Northeastern University

Final Project Report

Topic: Big Data Analysis on Airbnb Dataset

Course Name: Engineering of Big Data
Academic Year: Summer 2020

Submitted By:
Dhruvil Shah
NUID: 001873473

Summarization:

For this project, I am working on 'Airbnb Boston Dataset Analysis'.

I downloaded the dataset from Kaggle from below link:
https://www.kaggle.com/airbnb/boston

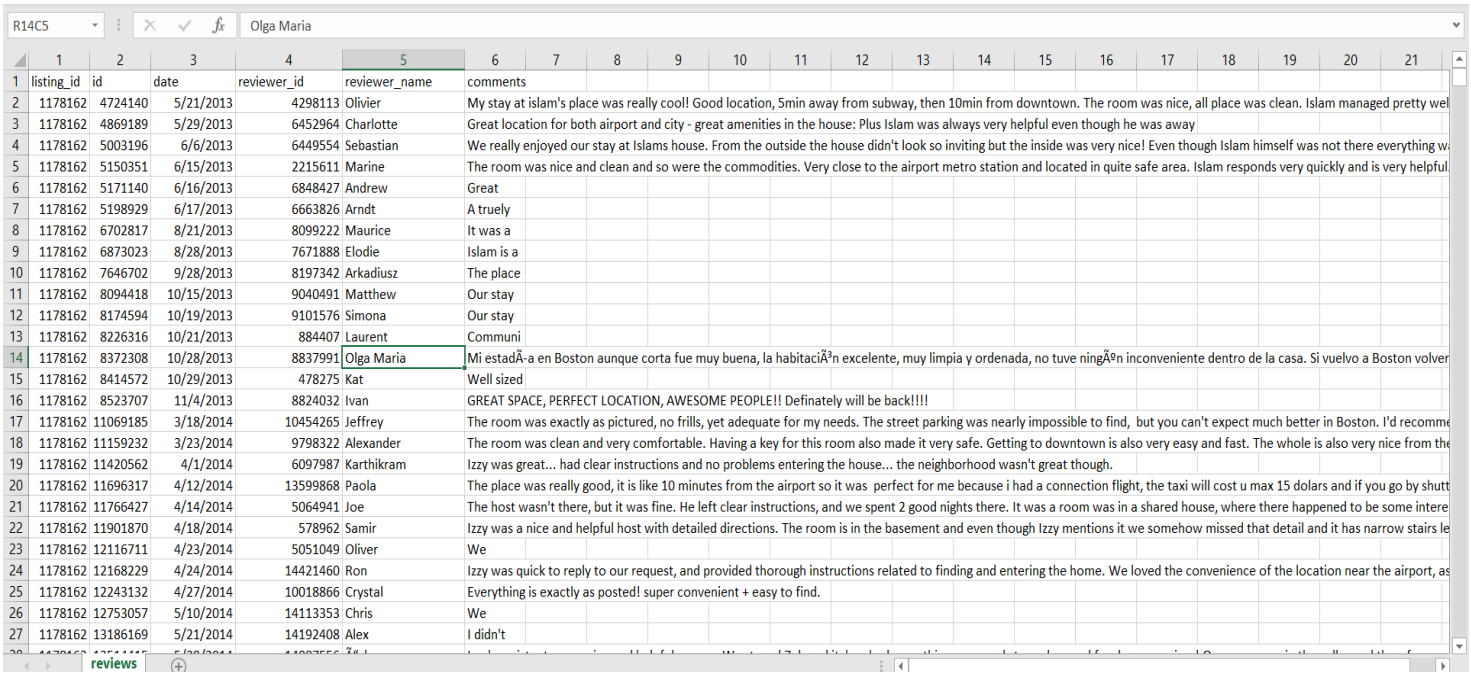
The dataset has 3 csv files namely reviews.csv, calendar.csv and listings.csv

❖ reviews.csv

This file has around a million records.

Attribute used:

- Listing Id
- Review Id
- Reviewer Id
- Reviewer Name
- Comments
- Review Date



R14C5 | Olga Maria

	1	2	3	4	5	6
1	listing_id	id	date	reviewer_id	reviewer_name	comments
2	1178162	4724140	5/21/2013	4298113	Olivier	My stay at islam's place was really cool! Good location, 5min away from subway, then 10min from downtown. The room was nice, all place was clean. Islam managed pretty wel
3	1178162	4869189	5/29/2013	6452964	Charlotte	Great location for both airport and city - great amenities in the house: Plus Islam was always very helpful even though he was away
4	1178162	5003196	6/6/2013	6449554	Sebastian	We really enjoyed our stay at Islams house. From the outside the house didn't look so inviting but the inside was very nice! Even though Islam himself was not there everything w
5	1178162	5150351	6/15/2013	2215611	Marine	The room was nice and clean and so were the commodities. Very close to the airport metro station and located in quite safe area. Islam responds very quickly and is very helpful
6	1178162	5171140	6/16/2013	6848427	Andrew	Great
7	1178162	5198929	6/17/2013	6663826	Arndt	A truely
8	1178162	6702817	8/21/2013	8099222	Maurice	It was a
9	1178162	6873023	8/28/2013	7671888	Elodie	Islam is a
10	1178162	7646702	9/28/2013	8197342	Arkadiusz	The place
11	1178162	8094418	10/15/2013	9040491	Matthew	Our stay
12	1178162	8174594	10/19/2013	9101576	Simona	Our stay
13	1178162	8226316	10/21/2013	884407	Laurent	Communi
14	1178162	8372308	10/28/2013	8837991	Olga Maria	Mi estadÃ-a en Boston aunque corta fue muy buena, la habitaciÃ³n excelente, muy limpia y ordenada, no tuve ningÃºn inconveniente dentro de la casa. Si vuelvo a Boston volver
15	1178162	8414572	10/29/2013	478275	Kat	Well sized
16	1178162	8523707	11/4/2013	8824032	Ivan	GREAT SPACE, PERFECT LOCATION, AWESOME PEOPLE!! Definately will be back!!!!
17	1178162	11069185	3/18/2014	10454265	Jeffrey	The room was exactly as pictured, no frills, yet adequate for my needs. The street parking was nearly impossible to find, but you can't expect much better in Boston. I'd recomme
18	1178162	11159232	3/23/2014	9798322	Alexander	The room was clean and very comfortable. Having a key for this room also made it very safe. Getting to downtown is also very easy and fast. The whole is also very nice from the
19	1178162	11420562	4/1/2014	6097987	Karthikram	Izzy was great... had clear instructions and no problems entering the house... the neighborhood wasn't great though.
20	1178162	11696317	4/12/2014	13599868	Paola	The place was really good, it is like 10 minutes from the airport so it was perfect for me because i had a connection flight, the taxi will cost u max 15 dolars and if you go by shutt
21	1178162	11766427	4/14/2014	5064941	Joe	The host wasn't there, but it was fine. He left clear instructions, and we spent 2 good nights there. It was a room was in a shared house, where there happened to be some intere
22	1178162	11901870	4/18/2014	578962	Samir	Izzy was a nice and helpful host with detailed directions. The room is in the basement and even though Izzy mentions it we somehow missed that detail and it has narrow stairs le
23	1178162	12116711	4/23/2014	5051049	Oliver	We
24	1178162	12168229	4/24/2014	14421460	Ron	Izzy was quick to reply to our request, and provided thorough instructions related to finding and entering the home. We loved the convenience of the location near the airport, as
25	1178162	12243132	4/27/2014	10018866	Crystal	Everything is exactly as posted! super convenient + easy to find.
26	1178162	12753057	5/10/2014	14113353	Chris	We
27	1178162	13186169	5/21/2014	14192408	Alex	I didn't

reviews

❖ listings.csv

Attributes Used:

- Listing Id
- Host Since
- Neighborhood
- Listing URL
- Price
- Review Scores Ratings
- Name
- Summary
- Description

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	id	host_since	neighbourhood_	listing_url	scrape_id	last_scraped	price	review_scores_ratin	name	summary	space	descriptior	experience	neighborh	notes
2	12147973	4/15/2015	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$250.00		Sunny Bungalow in the City	Cozy, sunn	The house	Cozy, sunn	none	Roslindale	is quiet
3	3075044	6/7/2012	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$65.00	94	Charming room in pet friendly apt	Charming (	Small but (	Charming (	none	The room	If you (
4	6976	5/11/2009	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$65.00	98	Mexican Folk Art Haven in Boston	Come stay	Come stay	Come stay	none	The LOCAT	I am in
5	1436513	4/21/2013	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$75.00	100	Spacious Sunny Bedroom Suite in Historic Home	Come exp	Most place	Come exp	none	Roslindale	Please
6	7651065	5/11/2014	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$79.00	99	Come Home to Boston	My comfy,	Clean, attr	My comfy,	none	I love the p	I have
7	12386020	3/23/2016	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$75.00	100	Private Bedroom + Great Coffee	Super com	Our sunny	Super com	none	We love our	corn
8	5706985	5/25/2013	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$100.00	90	New Lrg Studio apt 15 min to Boston	It's a 5 mir	The whole	It's a 5 mir	none	Roslindale	Inform
9	2843445	8/5/2012	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$75.00	96	"Tranquility" on "Top of the Hill"	We can ac	We provid	We can ac	none	Our neighk	We lov
10	753446	10/24/2012	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$58.00	96	6 miles away from downtown Boston!	Nice and c	Nice and c	Nice and c	none	Roslindale	is a pri
11	849408	12/18/2012	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$229.00	94	Perfect & Practical Boston Rental	This is a cc	Perfect ap	This is a cc	none	This neigh	Please
12	12023024	3/24/2016	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$60.00	80	Cozy room in a well located house	The room is in a singl		The room	none		
13	1668313	9/14/2013	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$57.00	94	Room in Rozzie-Twin Bed-Full Bath	Quiet second floor be		Quiet secc	none	Our neighk	On wo
14	2684840	3/31/2014	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$93.00	100	Updated, spacious living in Rozzie	Clean, sunny 2 bedroc		Clean, sun	none		
15	13547301	2/4/2016	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$150.00	97	2 Bedroom Apartment in Boston	My place is close to P		My place i	none		
16	5434353	10/27/2013	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$145.00	91	Quiet Beauty in Boston	Enjoy the f	Large livin	Enjoy the f	none	Quiet	Roslindale,
17	225979	9/18/2011	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$60.00	96	Cozy Room & Fresh Roasted Coffee	Fresh hom	This is a ch	Fresh hom	none	This is a ve	If you
18	3420384	3/9/2011	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$165.00	100	Convient, Safe and Comfortable	Located al	Extra large	Located al	none	Village 1 m	Access
19	13512930	3/31/2014	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$75.00	100	Roslindale Beauty next to Arboretum	Clean, sunny 1 bedroc		Clean, sun	none		
20	7482195	5/30/2015	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$49.00		Private room near bus stop	A handsome colonial		A handson	none		
21	7252607	5/30/2015	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$49.00	95	Private Room near Public Transport	A handsome colonial		A handson	none		
22	2583074	3/17/2014	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$40.00	99	Cozy room in a charming villa.	My home i	The room	My home i	none	This a nice	I speak
23	13251243	5/28/2016	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$120.00	98	Arborside Guest Cottage	Lovely sun	Today the	Lovely sun	none	This Bosto	Enclos
24	225834	9/18/2011	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$70.00	98	Skyline View to Boston	High atop	After a go	High atop	none	This is a ve	If you
25	6400432	10/29/2014	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$150.00	88	Spacious 3 bedroom cape cod style	Three bedı	This Cape (	Three bedı	none		
26	5498472	2/28/2015	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$175.00	100	4BD/3.5BA, Perfect for families!	Our spacio	Our 2,400	Our spacio	none	Roslindale	Our to
27	894539	1/19/2013	Roslindale	https://www.airbnl	2.02E+13	9/7/2016	$95.00	96	Private room in house		Large prive	Large prive	none		

listings

❖ calendar.csv

Attribute Used:

