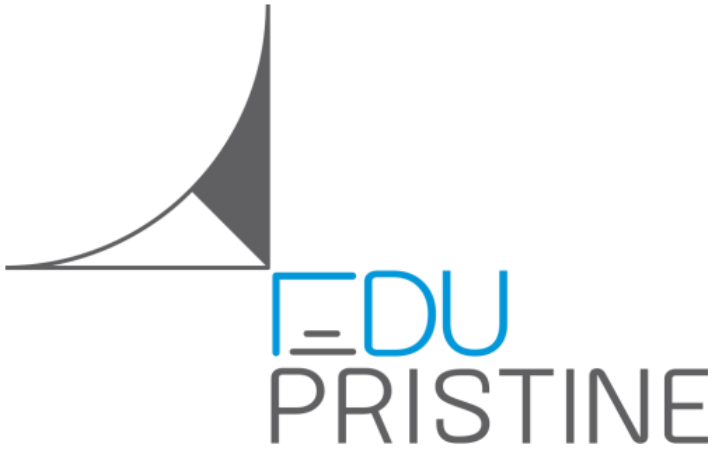


Live Project 1

Retail Data Analysis using Pig and Hive

© **EduPristine** – www.edupristine.com

Retail Transactions

- Retail stores daily generate millions of transactions logs.
- Analyzing these logs would generate beautiful insights and improve business.
- Storing these logs on traditional databases would be costly and scalability will be a big challenge.

Volume of transactions

- Stores like walmart are spread across different locations.
- Daily millions of customers visit these stores and generate billions of logs.
- This billions of logs contribute to huge volume of data.

Velocity of transactions

- In peak hours, 1000’s of transactions will happen in any given second.

- 1000’s of trasactions/sec across all stores contribute to high velocity.

Variety of transactions

- The most widely known varities of data generated by transactions:
  - Json Format
  - Xml Format
  - Csv Format

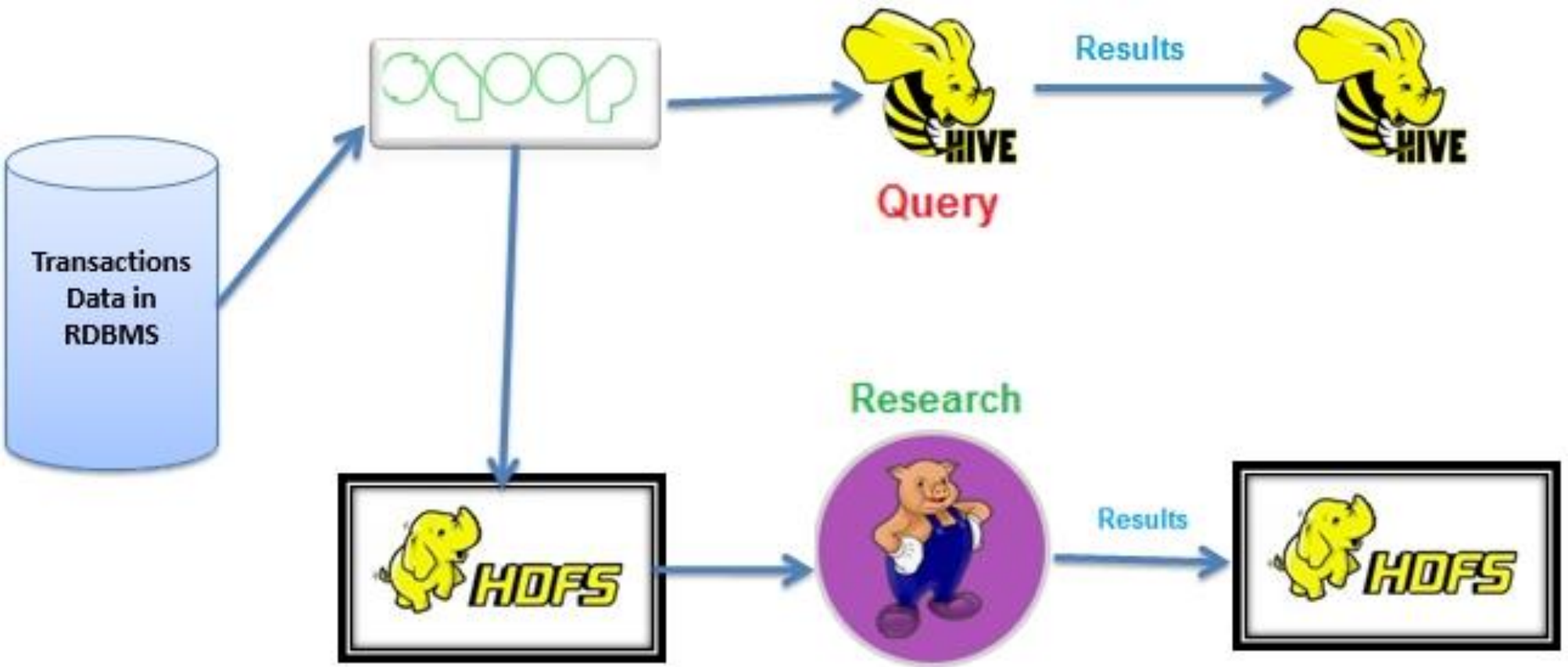
Its Bigggg Data

- Having Huge Volume, High velocity and variety will make this data Big Data.
  - Challenges:
    - Storage
    - Scalability
    - Processing
    - Sharing

Goal

- Process these logs over night as a batch to understand:
  - Demand of a given product.
  - Trend and seasonality of sales.
  - Understand performance of chain.
  - Loyal Customer identification.

