ASSIGNMENT 4.2

Dataset used:

television.txt

```
2017-12-02 01:32:03,624 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(Samsung,Optima,14,Madhya Pradesh,132401,14200)
(Onida,Lucid,18,Uttar Pradesh,232401,16200)
(Akai,Decent,16,Kerala,922401,12200)
(Lava,Attention,20,Assam,454601,24200)
(Zen,Super,14,Maharashtra,619082,9200)
(Samsung,Optima,14,Madhya Pradesh,132401,14200)
(Onida,Lucid,18,Uttar Pradesh,232401,16200)
(Onida,Decent,14,Uttar Pradesh,232401,16200)
(Lava,Attention,20,Assam,454601,24200)
(Zen,Super,14,Maharashtra,619082,9200)
(Samsung,Optima,14,Madhya Pradesh,132401,14200)
(Samsung,Decent,16,Kerala,922401,12200)
(Lava,Attention,20,Assam,454601,24200)
(Samsung,Super,14,Maharashtra,619082,9200)
(Samsung,Super,14,Maharashtra,619082,9200)
(Samsung,Super,14,Maharashtra,619082,9200)
grunt> DESCRIBE filtered_tv;
filtered_tv: {company_name: chararray,product_name: chararray,size_in_inches: int,state: chararray,pincode: int,price: int}
grunt>
acadgild@localhost:/u...  *Unsaved Document 1...
```

Data in the text file had some invalid rows. So I have used filter operation to get rid of them. **filtered_tv** is the resulted relation/schema.

Apache Pig Commands:

1. CONCAT(): It is used to two or more expressions or columns of same type.

Example query: concat_company_and_product = FOREACH filtered_tv GENERATE CONCAT (company_name, ' ', product_name) AS (company_with_product);

Output: DUMP concat_company_and_product;

```
2017-12-02 02:05:03,561 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2017-12-02 02:05:03,562 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to pro
cess : 1
(Samsung Optima)
(Onida Lucid)
(Akai Decent)
(Lava Attention)
(Zen Super)
(Samsung Optima)
(Onida Lucid)
(Onida Decent)
(Lava Attention)
(Zen Super)
(Samsung Optima)
(Samsung Decent)
(Lava Attention)
(Samsung Super)
(Samsung Super)
(Samsung Super)
grunt> DESCRIBE concat_company_and_product;
concat_company_and_product: {company_with_product: chararray}
grunt>
acadgild@localhost:/u...  *Unsaved Document 1...
```

2. TOKENIZE(): It is used to split a string having many words in a tuple. It returns a bag of token tuples of type same as input.

Example: tokenize_company_and_product = FOREACH concat_company_and_product GENERATE TOKENIZE(company_with_product);

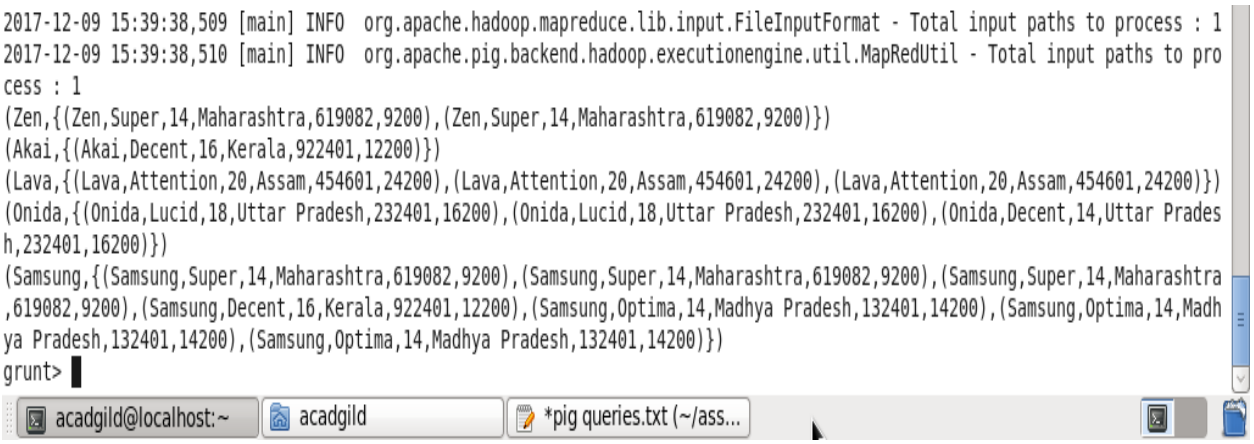
Output: DUMP tokenize_company_and_product;

