Apache PIG

**Intro:**

**Apache Pig** 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. The salient 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 provides an engine for processing data flow parallelization on hadoop using a data flow language called Pig Latin.

It uses HDFS for storage and MR for processing.

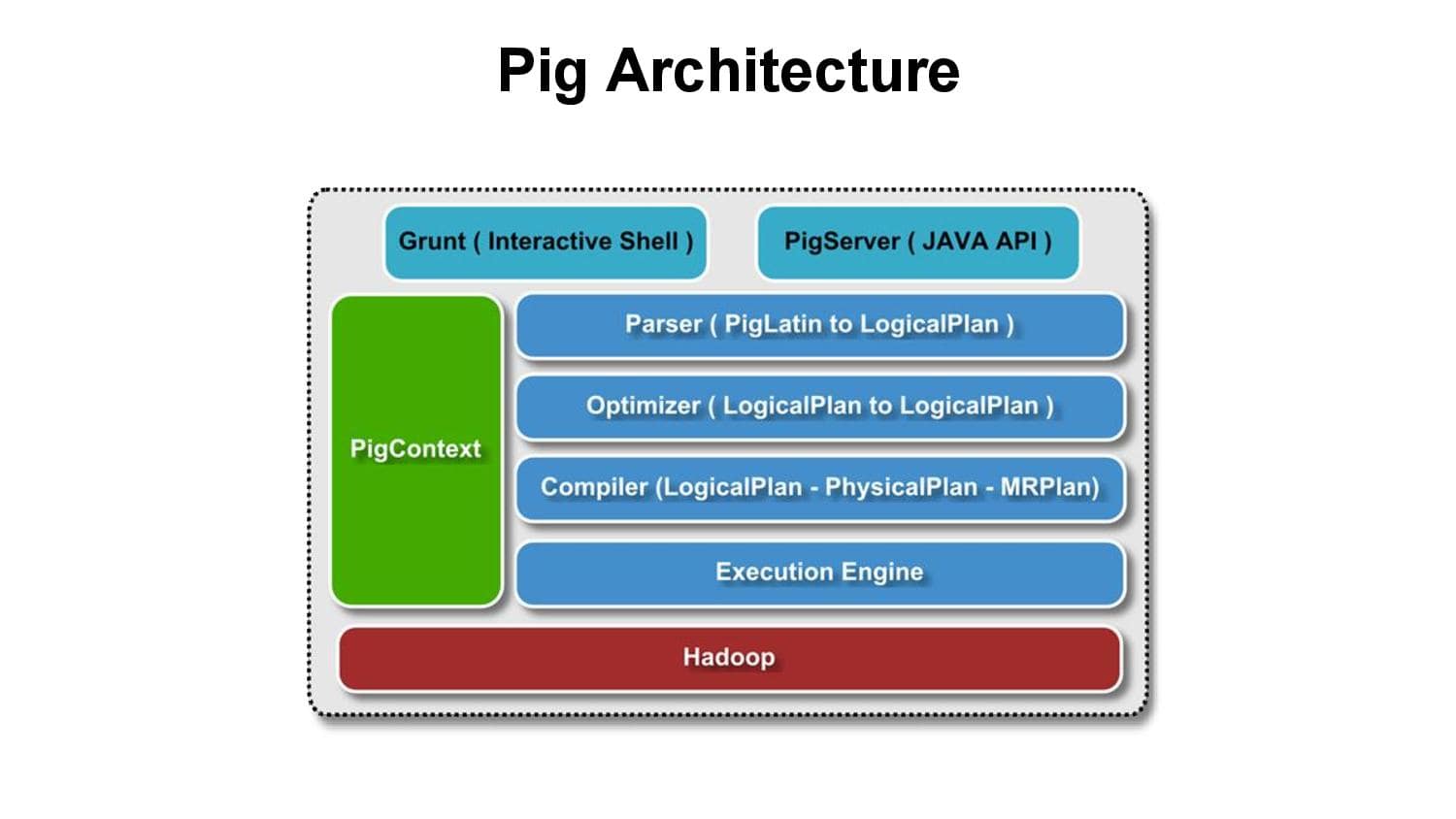
**Why PIG:**

* Layered approach
* Slice and dice the task into smaller chunks and process it in layered fashion.
* Flexible
* With and Without knowing the schema we can process the data.
* Ease of programming
* Just 5% of complexity and code when compared to MR.
* Extensible
* Optimization

**Applications in PIG:**

* ETL
* Raw data analysis (for eg: web log)
* Interactive processing

**Architecture:**



1. **Pig Latin** or **Grunt shell** will be installed in the edge node by which the user will be interacting to the system.

