS' (Assume : /home/user/Documents/PigData/)
-- HDFS Location of datasets : '/PigData'
-- HDFS Location of JOIN datsets : '/PigData/joinDatasets'
-- HDFS Location for storing relations : '/PigRelations'
-- HDFS Location for storing outputs : '/PigOutput'

Note : You can the dataset from the following github repository
https://github.com/jainpayal12/UseOfPigOperators/tree/master/PigPracticalData

Thus summarising,
$ start-all.sh
$ pig
grunt > fs -put /home/user/Documents/PigData/ /PigData/
grunt > fs -lsr /PigData

```
-rw-r--r--   1 payal supergroup       233509 2017-11-16 13:20 /PigData/baseball
-rw-r--r--   1 payal supergroup       479 2017-11-16 13:20 /PigData/baseball2
-rw-r--r--   1 payal supergroup       511 2017-11-16 13:20 /PigData/class.csv
-rw-r--r--   1 payal supergroup       129 2017-11-16 13:20 /PigData/class2
```

grunt > fs -put /home/user/Documents/PigData/joinDatasets /PigData/joinDatasets
grunt > fs -lsr /PigData/joinDatasets

```
-rw-r--r--   1 payal supergroup       165 2017-11-23 11:10 /PigData/joinDatasets/customer.txt
-rw-r--r--   1 payal supergroup       298 2017-11-23 12:26 /PigData/joinDatasets/employee.txt
-rw-r--r--   1 payal supergroup       362 2017-11-23 12:26 /PigData/joinDatasets/employeeContact.txt
-rw-r--r--   1 payal supergroup       124 2017-11-23 11:10 /PigData/joinDatasets/orders.txt
```

LIST OF COMMONLY USED OPERATORS

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.

LOAD AND STORE OPERATORS

LOAD

SYNTAX : Relation_Name = LOAD '/Input/File/Path/' USING Load Function AS schema(optional);

where **Relation_Name** is the name of the relation which has to be created
'**/Input/File/Path/**' is file from HDFS as pig is launched in MAP_REDUCE mode
Load Function : Specifies file format to be read : It can be either of the following :

1. PigStorage() [By default] : It loads and stores data as structured text files.It takes a delimiter using which each entity of a tuple is separated, as a parameter. By default, it takes '\t' as a parameter. UTF-8 format.
2. BinStorage() : Works with data that is represented on disk in machine-readable format
3. JsonLoader() : Works with data that is represented in JSON format. It takes Pig schema for the data as a parameter.

4. TextLoader() : Loads unstructured data in UTF-8 format.

schema : schema definition, listing fields and type for a relation.

USING, AS : Keywords

Case1. Load with schema

```
baseball = LOAD '/PigData/baseball'
           AS (Name : chararray, Team : chararray, Position : bag{}, Bat : map[]);
```

Note:
1. We haven't used any Load function. Pig interpretes it to be PigStorage('\t') by default.
2. In case of LOAD, Pig validates the syntax and semantics of all statements(compilation). It is executed only after using DUMP (display results on screen) or STORE (store results in HDFS). Hence we will see that even non-existence of file will not throw error(as runtime error) while we use LOAD / Create a relation.

Case2. Load without schema

```
baseball = LOAD '/PigData/baseball' ;
```

Case3. Load with PigStorage()

```
class = LOAD '/PigData/class.csv'
        USING PigStorage(',')
        AS (ID : chararray, Name : chararray, Gender : chararray, Age : int, Height : float, Weight : float);
```

Case4. Load with JsonLoader()

```
categories = Load '/PigData/JSON_Datasets/categories'
             USING JsonLoader('category_id:int,category_department_id:int,category_name:chararray');
```

Case5. Load with TextLoader()

```
worCount = LOAD '/PigData/red'
           USING TextLoader();
```

Case6. Load with BinStorage()

```
class =  LOAD '/PigOutput/binStorage'
         USING BinStorage()
         AS (ID : chararray, Name : chararray, Gender : chararray, Age : int, Height : float, Weight : float);
```

Note : You can get a binary file using following procedure:

1. Load data using PigStorage()
2. Store data using BinStorage()
3. Load data using BinStorage()

STORE

SYNTAX : STORE Relation_Name INTO '/directory/path/' USING Store_Function;

where **'/directory/Path/'** is path in HDFS as pig is launched in MAP_REDUCE mode. This specifies path where output has to be stored.

Store Function : Specifies file format for data to be wriiten . It can be either of the following :

1. PigStorage() [By default]
2. BinStorage()
3. JsonStorage()

Case 1. Load and store using PigStorage()

```
class = LOAD '/PigData/class.csv'
        USING PigStorage(',')
        AS (ID : chararray, Name : chararray, Gender : chararray, Age : int, Height : float, Weight : float);

STORE class
INTO '/PigOutput/classOutput';
```

Viewing the web UI and browsing the file system, we have 2 files and one flag:

Contents of directory /PigOutput/classOutput

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
_SUCCESS	file	0 KB	1	64 MB	2017-12-02 00:31	rw-r--r--	payal	supergroup
_logs	dir				2017-12-02 00:31	rwxr-xr-x	payal	supergroup
part-m-00000	file	0.48 KB	1	64 MB	2017-12-02 00:31	rw-r--r--	payal	supergroup

Partial Output:

```
A11     Alfred  M       14      69.0    112.5
A22     Alice   F       13      56.5    84.0
B11     Barbara F       13      65.3    98.0
C11     Carol   F       14      62.8    102.5
H11     Henry   M       14      63.5    102.5
```

Note default store function taken is PigStorage()

Case 2. Load using PigStorage() and store using JsonStorage()

```
class = LOAD '/PigData/class.csv'
        USING PigStorage(',')
        AS (ID : chararray, Name : chararray, Gender : chararray, Age : int, Height : float, Weight : float);

STORE class
INTO '/PigOutput/jsonStorage'
USING JsonStorage();
```

Viewing the web UI and browsing the file system, we have 2 files and one flag:

Contents of directory /PigOutput/jsonStorage

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
.pig_header	file	0.03 KB	1	64 MB	2017-12-02 00:15	rw-r--r--	payal	supergroup
.pig_schema	file	0.6 KB	1	64 MB	2017-12-02 00:15	rw-r--r--	payal	supergroup
_SUCCESS	file	0 KB	1	64 MB	2017-12-02 00:15	rw-r--r--	payal	supergroup
_logs	dir				2017-12-02 00:14	rwxr-xr-x	payal	supergroup
part-m-00000	file	1.46 KB	1	64 MB	2017-12-02 00:14	rw-r--r--	payal	supergroup