3. SUM(): As the name implies, this function is used to get sum of all values of a specific column in a single column bag. We need to perform GROUP operation on a column or on all columns, after which we can apply SUM() to get global sum among the groups.

Example query to group data by company_name :

grouped_tv = GROUP filtered_tv BY company_name;

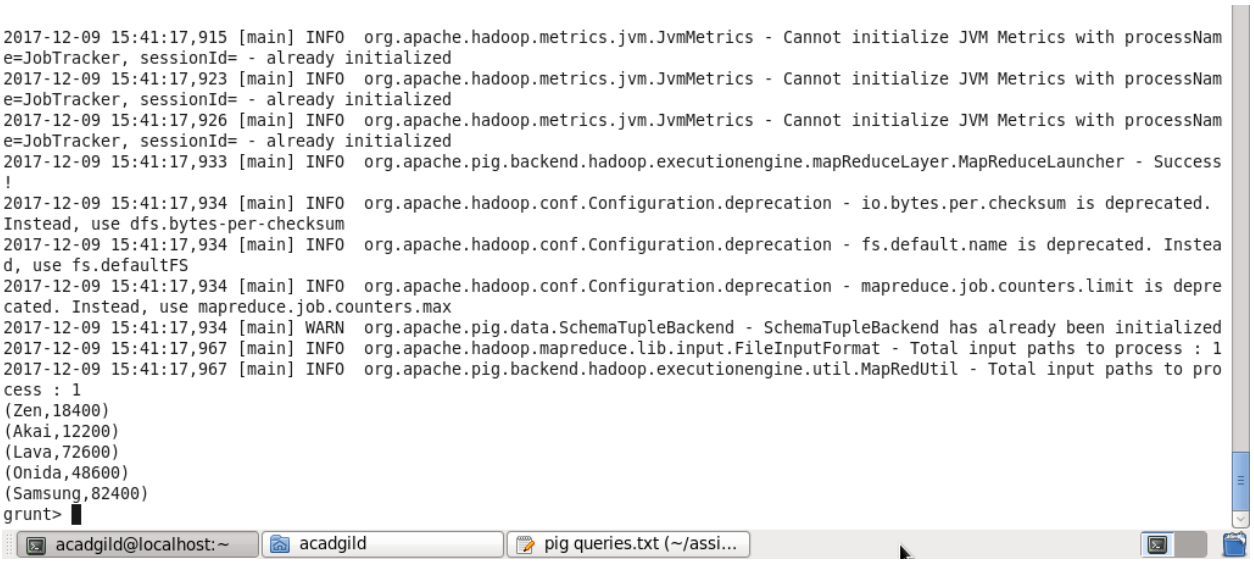
Output: DUMP grouped_tv;



SUM query on each company group based on price:

sum_of_prices = FOREACH grouped_tv GENERATE group, SUM(filtered_tv.price);

Output: DUMP sum_of_prices;