* Pig Latin is the script file in which the pig scripts will be written in layered fashion.
* Grunt shell is a kind of CLI.

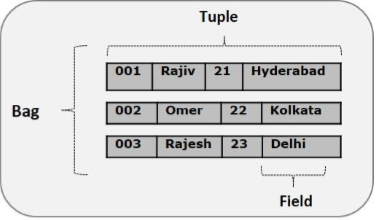
1. **Parser**:

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

1. The logical plan (DAG) is passed to the logical **optimizer**, which carries out the logical optimizations such as projection and pushdown.
2. The **compiler** compiles the optimized logical plan into a series of MapReduce jobs.
3. Finally the MapReduce jobs are submitted to Hadoop in a sorted order. Finally, these MapReduce jobs are **executed** on Hadoop producing the desired results.

**Pig Latin Data Model**

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



**Atom**

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

Example − ‘raja’ or ‘30’

**Tuple**

A record that is formed by an ordered set of fields is known as a tuple, the fields can be of any type. A tuple is similar to a row in a table of RDBMS.

Example − (Raja, 30)

**Bag**

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

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

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

Example − {Raja, 30, {9848022338, [raja@gmail.com,}](mailto:raja@gmail.com,%7d)}

**Map**

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

Example − [name#Raja, age#30]

**Relation**

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

**Let’s see some hands on part in PIG.**

**Installation and Configuration**

# Goto the below path cd /home/hduser/install/

1. Extract the tarfile

tar xvzf pig-0.16.0.tar.gz

1. Rename and move the pig folder

sudo mv pig-0.16.0 /usr/local/pig

as usual we are extracting the tar.gz file and moving it to the pig folder in local.

1. Start the History Server:

mr-jobhistory-daemon.sh start historyserver

we are initiating the job history daemon to get the logs of all activities happening in PIG.

**Workouts:**

1. Run Pig in different modes in Grunt shell

pig -x local

(Pig -x local will make the PIG run on local which will use the local hard disk for storage).

## OR

pig -x mapreduce pig

(Pig -x local will make the PIG run on local which will use the local hard disk for storage).

1. Load, Store and Dump.

game = load '/home/hduser/Datasets/game1.csv' using PigStorage(',');

here we are loading the data from game1.csv into the relation game.

## OR

game = load '/home/hduser/Datasets/game1.csv' using PigStorage(',')

as (name:chararray,team:chararray,position:chararray,weight:int,height:int,age:int);

if we know the structure of the data in the source file, we can define that structure while loading into the relation.

dump game;

fs -rm -r /home/hduser/kanna1/pigworks/game

store game into ‘/home/hduser/kanna1/pigworks/game’;

In PIG only the keywords dump and store are executable.

In the above example only when the dump is issued on game, the relation game will be loaded with data from game1.scv.

Similarly only when store game is issued, its parent relation game will be loaded and then its output will be stored in the given location. (the given location should not exist).