Solution


Transactions Data in RDBMS
sqoop
HIVE
Query
Results
HIVE
HDFS
Research
Results
HDFS

Transactions dataset

Field	Type	Null	Key	Default	Extra
id	varchar(20)	YES		NULL	
chain	varchar(20)	YES		NULL	
dept	varchar(20)	YES		NULL	
category	varchar(20)	YES		NULL	
company	varchar(20)	YES		NULL	
brand	varchar(20)	YES		NULL	
date1	varchar(10)	YES		NULL	
productsize	int(11)	YES		NULL	
productmeasure	varchar(10)	YES		NULL	
purchasequantity	int(11)	YES		NULL	
purchaseamount	float	YES		NULL	

11 rows in set (0.00 sec)

Load data into MySQL

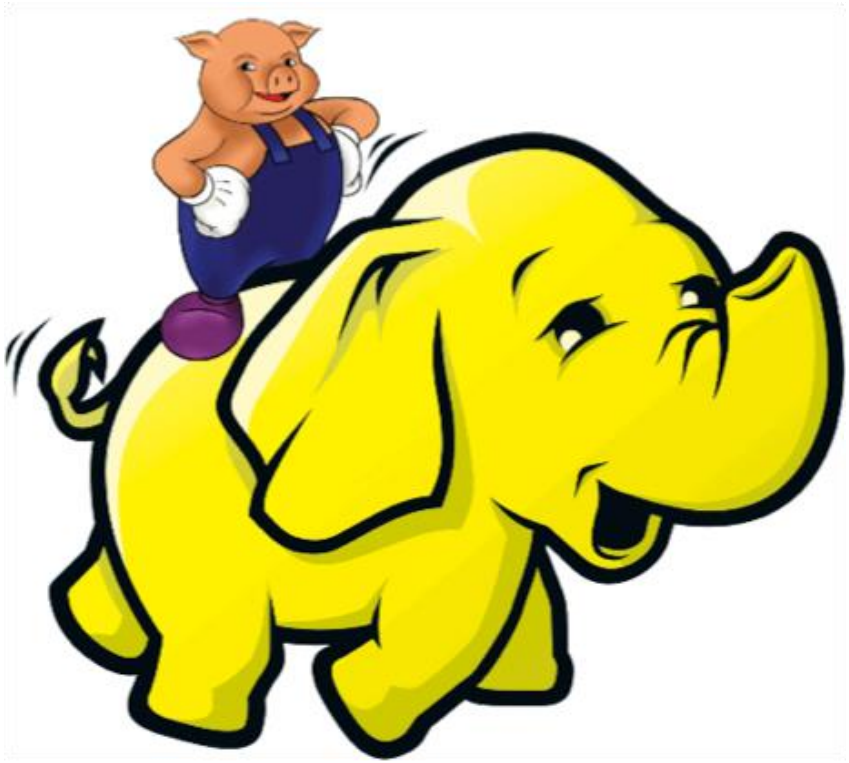
```
-- select database
use training

-- create table
CREATE TABLE transactions(id varchar(20),chain varchar(20), dept varchar(20),category varchar(20), company varchar(20),
brand varchar(20), date1 varchar(10), productsize int, productmeasure varchar(10), purchasequantity int, purchaseamount FLOAT);

-- load data
LOAD DATA INFILE '/home/cloudera/Desktop/transactions.csv' INTO TABLE transactions
FIELDS TERMINATED BY ',' ENCLOSED BY '""'
LINES TERMINATED BY '\r\n';

-- check

select * from transactions limit 5;
```



- describe
- illustrate
- explain

Use the above commands for better understanding of schema and data flow.

SQOOP data into HDFS

```
-- SQOOP data into HDFS

sqoop import --connect jdbc:mysql://localhost/training
--username training -P --table transactions -m 1 --target-dir TransactionsData
```

Contents of directory /user/training/TransactionsData

Goto : /user/training/TransactionsD go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
_SUCCESS	file	0 KB	1	64 MB	2015-04-05 21:52	rw-r--r--	training	supergroup
part-m-00000	file	57.52 MB	1	64 MB	2015-04-05 21:52	rw-r--r--	training	supergroup

Load data into pig

```
transactions = LOAD 'TransactionsData/part-m-00000' USING PigStorage(',') as
(id:chararray,chain:chararray,dept:chararray,category:chararray,company:chararray,brand:ch
ararray,date:chararray, productsize:float, productmeasure:chararray, purchasequantity:int,
purchaseamount:float);
```

Top 10 customers

```
custGroup = GROUP transactions BY id; --grouping
```

```
custSpendings = FOREACH custGroup GENERATE group,
SUM(transactions.purchaseamount) as spendings; --sum operation
```

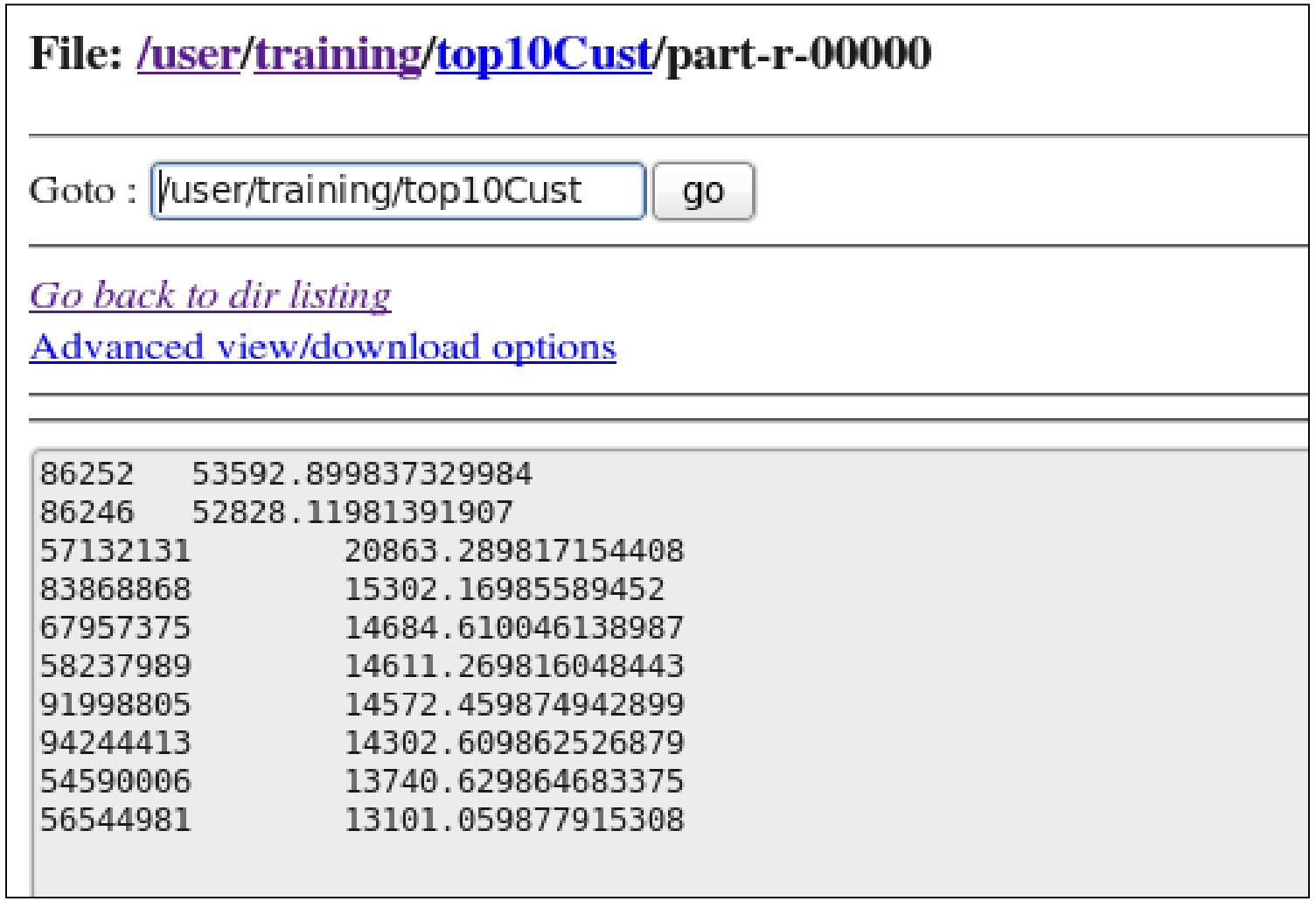
```
custSpendingsSort = ORDER custSpendings BY spendings desc;
```

```
top10Cust = LIMIT custSpendingsSort 10;
```

```
DUMP top10Cust;
```

```
STORE top10Cust INTO 'top10Cust'
```