Partial Output:


```
{"ID":"A11","Name":"Alfred","Gender":"M","Age":14,"Height":69.0,"Weight":112.5}
{"ID":"A22","Name":"Alice","Gender":"F","Age":13,"Height":56.5,"Weight":84.0}
{"ID":"B11","Name":"Barbara","Gender":"F","Age":13,"Height":65.3,"Weight":98.0}
{"ID":"C11","Name":"Carol","Gender":"F","Age":14,"Height":62.8,"Weight":102.5}
{"ID":"H11","Name":"Henry","Gender":"M","Age":14,"Height":63.5,"Weight":102.5}
```


.pig_header

```
ID      Name    Gender  Age     Height  Weight
```

Case 3. Load using JsonLoader() and store using JsonStorage()

```
jsonData = LOAD '/PigOutput/jsonStorage'
            USING JsonLoader('ID : chararray, Name : chararray, Gender : chararray, Age : int, Height : float, Weight :
float');

STORE jsonData
INTO '/PigOutput/jsonLoaderStorage'
USING JsonStorage();
```

Partial Output:

```
{"ID":"A11","Name":"Alfred","Gender":"M","Age":14,"Height":69.0,"Weight":112.5}
{"ID":"A22","Name":"Alice","Gender":"F","Age":13,"Height":56.5,"Weight":84.0}
{"ID":"B11","Name":"Barbara","Gender":"F","Age":13,"Height":65.3,"Weight":98.0}
{"ID":"C11","Name":"Carol","Gender":"F","Age":14,"Height":62.8,"Weight":102.5}
{"ID":"H11","Name":"Henry","Gender":"M","Age":14,"Height":63.5,"Weight":102.5}
```

Note :
Output structure remains same for all store with mininmum of three files.

Case 4. Load using PigStorage() and store using BinStorage()

```
class = LOAD '/PigData/class.csv'
        USING PigStorage(',')
        AS (ID : chararray, Name : chararray, Gender : chararray, Age : int, Height : float, Weight : float);

STORE class
INTO '/PigOutput/binStorage'
USING BinStorage();
```

Partial Output

```
###n���#7�#A117�#Alfred7�#M
���##B���#B���###n���#7�#A227�#Alice7�#F
���
#Bb��#B###n###7#B117#Barbara7#F
```

Case 5. Load using BinStorage() and store using PigStorage()

```
class = LOAD '/PigOutput/binStorage'
USING BinStorage()
AS (ID : chararray, Name : chararray, Gender : chararray, Age : int, Height : float, Weight : float);

STORE class
INTO '/PigOutput/binPigStorage'
USING PigStorage();
```

Partial Output :

A11	Alfred	M	14	69.0	112.5
A22	Alice	F	13	56.5	84.0
B11	Barbara	F	13	65.3	98.0
C11	Carol	F	14	62.8	102.5
H11	Henry	M	14	63.5	102.5

DIAGNOSTIC OPERATORS

DUMP

SYNTAX : DUMP RelationName;
where, RelationName is the relation which has to be executed and output is displayed on std output.

```
class = LOAD '/PigData/class.csv'
        USING PigStorage(',')
        AS (ID : chararray, Name : chararray, Gender : chararray, Age : int, Height : float, Weight : float);

DUMP class;
```

Partial Output:

```
(A11,Alfred,M,14,69.0,112.5)
(A22,Alice,F,13,56.5,84.0)
(B11,Barbara,F,13,65.3,98.0)
(C11,Carol,F,14,62.8,102.5)
(H11,Henry,M,14,63.5,102.5)
```

DESCRIBE

SYNTAX : DESCRIBE RelationName;
where, RelationName is the relation whose schema definition has to be returned

```
class = LOAD '/PigData/class.csv'
        USING PigStorage(',')
        AS (ID : chararray, Name : chararray, Gender : chararray, Age : int, Height : float, Weight : float);

DESCRIBE class;
```

OUTPUT

```
class: {ID: chararray,Name: chararray,Gender: chararray,Age: int,Height: float,Weight: float}
```

EXPLAIN

SYNTAX : EXPLAIN RelationName;
where, RelationName is the relation whose execution plans has to be returned

```
class = LOAD '/PigData/class.csv'
        USING PigStorage(',')
        AS (ID : chararray, Name : chararray, Gender : chararray, Age : int, Height : float, Weight : float);

EXPLAIN class;
```

```
#-----------------------------------------------
# New Logical Plan:
#-----------------------------------------------
class: (Name: LOStore Schema: ID#361:chararray,Name#362:chararray,Gender#363:chararray,Age#364:int,Height#365:float,Weight#366:float)
|
|---class: (Name: LOLoad Schema:
ID#361:chararray,Name#362:chararray,Gender#363:chararray,Age#364:int,Height#365:float,Weight#366:float)RequiredFields:null
#-----------------------------------------------
# Physical Plan:
#-----------------------------------------------
class: Store(fakefile:org.apache.pig.builtin.PigStorage) - scope-60
|
|---class: Load(/PigOutput/binStorage:BinStorage) - scope-59

2017-12-02 01:29:09,630 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation
threshold: 100 optimistic? false
2017-12-02 01:29:09,630 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size
before optimization: 1
2017-12-02 01:29:09,630 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after
optimization: 1
#-----------------------------------------------
# Map Reduce Plan
#-----------------------------------------------
MapReduce node scope-61
Map Plan
class: Store(fakefile:org.apache.pig.builtin.PigStorage) - scope-60
|
|---class: Load(/PigOutput/binStorage:BinStorage) - scope-59--------
Global sort: false
```

ILLUSTRATE

SYNTAX : ILLUSTRATE RelationName;
where, RelationName is the relation whose step by step execution plans has to be returned

```
class = LOAD '/PigData/class.csv'
        USING PigStorage(',')
        AS (ID : chararray, Name : chararray, Gender : chararray, Age : int, Height : float, Weight : float);

ILLUSTRATE class;
```