**When issued dump,**

dump game;

*..............*

*.....................*

*2019-04-16 11:29:27,366 [LocalJobRunner Map Task Executor #0] INFO org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter - Saved output of task attempt\_local1884666946\_0001\_m\_000000\_0' to file:/tmp/temp-1526005057/tmp607447141/\_temporary/0/task\_local1884666946\_0001\_m\_000000*

*................*

*................*

*Input(s):*

*Successfully read 140 records from: "/home/hduser/Datasets/game1.csv"*

*Output(s):*

*Successfully stored 140 records in: "file:/tmp/temp-1526005057/tmp607447141"*

*Counters:*

*Total records written : 140*

*Total bytes written : 0*

*Spillable Memory Manager spill count : 0*

*Total bags proactively spilled: 0*

*Total records proactively spilled: 0*

*Job DAG:*

*job\_local1884666946\_0001*

*..................*

*........*

*(Adam Donachie~BAL~Catcher ~74,180,23,,,)*

*(Paul Bako, BAL , Catcher ,74,215,35)*

*(Ramon Hernandez, BAL , Catcher ,72,210,31)*

*(Kevin Millar, BAL , First Baseman ,72,210,35)*

*(Chris Gomez, BAL , First Baseman ,73,188,36)*

*(Brian Roberts, BAL , Second Baseman ,69,176,29)*

*(Miguel Tejada, BAL , Shortstop ,69,209,31)*

*(Melvin Mora, BAL , Third Baseman ,71,200,35)*

*(Aubrey Huff, BAL , Third Baseman ,76,231,30)*

*(Adam Stern, BAL , Outfielder ,71,180,27)*

*...........................*

*.........*

Note: by default the load command will read data delimited by comma ( , ). Here in this example since the first row is partially delimited by ( ~ ), it reflects in the above first column.

1. Complex Data types.

**TOTUPLE**

Here let’s see how to convert the record into tuple.

By the below command, each row can be converted into a tuple.

gametup = foreach game generate TOTUPLE(name,team,position,weight,height,age);

*.....................*

*.........................*

*((Adam Donachie~BAL~Catcher ~74,180,23,,,))*

*((Paul Bako, BAL , Catcher ,74,215,35))*

*((Ramon Hernandez, BAL , Catcher ,72,210,31))*

*((Kevin Millar, BAL , First Baseman ,72,210,35))*

*((Chris Gomez, BAL , First Baseman ,73,188,36))*

*((Brian Roberts, BAL , Second Baseman ,69,176,29))*

*((Miguel Tejada, BAL , Shortstop ,69,209,31))*

*((Melvin Mora, BAL , Third Baseman ,71,200,35))*

*..........................*

*.............*

**BAG**

A bag contains multiple tuples.

gametup1 = foreach game generate TOTUPLE(TOTUPLE(name,team,position),TOTUPLE(weight,height,age));

*......................*

*................................*

*(((Adam Donachie~BAL~Catcher ~74,180,23),(,,)))*

*(((Paul Bako, BAL , Catcher ),(74,215,35)))*

*(((Ramon Hernandez, BAL , Catcher ),(72,210,31)))*

*(((Kevin Millar, BAL , First Baseman ),(72,210,35)))*

*(((Chris Gomez, BAL , First Baseman ),(73,188,36)))*

*(((Brian Roberts, BAL , Second Baseman ),(69,176,29)))*

*(((Miguel Tejada, BAL , Shortstop ),(69,209,31)))*

*(((Melvin Mora, BAL , Third Baseman ),(71,200,35)))*

*..................................*

*.................*

**MAP**

It’s a kind of key,value pair.

gametup = foreach game generate TOMAP(name,age);

*..............*

*.................*

*([Adam Donachie~BAL~Catcher ~74#])*

*([Paul Bako#35])*

*([Ramon Hernandez#31])*

*([Kevin Millar#35])*

*([Chris Gomez#36])*

*([Brian Roberts#29])*

*([Miguel Tejada#31])*

*([Melvin Mora#35])*

*([Aubrey Huff#30])*

*...........*

*.......*

1. Filtering columns/rows using foreach and filter commands.

**Column level filter:**

game = load '/home/hduser/Datasets/game1.csv' using PigStorage(',')

as (name:chararray,team:chararray,position:chararray,weight:int,height:int,age:int);

gamefil1 = foreach game generate name,team;

dump gamefill;