Snap shot of output



Chain wise sales

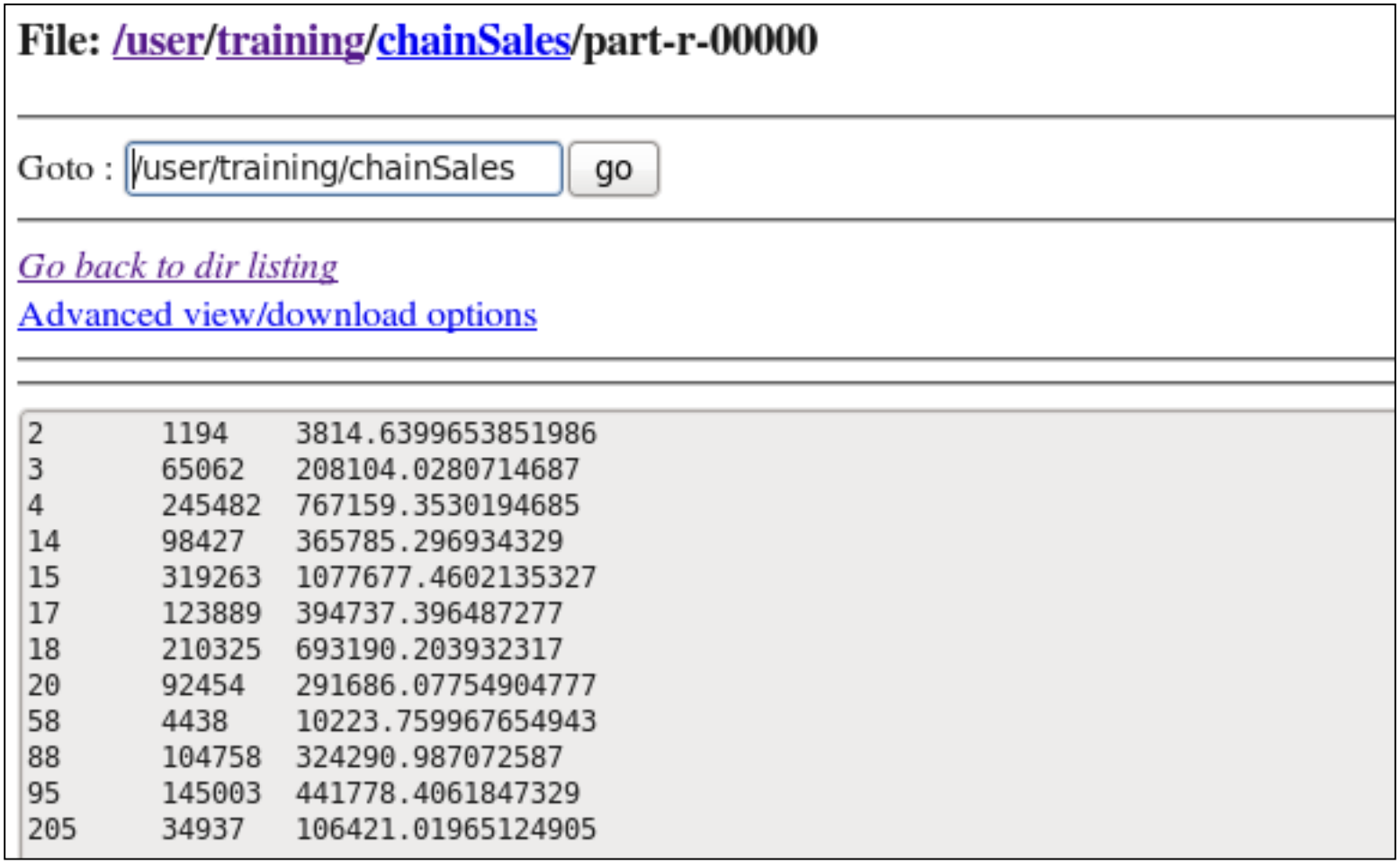
```
chainGroup = GROUP transactions BY chain;
```

```
chainSales = FOREACH chainGroup GENERATE group,
SUM(transactions.purchasequantity) as totalQuantity,
SUM(transactions.purchaseamount) as totalSales;
```

```
dump chainSales;
```

```
STORE chainSales INTO 'chainSales‘;
```