- ➢ Listing Id
- ➢ Available
- ➢ Date
- ➢ Price

	1	2	3	4	5
1	listing_id	availabl	date	available	price
367	3075044	t	8/22/2017	t	$65.00
368	3075044	t	8/21/2017	t	$65.00
369	3075044	t	8/20/2017	t	$65.00
370	3075044	t	8/19/2017	t	$75.00
371	3075044	t	8/18/2017	t	$75.00
372	3075044	t	8/17/2017	t	$65.00
373	3075044	t	8/16/2017	t	$65.00
374	3075044	t	8/15/2017	t	$65.00
375	3075044	t	8/14/2017	t	$65.00
376	3075044	t	8/13/2017	t	$65.00
377	3075044	t	8/12/2017	t	$75.00
378	3075044	t	8/11/2017	t	$75.00
379	3075044	t	8/10/2017	t	$65.00
380	3075044	t	8/9/2017	t	$65.00
381	3075044	t	8/8/2017	t	$65.00
382	3075044	t	8/7/2017	t	$65.00
383	3075044	t	8/6/2017	t	$65.00
384	3075044	t	8/5/2017	t	$75.00
385	3075044	t	8/4/2017	t	$75.00
386	3075044	t	8/3/2017	t	$65.00
387	3075044	t	8/2/2017	t	$65.00
388	3075044	t	8/1/2017	t	$65.00
389	3075044	t	7/31/2017	t	$65.00
390	3075044	t	7/30/2017	t	$65.00
391	3075044	t	7/29/2017	t	$75.00
392	3075044	t	7/28/2017	t	$75.00

calendar

Overview of Analysis:

No	Analysis Description	Analysis Used
1	Number of business in a city	**MongoDB**
2	Replicated join to join listings and review datasets	**Apache Pig – Replicated Join**
3	Find listings with more than 50 reviews	**Apache Pig**
4	To find top 7 listings based on ratings for each neighborhood.	**Data Cleaning using PIG + Apache Hive**
5	To perform Partitioning based on each year in the dataset	**MapReduce – Data Organization Partitioning**
6	To find number of hosts joining Airbnb for each year in the dataset	**MapReduce – Chaining Map Reduce Summarization**
7	To Sort the Listings according to reviews received per year	**MapReduce – Secondary Sort**
8	To find the top 20 listings according to ratings given	**MapReduce – Filtering Techniques Top n Filtering Pattern**
9	To perform Inner Join using Apache PySpark	**Apache Spark – PySpark**
10	To find total available days for each Listing	**Apache Spark – PySpark**
11	Mahout Recommendation Analysis	**Data Cleaning using PIG + Mahout Recommendation**

Getting Started

STARTING HADOOP

1. Navigate to Hadoop Directory:
cd /usr/local/bin/hadoop-2.10.0/sbin

2. Start Hadoop daemons
./start-all.sh

3. Check Hadoop Daemons
JPS

COPYING FILE FROM LOCAL FILE SYSTEM TO HDFS

```
7073 SecondaryNameNode
6658 NameNode
7683 Jps
7557 NodeManager
7211 ResourceManager
6845 DataNode
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/sbin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/sbin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/sbin$ cd ..
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0$ cd bin
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$ ./hadoop fs -mkdir /Airbnb/Data
set
mkdir: `/Airbnb/Dataset': No such file or directory
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$ ./hadoop fs -mkdir /Airbnb
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$ ./hadoop fs -mkdir /Airbnb/Data
set
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
```

```
dns7665@ubuntu: /usr/local/bin/hadoop-2.10.0/bin
mkdir: `/Airbnb/Dataset': No such file or directory
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$ ./hadoop fs -mkdir /Airbnb
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$ ./hadoop fs -mkdir /Airbnb/Data
set
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$ ./hadoop fs -put ~/D
Desktop/   Documents/ Downloads/
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$ ./hadoop fs -put ~/D
Desktop/   Documents/ Downloads/
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$ ./hadoop fs -put ~/Downloads/Ai
rbnb_boston_Dataset/
calendar.csv  listings.csv  reviews.csv
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$ ./hadoop fs -put ~/Downloads/Ai
rbnb_boston_Dataset/
calendar.csv  listings.csv  reviews.csv
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$ ./hadoop fs -put ~/Downloads/Ai
rbnb_boston_Dataset/
calendar.csv  listings.csv  reviews.csv
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$ ./hadoop fs -put ~/Downloads/Ai
rbnb_boston_Dataset/ /Airbnb/Dataset
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
```

Hadoop Overview Datanodes Datanode Volume Failures Snapshot Startup Progress Utilities

Browse Directory

/Airbnb/Dataset/Airbnb_boston_Dataset Go!

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	dns7665	supergroup	31.97 MB	Aug 03 15:56	1	128 MB	calendar.csv
-rw-r--r--	dns7665	supergroup	14.07 MB	Aug 03 15:56	1	128 MB	listings.csv
-rw-r--r--	dns7665	supergroup	26.05 MB	Aug 03 15:56	1	128 MB	reviews.csv

Showing 1 to 3 of 3 entries Previous 1 Next

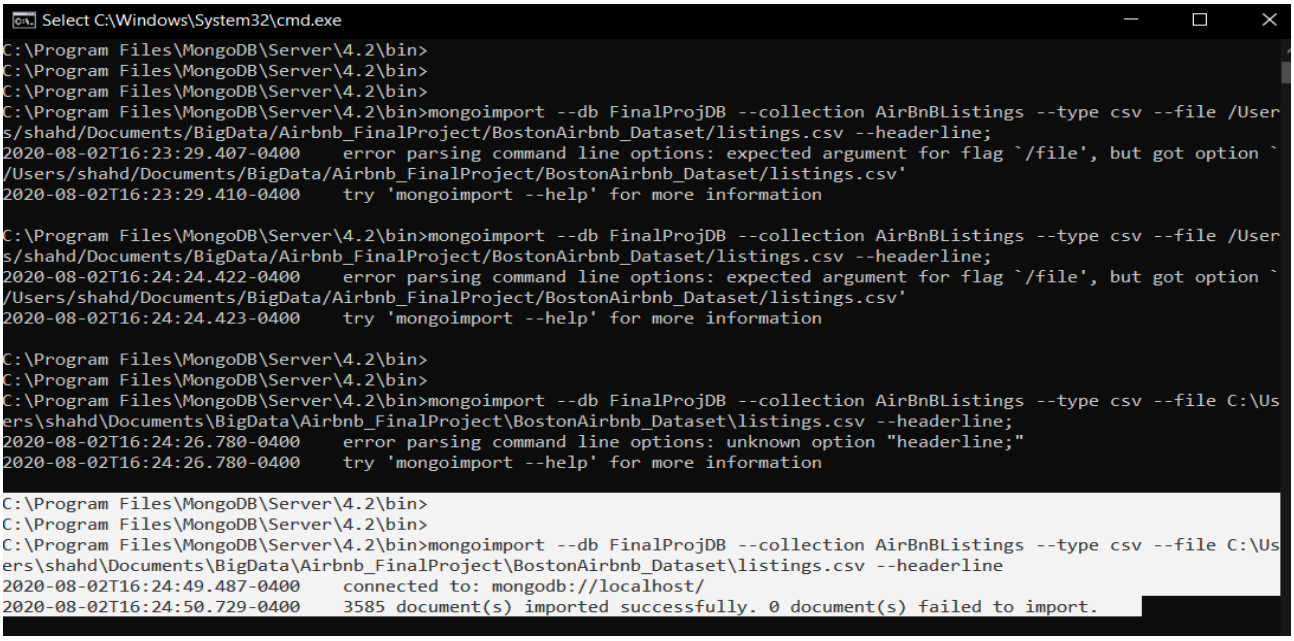
Hadoop, 2019.

MONGODB ANALYSIS:

Analysis 1: To find the count of number of business per city:

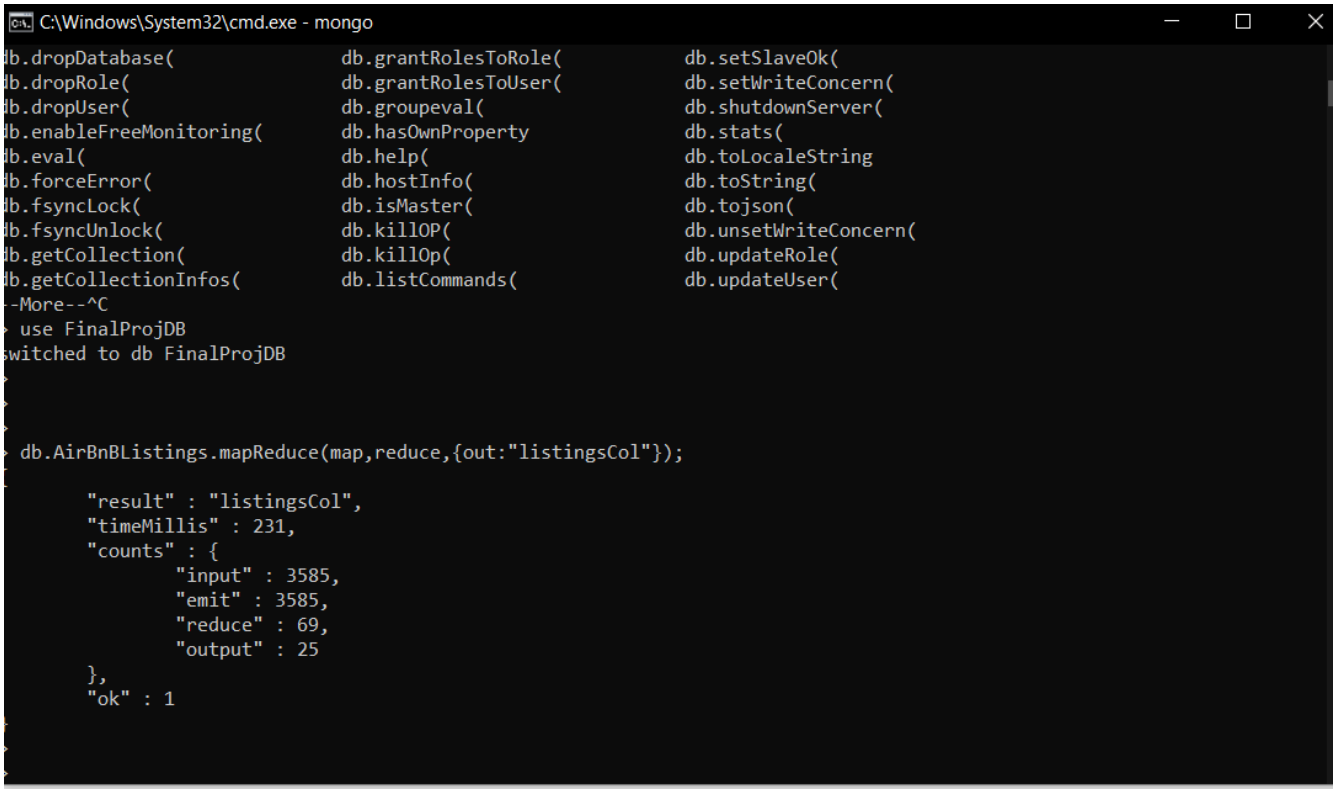
1. Create Mongo Database and Create Collection:

2. Import Data from CSV File



3. Write Map & Reduce Functions to perform Analysis and execute:

```
Map:
      function (){
            emit({neighborhood:
      this.neighbourhood_cleansed},{Number_of_listings:1})
      }

Reduce:
      function(key,countObjVals){

      reducedVal = { Number_of_listings:0 };
       for (var idx = 0; idx < countObjVals.length; idx++) {

            if(typeof countObjVals[idx] != "undefined")
                  reducedVal.Number_of_listings =
      reducedVal.Number_of_listings+countObjVals[idx].Number_of_listings;
            }

       return{Number_of_listings:reducedVal.Number_of_listings};
      }
```



4. Output:

```
> db.listingsCol.find();
{ "_id" : { "neightbourhood" : "Allston" }, "value" : { "Number_of_listings" : 260 } }
{ "_id" : { "neightbourhood" : "Back Bay" }, "value" : { "Number_of_listings" : 302 } }
{ "_id" : { "neightbourhood" : "Bay Village" }, "value" : { "Number_of_listings" : 24 } }
{ "_id" : { "neightbourhood" : "Beacon Hill" }, "value" : { "Number_of_listings" : 194 } }
{ "_id" : { "neightbourhood" : "Brighton" }, "value" : { "Number_of_listings" : 185 } }
{ "_id" : { "neightbourhood" : "Charlestown" }, "value" : { "Number_of_listings" : 111 } }
{ "_id" : { "neightbourhood" : "Chinatown" }, "value" : { "Number_of_listings" : 71 } }
{ "_id" : { "neightbourhood" : "Dorchester" }, "value" : { "Number_of_listings" : 269 } }
{ "_id" : { "neightbourhood" : "Downtown" }, "value" : { "Number_of_listings" : 172 } }
{ "_id" : { "neightbourhood" : "East Boston" }, "value" : { "Number_of_listings" : 150 } }
{ "_id" : { "neightbourhood" : "Fenway" }, "value" : { "Number_of_listings" : 290 } }
{ "_id" : { "neightbourhood" : "Hyde Park" }, "value" : { "Number_of_listings" : 31 } }
{ "_id" : { "neightbourhood" : "Jamaica Plain" }, "value" : { "Number_of_listings" : 343 } }
{ "_id" : { "neightbourhood" : "Leather District" }, "value" : { "Number_of_listings" : 5 } }
{ "_id" : { "neightbourhood" : "Longwood Medical Area" }, "value" : { "Number_of_listings" : 9 } }
{ "_id" : { "neightbourhood" : "Mattapan" }, "value" : { "Number_of_listings" : 24 } }
{ "_id" : { "neightbourhood" : "Mission Hill" }, "value" : { "Number_of_listings" : 124 } }
{ "_id" : { "neightbourhood" : "North End" }, "value" : { "Number_of_listings" : 143 } }
{ "_id" : { "neightbourhood" : "Roslindale" }, "value" : { "Number_of_listings" : 56 } }
{ "_id" : { "neightbourhood" : "Roxbury" }, "value" : { "Number_of_listings" : 144 } }
Type "it" for more
> it
{ "_id" : { "neightbourhood" : "South Boston" }, "value" : { "Number_of_listings" : 174 } }
{ "_id" : { "neightbourhood" : "South Boston Waterfront" }, "value" : { "Number_of_listings" : 83 } }
{ "_id" : { "neightbourhood" : "South End" }, "value" : { "Number_of_listings" : 326 } }
{ "_id" : { "neightbourhood" : "West End" }, "value" : { "Number_of_listings" : 49 } }
{ "_id" : { "neightbourhood" : "West Roxbury" }, "value" : { "Number_of_listings" : 46 } }
>
```