.*.......*

*...............*

*(Paul Bako, BAL )*

*(Ramon Hernandez, BAL )*

*(Kevin Millar, BAL )*

*(Chris Gomez, BAL )*

*(Brian Roberts, BAL )*

*(Miguel Tejada, BAL )*

*(Melvin Mora, BAL )*

*(Aubrey Huff, BAL )*

*(Adam Stern, BAL )*

*(Jeff Fiorentino, BAL )*

*(Freddie Bynum, BAL )*

*..........*

*....*

**Row level filter:**

gamefilbal = filter game by team == ' BAL ';

dump gamefilbal;

*.............*

*...................*

*(Paul Bako, BAL , Catcher ,74,215,35)*

*(Ramon Hernandez, BAL , Catcher ,72,210,31)*

*(Kevin Millar, BAL , First Baseman ,72,210,35)*

*(Chris Gomez, BAL , First Baseman ,73,188,36)*

*(Brian Roberts, BAL , Second Baseman ,69,176,29)*

*(Miguel Tejada, BAL , Shortstop ,69,209,31)*

*(Melvin Mora, BAL , Third Baseman ,71,200,35)*

*(Aubrey Huff, BAL , Third Baseman ,76,231,30)*

*(Adam Stern, BAL , Outfielder ,71,180,27)*

*(Jeff Fiorentino, BAL , Outfielder ,73,188,24)*

*(Freddie Bynum, BAL , Outfielder ,73,180,27)*

*(Nick Markakis, BAL , Outfielder ,74,185,23)*

*(Brandon Fahey, BAL , Outfielder ,74,160,26)*

*(Corey Patterson, BAL , Outfielder ,69,180,28)*

*......................*

*..........*

gamefilbal = filter game by team matches ' BAL.\*';

dump gamefilbal;

.............

*...................*

*(Paul Bako, BAL , Catcher ,74,215,35)*

*(Ramon Hernandez, BAL , Catcher ,72,210,31)*

*(Kevin Millar, BAL , First Baseman ,72,210,35)*

*(Chris Gomez, BAL , First Baseman ,73,188,36)*

*(Brian Roberts, BAL , Second Baseman ,69,176,29)*

*(Miguel Tejada, BAL , Shortstop ,69,209,31)*

*(Melvin Mora, BAL , Third Baseman ,71,200,35)*

*(Aubrey Huff, BAL , Third Baseman ,76,231,30)*

*(Adam Stern, BAL , Outfielder ,71,180,27)*

*(Jeff Fiorentino, BAL , Outfielder ,73,188,24)*

*(Freddie Bynum, BAL , Outfielder ,73,180,27)*

*(Nick Markakis, BAL , Outfielder ,74,185,23)*

*(Brandon Fahey, BAL , Outfielder ,74,160,26)*

*(Corey Patterson, BAL , Outfielder ,69,180,28)*

*......................*

*..........*

gamefilbal = filter game by team matches ' BAL.\*' and age > 30;

dump gamefilbal;

*(Paul Bako, BAL , Catcher ,74,215,35)*

*(Ramon Hernandez, BAL , Catcher ,72,210,31)*

*(Kevin Millar, BAL , First Baseman ,72,210,35)*

*(Chris Gomez, BAL , First Baseman ,73,188,36)*

*(Miguel Tejada, BAL , Shortstop ,69,209,31)*

*(Melvin Mora, BAL , Third Baseman ,71,200,35)*

*(Jay Payton, BAL , Outfielder ,70,185,34)*

*(Steve Trachsel, BAL , Starting Pitcher ,76,205,36)*

*(Jaret Wright, BAL , Starting Pitcher ,74,230,31)*

*(Kris Benson, BAL , Starting Pitcher ,76,195,32)*

*(Scott Williamson, BAL , Relief Pitcher ,72,180,31)*

*(Chad Bradford, BAL , Relief Pitcher ,77,203,32)*

*(Jamie Walker, BAL , Relief Pitcher ,74,195,36)*

1. Trending technologies example with datatype as char array, commands such as foreach generate flatten - tokenize, group by, count etc.