Snapshot of output



Each chain, top 10 customers

```
chainGroupCust = GROUP transactions BY (chain,id);
```

```
chainGroupCustSpedings1 = FOREACH chainGroupCust GENERATE group, SUM(transactions.purchaseamount)
as spendings;
```

```
chainGroupCustSpendings2= FOREACH chainGroupCustSpedings1 generate group.chain as
chain,group.id as id, spendings;
```

```
chainGroupCustSpendings3= GROUP chainGroupCustSpendings2 BY chain;
```

```
chainTop10Cust = FOREACH chainGroupCustSpendings3{
chainGroupCustSpedingsSort = ORDER chainGroupCustSpendings2 BY spendings DESC;
top10Cust = LIMIT chainGroupCustSpedingsSort  10;
GENERATE top10Cust;
}
```

```
chainTop10Cust = FOREACH chainTop10Cust GENERATE FLATTEN(top10Cust);
```

```
STORE chainTop10Cust INTO ‘chainTop10Cust ’
```

Snapshot of output

File: /user/training/chainTop10Cust/part-r-00000

Goto : /user/training/chainTop10Cu go

Go back to dir listing

Advanced view/download options

```
14      67957375        14684.610046138987
14      58237989        14611.269816048443
14      17456470        12878.939868450165
14      13074629        12329.949950814247
14      20552767        11653.639915667474
14      97110346        11154.869882548228
14      90548180        10474.799921853468
14      55304581        10256.949917048216
14      59816528        9873.22992990911
14      91883639        8436.549869716167
15      57132131        20863.289817154408
15      54590006        13740.629864683375
15      93752475        12884.169838543981
15      51531106        12739.759873157367
15      13251776        12512.589874466881
15      85851216        12289.38995597884
```

This operation cannot be done using hive queries

Each customer most bought brand

```
CustBrandGroup = GROUP transactions BY (id,brand);
```

```
CustBrandQuantity = FOREACH CustBrandGroup GENERATE group,
SUM(transactions.purchasequantity) as sales;
```

```
CustBrandQuantity = FOREACH CustBrandQuantity GENERATE
group.brand as brand, group.id as id,  sales;
```

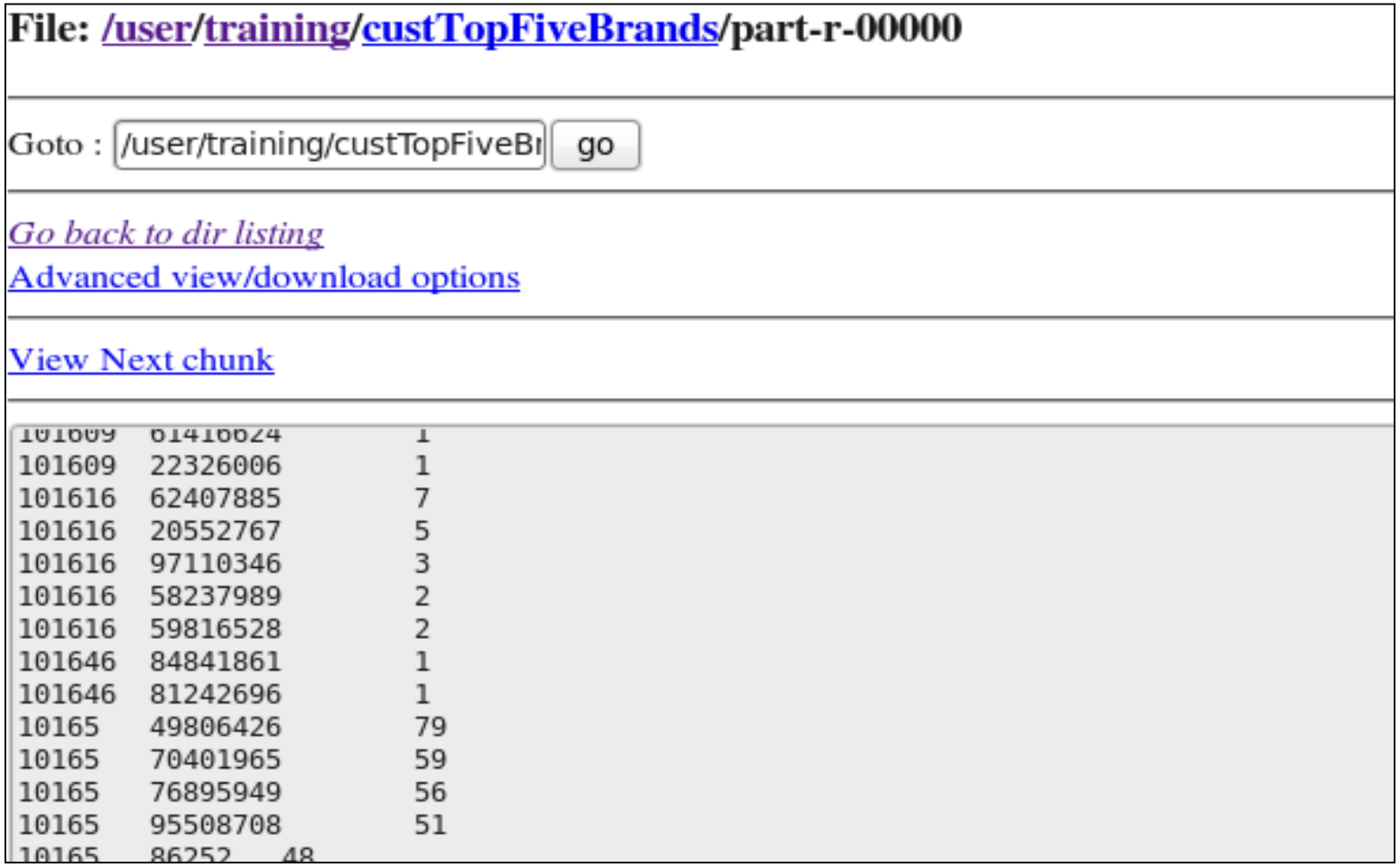
```
CustBrandQuantityGroup = GROUP CustBrandQuantity BY brand;
```

```
custTop5Brands = FOREACH CustBrandQuantityGroup{
        CustBrandQuantityGroupSort = ORDER CustBrandQuantity BY sales DESC;
                top5Brand = LIMIT CustBrandQuantityGroupSort  5;
        GENERATE top5Brand;}
```

```
custTop5Brands = FOREACH custTop5Brands GENERATE FLATTEN(top5Brand);
```

```
STORE custTop5Brands INTO 'custTopFiveBrands';
```