OUTPUT

```
2017-12-02 01:35:02,170 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-12-02 01:35:02,173 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-02 01:35:02,173 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
2017-12-02 01:35:02,200 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to hadoop file system at:
hdfs://localhost:9000
2017-12-02 01:35:02,201 [main] INFO  org.apache.pig.backend.hadoop.executionengine.HExecutionEngine - Connecting to map-reduce job tracker
at: localhost:9001
2017-12-02 01:35:02,217 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-12-02 01:35:02,218 [main] INFO  org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer -
{RULES_ENABLED=[ConstantCalculator, LoadTypeCastInserter, PredicatePushdownOptimizer, StreamTypeCastInserter],
RULES_DISABLED=[AddForEach, ColumnMapKeyPrune, GroupByConstParallelSetter, LimitOptimizer, MergeFilter, MergeForEach,
PartitionFilterOptimizer, PushDownForEachFlatten, PushUpFilter, SplitFilter]}
2017-12-02 01:35:02,226 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation
threshold: 100 optimistic? false
2017-12-02 01:35:02,227 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size
before optimization: 1
2017-12-02 01:35:02,227 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after
optimization: 1
2017-12-02 01:35:02,227 [main] INFO  org.apache.pig.tools.pigstats.mapreduce.MRScriptState - Pig script settings are added to the job
2017-12-02 01:35:02,227 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler -
mapred.job.reduce.markreset.buffer.percent is not set, set to default 0.3
2017-12-02 01:35:02,236 [main] INFO  org.apache.pig.impl.util.SpillableMemoryManager - Selected heap (PS Old Gen) of size 699400192 to
monitor. collectionUsageThreshold = 489580128, usageThreshold = 489580128
2017-12-02 01:35:02,236 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-12-02 01:35:02,238 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.PigMapOnly$Map - Aliases being
processed per job phase (AliasName[line,offset]): M: class[14,8] C:  R:
2017-12-02 01:35:02,238 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-12-02 01:35:02,240 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-02 01:35:02,240 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
2017-12-02 01:35:02,248 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation
threshold: 100 optimistic? false
2017-12-02 01:35:02,248 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size
before optimization: 1
2017-12-02 01:35:02,248 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after
optimization: 1
2017-12-02 01:35:02,249 [main] INFO  org.apache.pig.tools.pigstats.mapreduce.MRScriptState - Pig script settings are added to the job
2017-12-02 01:35:02,249 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler -
mapred.job.reduce.markreset.buffer.percent is not set, set to default 0.3
2017-12-02 01:35:02,256 [main] INFO  org.apache.pig.impl.util.SpillableMemoryManager - Selected heap (PS Old Gen) of size 699400192 to
monitor. collectionUsageThreshold = 489580128, usageThreshold = 489580128
2017-12-02 01:35:02,256 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-12-02 01:35:02,259 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.PigMapOnly$Map - Aliases being
processed per job phase (AliasName[line,offset]): M: class[14,8] C:  R:
2017-12-02 01:35:02,261 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation
threshold: 100 optimistic? false
2017-12-02 01:35:02,262 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size
before optimization: 1
2017-12-02 01:35:02,262 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after
optimization: 1
2017-12-02 01:35:02,262 [main] INFO  org.apache.pig.tools.pigstats.mapreduce.MRScriptState - Pig script settings are added to the job
2017-12-02 01:35:02,262 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler -
mapred.job.reduce.markreset.buffer.percent is not set, set to default 0.3
2017-12-02 01:35:02,268 [main] INFO  org.apache.pig.impl.util.SpillableMemoryManager - Selected heap (PS Old Gen) of size 699400192 to
monitor. collectionUsageThreshold = 489580128, usageThreshold = 489580128
2017-12-02 01:35:02,268 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-12-02 01:35:02,271 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.PigMapOnly$Map - Aliases being
processed per job phase (AliasName[line,offset]): M: class[14,8] C:  R:
2017-12-02 01:35:02,272 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation
threshold: 100 optimistic? false
2017-12-02 01:35:02,272 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size
before optimization: 1
2017-12-02 01:35:02,272 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after
optimization: 1
2017-12-02 01:35:02,273 [main] INFO  org.apache.pig.tools.pigstats.mapreduce.MRScriptState - Pig script settings are added to the job
```

2017-12-02 01:35:02,273 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - mapred.job.reduce.markreset.buffer.percent is not set, set to default 0.3
2017-12-02 01:35:02,278 [main] INFO org.apache.pig.impl.util.SpillableMemoryManager - Selected heap (PS Old Gen) of size 699400192 to monitor. collectionUsageThreshold = 489580128, usageThreshold = 489580128
2017-12-02 01:35:02,278 [main] WARN org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-12-02 01:35:02,281 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.PigMapOnly$Map - Aliases being processed per job phase (AliasName[line,offset]): M: class[14,8] C: R:
2017-12-02 01:35:02,282 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? false
2017-12-02 01:35:02,282 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size before optimization: 1
2017-12-02 01:35:02,282 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after optimization: 1
2017-12-02 01:35:02,282 [main] INFO org.apache.pig.tools.pigstats.mapreduce.MRScriptState - Pig script settings are added to the job
2017-12-02 01:35:02,283 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - mapred.job.reduce.markreset.buffer.percent is not set, set to default 0.3
2017-12-02 01:35:02,289 [main] INFO org.apache.pig.impl.util.SpillableMemoryManager - Selected heap (PS Old Gen) of size 699400192 to monitor. collectionUsageThreshold = 489580128, usageThreshold = 489580128
2017-12-02 01:35:02,289 [main] WARN org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-12-02 01:35:02,295 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.PigMapOnly$Map - Aliases being processed per job phase (AliasName[line,offset]): M: class[14,8] C: R:
(J44,Jeffrey,M,13,62.5,84.0)
2017-12-02 01:35:02,299 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? false
2017-12-02 01:35:02,299 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size before optimization: 1
2017-12-02 01:35:02,299 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after optimization: 1
2017-12-02 01:35:02,299 [main] INFO org.apache.pig.tools.pigstats.mapreduce.MRScriptState - Pig script settings are added to the job
2017-12-02 01:35:02,299 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - mapred.job.reduce.markreset.buffer.percent is not set, set to default 0.3
2017-12-02 01:35:02,303 [main] INFO org.apache.pig.impl.util.SpillableMemoryManager - Selected heap (PS Old Gen) of size 699400192 to monitor. collectionUsageThreshold = 489580128, usageThreshold = 489580128
2017-12-02 01:35:02,303 [main] WARN org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-12-02 01:35:02,306 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.PigMapOnly$Map - Aliases being processed per job phase (AliasName[line,offset]): M: class[14,8] C: R:
2017-12-02 01:35:02,306 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? false
2017-12-02 01:35:02,307 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size before optimization: 1
2017-12-02 01:35:02,307 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after optimization: 1
2017-12-02 01:35:02,314 [main] INFO org.apache.pig.tools.pigstats.mapreduce.MRScriptState - Pig script settings are added to the job
2017-12-02 01:35:02,314 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - mapred.job.reduce.markreset.buffer.percent is not set, set to default 0.3
2017-12-02 01:35:02,319 [main] INFO org.apache.pig.impl.util.SpillableMemoryManager - Selected heap (PS Old Gen) of size 699400192 to monitor. collectionUsageThreshold = 489580128, usageThreshold = 489580128
2017-12-02 01:35:02,319 [main] WARN org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-12-02 01:35:02,323 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.PigMapOnly$Map - Aliases being processed per job phase (AliasName[line,offset]): M: class[14,8] C: R:
2017-12-02 01:35:02,324 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? false
2017-12-02 01:35:02,324 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size before optimization: 1
2017-12-02 01:35:02,324 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after optimization: 1
2017-12-02 01:35:02,325 [main] INFO org.apache.pig.tools.pigstats.mapreduce.MRScriptState - Pig script settings are added to the job
2017-12-02 01:35:02,325 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - mapred.job.reduce.markreset.buffer.percent is not set, set to default 0.3
2017-12-02 01:35:02,331 [main] INFO org.apache.pig.impl.util.SpillableMemoryManager - Selected heap (PS Old Gen) of size 699400192 to monitor. collectionUsageThreshold = 489580128, usageThreshold = 489580128
2017-12-02 01:35:02,331 [main] WARN org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-12-02 01:35:02,333 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.PigMapOnly$Map - Aliases being processed per job phase (AliasName[line,offset]): M: class[14,8] C: R:
2017-12-02 01:35:02,334 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimistic? false
2017-12-02 01:35:02,334 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size before optimization: 1
2017-12-02 01:35:02,334 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after optimization: 1
2017-12-02 01:35:02,335 [main] INFO org.apache.pig.tools.pigstats.mapreduce.MRScriptState - Pig script settings are added to the job
2017-12-02 01:35:02,335 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - mapred.job.reduce.markreset.buffer.percent is not set, set to default 0.3
2017-12-02 01:35:02,339 [main] INFO org.apache.pig.impl.util.SpillableMemoryManager - Selected heap (PS Old Gen) of size 699400192 to monitor. collectionUsageThreshold = 489580128, usageThreshold = 489580128
2017-12-02 01:35:02,339 [main] WARN org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2017-12-02 01:35:02,342 [main] INFO org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.PigMapOnly$Map - Aliases being processed per job phase (AliasName[line,offset]): M: class[14,8] C: R:

class	ID:chararray	Name:chararray	Gender:chararray	Age:int	Height:float	Weight:float
	J44	Jeffrey	M	13	62.5	84.0

LIMIT

SYNTAX : New Rel_name = LIMIT RelationName <No_Of_Records_to_be_displayed_OR_Stored>;
where, Relation name is the relation whose output has to be restricted

```
class = LOAD '/PigData/class.csv'
        USING PigStorage(',')
        AS (ID : chararray, Name : chararray, Gender : chararray, Age : int, Height : float, Weight : float);

class2 =  LIMIT class 2;

dump class2;
```

Output:
(A11,Alfred,M,14,69.0,112.5)
(A22,Alice,F,13,56.5,84.0)

FILTER OPERATORS

DISTINCT

SYNTAX : Output_Rel = DISTINCT RelationName;
where, RelationName is the relation whose duplicate ccontent has to be removed

```
class = LOAD '/PigData/duplicatesclass.csv'
        USING PigStorage(',')
        AS (ID : chararray, Name : chararray, Gender : chararray, Age : int, Height : float, Weight : float);

distinctClass = DISTINCT class;

rel = LIMIT distinctClass 5;

dump distinctClass;
```

Output:
(A11,Alfred,M,14,69.0,112.5)
(A22,Alice,F,13,56.5,84.0)
(B11,Barbara,F,13,65.3,98.0)
(C11,Carol,F,14,62.8,102.5)
(H11,Henry,M,14,63.5,102.5)

FOREACH

....SchemaBased Relation
SYNTAX : New_Rel_name=FOREACH Old_Rel_Name GENERATE Key/Key1,Key2...keyN;

....SchemaLess Relation
SYNTAX : New_Rel_name=FOREACH Old_Rel_Name GENERATE PositionParameters i:e $0,$1...;

where **old_Rel_Name** : is the relation whose each row has to be read
key : It is the column name by which the records have to be filtered

Case 1. FOREACH FOR A SCHEMA RELATION

```
baseball = LOAD '/PigData/baseball'
           AS (Name : chararray, Team : chararray, Position : bag{}, Bat : map[]);
```

```
Player_Position = FOREACH baseball GENERATE Name, Position;

rel = LIMIT Player_Position 3;

DUMP  rel;
```

Output:
(Jorge Posada,{(Catcher),(Designated_hitter)})
(Landon Powell,{(Catcher),(First_baseman)})
(Martín Prado,{(Second_baseman),(Infielder),(Left_fielder)})

Case 2. FOREACH FOR A SCHEMALESS RELATION

```
baseball = LOAD '/PigData/baseball' ;
Param1_Param2 = FOREACH baseball GENERATE $0, $1;
rel = LIMIT Param1_Param2 3;

dump rel;
```

Output:
(Jorge Posada,{(Catcher),(Designated_hitter)})
(Landon Powell,{(Catcher),(First_baseman)})
(Martín Prado,{(Second_baseman),(Infielder),(Left_fielder)})

Case 3. FOREACH FOR ACCESSING COLUMN OF COMPLEX TYPE ..... TUPLES

```
name_subjects = FOREACH tupleSample
                        GENERATE t.name, t.Subjects;

rel = LIMIT name_subjects 2;

DUMP rel;
```

Output:
('Ron',{(English),(Maths)})
('Amit',{(Science),(Maths)})

Case 4. FOREACH FOR ACCESSING COLUMN OF COMPLEX TYPE ..... MAP

```
baseball = LOAD '/PigData/baseball'
                AS (Name : chararray, Team : chararray, Position : bag{}, Bat : map[]);

player_games_hits = FOREACH baseball generate Name,Bat#'games', Bat#'hits';

rel = LIMIT  player_games_hits 4;

DUMP rel;
```

Output:
(Jorge Posada,1594,1488)
(Landon Powell,46,32)
(Martín Prado,258,239)
(David Price,28,1)

Case 5. ACCESSING COLUMNS WITH ADDITIONAL ARITHMETIC OPERATION

```
class = LOAD '/PigData/class.csv'
        USING PigStorage(',')
        AS (ID : chararray, Name : chararray, Gender : chararray, Age : int, Height : float, Weight : float);

StuName_Age_AddAge_2 = FOREACH class GENERATE Name, Age, Age + 2;

DESCRIBE StuName_Age_AddAge_2;

StuName_Age_AddAge_2: {Name: chararray,Age: int,int}
```

-- Note Output : Age + 2 isn't assigned field name. By default based on operand of arithmetic expression type is given the value, but not the field name