APACHE PIG ANALYSIS

Analysis 2: To implement Replicated join to join reviews and listings dataset

PIG Commands:

```
1. reviews = LOAD
   '/home/dns7665/Downloads/Airbnb_boston_Dataset/reviews.csv' USING
   org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'NO_MULTILINE',
   'UNIX', 'SKIP_INPUT_HEADER');

2. filterReview = FOREACH reviews GENERATE (chararray) $0 as listing_Id,
   (chararray) $1 as review_Id, (chararray) $2 as date, (chararray) $3
   as reviewer_Id, (chararray) $4 as reviewer_name, (chararray) $5 as
   ReviewContent;


3. limitFilter = LIMIT filterReview 5;

4. describe filterReview;

5. dump limitFilter;

6. listings = LOAD
   '/home/dns7665/Downloads/Airbnb_boston_Dataset/listings.csv' USING
   org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'NO_MULTILINE',
   'UNIX', 'SKIP_INPUT_HEADER');

7. filterList = FOREACH listings GENERATE (chararray) $0 as listing_Id,
   (chararray) $1 as listing_Url, (chararray) $4 as listing_name,
   (chararray) $7 as listing_desc;

8. describe filterList;

9. limitList = LIMIT filterList 5;

10.     dump limitList;

11.     repJoin = Join filterReview by listing_Id, filterList by
   listing_Id USING 'replicated';

12.     limitJoin = LIMIT repJoin 5;

13.     dump limitJoin;

14.     STORE repJoin INTO
   '/home/dns7665/Documents/ReplicatedJoin_Pig.txt' Using
   PigStorage(',');
```

```
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt> reviews = LOAD '/home/dns7665/Downloads/Airbnb_boston_Dataset/reviews.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'NO_MULTIL
INE', 'UNIX', 'SKIP_INPUT_HEADER');
2020-08-07 19:22:04,927 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-
checksum
grunt>
```

```
grunt> filterReview = FOREACH reviews GENERATE (chararray) $0 as listing_Id, (chararray) $1 as review_Id, (chararray) $2 as date, (chararray) $3 as revi
ewer_Id, (chararray) $4 as reviewer_name, (chararray) $5 as ReviewContent;
grunt> limitFilter = LIMIT filterReview 5;
grunt> describe filterReview;
filterReview: {listing_Id: chararray,review_Id: chararray,date: chararray,reviewer_Id: chararray,reviewer_name: chararray,ReviewContent: chararray}
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
grunt>
```

```
grunt> listings = LOAD '/home/dns7665/Downloads/Airbnb_boston_Dataset/listings.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'NO_MULT
ILINE', 'UNIX', 'SKIP_INPUT_HEADER');
2020-08-07 19:24:15,087 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-
checksum
grunt>
```

```
grunt> filterList = FOREACH listings GENERATE (chararray) $0 as listing_Id, (chararray) $1 as listing_Url, (chararray) $4 as listing_name, (chararray) $
7 as listing_desc;
grunt> describe filterList;
filterList: {listing_Id: chararray,listing_Url: chararray,listing_name: chararray,listing_desc: chararray}
grunt>  limitList = LIMIT filterList 5;
grunt>
grunt>
grunt>
```

```
grunt> dump limitList;
2020-08-07 19:25:04,384 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: LIMIT
2020-08-07 19:25:04,390 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-
checksum
2020-08-07 19:25:04,390 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2020-08-07 19:25:04,390 [main] INFO  org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, Cons
tantCalculator, GroupByConstParallelSetter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, MergeForEach, NestedLimitOptimizer, PartitionFilterOptimi
zer, PredicatePushdownOptimizer, PushDownForEachFlatten, PushUpFilter, SplitFilter, StreamTypeCastInserter]}
2020-08-07 19:25:04,394 [main] INFO  org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter - File Output Committer Algorithm version is 1
2020-08-07 19:25:04,408 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2020-08-07 19:25:04,409 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2020-08-07 19:25:04,409 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
2020-08-07 19:25:04,410 [main] INFO  org.apache.hadoop.mapreduce.lib.output.FileOutputCommitter - Saved output of task 'attempt__0001_m_000001_1' to fil
e:/tmp/temp-208891927/tmp900052049/_temporary/0/task__0001_m_000001
2020-08-07 19:25:04,418 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2020-08-07 19:25:04,423 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2020-08-07 19:25:04,423 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(12147973,4/15/2015,2.02E+13,Sunny Bungalow in the City)
(3075044,6/7/2012,2.02E+13,Charming room in pet friendly apt)
(6976,5/11/2009,2.02E+13,Mexican Folk Art Haven in Boston)
(,,,)
(In addition to the above, I run web sites for small companies,,)
grunt>
```

```
already initialized
2020-08-07 19:26:25,834 [main] INFO  org.apache.hadoop.metrics.jvm.JvmMetrics - Cannot initialize JVM Metrics with processName=JobTracker, sessionId= -
already initialized
2020-08-07 19:26:25,835 [main] WARN  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Encountered Warning ACCESSING_NON_
EXISTENT_FIELD 11690 time(s).
2020-08-07 19:26:25,835 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MapReduceLauncher - Success!
2020-08-07 19:26:25,836 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-
checksum
2020-08-07 19:26:25,836 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2020-08-07 19:26:25,851 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2020-08-07 19:26:25,851 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(1178162,4724140,2013-05-21,4298113,Olivier,My stay at islam's place was really cool! Good location, 5min away from subway, then 10min from downtown. Th
e room was nice, all place was clean. Islam managed pretty well our arrival, even if it was last minute ;) i do recommand this place to any airbnb user
;),1178162,11/14/2011,2.02E+13,Private Room Great Location Boston)
(1178162,4869189,2013-05-29,6452964,Charlotte,Great location for both airport and city - great amenities in the house: Plus Islam was always very helpfu
l even though he was away,1178162,11/14/2011,2.02E+13,Private Room Great Location Boston)
(1178162,5003196,2013-06-06,6449554,Sebastian,We really enjoyed our stay at Islams house. From the outside the house didn't look so inviting but the ins
ide was very nice! Even though Islam himself was not there everything was prepared for our arrival. The airport T Station is only a 5-10 min walk away.
The only little issue was that all the people in the house had to share one bathroom. But it was not really a problem and it worked out fine. We would r
ecommend Islams place for a stay in Boston. ,1178162,11/14/2011,2.02E+13,Private Room Great Location Boston)
(1178162,5150351,2013-06-15,2215611,Marine,The room was nice and clean and so were the commodities. Very close to the airport metro station and located
in quite safe area. Islam responds very quickly and is very helpful. I would recommend it. ,1178162,11/14/2011,2.02E+13,Private Room Great Location Bost
on)
(1178162,5171140,2013-06-16,6848427,Andrew,Great location. Just 5 mins walk from the Airport Station. Good food nearby.,1178162,11/14/2011,2.02E+13,Priv
ate Room Great Location Boston)
grunt>
```

```
grunt>
grunt>
grunt> repJoin = Join filterReview by listing_Id, filterList by listing_Id USING 'replicated';
grunt> limitJoin = LIMIT repJoin 5;
grunt>
grunt>
grunt>
grunt>
grunt>
```

```
grunt> STORE repJoin INTO '/home/dns7665/Documents/ReplicatedJoin_Pig.txt' Using PigStorage(',');
2020-08-07 19:28:13,999 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-
checksum
2020-08-07 19:28:14,020 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - mapred.textoutputformat.separator is deprecated. Instead, use ma
preduce.output.textoutputformat.separator
2020-08-07 19:28:14,038 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: REPLICATED_JOIN
2020-08-07 19:28:14,046 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-
checksum
2020-08-07 19:28:14,046 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2020-08-07 19:28:14,046 [main] INFO  org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, Cons
tantCalculator, GroupByConstParallelSetter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, MergeForEach, NestedLimitOptimizer, PartitionFilterOptimi
zer, PredicatePushdownOptimizer, PushDownForEachFlatten, PushUpFilter, SplitFilter, StreamTypeCastInserter]}
2020-08-07 19:28:14,051 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimis
tic? false
2020-08-07 19:28:14,052 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size before optimization
: 2
2020-08-07 19:28:14,052 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after optimization:
```

Analysis 3: To find the Listings with more than 50 reviews

1. LOAD the review dataset using PigStorage and skip header
2. Filtering the review dataset
3. LOAD the Listings dataset using PigStorage and skip header
4. Filtering the listing dataset
5. Generating distinct data from filtered listings
6. Grouping the filtered review by listing Id
7. Generating review data count
8. Filtering review data according to the count
9. Storing the output file to HDFS

```
1  reviews = LOAD '/home/dns7665/Downloads/Airbnb_boston_Dataset/reviews.csv' USING
org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'NO_MULTILINE', 'UNIX', 'SKIP_INPUT_HEADER');
2  filterReview = FOREACH reviews GENERATE (chararray) $0 as listing_Id, (chararray) $1 as review_Id;
3  limitR = LIMIT filterReview 5;
4  limitR = LIMIT filterReview 50;
5  filterReview = FOREACH reviews GENERATE (int) $0 as listing_Id, (int) $1 as review_Id;
6  limitR = LIMIT filterReview 50;
7  distinct_data = DISTINCT limitR;
8  distinct_data = DISTINCT filterReview;
9  reviewGroup = Group distinct_data by listing_Id;
10  reviewCount = FOREACH reviewGroup GENERATE group as listing_Id, COUNT(distinct_data) as
reviews;
11  resultGenerator = FOREACH reviewCount GENERATE listing_Id, reviews;
12  limitres = LIMIT resultGenerator 10;
13  finalResult = FILTER resultGenerator by reviews > 50;
14 STORE finalResult INTO '/home/dns7665/Documents/ListingsWithMoreThan50Rev.txt' Using
PigStorage(',');
```