* Here we are going to give a text file as input
* Going to convert each sentence in the file as tuple
* Then gonna further reduce each work as a tuple
* Then gonna group the word and take the count of occurance
* Then sort the count along with group in desc order
* Then limit the select by first 5 rows which gives the top 5 most occurred words in this text file.

lines = load '/home/hduser/Datasets/hachiko.txt' as (line:chararray);

dump lines;

*(Life with the professor)*

*(Hachiko was an Akita Inu who was born in 1923 in Akita prefecture, Japan. A year later he became a gift for the daughter of a professor of agronomic engineering at Tokyo University. When the professor, Eisaburo Ueno, saw him for the first time, he realized that his legs were slightly crooked. They resembled the kanji that represents the number 8 (?, which in Japanese is pronounced hachi), so he decided to name him Hachiko .)*

*()*

*(When Ueno's daughter grew up, she married and went to live with her husband, leaving the dog behind. The teacher had grown very fond of him, so he decided to keep Hachi instead of giving him away.)*

*()*

*(Ueno went to work by train every day and Hachiko became his faithful companion. Every morning he accompanied him to the Shibuya train station and in the afternoon he went back to meet him when he returned. True loyalty!)*

*...............................*

*.................*

words = foreach lines generate flatten(TOKENIZE(line)) as word;

dump words;

*(Life)*

*(with)*

*(the)*

*(professor)*

*(Hachiko)*

*(was)*

*(an)*

*(Akita)*

*(Inu)*

*(who)*

*(was)*

*(born)*

*(in)*

*(1923)*

*(in)*

*(Akita)*

*(prefecture)*

*(Japan.)*

*...........*

*.....*

grouped = group words by word;

dump grouped;

*......*

*..........(giving,{(giving)})*

*(legacy,{(legacy)})*

*(master,{(master),(master)})*

*(months,{(months),(months)})*

*(number,{(number)})*

*(place.,{(place.)})*

*(return,{(return)})*

*(rework,{(rework)})*

*(statue,{(statue),(statue),(statue)})*

*(sudden,{(sudden)})*

*(taking,{(taking)})*

*(today.,{(today.)})*

*(turned,{(turned)})*

*(waited,{(waited),(waited),(waited),(waited),(waited)})*

*(years.,{(years.),(years.),(years.)})*

*(Finally,{(Finally)})*

*(Hachiko,{(Hachiko),(Hachiko),(Hachiko),(Hachiko),(Hachiko),(Hachiko),(Hachiko),(Hachiko),(Hachiko),(Hachiko),(Hachiko),(Hachiko),(Hachiko),(Hachiko)})*

*....................*

*.........*

techcount = foreach grouped generate group,COUNT(words);

dump techcount;

*orderedout = order techcount by $1 desc;*

*DUMP orderedout;*

*(Hachiko,14)*

*(for,13)*

*(and,12)*

*(him,10)*

*(he,10)*

*(The,9)*

*(a,8)*

*(was,6)*

*(owner,5)*

*(waited,5)*

*(went,4)*

*(that,4)*

*(same,4)*

*……..*

lmt = limit orderedout 5;

dump lmt;

*(the,32)*

*(of,28)*

*(his,15)*

*(in,15)*

*(to,15)*

## Customer Professional Segmentation usecase:

1. Load Customer records

customer = LOAD '/home/hduser/pigdata/custs' using PigStorage(',') AS (

custid:chararray,firstname:chararray, lastname:chararray, age:long, profession:chararray);

dump customer;

*...........*

*....................*

*(4008978,Steven,Howe,63,Actor)*

*(4008979,Samantha,Harrell,55,Therapist)*

*(4008980,Tonya,Bridges,27,Lawyer)*

*(4008981,Shirley,Gordon,37,Computer support specialist)*

*(4008982,Samuel,George,69,Lawyer)*

*(4008983,Bob,Herndon,68,Reporter)*

*(4008984,Kent,Cochran,29,Judge)*

*(4008985,Marion,Hauser,71,Statistician)*

*(4008986,Elaine,Stanton,42,Social worker)*

*(4008987,Elsie,McLaughlin,69,Chemist)*

*(4008988,Tom,McGuire,70,Politician)*

*(4008989,Harry,Bowden,42,Pharmacist)*

*(4008990,Shawn,Weinstein,30,Human resources assistant)*

*..........................*

*..........*

1. Select only 10 records

lmtcustomer = limit customer 10;

grunt> dump lmtcustomer;