Snapshot of output



This operation cannot be done using hive queries

Top 10 brands

```
brandGroup = GROUP transactions BY brand;
```

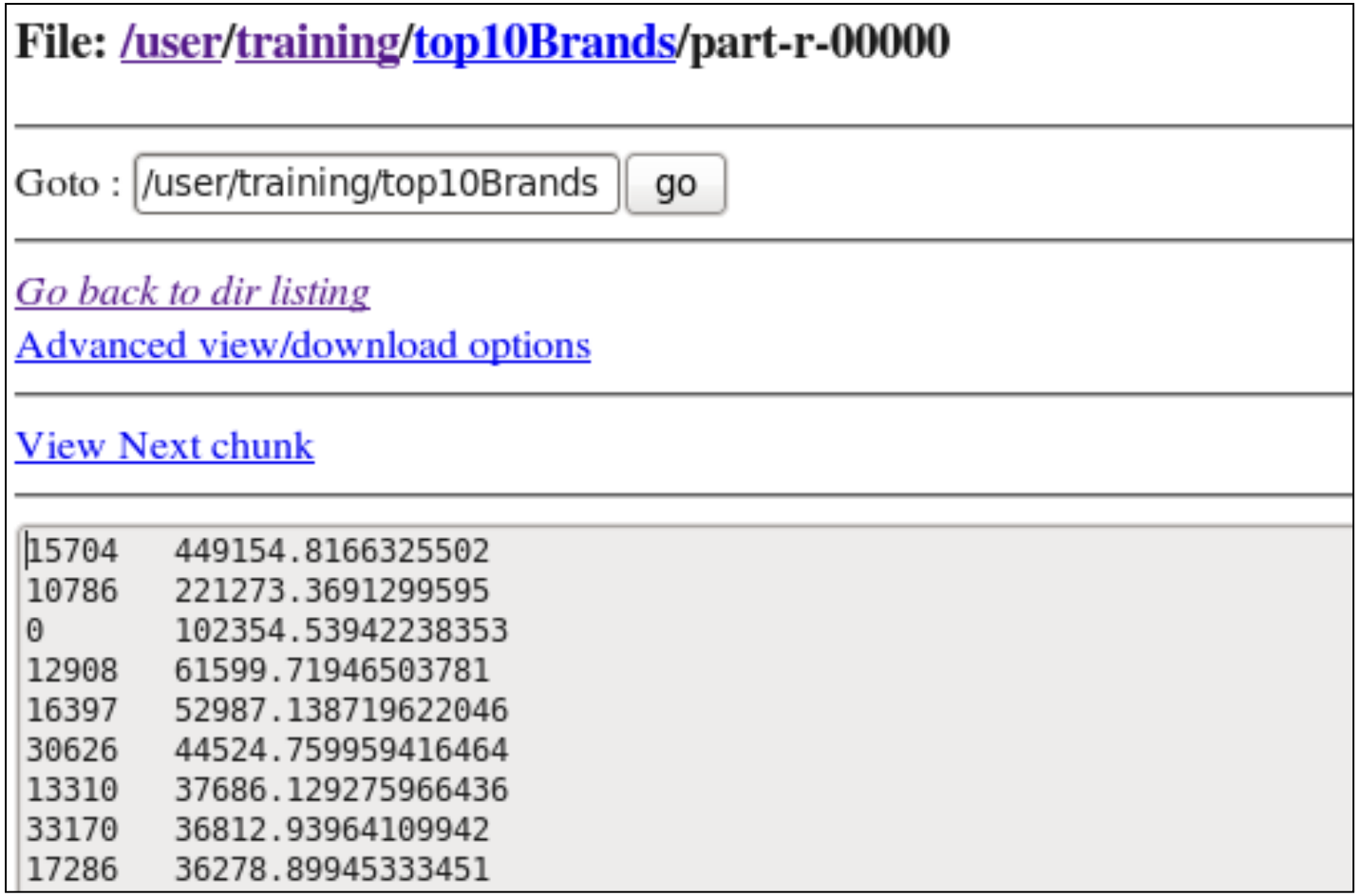
```
brandPurchase = FOREACH brandGroup GENERATE group,
SUM(transactions.purchaseamount) as purchase;
```

```
brandPurchaseSort = ORDER brandPurchase BY purchase desc;
```

```
top10Brands = LIMIT brandPurchaseSort 10;
```

```
STORE top10Brands INTO 'top10Brands';
```