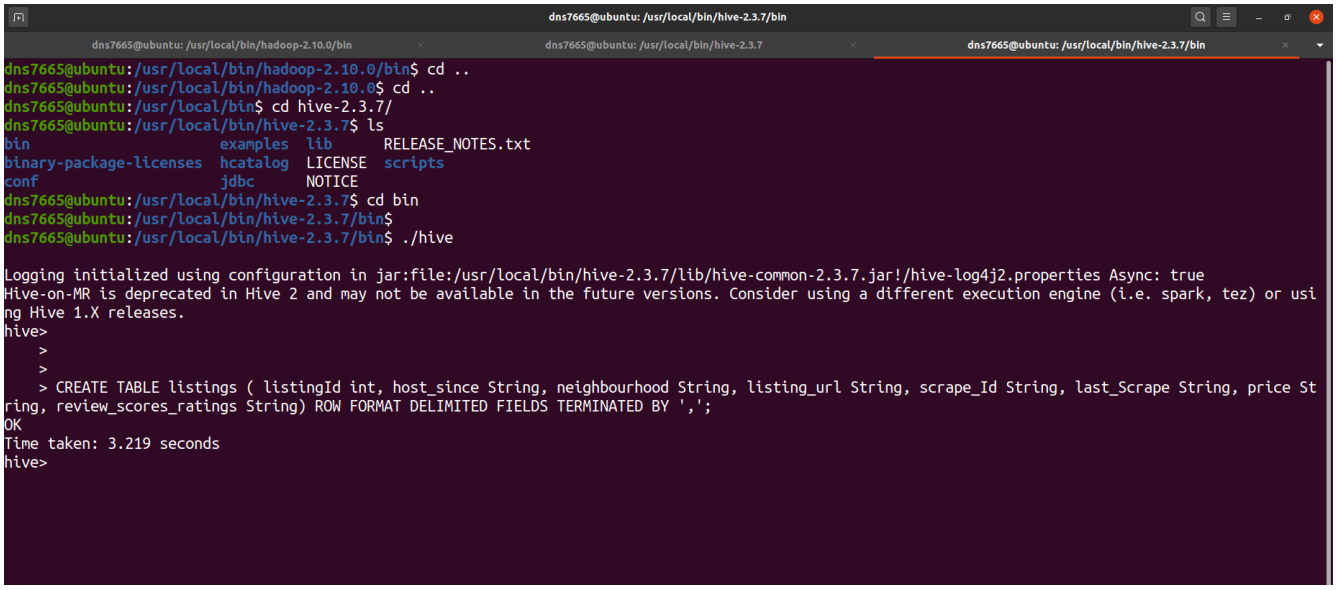
```
grunt> finalResult = FILTER resultGenerator by reviews > 50;
2020-08-07 19:46:14,014 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_LONG 1 time(s).
grunt>
grunt>
grunt>
grunt> dump finalResult;
2020-08-07 19:46:18,739 [main] WARN  org.apache.pig.newplan.BaseOperatorPlan - Encountered Warning IMPLICIT_CAST_TO_LONG 1 time(s).
2020-08-07 19:46:18,740 [main] INFO  org.apache.pig.tools.pigstats.ScriptState - Pig features used in the script: GROUP_BY,DISTINCT,FILTER
2020-08-07 19:46:18,748 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-
checksum
2020-08-07 19:46:18,748 [main] INFO  org.apache.pig.data.SchemaTupleBackend - Key [pig.schematuple] was not set... will not generate code.
2020-08-07 19:46:18,748 [main] INFO  org.apache.pig.newplan.logical.optimizer.LogicalPlanOptimizer - {RULES_ENABLED=[AddForEach, ColumnMapKeyPrune, Cons
tantCalculator, GroupByConstParallelSetter, LimitOptimizer, LoadTypeCastInserter, MergeFilter, MergeForEach, NestedLimitOptimizer, PartitionFilterOptimi
zer, PredicatePushdownOptimizer, PushDownForEachFlatten, PushUpFilter, SplitFilter, StreamTypeCastInserter]}
2020-08-07 19:46:18,764 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-
checksum
2020-08-07 19:46:18,766 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MRCompiler - File concatenation threshold: 100 optimis
tic? false
2020-08-07 19:46:18,767 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.CombinerOptimizerUtil - Choosing to move algebraic foreach to co
mbiner
2020-08-07 19:46:18,767 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size before optimization
: 2
2020-08-07 19:46:18,768 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.MultiQueryOptimizer - MR plan size after optimization:
 2
2020-08-07 19:46:18,773 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-
checksum
2020-08-07 19:46:18,774 [main] INFO  org.apache.hadoop.metrics.jvm.JvmMetrics - Cannot initialize JVM Metrics with processName=JobTracker, sessionId= -
already initialized
2020-08-07 19:46:18,775 [main] INFO  org.apache.pig.tools.pigstats.mapreduce.MRScriptState - Pig script settings are added to the job
2020-08-07 19:46:18,775 [main] INFO  org.apache.pig.backend.hadoop.executionengine.mapReduceLayer.JobControlCompiler - mapred.job.reduce.markreset.buffe
```

```
2020-08-07 19:46:20,545 [main] INFO  org.apache.hadoop.conf.Configuration.deprecation - io.bytes.per.checksum is deprecated. Instead, use dfs.bytes-per-
checksum
2020-08-07 19:46:20,546 [main] WARN  org.apache.pig.data.SchemaTupleBackend - SchemaTupleBackend has already been initialized
2020-08-07 19:46:20,550 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileInputFormat - Total input paths to process : 1
2020-08-07 19:46:20,550 [main] INFO  org.apache.pig.backend.hadoop.executionengine.util.MapRedUtil - Total input paths to process : 1
(13589,52)
(20000,271)
(22354,215)
(23619,127)
(29765,90)
(31796,291)
(36885,125)
(40601,55)
(44205,76)
(45987,98)
(47521,241)
(57800,193)
(60029,112)
(66288,404)
(69369,83)
(72811,123)
(76073,54)
(77691,194)
(93505,166)
(163941,138)
(169430,105)
(176006,88)
(177129,131)
(179244,81)
```

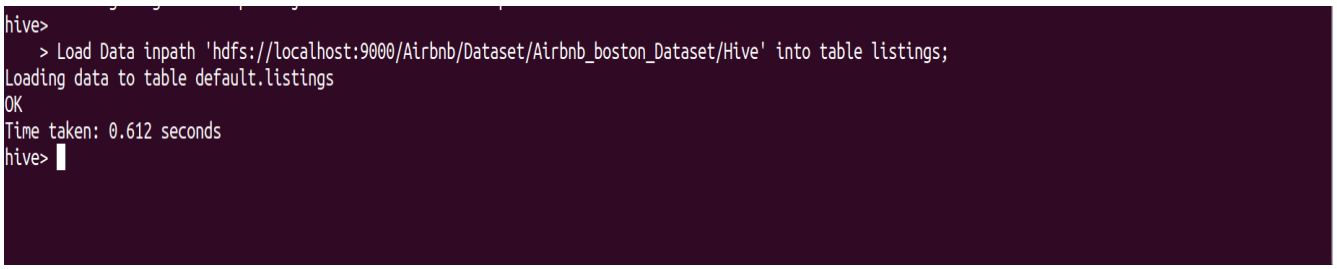
APACHE HIVE ANALYSIS

Analysis 4: To find Top 7 listings based on rating for each neighborhood.

1. Create Table 'listings'



2. Load csv Data into listings table



3. Create assortedListings Table

```
hive> Create Table assortedListings as Select listingid, neighbourhood, Cast(review_scores_ratings as int) AS rating_Data from listings where listingid
is not NULL AND length(review_scores_ratings)==2 ;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, te
z) or using Hive 1.X releases.
Query ID = dns7665_20200807190318_37c58c01-21a8-4486-b803-ae2eb749c5a3
Total jobs = 3
Launching Job 1 out of 3
Number of reduce tasks is set to 0 since there's no reduce operator
Starting Job = job_1596851359989_0001, Tracking URL = http://ubuntu:8088/proxy/application_1596851359989_0001/
Kill Command = /usr/local/bin/hadoop-2.10.0/bin/hadoop job  -kill job_1596851359989_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 0
2020-08-07 19:03:26,346 Stage-1 map = 0%,  reduce = 0%
2020-08-07 19:03:31,515 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.59 sec
MapReduce Total cumulative CPU time: 1 seconds 590 msec
Ended Job = job_1596851359989_0001
Stage-4 is selected by condition resolver.
Stage-3 is filtered out by condition resolver.
Stage-5 is filtered out by condition resolver.
Moving data to directory hdfs://localhost:9000/hive/warehouse/.hive-staging_hive_2020-08-07_19-03-18_778_7245017806288800195-1/-ext-10002
Moving data to directory hdfs://localhost:9000/hive/warehouse/assortedlistings
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1   Cumulative CPU: 1.59 sec   HDFS Read: 14749629 HDFS Write: 47017 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 590 msec
OK
Time taken: 14.149 seconds
hive>
```

4. Top 7 listings based on rating for each neighbourhood using rank()

```
hive> select * from( Select listingid, neighbourhood, rating_Data, rank() over (partition  by neighbourhood order by rating_Data desc) as rank from asso
rtedListings) r where rank <7;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, te
z) or using Hive 1.X releases.
Query ID = dns7665_20200807191222_4d48763f-d07c-4d5b-852c-679979a754d9
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1596851359989_0007, Tracking URL = http://ubuntu:8088/proxy/application_1596851359989_0007/
Kill Command = /usr/local/bin/hadoop-2.10.0/bin/hadoop job  -kill job_1596851359989_0007
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-08-07 19:12:27,972 Stage-1 map = 0%,  reduce = 0%
2020-08-07 19:12:32,061 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.05 sec
2020-08-07 19:12:37,159 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.58 sec
MapReduce Total cumulative CPU time: 2 seconds 580 msec
Ended Job = job_1596851359989_0007
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.58 sec   HDFS Read: 57490 HDFS Write: 8051 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 580 msec
OK
10032327        Allston 99      1
95453   Allston 98      2
5889190 Allston 98      2
```

5. Inserting the Hive Output to HDFS using Insert Overwrite Directory command:

```
    >
    > Insert Overwrite Directory  'hdfs://localhost:9000/Airbnb_Analysis_Output/analysis_Hive_Top7ListingsForEachNeighbourhood/' select * from( Select l
istingid, neighbourhood, rating_Data, rank() over (partition  by neighbourhood order by rating_Data desc) as rank from assortedListings) r where rank <7
;
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the future versions. Consider using a different execution engine (i.e. spark, te
z) or using Hive 1.X releases.
Query ID = dns7665_20200807191707_b3951bf1-a73a-4331-869a-536bbda0cdfe
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1596851359989_0008, Tracking URL = http://ubuntu:8088/proxy/application_1596851359989_0008/
Kill Command = /usr/local/bin/hadoop-2.10.0/bin/hadoop job  -kill job_1596851359989_0008
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-08-07 19:17:13,310 Stage-1 map = 0%,  reduce = 0%
2020-08-07 19:17:17,388 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.11 sec
2020-08-07 19:17:22,475 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 2.39 sec
MapReduce Total cumulative CPU time: 2 seconds 390 msec
Ended Job = job_1596851359989_0008
Moving data to directory hdfs://localhost:9000/Airbnb_Analysis_Output/analysis_Hive_Top7ListingsForEachNeighbourhood
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 2.39 sec   HDFS Read: 57208 HDFS Write: 5300 SUCCESS
Total MapReduce CPU Time Spent: 2 seconds 390 msec
OK
Time taken: 16.373 seconds
hive>
```

Output:

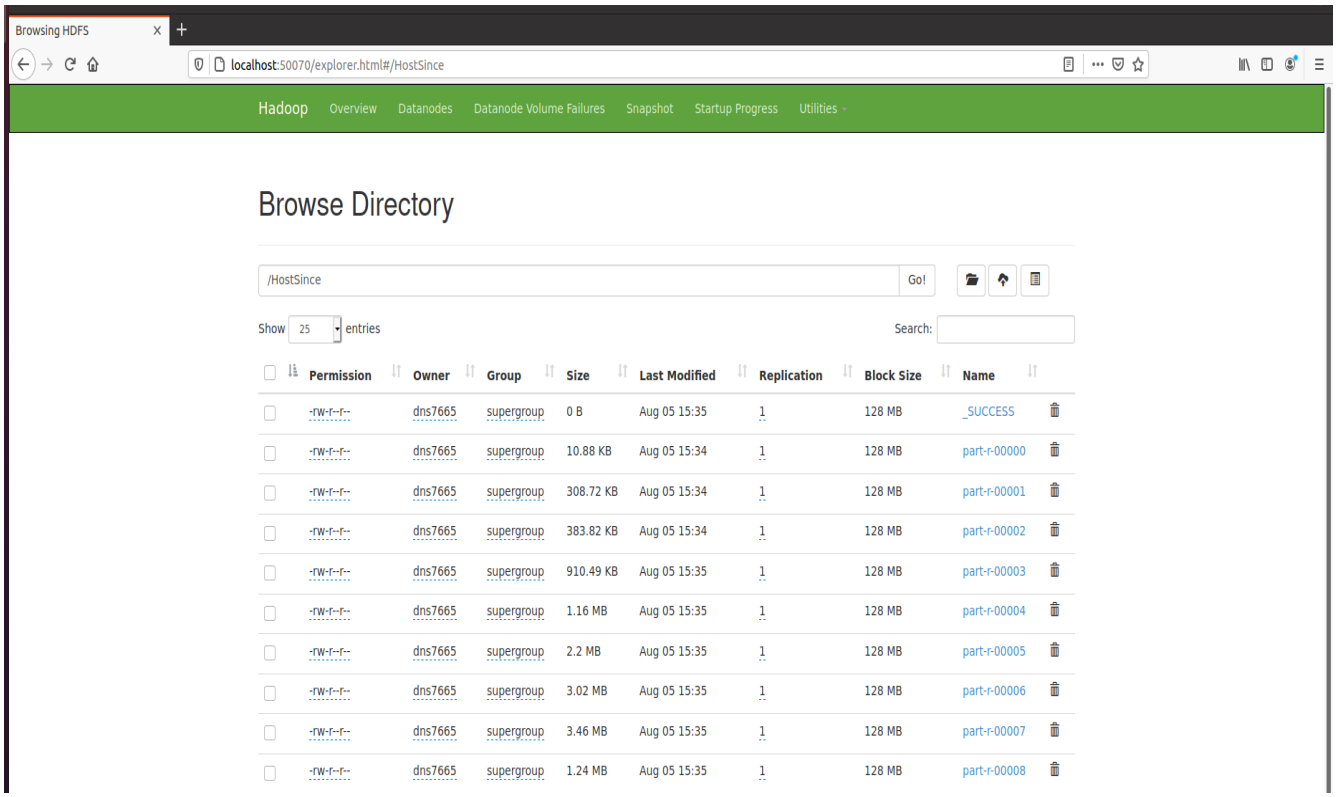
```
OK
10032327        Allston 99      1
95453   Allston 98      2
5889190 Allston 98      2
8579799 Allston 98      2
3519768 Allston 98      2
13733887        Allston 98      2
2277821 Back Bay        99      1
7740436 Back Bay        99      1
9231486 Back Bay        99      1
10542140        Back Bay        99      1
11082235        Back Bay        99      1
7804358 Back Bay        99      1
7194793 Back Bay        99      1
12092499        Back Bay        99      1
9725178 Back Bay        99      1
6268082 Bay Village     97      1
7931882 Bay Village     96      2
12265433        Bay Village     95      3
8422876 Bay Village     94      4
6292261 Bay Village     93      5
9825415 Bay Village     93      5
4995033 Beacon Hill     99      1
6066455 Beacon Hill     99      1
8471852 Beacon Hill     99      1
2947662 Beacon Hill     99      1
3377100 Beacon Hill     98      5
5743939 Beacon Hill     98      5
13788865        Beacon Hill     98      5
13101775        Beacon Hill     98      5
6677640 Beacon Hill     98      5
8662258 Beacon Hill     98      5
3890373 Beacon Hill     98      5
```