-- However we can have the field name using following,

```
StuName_Age_AddAge_2 = FOREACH class generate Name, Age, Age + 2 AS AddedAge;
DESCRIBE StuName_Age_AddAge_2;

StuName_Age_AddAge_2: {Name: chararray,Age: int,AddedAge: int}

rel =  LIMIT StuName_Age_AddAge_2 5;

DUMP rel;

Output:
(Alfred,14,16)
(Alice,13,15)
(Barbara,13,15)
(Carol,14,16)
(Henry,14,16)
```

FILTER

SYNTAX : New_Rel_name=FILTER Old_Rel_Name BY Condition;

where, **Old_Rel_Name** : relation whose records have to filtered based on some condition
condition : condition using which records have to be filterd

```
class = LOAD '/PigData/class.csv'
        USING PigStorage(',')
        AS (ID : chararray, Name : chararray, Gender : chararray, Age : int, Height : float, Weight : float);
```

Case 1. Filter by single condition

```
 FilterBy_Gender = FILTER class BY Gender == 'F';

rel = LIMIT FilterBy_Gender

DUMP rel;
Output:
(A22,Alice,F,13,56.5,84.0)
(B11,Barbara,F,13,65.3,98.0)
(C11,Carol,F,14,62.8,102.5)
(J22,Jane,F,12,59.8,84.5)
(J33,Janet,F,15,62.5,112.5)
```

Case 2. Filter by multiple condition

SYNTAX : New_Rel_name=FILTER Old_Rel_Name BY Condition1 OR|AND|NOT Condition2.....;

```
FilterBy_Gender_Age = FILTER class BY Gender == 'F' AND Age > 15;

rel = LIMIT FilterBy_Gender_Age;
```

```
DUMP rel;
```

Output:
(J33,Janet,F,15,62.5,112.5)
(M11,Mary,F,15,66.5,112.0)

Case 3. Filter using regular expression

```
SYNTAX : New_Rel_name=FILTER Old_Rel_Name BY Key matches 'expression';

Names_Starting_with_P = FILTER class BY Name matches 'P.*';

DUMP Names_Starting_with_P;
```

Output:
(P11,Philip,M,16,72.0,150.0)

GROUPING ANG JOINING OPERATORS

GROUP

```
SYNTAX : New_Rel_name=GROUP Old_Rel_Name BY Key/(Key1,Key2,Key3.....KeyN);
where,  Old_Rel_Name :
          Key :

sales = LOAD '/PigData/sales.txt'
          USING PigStorage(',')
          AS (region chararray, product chararray, subsidiary chararray, stores chararray, sales int, inventory int, returns
int)
```

Case 1. Group By using single column

```
Group_By_Product = GROUP sales BY product;
```

Case 2. Group By using multipe column

```
Group_By_subsidiary_Product = GROUP sales BY (subsidiary,product);
```

Case 3. Group By All

```
Group_By_All = GROUP sales ALL;
```

JOIN

```
customers = LOAD ' /PigData/joinDatasets/customer.txt'
                USING PigStorage(',')
                AS (id:int, name:chararray, age:int, address:chararray, salary:int);

orders = LOAD '/PigData/joinDatasets/orders.txt'
           USING PigStorage(',')
           AS (oid:int, date:chararray, customer_id:int, amount:int);
```

Case 1. INNER JOIN

```
SYNTAX : Output_Rel = JOIN Relation_1 by KEY, Relation_2 by KEY;

Cust_Order_Join =  JOIN customers by id, orders by customer_id;

DUMP  Cust_Order_Join;
```

Output:
(2,Khilan,25,Delhi,150,101,2009-11-20 00:00:00,2,1560)
(3,kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)
(3,kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)

Case 2. SELF JOIN

SYNTAX : Output_Rel = JOIN Relation_1 by KEY, Relation_2 by KEY;

Note : Although self join is join on same relation , Relation_1, Relation_2, are different relations with same schema definition. Using the same name for both the relation throws "Pig does not accept same alias as input for JOIN operation" error

```
customers = LOAD ' /PigData/joinDatasets/customer.txt'
            USING PigStorage(',')
            AS (id:int, name:chararray, age:int, address:chararray, salary:int);

customers1 =  customers;

Cust_Cust1_SelfJoin =  JOIN customers by id, customers1 by id;
```

Partial Output:
(1,Ramesh,32,Ahmedabad,2000,1,Ramesh,32,Ahmedabad,2000)
(2,Khilan,25,Delhi,150,2,Khilan,25,Delhi,150)
(3,kaushik,23,Kota,2000,3,kaushik,23,Kota,2000)
(4,Chaitali,25,Mumbai,6500,4,Chaitali,25,Mumbai,6500)
(5,Hardik,27,Bhopal,8500,5,Hardik,27,Bhopal,8500)

Case 3. OUTER LEFT JOIN

SYNTAX : Output_Rel = JOIN Relation_1 by KEY LEFT OUTER, Relation_2 by KEY;

```
Cust_OUTER_LEFT_JOIN_order  = JOIN customers by id LEFT OUTER, orders by customer_id;
```

OUTPUT:
(1,Ramesh,32,Ahmedabad,2000,,,,)
(2,Khilan,25,Delhi,150,101,2009-11-20 00:00:00,2,1560)
(3,kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)
(3,kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)
(5,Hardik,27,Bhopal,8500,,,,)
(6,Komal,22,MP,4500,,,,)
(7,Muffy,24,Indore,10000,,,,)

Case 4. OUTER RIGHT JOIN

SYNTAX : Output_Rel = JOIN Relation_1 by KEY RIGHT OUTER, Relation_2 by KEY;

```
Cust_OUTER_RIGHT_JOIN_order  = JOIN customers by id RIGHT OUTER, orders by customer_id;
```

Output:
(2,Khilan,25,Delhi,150,101,2009-11-20 00:00:00,2,1560)
(3,kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)
(3,kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)

Case 5. OUTER FULL JOIN

SYNTAX : Output_Rel = JOIN Relation_1 by KEY FULL OUTER, Relation_2 by KEY;

```
Cust_OUTER_FULL_JOIN_order  = JOIN customers by id FULL OUTER, orders by customer_id;
```

Output:
(1,Ramesh,32,Ahmedabad,2000,,,,)
(2,Khilan,25,Delhi,150,101,2009-11-20 00:00:00,2,1560)
(3,kaushik,23,Kota,2000,102,2009-10-08 00:00:00,3,3000)
(3,kaushik,23,Kota,2000,100,2009-10-08 00:00:00,3,1500)
(4,Chaitali,25,Mumbai,6500,103,2008-05-20 00:00:00,4,2060)
(5,Hardik,27,Bhopal,8500,,,,)
(6,Komal,22,MP,4500,,,,)
(7,Muffy,24,Indore,10000,,,,)