Snapshot of output



top 10 companies

```
companyGroup = GROUP transactions BY company;
```

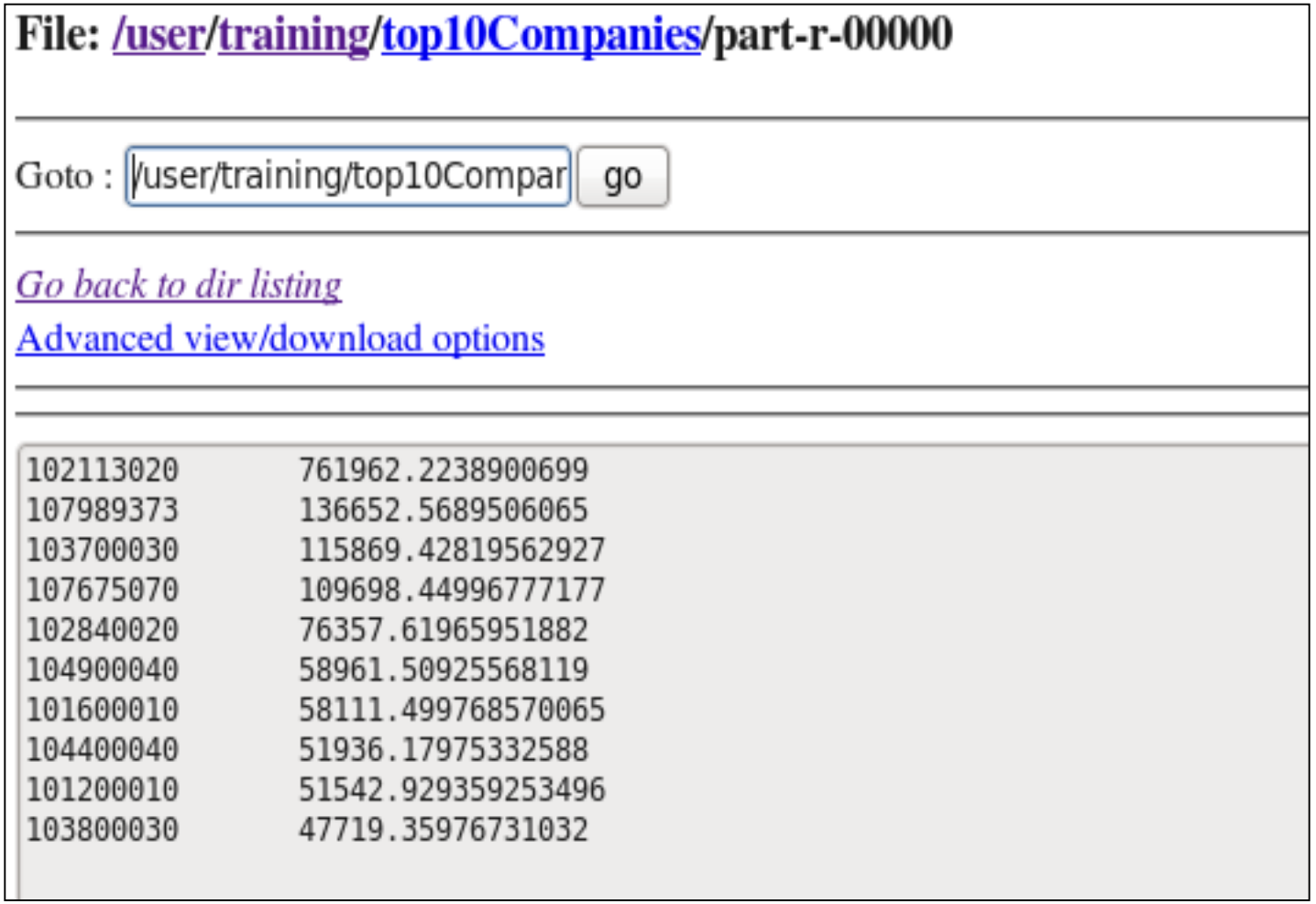
```
companyPurchase = FOREACH companyGroup GENERATE group,
SUM(transactions.purchaseamount) as purchase;
```

```
companyPurchaseSort = ORDER companyPurchase BY purchase desc;
```

```
top10Companies = LIMIT companyPurchaseSort 10;
```

```
STORE top10Companies INTO 'top10Companies';
```

Snapshot of output



Chain Year Monthly Sales

```
chainYearMonSales = FOREACH transactions GENERATE
chain,STRSPLIT(date,'/',3),purchaseamount as sales;
```

```
chainYearMonSales = FOREACH chainYearMonSales GENERATE chain, $1.$0 as
month, $1.$2 as year, sales;
```

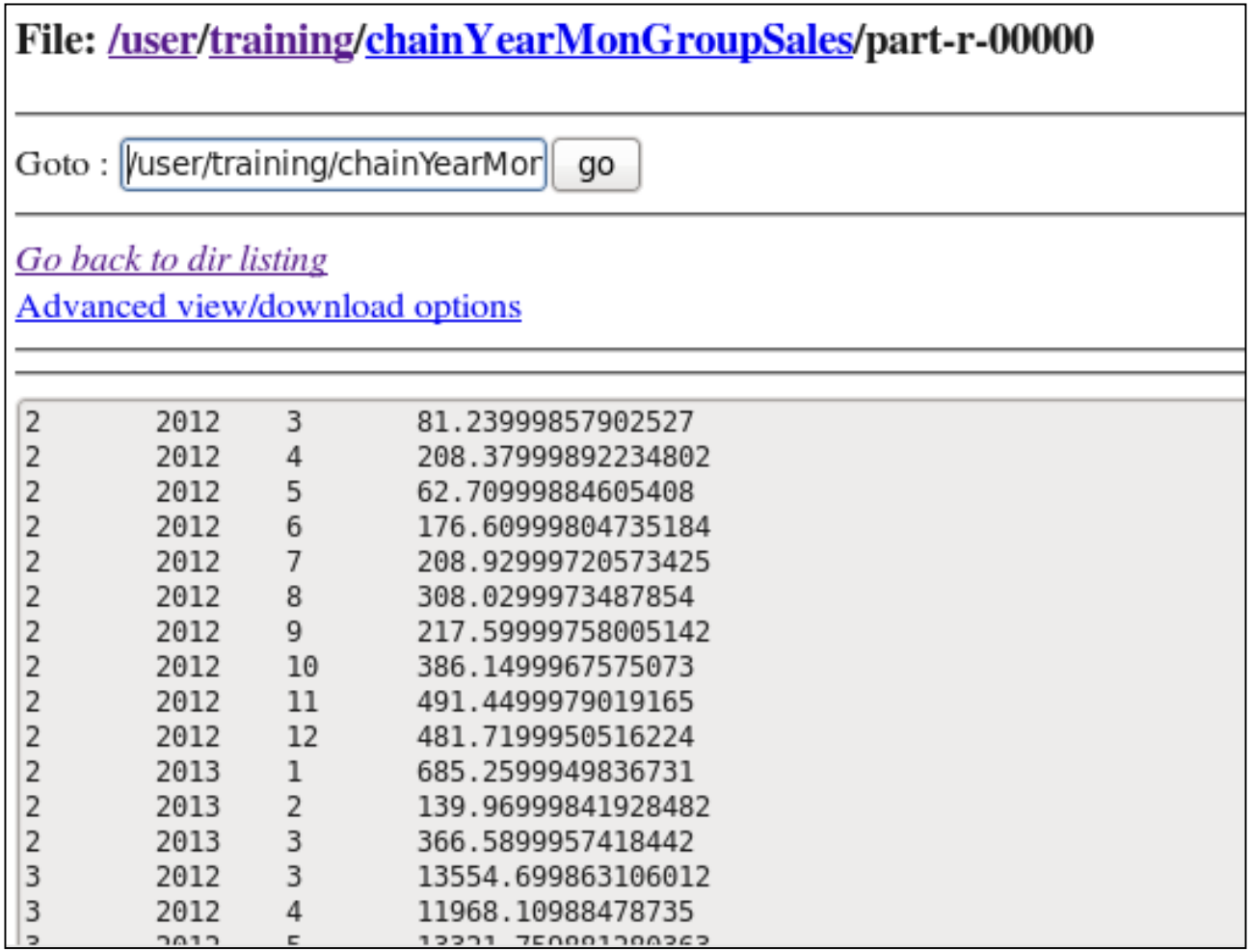
```
chainYearMonSalesGroup = GROUP chainYearMonSales by (chain,year,month);
```

```
chainYearMonGroupSales = FOREACH chainYearMonSalesGroup GENERATE group,
SUM(chainYearMonSales.sales) as totalsales;
```

```
chainYearMonGroupSales = FOREACH chainYearMonGroupSales GENERATE
group.chain as chain, group.year as year, group.month as month, totalsales;
```

```
STORE chainYearMonGroupSales INTO 'chainYearMonGroupSales';
```