MAPREDUCE ANALYSIS:

Analysis 5: To implement the partitioning based on year the host joined Airbnb.

Partitioning Pattern of Data Organization Technique Used

Partitioning is done based on year. Year is extracted from the host since column in listings dataset.



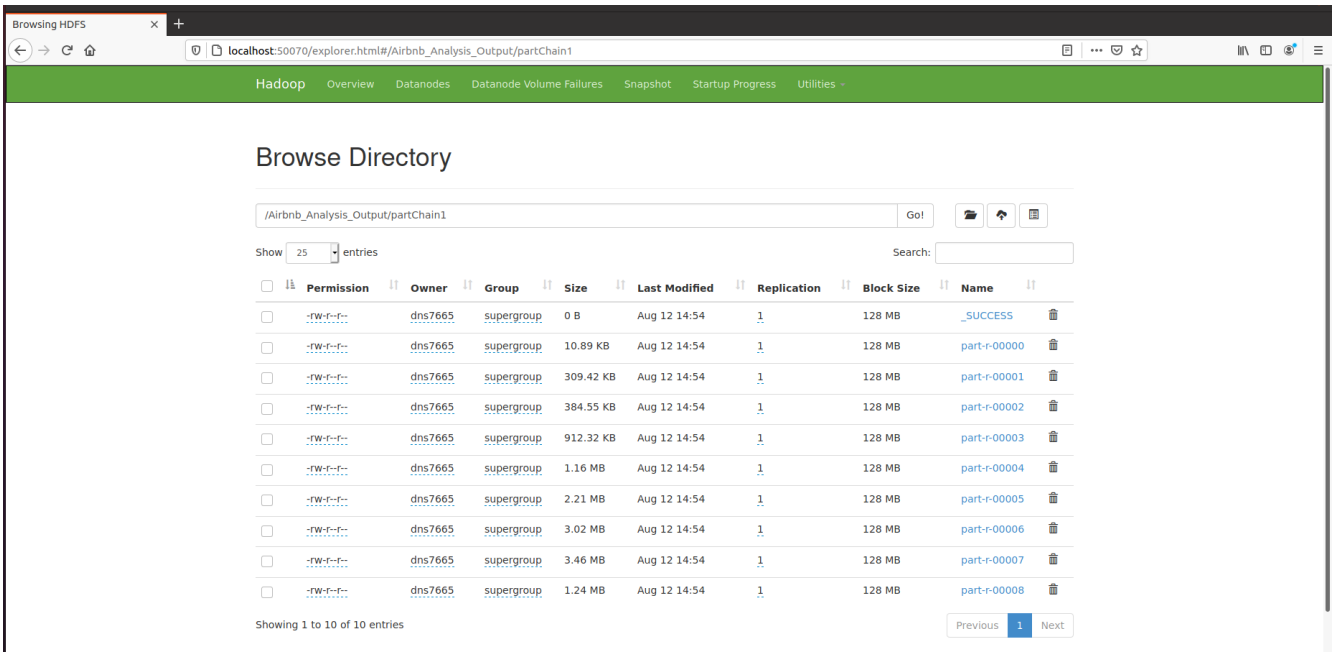
Analysis 6: MapReduce Chaining to calculate the no. of users signed up on yelp each year

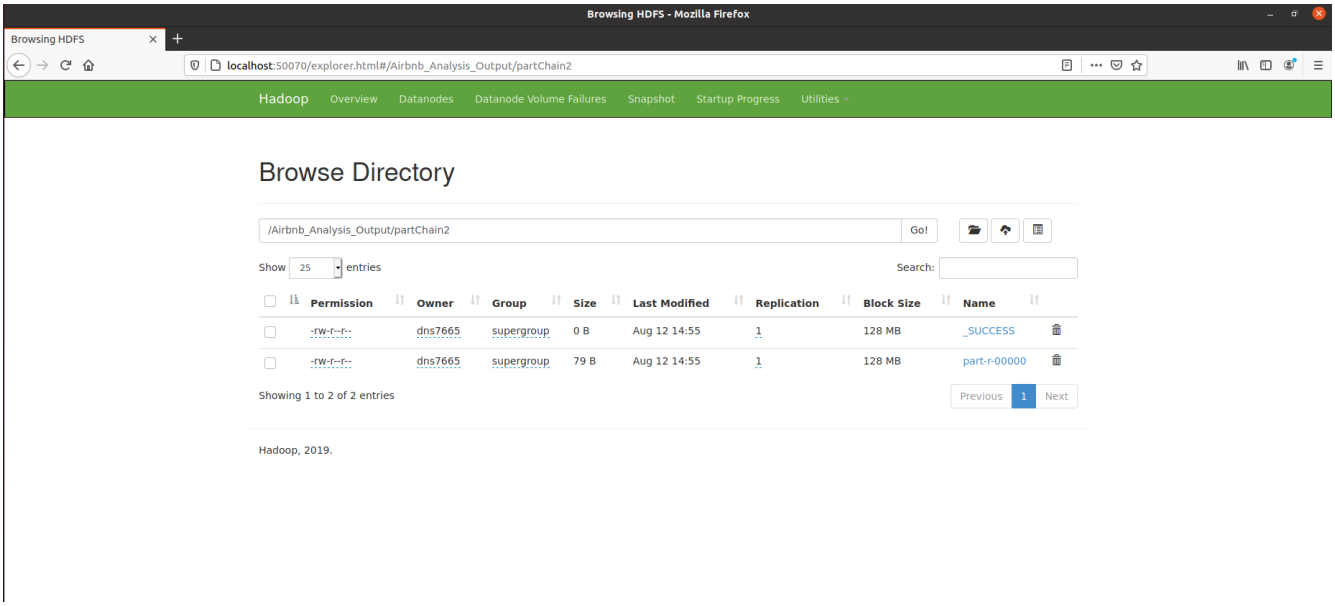
Chaining Operation Performed

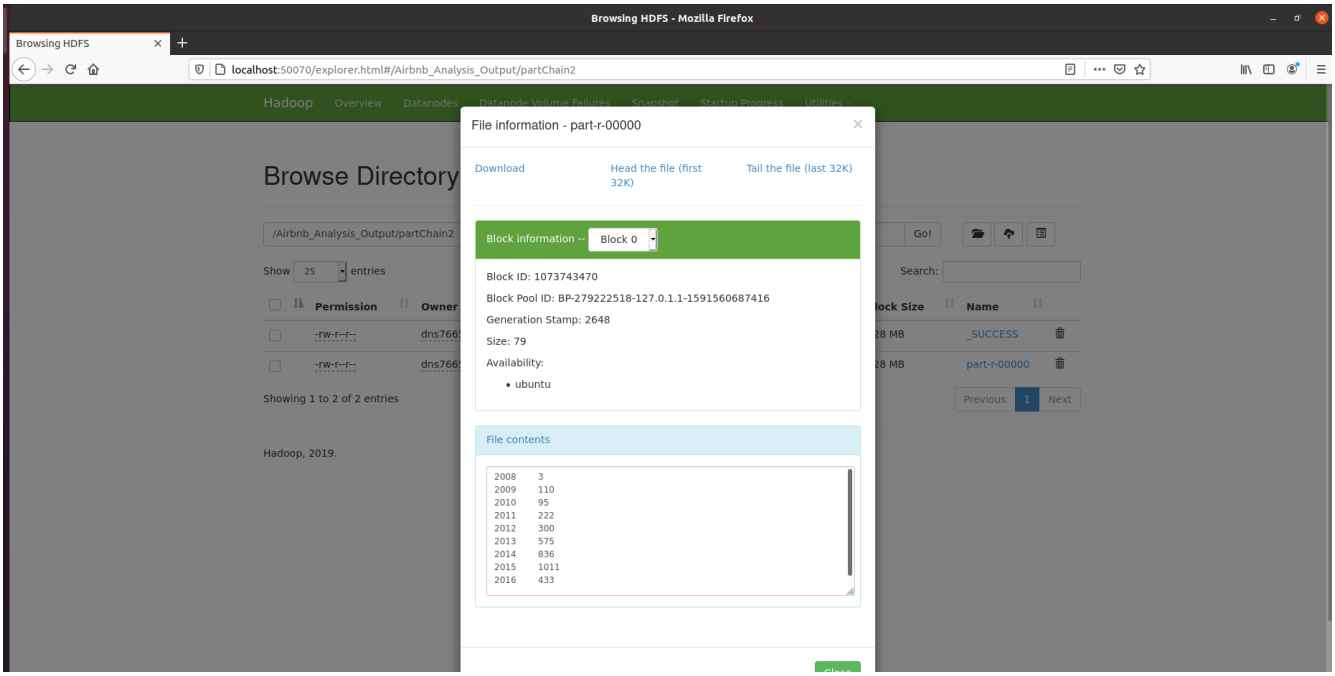
MapReduce 1: Partition the files based on the years in dataset – Partitioning Technique

MapReduce 2: Count the no of files in each partition

```
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$ ./hadoop jar ~/IdeaProjects/Analysis_AirbnbHosts_partitionAndChaining/target/Analysis_AirbnbHosts_partitionByYear-1.0-SNAPSHOT.jar DriverClass /Airbnb/Dataset/Airbnb_boston_Dataset/MapReduce/listings.csv /Airbnb_Analysis_Output/partChain1  /Airbnb_Analysis_Output/partChain2
20/08/12 14:54:13 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
20/08/12 14:54:13 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
20/08/12 14:54:13 INFO input.FileInputFormat: Total input files to process : 1
20/08/12 14:54:14 INFO mapreduce.JobSubmitter: number of splits:1
20/08/12 14:54:14 INFO Configuration.deprecation: yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-publisher.enabled
20/08/12 14:54:14 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1597265931865_0006
20/08/12 14:54:14 INFO conf.Configuration: resource-types.xml not found
20/08/12 14:54:14 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
20/08/12 14:54:14 INFO resource.ResourceUtils: Adding resource type - name = memory-mb, units = Mi, type = COUNTABLE
20/08/12 14:54:14 INFO resource.ResourceUtils: Adding resource type - name = vcores, units = , type = COUNTABLE
20/08/12 14:54:14 INFO impl.YarnClientImpl: Submitted application application_1597265931865_0006
20/08/12 14:54:14 INFO mapreduce.Job: The url to track the job: http://ubuntu:8088/proxy/application_1597265931865_0006/
20/08/12 14:54:14 INFO mapreduce.Job: Running job: job_1597265931865_0006
20/08/12 14:54:19 INFO mapreduce.Job: Job job_1597265931865_0006 running in uber mode : false
20/08/12 14:54:19 INFO mapreduce.Job:  map 0% reduce 0%
20/08/12 14:54:23 INFO mapreduce.Job:  map 100% reduce 0%
20/08/12 14:54:28 INFO mapreduce.Job:  map 100% reduce 11%
20/08/12 14:54:32 INFO mapreduce.Job:  map 100% reduce 22%
20/08/12 14:54:35 INFO mapreduce.Job:  map 100% reduce 33%
20/08/12 14:54:38 INFO mapreduce.Job:  map 100% reduce 56%
20/08/12 14:54:39 INFO mapreduce.Job:  map 100% reduce 67%
20/08/12 14:54:40 INFO mapreduce.Job:  map 100% reduce 78%
20/08/12 14:54:41 INFO mapreduce.Job:  map 100% reduce 100%
20/08/12 14:54:43 INFO mapreduce.Job: Job job_1597265931865_0006 completed successfully
20/08/12 14:54:43 INFO mapreduce.Job: Counters: 50
        File System Counters
```




Browsing HDFS - Mozilla Firefox
Browsing HDFS
localhost:50070/explorer.html#/Airbnb_Analysis_Output/partChain2
Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Browse Directory
/Airbnb_Analysis_Output/partChain2
Go!
Show 25 entries
Search:
Permission
Owner
Group
Size
Last Modified
Replication
Block Size
Name
-rw-r--r--
dns7665
supergroup
0 B
Aug 12 14:55
1
128 MB
_SUCCESS
-rw-r--r--
dns7665
supergroup
79 B
Aug 12 14:55
1
128 MB
part-r-00000
Showing 1 to 2 of 2 entries
Previous
1
Next
Hadoop, 2019.