*(4000001,Kristina,Chung,55,Pilot)*

*(4000002,Paige,Chen,74,Teacher)*

*(4000003,Sherri,Melton,34,Firefighter)*

*(4000004,Gretchen,Hill,66,Computer hardware engineer)*

*(4000005,Karen,Puckett,74,Lawyer)*

*(4000006,Patrick,Song,42,Veterinarian)*

*(4000007,Elsie,Hamilton,43,Pilot)*

*(4000008,Hazel,Bender,63,Carpenter)*

*(4000009,Malcolm,Wagner,39,Artist)*

*(4000010,Dolores,McLaughlin,60,Writer)*

1. Group customer records by profession

|  |
| --- |
| If we do group by a column, it will be grouped as shown below  Eg: grouped\_column1,{(rows1 from table with grouped\_column1),( rows2 from table with grouped\_column1,),( rows3 from table with grouped\_column1,),( rows4 from table with grouped\_column1,) |

custgrp = group customer by profession;

grunt> dump custgrp;

*.......*

*......................*

*(Agricultural and food scientist,{(4006883,Tina,Starr,48,Agricultural and food scientist),(4008439,Ellen,Kessler,52,Agricultural and food scientist),(4009177,Rose,Cole,64,Agricultural and food scientist),(4008836,Molly,Lu,53,Agricultural and food scientist),(4008448,George,Foster,73,Agricultural and food scientist),(4004101,Mildred,Hewitt,69,Agricultural and food scientist),(4004106,Florence,Wright,25,Agricultural and food scientist),(4004940,Maureen,Rodgers,71,Agricultural and food scientist),(4004667,Debbie,Livingston,56,Agricultural and food scientist),(4008916,Tamara,Norton,56,Agricultural and food scientist),(4002853,Julian,Parks,61,Agricultural and food scientist),(4009589,Virginia,Barbee,52,Agricultural and food scientist),(4003683,Joan,Moon,51,Agricultural and food scientist),(4009865,Betsy,Denton,46,Agricultural and food scientist),(4006861,Ryan,Roberts,42,Agricultural and food scientist),(4005876,W........................*

*...............................................................*

1. Count no of customers by profession

countcust = foreach custgrp generate group,COUNT(customer); dump countcust;

*(Actor,196)*

*(Coach,199)*

*(Judge,189)*

*(Nurse,191)*

*(Pilot,209)*

*(Artist,175)*

*(Dancer,178)*

*(Doctor,189)*

*(Farmer,196)*

*(Lawyer,201)*

*(Writer,95)*

*(Athlete,196)*

*(Chemist,206)*

*(Teacher,189)*

*(Designer,204)*

*(Musician,204)*

*(Reporter,199)*

*(Architect,202)*

*...................*

*........*

*...*

1. Load transaction records

transaction = LOAD '/home/hduser/pigdata/txns' using PigStorage(',') AS

(txnid:chararray,date:chararray,custid:chararray,amount:double,category:chararray,

product:chararray,city:chararray, state:chararray, type:chararray);

dump transaction;

*............*

*......................*

*(00095890,09-19-2011,4009746,63.72,Gymnastics,Balance Beams,San Jose,California,credit)*

*(00095891,08-30-2011,4001223,155.09,Team Sports,Lacrosse,Berkeley,California,credit)*

*(00095892,03-07-2011,4000347,134.21,Team Sports,Softball,Jackson,Mississippi,credit)*

*(00095893,11-14-2011,4005307,87.02,Outdoor Play Equipment,Water Tables,Des Moines,Iowa,credit)*

*(00095894,06-20-2011,4006999,195.08,Outdoor Recreation,Archery,Gilbert,Arizona,credit)*

*(00095895,11-19-2011,4008394,119.47,Air Sports,Parachutes,Miami,Florida,credit)*

*(00095896,04-18-2011,4007291,100.5,Gymnastics,Gymnastics Mats,El Paso,Texas,credit)*