Case 5. JOIN WITH MULTIPLE KEYS

SYNTAX : Output_Rel = JOIN Relation_1 by (KEY1,KEY2...), Relation_2 by (KEY1,KEY2.....);

```
Employee = LOAD '/PigData/joinDatasets/employee.txt'
                USING PigStorage(',')
                AS (id:int, firstname:chararray, lastname:chararray, age:int, designation:chararray, jobid:int);

Employee_Contact = LOAD '/PigData/joinDatasets/employeeContact.txt'
                           USING PigStorage(',')
                           AS (id:int, phone:chararray, email:chararray, city:chararray, jobid:int);

EMPLOYEE_EMP_CONTACT = JOIN Employee
                                    BY (id,jobid), Employee_Contact BY (id,jobid);
```

Output:
(1,Rajiv,Reddy,21,programmer,3,1,9848022337,Rajiv@gmail.com,Hyderabad,3)
(2,siddarth,Battacharya,22,programmer,3,2,9848022338,siddarth@gmail.com,Kolkata,3)
(3,Rajesh,Khanna,22,programmer,3,3,9848022339,Rajesh@gmail.com,Delhi,3)
(4,Preethi,Agarwal,21,programmer,3,4,9848022330,Preethi@gmail.com,Pune,3)
(5,Trupthi,Mohanthy,23,programmer,3,5,9848022336,Trupthi@gmail.com,Bhuwaneshwar,3)
(6,Archana,Mishra,23,programmer,3,6,9848022335,Archana@gmail.com,Chennai,3)
(7,Komal,Nayak,24,teamlead,2,7,9848022334,Komal@gmail.com,trivendram,2)
(8,Bharathi,Nambiayar,24,manager,1,8,9848022333,Bharathi@gmail.com,Chennai,1)

COGROUP

```
emp_sales = LOAD '/PigData/empSales.txt' USING PigStorage(',')
            AS(sno:int, name:chararray, age:int, salary:int, dept:chararray);

emp_bonus = LOAD '/PigData/empBonus.txt' USING PigStorage(',')
             AS (sno:int, name:chararray, age:int, bonus:int, dept:chararray);

cogroup_data = COGROUP emp_sales by sno, emp_bonus by sno;
```

Output:
(1,{(1,Robin,22,25000,sales)},{(1,Robin,22,25000,sales)})
(2,{(2,BOB,23,30000,sales)},{(2,Jaya,23,20000,admin)})
(3,{(3,Maya,23,25000,sales)},{(3,Maya,23,25000,sales)})
(4,{(4,Sara,25,40000,sales)},{(4,Alia,25,50000,admin)})
(5,{(5,David,23,45000,sales)},{(5,David,23,45000,sales)})
(6,{(6,Maggy,22,35000,sales)},{(6,Omar,30,30000,admin)})

SORTING OPERATORS

ORDER

SYNTAX : New_Rel_name=ORDER Old_Rel_Name BY Key;
SYNTAX : New_Rel_name=ORDER Old_Rel_Name BY Key DESC;

```
class = LOAD '/PigData/class.csv'
        USING PigStorage(',')
        AS (ID : chararray, Name : chararray, Gender : chararray, Age : int, Height : float, Weight : float);
```

Note : 1. default ascending numeric order for integer and lexical order for strings
2. Order By Ky which is of complex type is not allowed in Pig i:e Neither for Map, tuple, bag
A = LOAD '/PigData' ;
B = ORDER A BY y; -- this is not allowed because y is a complex type
B = ORDER A BY y#'id'; -- this is not allowed because y#'id' is an expression

3. Pig currently supports ordering on fields with simple types or by tuple designator (*). You cannot order on fields with complex types or by expressions.

Case 1. ORDER by single key for type integer/float

```
OrderByAge = ORDER class BY Age;
Partial Output:
(J66,Joyce,F,11,51.3,50.5)
(T11,Thomas,M,11,57.5,85.0)
(J55,John,M,12,59.0,99.5)
(L11,Louise,F,12,56.3,77.0)
(R11,Robert,M,12,64.8,128.0)
(J11,James,M,12,57.3,83.0)
(J22,Jane,F,12,59.8,84.5)
(J44,Jeffrey,M,13,62.5,84.0)
(A22,Alice,F,13,56.5,84.0)
(B11,Barbara,F,13,65.3,98.0)
```

Case 2. ORDER by single key for type chararray

```
OrderByName = ORDER class BY Name;
Partial Output:
(A11,Alfred,M,14,69.0,112.5)
(A11,Alfred,M,14,69.0,112.5)
(A11,Alfred,M,14,69.0,112.5)
(A22,Alice,F,13,56.5,84.0)
(B11,Barbara,F,13,65.3,98.0)
(C11,Carol,F,14,62.8,102.5)
(H11,Henry,M,14,63.5,102.5)
(J11,James,M,12,57.3,83.0)
(J22,Jane,F,12,59.8,84.5)
(J33,Janet,F,15,62.5,112.5)
(J44,Jeffrey,M,13,62.5,84.0)
(J55,John,M,12,59.0,99.5)
```

Case 3. ORDER by single key for type integers(Descending Order)

```
OrderByAge = ORDER class BY Age DESC;
Partial Output:
(P11,Philip,M,16,72.0,150.0)
(R22,Ronald,M,15,67.0,133.0)
(M11,Mary,F,15,66.5,112.0)
(W11,William,M,15,66.5,112.0)
(J33,Janet,F,15,62.5,112.5)
(A11,Alfred,M,14,69.0,112.5)
(A11,Alfred,M,14,69.0,112.5)
(A11,Alfred,M,14,69.0,112.5)
(C11,Carol,F,14,62.8,102.5)
```

Case 4. ORDER by single key for type chararray (Descending Order)

```
OrderByName = ORDER class BY Name DESC;
```

Partial Output:

```
(W11,William,M,15,66.5,112.0)
(T11,Thomas,M,11,57.5,85.0)
(R22,Ronald,M,15,67.0,133.0)
(R11,Robert,M,12,64.8,128.0)
(P11,Philip,M,16,72.0,150.0)
(M11,Mary,F,15,66.5,112.0)
(L11,Louise,F,12,56.3,77.0)
(J77,Judy,F,14,64.3,90.0)
(J66,Joyce,F,11,51.3,50.5)
```

Case 5. ORDER by multiple keys

```
SYNTAX : New_Rel_name=ORDER Old_Rel_Name BY Key1,key2....keyN;

OrderByNameAge = ORDER class BY Name,Age;
```