Snapshot of output




HIVE

Sqoop data into hive

```
-- sqoop data from mysql to hive

sqoop import --connect jdbc:mysql://localhost/training \
   --username training -P \
   --table transactions \
   --hive-import \
   --hive-table transactions_staging -m 1
```

Partioning and bucketing

- Partitioning and bucketing in hive will let you do faster querying.
- For dynamic partitioning, load the data in to staging table which is already done.
- Now create a production table, and insert data.

Open hive and set below properties

```
-- Partitioning and bucketing
set hive.enforce.bucketing = true;
SET hive.exec.dynamic.partition = true;
SET hive.exec.dynamic.partition.mode = nonstrict;
```

Create production table and load data

```
-- production table
CREATE TABLE transactions_production
( id string,
dept string,
category string,
company string,
brand string,
date1 string,
productsize int,
productmeasure string,
purchasequantity int,
purchaseamount double)
PARTITIONED BY (chain   string) CLUSTERED BY(id) INTO 5 BUCKETS
ROW FORMAT DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE;

INSERT OVERWRITE TABLE transactions_production PARTITION (chain)
select id,dept, category, company,brand,date1,productsize,productmeasure,
purchasequantity,purchaseamount,chain from transactions_staging;
```

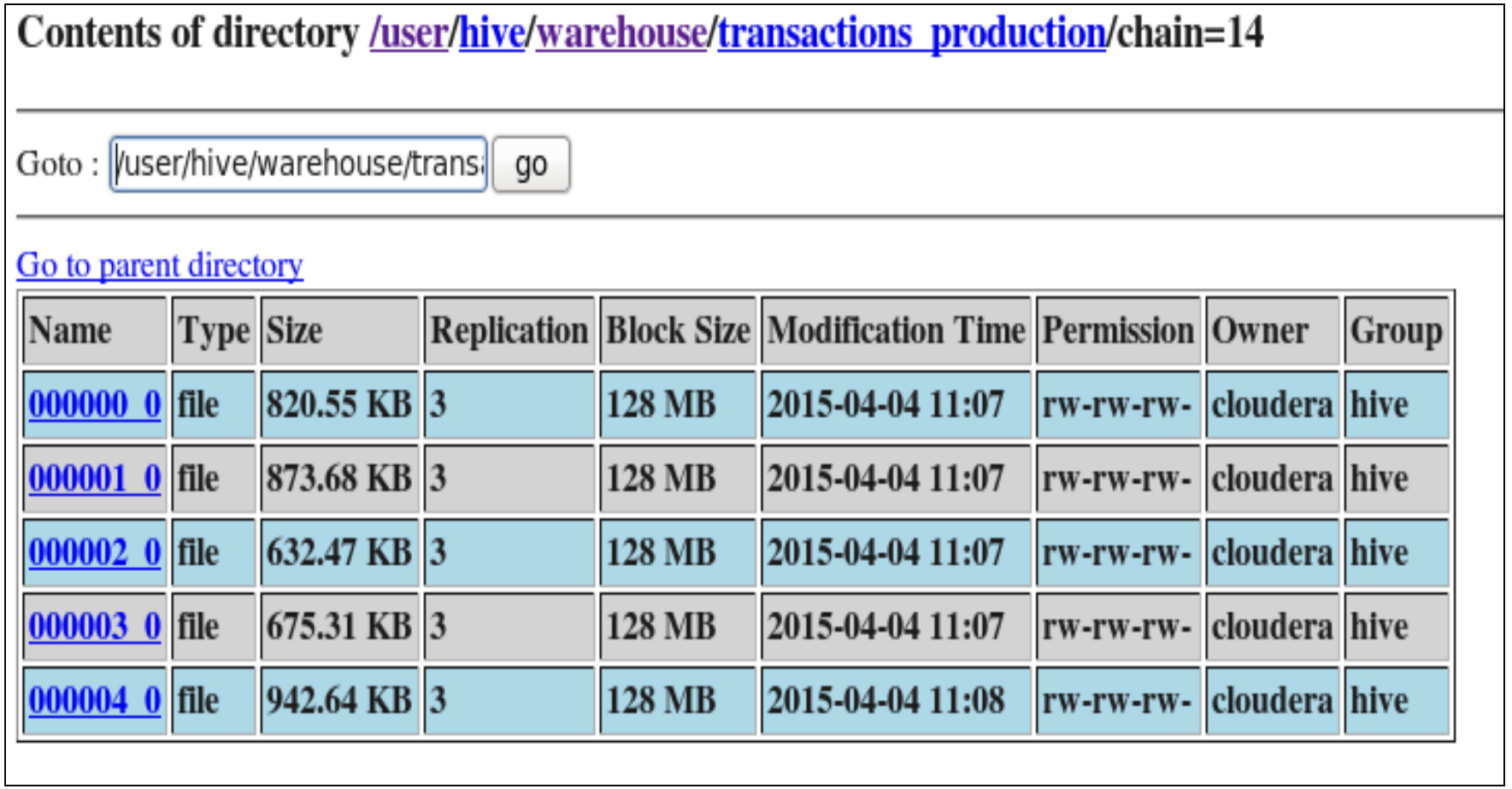
Partitioning snapshot

Goto : /user/hive/warehouse/trans go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
chain=14	dir				2015-04-04 11:08	rwxrwxrwt	cloudera	hive
chain=15	dir				2015-04-04 11:08	rwxrwxrwt	cloudera	hive
chain=17	dir				2015-04-04 11:08	rwxrwxrwt	cloudera	hive
chain=18	dir				2015-04-04 11:08	rwxrwxrwt	cloudera	hive
chain=2	dir				2015-04-04 11:08	rwxrwxrwt	cloudera	hive
chain=20	dir				2015-04-04 11:08	rwxrwxrwt	cloudera	hive
chain=205	dir				2015-04-04 11:08	rwxrwxrwt	cloudera	hive
chain=3	dir				2015-04-04 11:08	rwxrwxrwt	cloudera	hive
chain=4	dir				2015-04-04 11:08	rwxrwxrwt	cloudera	hive
chain=58	dir				2015-04-04 11:08	rwxrwxrwt	cloudera	hive
chain=88	dir				2015-04-04 11:08	rwxrwxrwt	cloudera	hive
chain=95	dir				2015-04-04 11:08	rwxrwxrwt	cloudera	hive

Bucketing snapshot in chain=14



Contents of directory /user/hive/warehouse/transactions_production/chain=14

Goto : /user/hive/warehouse/trans go

Go to parent directory

Name	Type	Size	Replication	Block Size	Modification Time	Permission	Owner	Group
000000_0	file	820.55 KB	3	128 MB	2015-04-04 11:07	rw-rw-rw-	cloudera	hive
000001_0	file	873.68 KB	3	128 MB	2015-04-04 11:07	rw-rw-rw-	cloudera	hive
000002_0	file	632.47 KB	3	128 MB	2015-04-04 11:07	rw-rw-rw-	cloudera	hive
000003_0	file	675.31 KB	3	128 MB	2015-04-04 11:07	rw-rw-rw-	cloudera	hive
000004_0	file	942.64 KB	3	128 MB	2015-04-04 11:08	rw-rw-rw-	cloudera	hive

Top 10 customers

```
-- Top 10 customers

select id, sum(purchaseamount) as custSpendings from transactions_production
group by id
sort by custSpendings DESC
limit 10;
```

Snapshot of output

```
86252    53592.89999999957
86246    52828.11999999952
57132131          20863.29000000036
83868868          15302.169999999785
67957375          14684.609999999933
58237989          14611.269999999817
91998805          14572.45999999987
94244413          14302.60999999986
54590006          13740.629999999846
56544981          13101.059999999881
Time taken: 178.254 seconds
```

Chain wise sales

```
-- chain wise sales
select chain, sum(purchaseamount), sum(purchasequantity) from transactions_production
group by chain;
```

Snapshot of output

```
OK