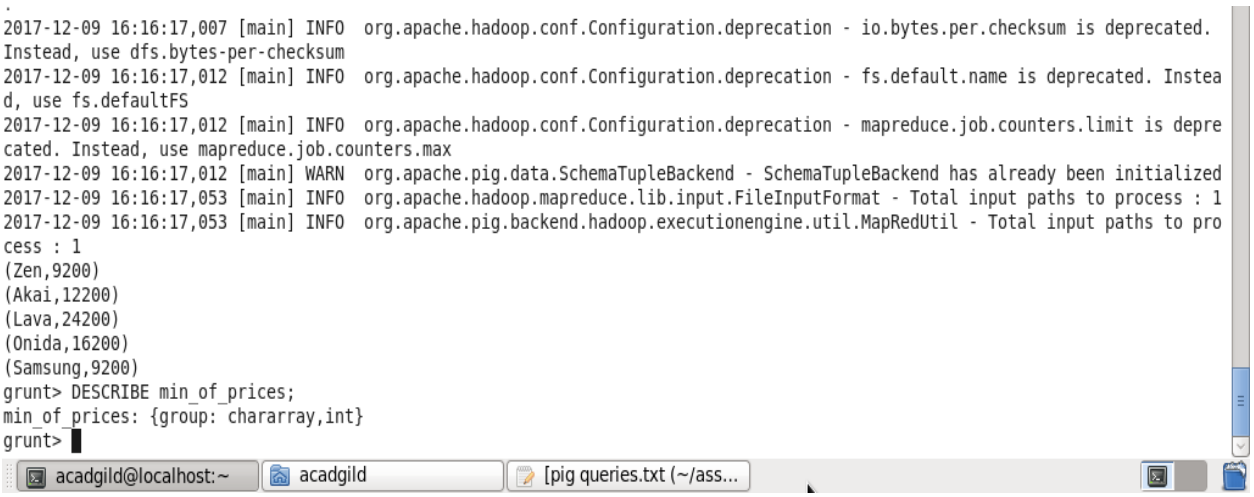


4. MIN(): This function is used to get minimum value for a particular column in a single column bag. Like SUM(), this function is also applied on grouped data to find lowest value of a column from each group.

Example query:

min_of_prices = FOREACH grouped_tv GENERATE group, MIN(filtered_tv.price);

Output: DUMP min_of_prices;

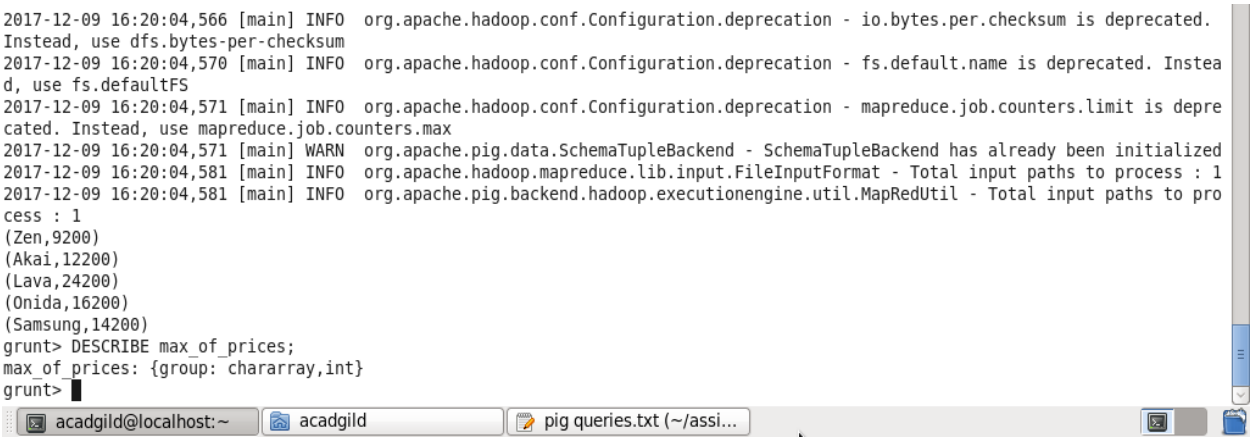


5. MAX(): This function is used to get maximum value for a particular column in a single column bag. This function is applied on grouped data to find highest value of a column from each group.