Browsing HDFS - Mozilla Firefox
Browsing HDFS
localhost:50070/explorer.html#/Airbnb_Analysis_Output/partChain2
Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Browse Directory
/Airbnb_Analysis_Output/partChain2
Go!
Show 25 entries
Search:
Permission
Owner
Name
-rw-r--r--
_SUCCESS
-rw-r--r--
part-r-00000
Showing 1 to 2 of 2 entries
Previous
1
Next
Hadoop, 2019.
File information - part-r-00000
Download
Head the file (first 32K)
Tail the file (last 32K)
Block information --
Block 0
Block ID: 1073743470
Block Pool ID: BP-279222518-127.0.1.1-1591560687416
Generation Stamp: 2648
Size: 79
Availability:
ubuntu
File contents
2008 3
2009 110
2010 95
2011 222
2012 300
2013 575
2014 836
2015 1011
2016 433

Analysis 7: To Sort the Listings according to reviews received per year

Run:

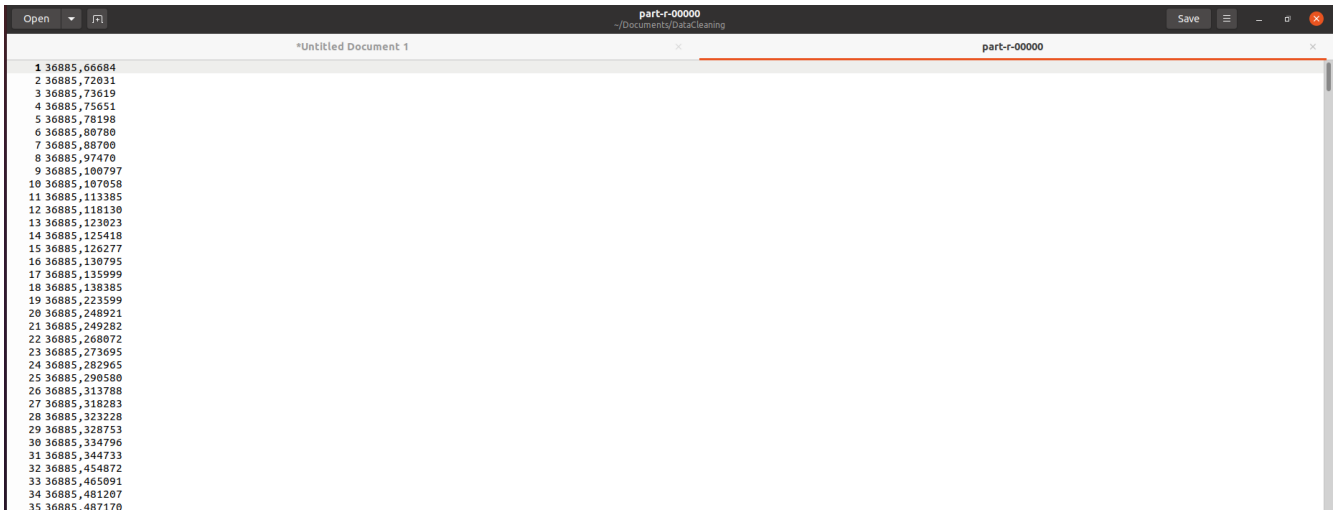
```
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$ ./hadoop jar ~/IdeaProjects/Analysis_Airbnb_SecondarySort/target/Analysis_Airbnb_SecondarySort_Price-1.
0-SNAPSHOT.jar DriverClass /Airbnb/Dataset/Airbnb_boston_Dataset/MapReduce/listings.csv /Airbnb_Analysis_Output/secondSort
20/08/12 15:08:23 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
20/08/12 15:08:23 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your ap
plication with ToolRunner to remedy this.
20/08/12 15:08:23 INFO input.FileInputFormat: Total input files to process : 1
20/08/12 15:08:23 INFO mapreduce.JobSubmitter: number of splits:1
20/08/12 15:08:23 INFO Configuration.deprecation: yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-
publisher.enabled
20/08/12 15:08:23 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1597265931865_0009
20/08/12 15:08:23 INFO conf.Configuration: resource-types.xml not found
20/08/12 15:08:23 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
20/08/12 15:08:23 INFO resource.ResourceUtils: Adding resource type - name = memory-mb, units = Mi, type = COUNTABLE
20/08/12 15:08:23 INFO resource.ResourceUtils: Adding resource type - name = vcores, units = , type = COUNTABLE
20/08/12 15:08:23 INFO impl.YarnClientImpl: Submitted application application_1597265931865_0009
20/08/12 15:08:24 INFO mapreduce.Job: The url to track the job: http://ubuntu:8088/proxy/application_1597265931865_0009/
20/08/12 15:08:24 INFO mapreduce.Job: Running job: job_1597265931865_0009
20/08/12 15:08:29 INFO mapreduce.Job: Job job_1597265931865_0009 running in uber mode : false
20/08/12 15:08:29 INFO mapreduce.Job:  map 0% reduce 0%
20/08/12 15:08:33 INFO mapreduce.Job:  map 100% reduce 0%
20/08/12 15:08:37 INFO mapreduce.Job:  map 100% reduce 100%
20/08/12 15:08:37 INFO mapreduce.Job: Job job_1597265931865_0009 completed successfully
20/08/12 15:08:37 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=96932
                FILE: Number of bytes written=606415
                FILE: Number of read operations=0
```

Output:

```
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$ ./hadoop fs -cat /secondSort/part-r-00000
12269155,2016   5
5601743,2016    10
8153848,2016    4
8981656,2016    2
11333422,2016   1
7161203,2016    2
11333422,2016   1
11304657,2016   1
9256035,2016    7
11333422,2016   1
10964552,2016   1
2215762,2016    1
1303261,2016    1
2215762,2016    6
3419711,2016    9
4304969,2016    3
8757949,2016    12
10964552,2016   4
7527864,2016    2
12998582,2016   1
13709190,2016   3
13374617,2016   3
4190704,2016    28
13709190,2016   17
5690640,2016    2
9900315,2016    11
13032590,2016   4
3419711,2016    1
13032590,2016   14
8782991,2016    4
4632511,2016    12
```

Analysis 8: To find the top 20 listings based on the ratings

Data Cleaning using PIG:



Step1:

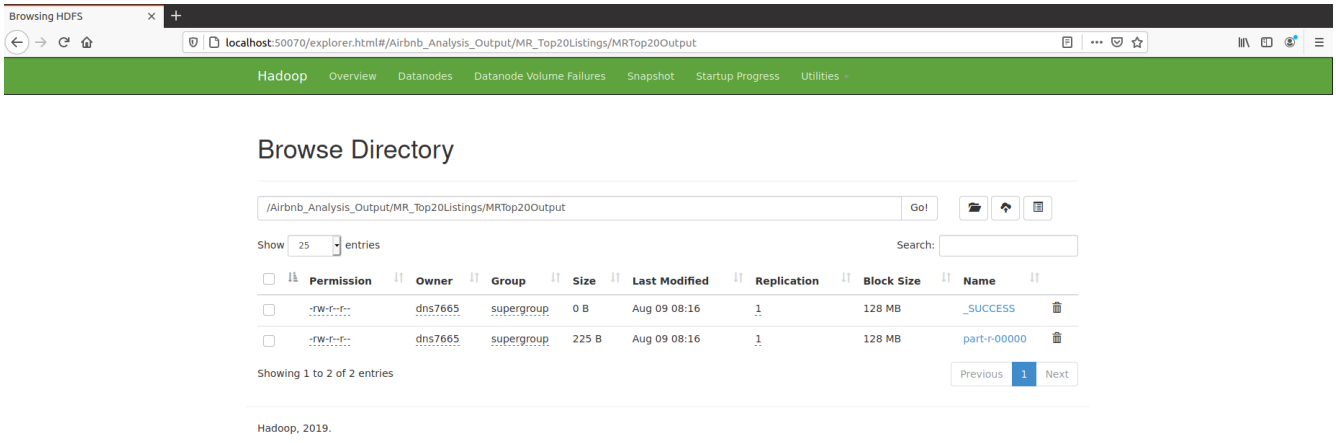
```
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$ ./hadoop fs -mkdir /Airbnb_Analysis_Output/MR_Top20Listings
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$ ./hadoop jar ~/IdeaProjects/Analysis_Airbnb_MR_Top20Listings/target/Analysis_Airbnb_MR_Top20Listings-1.
0-SNAPSHOT.jar DriverClass /Airbnb/Dataset/Airbnb_boston_Dataset/DataCleaningPig.txt/part-r-00000 /MRTop20 /MRTop20.2
```

Step2:

```
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$ ./hadoop jar ~/IdeaProjects/Analysis_Airbnb_MR_Top20Listings/target/Analysis_Airbnb_MR_Top20Listings-1.
0-SNAPSHOT.jar DriverClass /Airbnb/Dataset/Airbnb_boston_Dataset/DataCleaningPig.txt/part-r-00000 /Airbnb_Analysis_Output/MR_Top20Listings/MRChain /Airb
nb_Analysis_Output/MR_Top20Listings/MRTop20Output
20/08/09 08:15:36 INFO client.RMProxy: Connecting to ResourceManager at /0.0.0.0:8032
20/08/09 08:15:37 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your ap
plication with ToolRunner to remedy this.
20/08/09 08:15:38 INFO input.FileInputFormat: Total input files to process : 1
20/08/09 08:15:38 INFO mapreduce.JobSubmitter: number of splits:1
20/08/09 08:15:39 INFO Configuration.deprecation: yarn.resourcemanager.system-metrics-publisher.enabled is deprecated. Instead, use yarn.system-metrics-
publisher.enabled
20/08/09 08:15:39 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1596985734028_0001
20/08/09 08:15:39 INFO conf.Configuration: resource-types.xml not found
20/08/09 08:15:39 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
20/08/09 08:15:39 INFO resource.ResourceUtils: Adding resource type - name = memory-mb, units = Mi, type = COUNTABLE
20/08/09 08:15:39 INFO resource.ResourceUtils: Adding resource type - name = vcores, units = , type = COUNTABLE
20/08/09 08:15:40 INFO impl.YarnClientImpl: Submitted application application_1596985734028_0001
20/08/09 08:15:40 INFO mapreduce.Job: The url to track the job: http://ubuntu:8088/proxy/application_1596985734028_0001/
20/08/09 08:15:40 INFO mapreduce.Job: Running job: job_1596985734028_0001
20/08/09 08:15:54 INFO mapreduce.Job: Job job_1596985734028_0001 running in uber mode : false
20/08/09 08:15:54 INFO mapreduce.Job:  map 0% reduce 0%
20/08/09 08:16:02 INFO mapreduce.Job:  map 100% reduce 0%
20/08/09 08:16:10 INFO mapreduce.Job:  map 100% reduce 100%
20/08/09 08:16:11 INFO mapreduce.Job: Job job_1596985734028_0001 completed successfully
20/08/09 08:16:11 INFO mapreduce.Job: Counters: 49
        File System Counters
                FILE: Number of bytes read=947351
                FILE: Number of bytes written=2306181
                FILE: Number of read operations=0
                FILE: Number of large read operations=0
                FILE: Number of write operations=0
                HDFS: Number of bytes read=1145952
```