*(00095897,05-07-2011,4005819,185.54,Indoor Games,Air Hockey,Durham,North Carolina,credit)*

*(00095898,03-07-2011,4007288,141.67,Team Sports,Team Handball,Pasadena,California,credit)*

*(00095899,03-09-2011,4001481,76.01,Water Sports,Life Jackets,New York,New York,credit)*

*(00095900,04-14-2011,4007608,33.94,Exercise & Fitness,Cardio Machine Accessories,Denver ,Colorado,credit)*

*(00095901,01-02-2011,4007334,138.36,Outdoor Play Equipment,Outdoor Playsets,Huntsville,Alabama,credit)*

*(00095902,01-03-2011,4009230,32.84,Team Sports,Hockey,Everett,Washington,credit)*

*(00095903,09-05-2011,4005514,52.82,Jumping,Pogo Sticks,Scottsdale,Arizona,credit)*

1. Group transactions by customer

txngroup = group transaction by custid;

grunt> dump txngroup;

*..............*

*...............................*

*(4009999,{(00088664,10-31-2011,4009999,91.38,Exercise & Fitness,Exercise Balls,Gresham,Oregon,credit),(00038171,10-23-2011,4009999,124.63,Water Sports,Wetsuits,Sunnyvale,California,credit),(00038594,04-30-2011,4009999,109.87,Outdoor Recreation,Rock Climbing,Green Bay,Wisconsin,credit),(00047083,07-20-2011,4009999,27.08,Racquet Sports,Badminton,San Diego,California,cash),(00036163,03-02-2011,4009999,74.67,Team Sports,Football,Cambridge,Massachusetts,credit),(00036126,07-12-2011,4009999,33.06,Outdoor Recreation,Disc Golf,Cleveland,Ohio,credit),(00026724,12-03-2011,4009999,25.24,Outdoor Play Equipment,Playhouses,McKinney,Texas,credit),(00066732,11-06-2011,4009999,182.5,Outdoor Recreation,Geocaching,El Paso,Texas,credit),(00052783,05-14-2011,4009999,142.63,Jumping,Trampoline Accessories,Irvine,California,credit),(00002155,04-14-2011,4009999,176.0,Water Sports,Wetsuits,Columbia,South Carolina,credit),(00026984,05-22-2011,4009999,111.47,Water Sports,Bodyboarding,Dayton,Ohio,credit),(00083186,05-14-2011,4009999,167.11,Racquet Sports,Squash,Detroit,Michigan,credit)})*

1. Sum total amount spent by each customer

txnamtsum = foreach txngroup generate group,SUM(transaction.amount);

*.......*

*............*

*(4009988,620.51)*

*(4009989,974.6899999999999)*

*(4009990,971.0699999999999)*

*(4009991,941.1099999999999)*

*(4009992,913.13)*

*(4009993,1285.5200000000002)*

*(4009994,1021.3699999999999)*

*(4009995,810.75)*

*(4009996,1538.5499999999997)*

*(4009997,692.0999999999999)*

*(4009998,924.87)*

*(4009999,1265.6399999999999)*

1. Order the customer records beginning from highest spender
2. Select only top 10 customers

tnxorderamt = order txnamtsum by $1 desc;

grunt> lmtmaxamt = limit tnxorderamt 10;

grunt> dump lmtmaxamt;

*(4006425,2543.16)*

*(4001389,2521.62)*

*(4000663,2481.3800000000006)*

*(4005227,2457.17)*

*(4001779,2434.21)*

*(4007790,2433.72)*

*(4001051,2401.4300000000003)*

*(4001740,2401.43)*

*(4008933,2388.4300000000003)*

*(4006467,2376.45)*

1. Join the transactions with customer details

joincusttxn = join lmtmaxamt by $0, customer by $0;

lmtjoincusttxn = limit joincusttxn 10;

dump lmtjoincusttxn;