Example query:

max_of_prices = FOREACH grouped_tv GENERATE group, MAX(filtered_tv.price);

Output: DUMP max_of_prices;

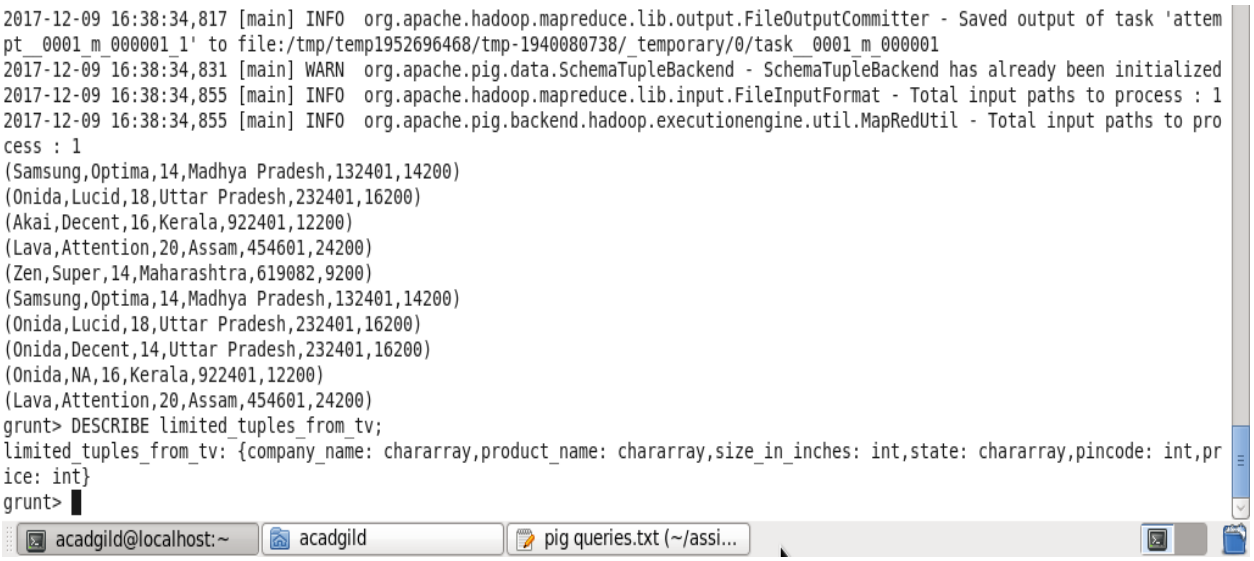


6. LIMIT: This operator is used to get limited number of tuples from a relation. The resulting relation will have same schema as that of source relation.

Example query:

limited_tuples_from_tv = LIMIT tv 10;

Output: DUMP limited_tuples_from_tv;