14      365785.2999998729       98427
15      1077677.4699990302      319263
17      394737.39999981265      123889
18      693190.209999508        210325
2       3814.6399999999644      1194
20      291686.0799999232       92454
205     106421.01999999923      34937
3       208104.02999998012      65062
4       767159.3599993604       245482
58      10223.759999999922      4438
88      324290.9899998967       104758
95      441778.4099997955       145003
Time taken: 62.781 seconds
```

Top 10 brands

```
-- top 10 brands
select brand, sum(purchaseamount) as custSpendings from transactions_production
group by brand
sort by custSpendings DESC
limit 10;
```

Snapshot of output

```
OK
15704    449154.8199997314
10786    221273.3699999814 9
0        102354.5400000084 9
12908    61599.71999999789
16397    52987.13999999947
30626    44524.76000000001
13310    37686.13000000184 5
33170    36812.94000000226 5
17286    36278.90000000090 4
9886     30633.26000000218 5
Time taken: 141.856 seconds
```

Top 10 companies

```
-- top 10 companies
select company, sum(purchaseamount) as custSpendings from transactions_production
group by company
sort by custSpendings DESC
limit 10;
```

Snapshot of output

```
Total MapReduce CPU Time Spent: 19 seconds 760 msec
OK
102113020       761962.2299992407
107989373       136652.57000001118
103700030       115869.43000001504
107675070       109698.45000000006
102840020       76357.6199999992
104900040       58961.5100000003
101600010       58111.500000000546
104400040       51936.17999999996
101200010       51542.92999999974
103800030       47719.360000000524
Time taken: 134.221 seconds
```

Chain Year Monthly Sales

```
-- Chain Year Monthly Sales
select chain, split(date1,'/')[2] as year1, split(date1,'/')[0] as month1,
sum(purchaseamount) as totalsales from transactions_production
group by chain,split(date1,'/')[0],split(date1,'/')[2];
```

Snapshot of output

```
OK
14          2013        1           25989.100000000908
14          2012        10          24914.220000000976
14          2012        11          26941.65000000088
14          2012        12          34061.4400000014
14          2013        2           24553.660000000826
14          2012        3           22811.400000000634
14          2013        3           25149.64000000094
14          2012        4           23113.890000000614
14          2013        4           17588.63999999994
14          2012        5           26469.71000000088
14          2013        5           10979.339999999844
14          2012        6           24692.90000000072
14          2013        6           8059.429999999861
14          2012        7           21334.34000000049
14          2013        7           543.8000000000003
14          2012        8           23286.67000000069
14          2012        9           25295.47000000095
15          2013        1           77346.02000000096
15          2012        10          77011.47000000112
15          2012        11          76465.90000000053
15          2012        12          97227.45000000458
15          2013        2           70020.98999999887
15          2012        3           69659.74999999933
15          2013        3           77026.080000001
```

Storing results

- **To store results**

```
-- store results

CREATE TABLE target
AS
SELECT col1,col2
FROM SOURCE
```

```
-- Example create table top10companies
CREATE TABLE top10companies
AS
select company, sum(purchaseamount) as custSpendings from transactions_production
group by company
sort by custSpendings DESC
```