*(4000663,2481.3800000000006,4000663,Claire,Henry,29,Police officer)*

*(4001051,2401.4300000000003,4001051,Arlene,Higgins,62,Police officer)*

*(4001389,2521.62,4001389,Maureen,Dixon,31,Electrical engineer)*

*(4001740,2401.43,4001740,Melanie,Kirkland,45,Civil engineer)*

*(4001779,2434.21,4001779,William,Cross,35,Writer)*

*(4005227,2457.17,4005227,Diane,Allred,57,Firefighter)*

*(4006425,2543.16,4006425,Joe,Burns,30,Economist)*

*(4006467,2376.45,4006467,Evelyn,Monroe,37,Financial analyst)*

*(4007790,2433.72,4007790,Clifford,Wooten,24,Librarian)*

*(4008933,2388.4300000000003,4008933,Regina,Conner,39,Automotive mechanic)*

1. Select the required fields from the join for final output

formatcusttxn = foreach lmtjoincusttxn generate $0, $3, $5, $1;

dump formatcusttxn;

*(4000663,Claire,29,2481.3800000000006)*

*(4001051,Arlene,62,2401.4300000000003)*

*(4001389,Maureen,31,2521.62)*

*(4001740,Melanie,45,2401.43)*

*(4001779,William,35,2434.21)*

*(4005227,Diane,57,2457.17)*

*(4006425,Joe,30,2543.16)*

*(4006467,Evelyn,37,2376.45)*

*(4007790,Clifford,24,2433.72)*

*(4008933,Regina,39,2388.4300000000003)*

1. Order the customer based on the highest spender.

orderformatcusttxn = order formatcusttxn by $3 desc;

dump orderformatcusttxn;

*(4006425,Joe,30,2543.16)*

*(4001389,Maureen,31,2521.62)*

*(4000663,Claire,29,2481.3800000000006)*

*(4005227,Diane,57,2457.17)*

*(4001779,William,35,2434.21)*

*(4007790,Clifford,24,2433.72)*

*(4001051,Arlene,62,2401.4300000000003)*

*(4001740,Melanie,45,2401.43)*

*(4008933,Regina,39,2388.4300000000003)*

*(4006467,Evelyn,37,2376.45)*

1. store the final output

store orderformatcusttxn into '/home/hduser/kanna1/piglocal/custtxnoutput' using PigStorage('~');

[hduser@Inceptez ~]$ ls /home/hduser/kanna1/piglocal/custtxnoutput

part-r-00000 \_SUCCESS

[hduser@Inceptez ~]$ cat /home/hduser/kanna1/piglocal/custtxnoutput/part-r-00000

*4006425~Joe~30~2543.16*

*4001389~Maureen~31~2521.62*

*4000663~Claire~29~2481.3800000000006*

*4005227~Diane~57~2457.17*

*4001779~William~35~2434.21*

*4007790~Clifford~24~2433.72*

*4001051~Arlene~62~2401.4300000000003*

*4001740~Melanie~45~2401.43*

*4008933~Regina~39~2388.4300000000003*

*4006467~Evelyn~37~2376.45*

1. UNION

top3custtxndesc = order formatcusttxn by $3 desc;

top3desc = limit top3custtxndesc 3;

top3custtxnasc = order formatcusttxn by $3;

top3asc = limit top3custtxnasc 3;

unioncusttxn = UNION top3desc, top3asc;

dump unioncusttxn;

*(4006467,Evelyn,37,2376.45)*

*(4008933,Regina,39,2388.4300000000003)*

*(4001740,Melanie,45,2401.43)*

*(4006425,Joe,30,2543.16)*

*(4001389,Maureen,31,2521.62)*

*(4000663,Claire,29,2481.3800000000006)*

**Run pig latin script with param.**

Create a script as testpig.pig with the below content

## vi testpig.pig

**raw = LOAD '$INPUTDATA' USING PigStorage('\t'); dump raw;**

**Execute the below script from linux command line**

**pig -x local -f testpig.pig -p INPUTDATA=/home/hduser/pigdata/testdata.txt**

|  |  |
| --- | --- |
| **PIG** | **HIVE** |
| * It’s a data flow language * Nested relation data model * Schema is not mandatory * Limited optimization opportunity | * It’s a declarative language * Flat relation data model * Schema is mandatory * Significantly opportunities are higher |