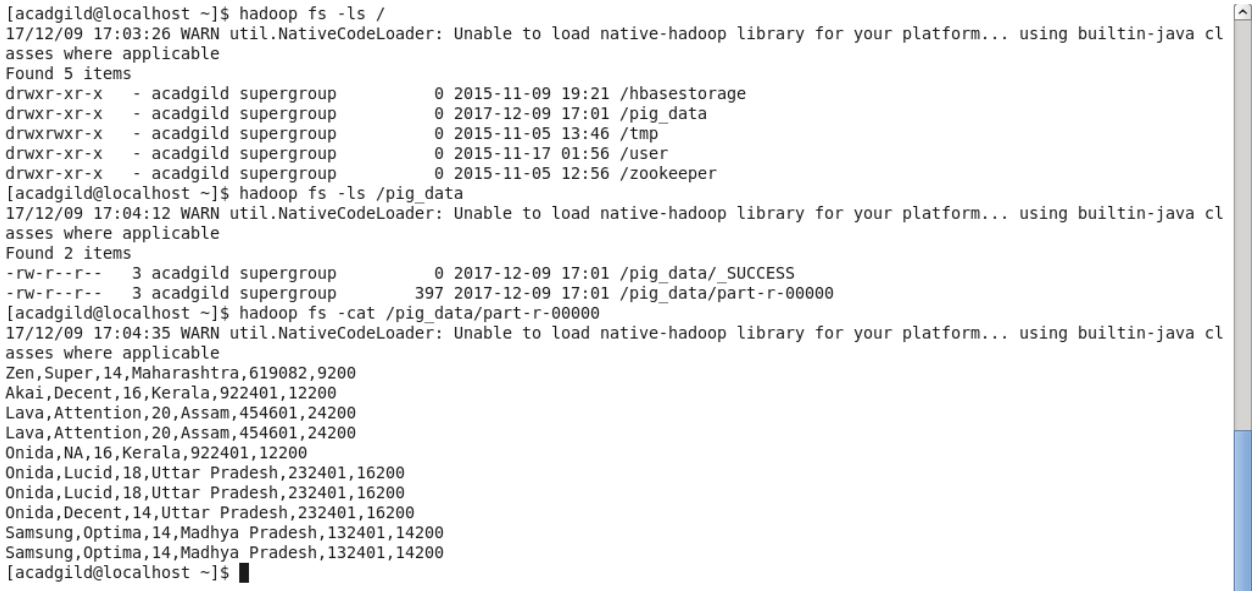


7. STORE: This operator is used to store data of a relation into local file system or to HDFS. Output directory should not exist as STORE command creates one and places the output file into it.

Example query to store data into HDFS:

STORE limited_tuples_from_tv INTO 'hdfs://localhost:9000/pig_data/' USING PigStorage(',');

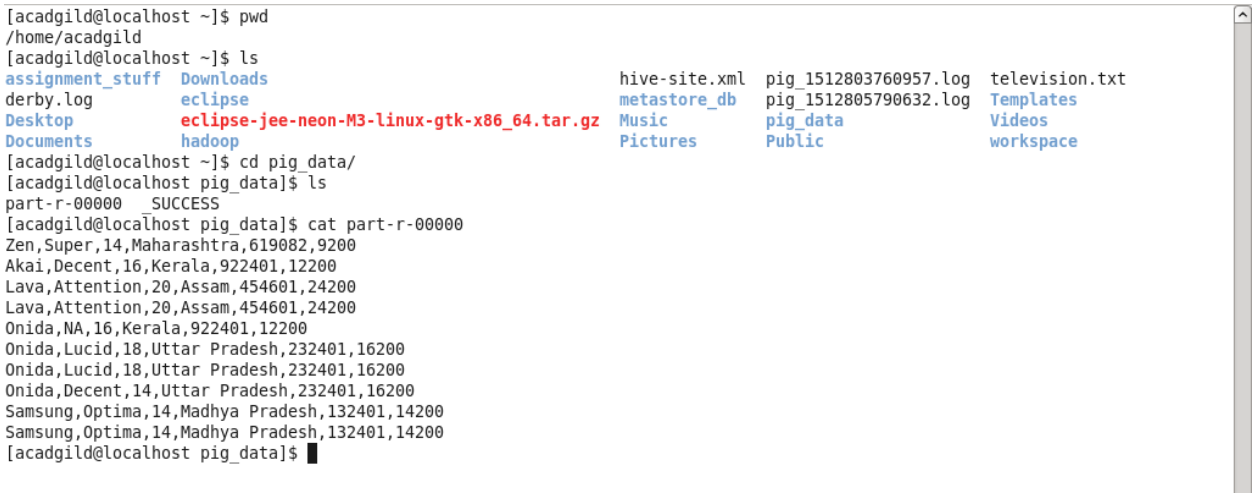
Output:



Example query to store data into local file system:

STORE limited_tuples_from_tv INTO 'file:///home/acadgild/pig_data' USING PigStorage(',');

Output:



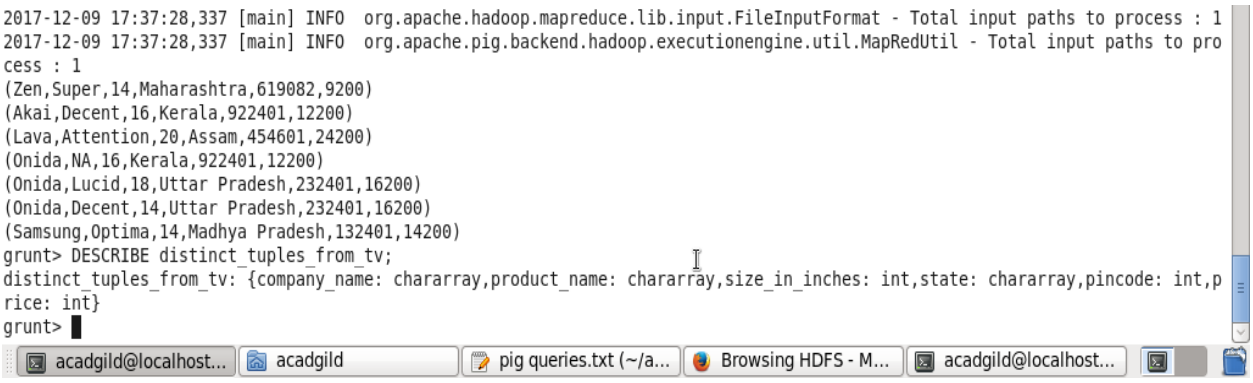
8. DISTINCT: This operator returns unique tuples from a relation removing the redundant ones.