Step 3:

```
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$ ./hadoop fs -cat /Airbnb_Analysis_Output/MR_Top20Listings/MRTop20Output/part-r-00000
66288   404
1497879 320
414419  312
31796   291
916123  281
1136972 280
20000   271
1695275 252
766700  244
2776143 242
47521   241
1141522 237
1615033 236
1584362 232
1472520 231
197972  230
1178106 226
1544702 222
708802  221
22354   215
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
```



APACHE SPARK

Analysis 9: Inner Join using Apache Spark and PySpark

PySpark is python library to execute spark commands using python

Step 1)
Load Data from reviews.csv file using spark context object
Clean data before storing using map and filter commands in pySpark
Take in account listing_Id and review_Id Columns from the dataset

```
>>data=sc.textFile("/home/dns7665/Downloads/Airbnb_boston_Dataset/reviews.csv")
\
   .map(lambda line: line.split(",")) \
   .filter(lambda line: len(line)>1) \
   .filter(lambda line: len(line[0])<8)\
   .filter(lambda line: len(line[0])>3)\
   .map(lambda line: (line[0],line[1])) \
   .collect()
```

```
>>
>>
>>> data=sc.textFile("/home/dns7665/Downloads/Airbnb_boston_Dataset/reviews.csv") \
...     .map(lambda line: line.split(",")) \
...     .filter(lambda line: len(line)>1) \
...     .filter(lambda line: len(line[0])<8)\
...     .filter(lambda line: len(line[0])>3)\
...     .map(lambda line: (line[0],line[1])) \
...     .collect()
>>
>>
>>
```

Create a Resilient Distributed Dataset using parallelize method in Spark
RDD stands for Resilient Distributed Dataset, these are the elements that run and operate on multiple nodes to do parallel processing on a cluster. RDDs are immutable elements, which means once you create an RDD you cannot change it. RDDs are fault tolerant as well, hence in case of any failure, they recover automatically. You can apply multiple operations on these RDDs to achieve a certain task.

```
>>rdd1 = sc.parallelize(data)
```

```
# Create PySpark DataFrame using spark createDataFrame function and display it
>>df1 = spark.createDataFrame(rdd1)
>>df1.show()
```

Step 2)

```
# Load Data from listings.csv file using spark context object
# Clean data before storing using map and filter commands in pySpark
# Take in account listing_Id and neighbourhood Columns from the dataset

>>data2=sc.textFile("/home/dns7665/Downloads/Airbnb_boston_Dataset/listings.csv"
) \
   .map(lambda line: line.split(",")) \
   .filter(lambda line: len(line)>40) \
   .filter(lambda line: len(line[0])<8)\
   .filter(lambda line: len(line[0])>3)\
   .map(lambda line: (line[0],line[2])) \
   .collect()

# Create a Resilient Distributed Dataset using parallelize method in Spark
# RDD stands for Resilient Distributed Dataset, these are the elements that run and
operate on multiple nodes to do parallel processing on a cluster. RDDs are immutable
elements, which means once you create an RDD you cannot change it. RDDs are fault
tolerant as well, hence in case of any failure, they recover automatically. You can apply
multiple operations on these RDDs to achieve a certain task.

>>rdd2 = sc.parallelize(data2)


# Create PySpark DataFrame using spark createDataFrame function and display it
>>df2 = spark.createDataFrame(rdd2)

>>df2.show()
```

Step 3)

```
# Rename Columns of joined table before performing join operation
```

```
>>df3 = df1.withColumnRenamed("_1","listingId") \
   .withColumnRenamed("_2","reviewId")
>>df3.printSchema()
```

```
>>> df3 = df1.withColumnRenamed("_1","listingId") \
...      .withColumnRenamed("_2","reviewId")
>>> df3.printSchema()
root
 |-- listingId: string (nullable = true)
 |-- reviewId: string (nullable = true)
```

```
>>df4 = df2.withColumnRenamed("_1","listingId") \
   .withColumnRenamed("_2","Neighbourhood")
>>df4.printSchema()
```

```
>>> df4 = df2.withColumnRenamed("_1","listingId") \
...      .withColumnRenamed("_2","Neighbourhood")
>>> df4.printSchema()
root
 |-- listingId: string (nullable = true)
 |-- Neighbourhood: string (nullable = true)

>>>
```

Step 4)

#Perform Inner Join operation using join function of PySpark DataFrames

```
>>df = df3.join(df4, on=['listingId'], how='inner')
>>df.show()
```

```
>>> df = df3.join(df4, on=['listingId'], how='inner')
>>> df.show()
+---------+--------+-------------+
|listingId|reviewId|Neighbourhood|
+---------+--------+-------------+
|  1223050| 5496451|  Charlestown|
|  1223050| 5687235|  Charlestown|
|  1223050| 5890036|  Charlestown|
|  1223050| 6761419|  Charlestown|
|  1223050| 6819814|  Charlestown|
|  1223050| 6975683|  Charlestown|
|  1223050| 8009730|  Charlestown|
|  1223050| 8283778|  Charlestown|
|  1223050| 8298682|  Charlestown|
|  1223050| 8348951|  Charlestown|
|  1223050| 8480744|  Charlestown|
|  1223050|10188650|  Charlestown|
|  1223050|10650876|  Charlestown|
|  1223050|10960656|  Charlestown|
|  1223050|11132936|  Charlestown|
|  1223050|11199332|  Charlestown|
|  1223050|11376202|  Charlestown|
|  1223050|11720770|  Charlestown|
|  1223050|12048508|  Charlestown|
|  1223050|12219780|  Charlestown|
+---------+--------+-------------+
only showing top 20 rows
```

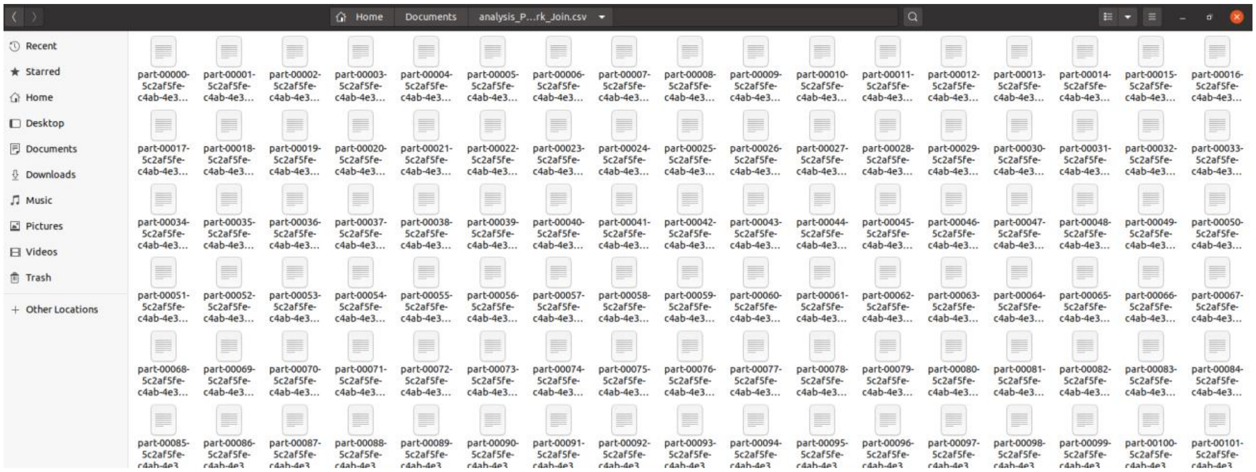
Step 5)

#Store the Output in csv form and upload on HDFS

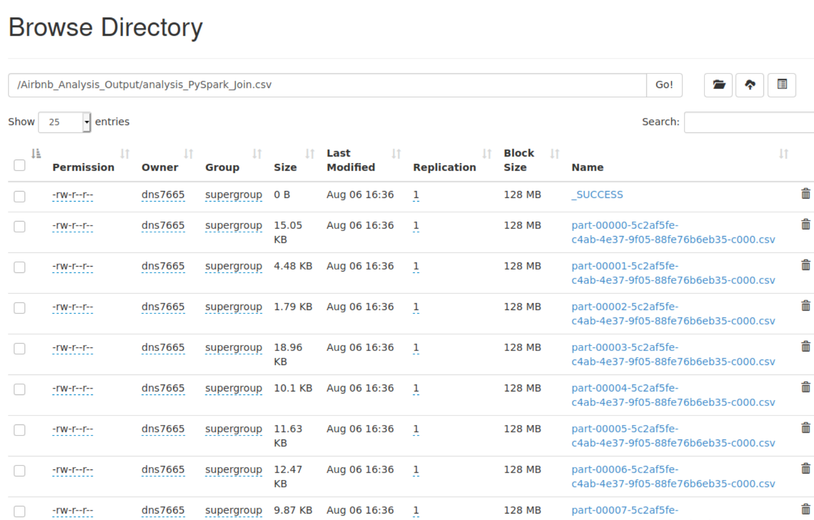
>>df.write.format('csv').option('header',True).mode('overwrite').option('sep',',').save('file:///home/dns7665/Documents/analysis_PySpark_Join.csv')

From hadoop terminal:
./hadoop fs -put ~/Documents/analysis_PySpark_Join.csv/ /Airbnb_Analysis_Output

Local:



HDFS

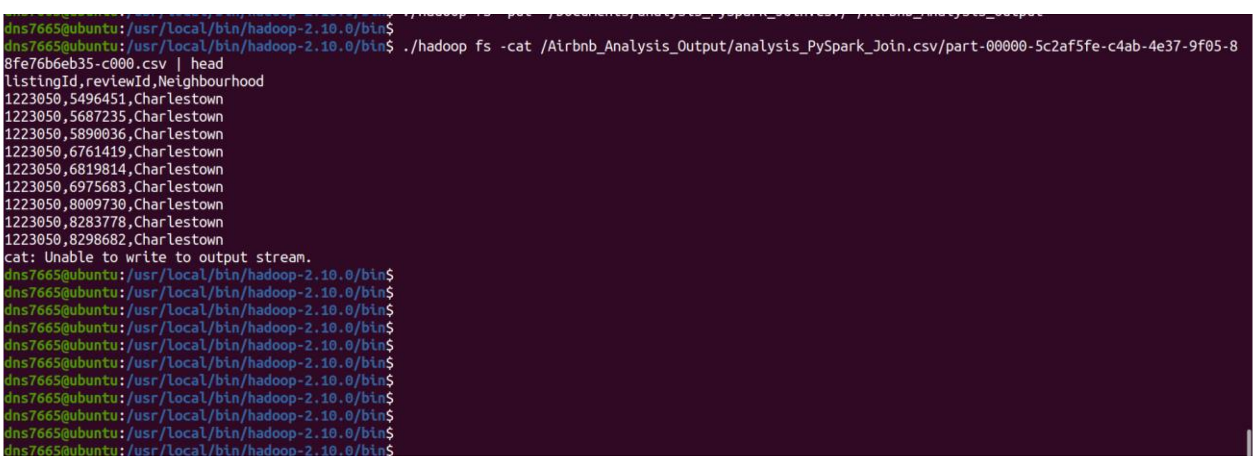


Browse Directory

/Airbnb_Analysis_Output/analysis_PySpark_Join.csv Go!