Partial Output:

```
(A11,Alfred,M,14,69.0,112.5)
(A11,Alfred,M,14,69.0,112.5)
(A11,Alfred,M,14,69.0,112.5)
(A22,Alice,F,13,56.5,84.0)
(B11,Barbara,F,13,65.3,98.0)
(C11,Carol,F,14,62.8,102.5)
(H11,Henry,M,14,63.5,102.5)
(J11,James,M,12,57.3,83.0)
(J22,Jane,F,12,59.8,84.5)
(J33,Janet,F,15,62.5,112.5)
(J44,Jeffrey,M,13,62.5,84.0)
(J55,John,M,12,59.0,99.5)
(J66,Joyce,F,11,51.3,50.5)
(J77,Judy,F,14,64.3,90.0)
```

Case 6. ORDER by multiple keys

```
baseball = LOAD '/PigData/baseball'
        AS (Name : chararray, Team : chararray, Position : bag{}, Bat : map[]);

OrderByNameGrandSlams= ORDER baseball BY Name, Bat#grand_slams;
```

Doesn't work with complex types;

FLATTEN

```
baseball = LOAD '/PigData/baseball'
        AS (Name : chararray, Team : chararray, Position : bag{}, Bat : map[]);

flattenuse= FOREACH baseball GENERATE Name,flatten(Position);
rel = LIMIT flattenuse 15;
dump rel;
```

OUTPUT:

```
(David Price,Pitcher)
(David Price,Starting_pitcher)
(Jason Pridie,Outfielder)
(Jason Pridie,Left_fielder)
(Jason Pridie,Center_fielder)
(Jorge Posada,Catcher)
(Jorge Posada,Designated_hitter)
```

(Albert Pujols,Third_baseman)
(Landon Powell,Catcher)
(Landon Powell,First_baseman)
(Martín Prado,Infielder)
(Martín Prado,Left_fielder)
(Martín Prado,Second_baseman)
(Scott Proctor,Pitcher)
(Scott Proctor,Relief_pitcher)

COMBINING AND SPLITTING OPERATORS

SPLIT

SYNTAX : SPLIT Relation1_name INTO Relation2_name IF (condition1), Relation2_name (condition2):

SPLIT class INTO MaleStu IF (Gender == 'M'), FemaleStu IF (Gender == 'F');
store rel INTO '/PigData/MaleStu';

Partial Output:
(A11,Alfred,M,14,69.0,112.5)
(R11,Robert,M,12,64.8,128.0)
(R22,Ronald,M,15,67.0,133.0)
(T11,Thomas,M,11,57.5,85.0)
(W11,William,M,15,66.5,112.0)

store rel INTO '/PigData/FemaleStu';

Partial Output:

(A22,Alice,F,13,56.5,84.0)
(B11,Barbara,F,13,65.3,98.0)
(C11,Carol,F,14,62.8,102.5)
(J22,Jane,F,12,59.8,84.5)
(J33,Janet,F,15,62.5,112.5)

UNION

SYNTAX : Relation_name3 = UNION Relation_name1, Relation_name2;

MaleStu = LOAD '/PigData/MaleStu' AS(ID: chararray,Name: chararray,Gender: chararray,Age: int,Height: float,Weight: float);
FemaleStu = LOAD '/PigData/FemaleStu' AS(ID: chararray,Name: chararray,Gender: chararray,Age: int,Height: float,Weight: float);

Stu = UNION MaleStu,FemaleStu;

PartialOutput:
(A11,Alfred,M,14,69.0,112.5)
(H11,Henry,M,14,63.5,102.5)
(R22,Ronald,M,15,67.0,133.0)
(T11,Thomas,M,11,57.5,85.0)
(W11,William,M,15,66.5,112.0)
(A22,Alice,F,13,56.5,84.0)
(B11,Barbara,F,13,65.3,98.0)
(C11,Carol,F,14,62.8,102.5)
(J22,Jane,F,12,59.8,84.5)
(J33,Janet,F,15,62.5,112.5)
(J66,Joyce,F,11,51.3,50.5)

BUILTIN FUNCTIONS

Eval Functions

Given below is the list of **eval** functions provided by Apache Pig.

S.N.	Function & Description
1	**AVG()** To compute the average of the numerical values within a bag.
2	**BagToString()** To concatenate the elements of a bag into a string. While concatenating, we can place a delimiter between these values (optional).
3	**CONCAT()** To concatenate two or more expressions of same type.
4	**COUNT()** To get the number of elements in a bag, while counting the number of tuples in a bag.
5	**COUNT_STAR()** It is similar to the **COUNT()** function. It is used to get the number of elements in a bag.
6	**DIFF()** To compare two bags (fields) in a tuple.
7	**IsEmpty()** To check if a bag or map is empty.
8	**MAX()** To calculate the highest value for a column (numeric values or chararrays) in a single-column bag.
9	**MIN()** To get the minimum (lowest) value (numeric or chararray) for a certain column in a single-column bag.
10	**PluckTuple()** Using the Pig Latin **PluckTuple()** function, we can define a string Prefix and filter the columns in a relation that begin with the given prefix.
11	**SIZE()** To compute the number of elements based on any Pig data type.
12	**SUBTRACT()** To subtract two bags. It takes two bags as inputs and returns a bag which contains the tuples of the first bag that are not in the second bag.
13	**SUM()** To get the total of the numeric values of a column in a single-column bag.
14	**TOKENIZE()** To split a string (which contains a group of words) in a single tuple and return a bag which contains the output of the split operation.

AVG()

SYNTAX : AVG(expression)
where, expression : Any expression whose result is a bag. The elements of the bag should be data type int, long, float, or double.

```
sales = LOAD '/PigData/sales1.txt'
        USING PigStorage(',')
        AS (region:chararray,product:chararray,subsidiary:chararray,stores:chararray,sales:int,inventory:int,returns:int);

groupByProduct = group sales by product;

avgSaleOfProduct = FOREACH groupByProduct GENERATE group, AVG(sales.sales);

dump avgSaleOfProduct;
```

Output:

```
(Boot,36443.61538461538)
(Sandal,12064.204081632653)
(Slipper,95350.59615384616)
(Sport Shoe,11601.078431372549)
(Men's Dress,89971.52)
(Men's Casual,122014.4)
```

(Women's Dress,95834.64705882352)
(Women's Casual,72924.93333333333)

CONCAT()

SYNTAX : CONCAT (expression, expression)
where, expression : Any expression, The result values of the two expressions must have identical types.

```
Employee = LOAD '/PigData/joinDatasets/employee.txt'
                USING PigStorage(',')
                AS (id:int, firstname:chararray, lastname:chararray, age:int, designation:chararray, jobid:int);

ConcatFirstAndLastName = FOREACH Employee GENERATE CONCAT(firstname, ' ', lastname);

rel = LIMIT ConcatFirstAndLastName  5
DUMP rel;
```

OUTPUT:

(Rajiv Reddy)
(siddarth Battacharya)
(Rajesh Khanna)
(Preethi Agarwal)
(Trupthi Mohanthy)

COUNT()
SYNTAX : COUNT (expression)
where, expression : An expression with data type bag. Ignores NULL values

```
class = LOAD '/PigData/class.csv'
          USING PigStorage(',')
          AS (ID : chararray, Name : chararray, Gender : chararray, Age : int, Height : float, Weight : float);

GroupByGender =  GROUP class BY Gender;
CountOfMalesfemales =  FOREACH GroupByGender GENERATE group,COUNT(class);
DUMP CountOfMalesfemales;
```

OUTPUT:

(F,9)
(M,12)

COUNT_STAR()

SYNTAX : COUNT_STAR (expression)
where, expression : An expression with data type bag. Accepts NULL values

DIFF()
SYNTAX : DIFF (expression, expression)
where, expression : Any expression, The result values of the two expressions must have identical types.