We can see from STORE query output that it has 3 redundant tuples.

Example query:

distinct_tuples_from_tv = DISTINCT limited_tuples_from_tv;

Output:

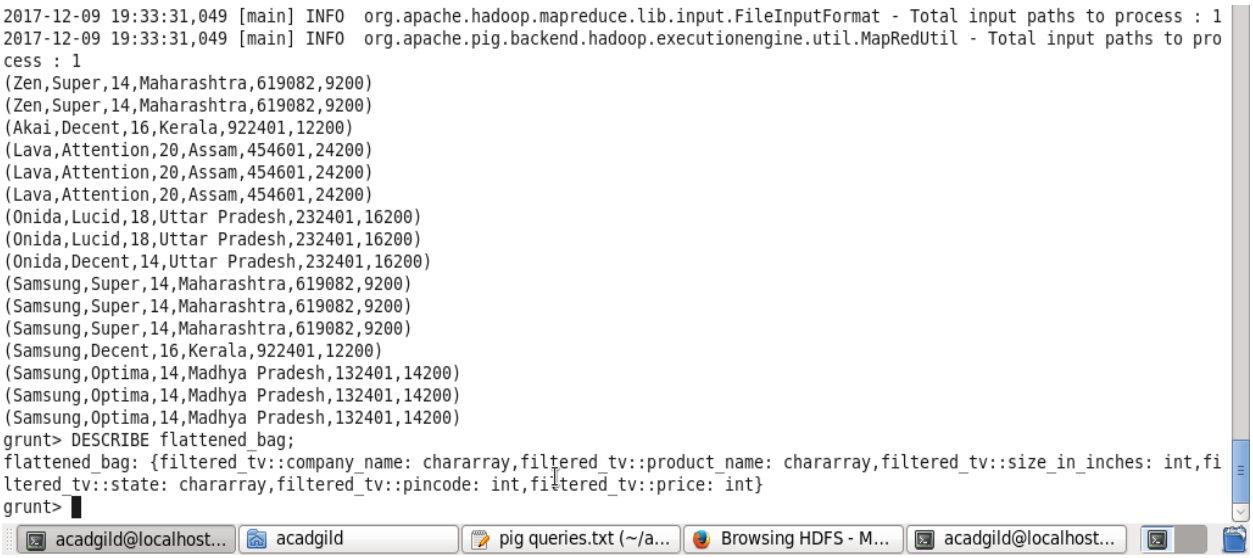


9. FLATTEN: This operator changes the structure of tuples and bags in a relation by un-nesting them. FLATTEN replaces a bag by tuples inside that bag and a tuple by fields of a tuple.

Example query on a bag:

flattened_bag = FOREACH grouped_tv GENERATE FLATTEN(filtered_tv);

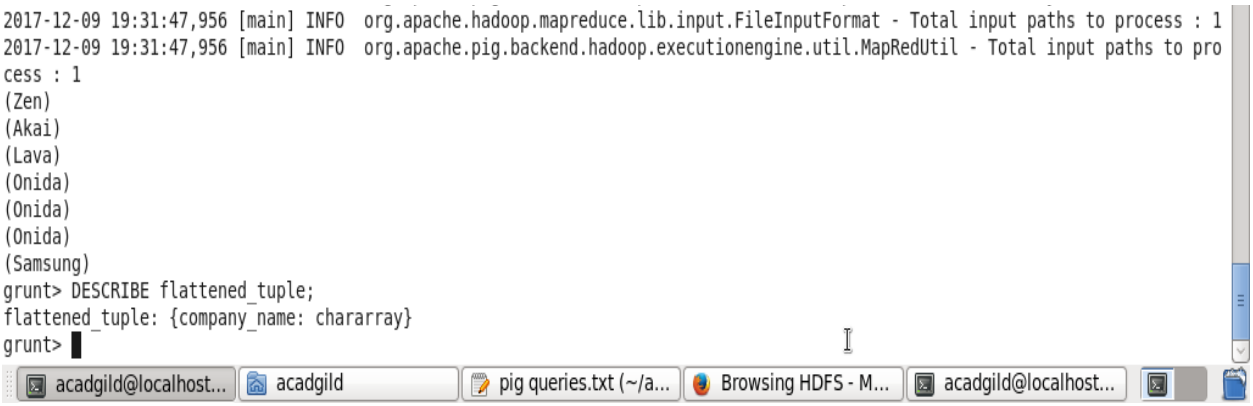
Output: DUMP flattened_bag;



Example query on a tuple:

flattened_tuple = FOREACH distinct_tuples_from_tv GENERATE FLATTEN($0);

Output: DUMP flattened_tuple;



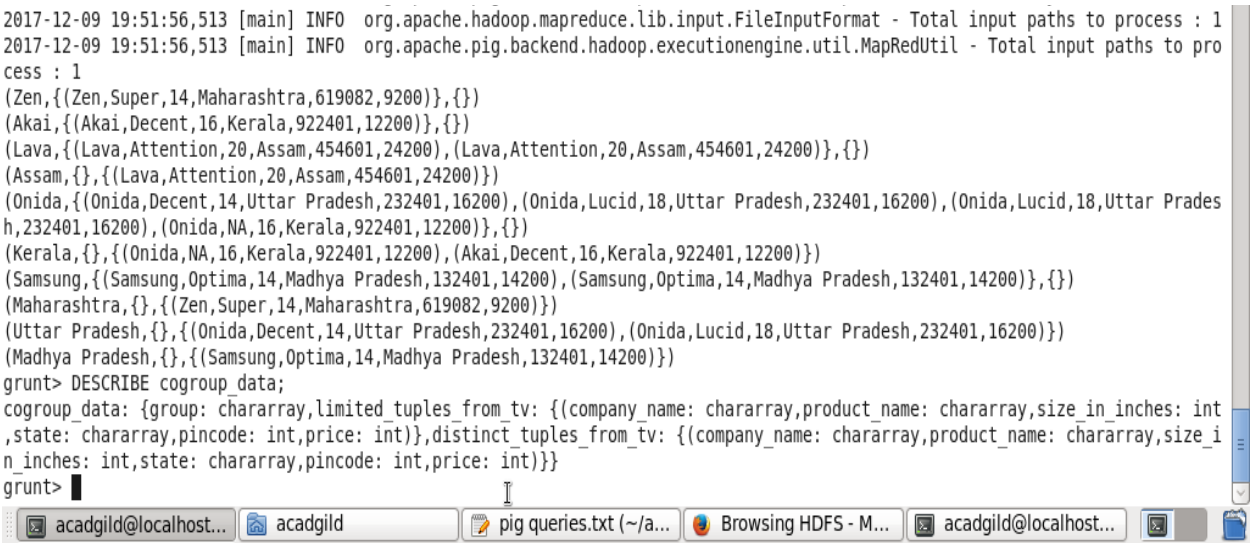
10. IsEmpty(): This function is used to check if a bag or map is empty. If a relation has at least one empty bag or map, it will be stored to an assigned relation.

Using COGROUP operator to group two relations on different columns we can obtain a relation that contains some empty bags.

Example query for COGROUP:

cogroup_data = COGROUP limited_tuples_from_tv BY company_name, distinct_tuples_from_tv by state;

Output: DUMP cogroup_data;



Example query for IsEmpty() taking cogroup_data as input relation:

isempty_result = FILTER cogroup_data BY IsEmpty(limited_tuples_from_tv);

Output: DUMP isempty_result;