If the bags match, an empty bag is returned. If the fields are not bags then they will be wrapped in tuples and returned in a bag if they do not match, or an empty bag will be returned if the two records match

```
emp_sales = LOAD '/PigData/empSales.txt' USING PigStorage(',')
                AS(sno:int, name:chararray, age:int, salary:int, dept:chararray);

emp_bonus = LOAD '/PigData/empBonus.txt' USING PigStorage(',')
                 AS (sno:int, name:chararray, age:int, bonus:int, dept:chararray);
```

```
cogroup_data = COGROUP emp_sales by sno, emp_bonus by sno;
```

Output:
```
({})
({(2,BOB,23,30000,sales )})
({})
({(4,Sara,25,40000,sales )})
({})
({(6,Maggy,22,35000,sales)})
```

SUBTRACT()

SYNTAX : SUBTRACT (expression, expression)

where, expression : Any expression, The result values of the two expressions must have identical types.

```
emp_sales = LOAD '/PigData/empSales.txt' USING PigStorage(',')
                AS(sno:int, name:chararray, age:int, salary:int, dept:chararray);

emp_bonus = LOAD '/PigData/empBonus.txt' USING PigStorage(',')
                 AS (sno:int, name:chararray, age:int, bonus:int, dept:chararray);

cogroup_data = COGROUP emp_sales by sno, emp_bonus by sno;

sub_data = FOREACH cogroup_data GENERATE SUBTRACT(emp_sales, emp_bonus);
dump sub_data;
```

Output:
```
({})
({(2,BOB,23,30000,sales )})
({})
({(4,Sara,25,40000,sales )})
({})
({(6,Maggy,22,35000,sales)})
```

MAX()

SYNTAX : MAX(expression)

where, expression : An expression with data types int, long, float, double, or chararray.

```
stu_details = LOAD '/PigData/student_details.txt' USING PigStorage(',') AS (id:int, FirstName:chararray,
LastName:chararray, Age:int, MobileNo:chararray, Place:chararray,TotalMarks:int);
GroupALL= GROUP stu_details ALL;
HighestMarks = FOREACH GroupALL GENERATE MAX(stu_details.TotalMarks);
DUMP HighestMarks;
```

Output:
```
(93)
```

MIN()

SYNTAX : MIN (expression)

where, expression : An expression with data types int, long, float, double, or chararray.

```
stu_details = LOAD '/PigData/student_details.txt' USING PigStorage(',') AS (id:int, FirstName:chararray,
LastName:chararray, Age:int, MobileNo:chararray, Place:chararray,TotalMarks:int);
GroupALL= GROUP stu_details ALL;
LowestMarks = FOREACH GroupALL GENERATE MIN(stu_details.TotalMarks);
DUMP LowestMarks;
```

Output:
```
(72)
```

SIZE()

SYNTAX : SIZE (expression)

where, expression : An expression with any data type.

int, double,float,long	:	Returns 1
chararray	:	Returns number of characters in the array
bytearray	:	Returns number of bytes in the array
tuple	:	Returns number of fields in the tuple
Map	:	Returns number of keyValue pairs in the map
Bag	:	Returns number of tuples in the bag

```
baseball = LOAD '/PigData/baseball'
        AS (Name : chararray, Team : chararray, Position : bag{}, Bat : map[]);

SizeOfTypes = FOREACH baseball GENERATE SIZE(Name), SIZE(Position),SIZE(Bat);
```

Output:

```
(12,2,19)
(13,2,15)
(12,3,18)
(11,2,6)
(12,3,7)
```

SUM()

SYNTAX : SUM (expression)

where, expression : An expression with data types int, long, float, double, or bytearray cast as double.

```
sales = LOAD '/PigData/sales1.txt'
        USING PigStorage(',')
        AS (region:chararray,product:chararray,subsidiary:chararray,stores:chararray,sales:int,inventory:int,returns:int);
groupByProduct = GROUP sales BY product;
dump groupByProduct;
TotalSalePerProduct = FOREACH groupByProduct GENERATE group, SUM(sales.sales);
dump TotalSalePerProduct;
```

Output:

```
(Boot,1895068)
(Sandal,591146)
(Slipper,4958231)
(Sport Shoe,591655)
(Men's Dress,4498576)
(Men's Casual,5490648)
(Women's Dress,4887567)
(Women's Casual,3281622)
```

TOKENIZE()

SYNTAX : TOKENIZE(expression [, 'field_delimiter'])

where, expression : An expression with data type chararray.

'field_delimiter' : An optional field delimiter (in single quotes). If field_delimiter is null or not passed, the following will be used as delimiters: space [ ], double quote ["], coma [,] parenthesis [()], star [*].

```
lines = LOAD '/PigData/red' AS (line:chararray);
words = FOREACH lines GENERATE TOKENIZE(line);
dump words;
```

Output:

```
({(red),(roses),(roses),(are),(red),(red),(apples),(apples),(are),(red)})
({(red),(roses),(roses),(are),(red),(red),(apples),(apples),(are),(red)})
({(red),(roses),(roses),(are),(red),(red),(apples),(apples),(are),(red)})
({(red),(roses),(roses),(are),(red),(red),(apples),(apples),(are),(red)})
```

({(red),(roses),(roses),(are),(red),(red),(apples),(apples),(are),(red)})

ISEMPTY()

SYNTAX : IsEmpty(expression)

```
baseball = LOAD '/PigData/baseball'
            AS (Name : chararray, Team : chararray, Position : bag{}, Bat : map[]);

checkEmpty = FOREACH baseball GENERATE IsEmpty(Position);

rel = LIMIT checkEmpty 5;

dump rel;
```

Output:
(false)
(false)
(false)
(false)
(false)