Show 25 entries Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rw-r--r--	dns7665	supergroup	0 B	Aug 06 16:36	1	128 MB	_SUCCESS
-rw-r--r--	dns7665	supergroup	15.05 KB	Aug 06 16:36	1	128 MB	part-00000-5c2af5fe-c4ab-4e37-9f05-88fe76b6eb35-c000.csv
-rw-r--r--	dns7665	supergroup	4.48 KB	Aug 06 16:36	1	128 MB	part-00001-5c2af5fe-c4ab-4e37-9f05-88fe76b6eb35-c000.csv
-rw-r--r--	dns7665	supergroup	1.79 KB	Aug 06 16:36	1	128 MB	part-00002-5c2af5fe-c4ab-4e37-9f05-88fe76b6eb35-c000.csv
-rw-r--r--	dns7665	supergroup	18.96 KB	Aug 06 16:36	1	128 MB	part-00003-5c2af5fe-c4ab-4e37-9f05-88fe76b6eb35-c000.csv
-rw-r--r--	dns7665	supergroup	10.1 KB	Aug 06 16:36	1	128 MB	part-00004-5c2af5fe-c4ab-4e37-9f05-88fe76b6eb35-c000.csv
-rw-r--r--	dns7665	supergroup	11.63 KB	Aug 06 16:36	1	128 MB	part-00005-5c2af5fe-c4ab-4e37-9f05-88fe76b6eb35-c000.csv
-rw-r--r--	dns7665	supergroup	12.47 KB	Aug 06 16:36	1	128 MB	part-00006-5c2af5fe-c4ab-4e37-9f05-88fe76b6eb35-c000.csv
-rw-r--r--	dns7665	supergroup	9.87 KB	Aug 06 16:36	1	128 MB	part-00007-5c2af5fe-



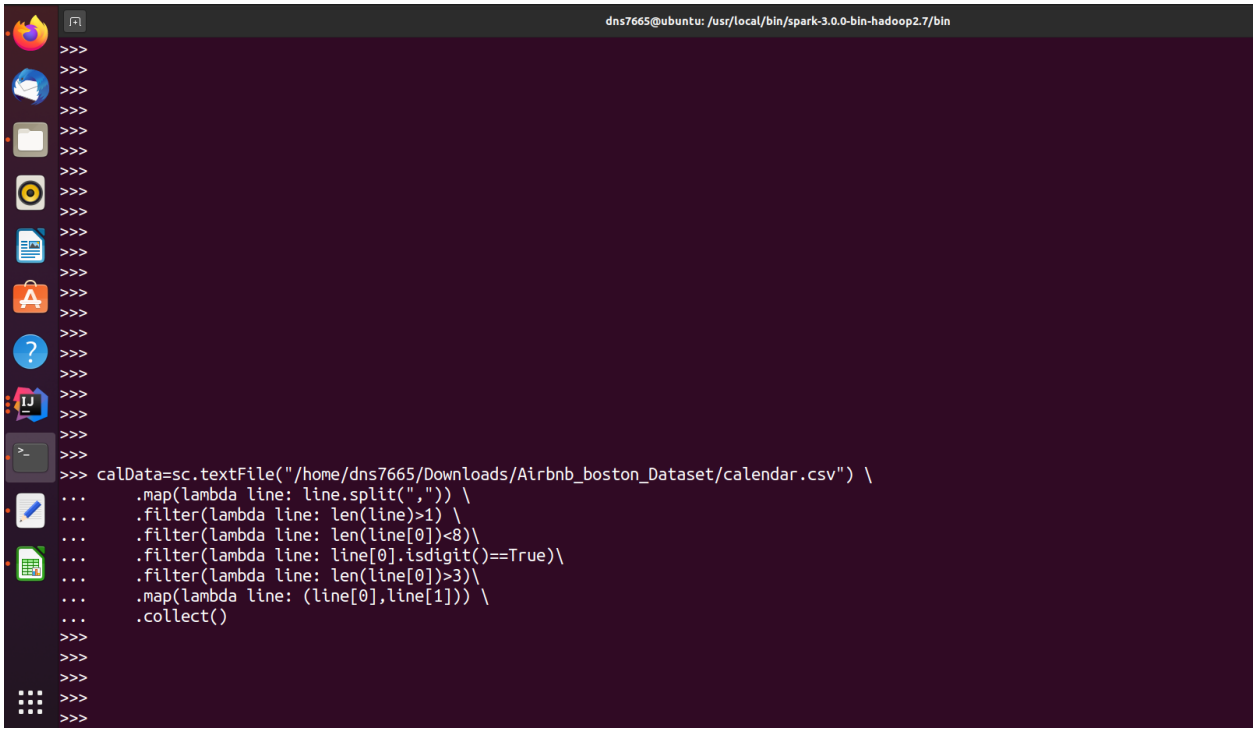
```
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$ ./hadoop fs -cat /Airbnb_Analysis_Output/analysis_PySpark_Join.csv/part-00000-5c2af5fe-c4ab-4e37-9f05-8
8fe76b6eb35-c000.csv | head
listingId,reviewId,Neighbourhood
1223050,5496451,Charlestown
1223050,5687235,Charlestown
1223050,5890036,Charlestown
1223050,6761419,Charlestown
1223050,6819814,Charlestown
1223050,6975683,Charlestown
1223050,8009730,Charlestown
1223050,8283778,Charlestown
1223050,8298682,Charlestown
cat: Unable to write to output stream.
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$
```

Analysis 10: To find total available days for each Listing

Find the available days from 2016-2017 for listings and then sort it before displaying:

Step 1)
Load Data from calendar.csv file using spark context object
Clean data before storing using map and filter commands in PySpark
Take in account listing_Id and available Columns from the dataset

```
calData=sc.textFile("/home/dns7665/Downloads/Airbnb_boston_Dataset/calendar.csv
") \
   .map(lambda line: line.split(",")) \
   .filter(lambda line: len(line)>1) \
   .filter(lambda line: len(line[0])<8)\
   .filter(lambda line: line[0].isdigit()==True)\
   .filter(lambda line: len(line[0])>3)\
   .map(lambda line: (line[0],line[1])) \
   .collect()
```



Create a Resilient Distributed Dataset using parallelize method in Spark

RDD stands for Resilient Distributed Dataset, these are the elements that run and operate on multiple nodes to do parallel processing on a cluster. RDDs are immutable elements, which means once you create an RDD you cannot change it. RDDs are fault tolerant as well, hence in case of any failure, they recover automatically. You can apply multiple operations on these RDDs to achieve a certain task.

Step 2)
rddCal = sc.parallelize(calData)

3)
# Create PySpark DataFrame using spark createDataFrame function and display it
calDf = spark.createDataFrame(rddCal)

calDf.show()

```
>>> rddCal = sc.parallelize(calData)
>>> calDf = spark.createDataFrame(rddCal)
20/08/06 16:01:01 WARN TaskSetManager: Stage 57 contains a task of very large size (4517 KiB). The maximum recommended task size is 1000 KiB.
>>>
>>> calDf.show()
20/08/06 16:01:03 WARN TaskSetManager: Stage 58 contains a task of very large size (4517 KiB). The maximum recommended task size is 1000 KiB.
+-------+---+
|     _1| _2|
+-------+---+
|3075044|  t|
|3075044|  t|
|3075044|  t|
|3075044|  t|
|3075044|  t|
|3075044|  t|
|3075044|  t|
|3075044|  t|
|3075044|  t|
|3075044|  t|
|3075044|  t|
|3075044|  t|
|3075044|  t|
|3075044|  t|
|3075044|  t|
|3075044|  t|
|3075044|  t|
|3075044|  t|
|3075044|  t|
|3075044|  t|
+-------+---+
only showing top 20 rows
```

Step 4)

Filter DataFrame for values (Only take days when listing is available) using filter command
Perform map-reduce on the dataFrame to get the count of available days for each listings
Sort the reduce output in descending order
Collect the output and store it in a new RDD called finalCal

```
finalCal=calDf.rdd\
        .filter(lambda x: x[1]=="t")\
        .map(lambda x:(x[0],1))\
        .reduceByKey(lambda a,b:a +b)\
        .sortBy(lambda x: x[1], ascending = False)\
        .collect()
```

Step 5)

Create a new DataFrame to store the Map-Reduce output and Display

```
finalCalDf = spark.createDataFrame(finalCal)
finalCalDf.show()
```

```
>>> finalCal=calDf.rdd\
...     .filter(lambda x: x[1]=="t")\
...     .map(lambda x:(x[0],1))\
...     .reduceByKey(lambda a,b:a +b)\
...     .sortBy(lambda x: x[1], ascending = False)\
...     .collect()
20/08/06 16:02:10 WARN TaskSetManager: Stage 59 contains a task of very large size (4517 KiB). The maximum recommended task size is 1000 KiB.
>>>
>>> finalCalDf = spark.createDataFrame(finalCal)
>>> finalCalDf.show()
+-------+---+
|     _1| _2|
+-------+---+
|2843445|365|
|5652147|365|
|7879708|365|
|4452800|365|
|4736217|365|
|8827268|365|
|1106555|365|
|6939126|365|
|5834930|365|
|1425973|365|
|2915643|365|
|5455004|365|
|3673688|365|
| 182049|365|
| 238846|365|
|4917583|365|
|1303261|365|
|6488924|365|
|1529393|365|
| 973695|365|
```

Step 6)

Rename Columns of DataFrame

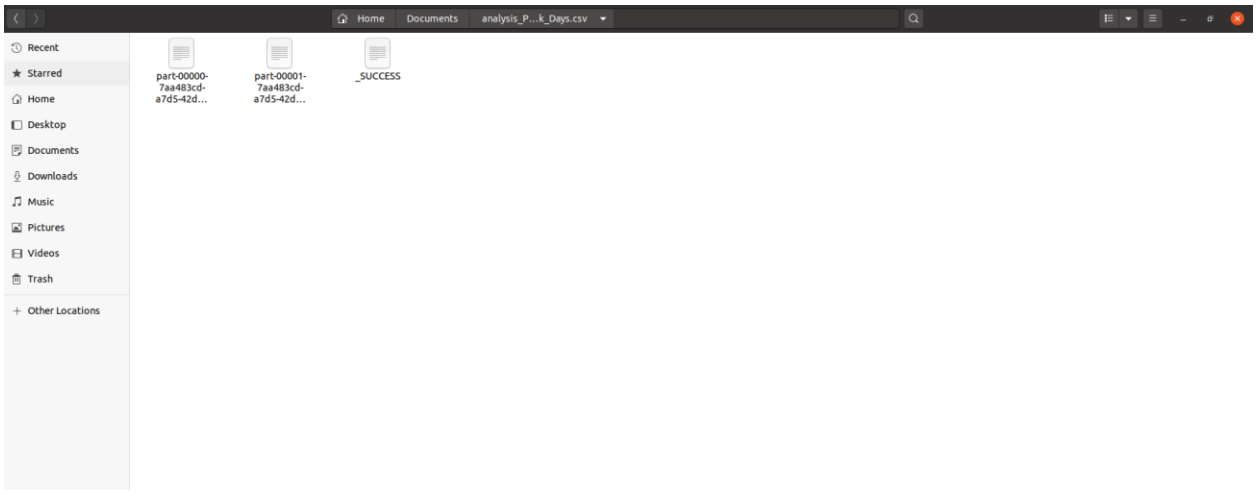
Store the DataFrame in form of CSV

```
finalCalDf = finalCalDf.withColumnRenamed("_1","listingId") \
   .withColumnRenamed("_2","Number_of_available_Days")
finalCalDf.printSchema()
```

finalCalDf.show()

```
>>> finalCalDf = finalCalDf.withColumnRenamed("_1","listingId") \
...     .withColumnRenamed("_2","Number_of_available_Days")
>>> finalCalDf.printSchema()
root
 |-- listingId: string (nullable = true)
 |-- Number_of_available_Days: long (nullable = true)

>>>
>>> finalCalDf.show()
+---------+------------------------+
|listingId|Number_of_available_Days|
+---------+------------------------+
|  2843445|                     365|
|  5652147|                     365|
|  7879708|                     365|
|  4452800|                     365|
|  4736217|                     365|
|  8827268|                     365|
|  1106555|                     365|
|  6939126|                     365|
|  5834930|                     365|
|  1425973|                     365|
|  2915643|                     365|
|  5455004|                     365|
|  3673688|                     365|
|   182049|                     365|
|   238846|                     365|
|  4917583|                     365|
|  1303261|                     365|
|  6488924|                     365|
|  1529393|                     365|
|   973695|                     365|
```

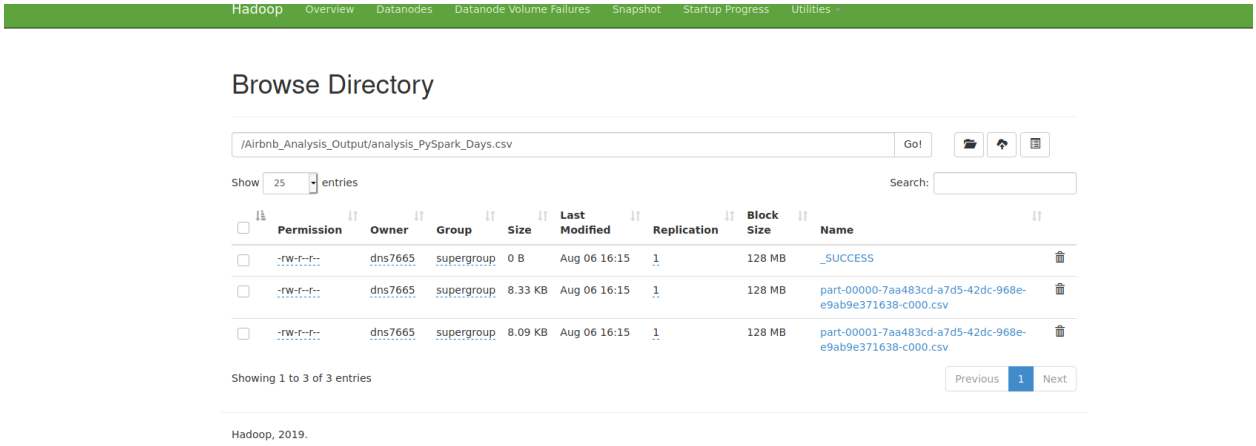
finalCalDf.write.format('csv').option('header',True).mode('overwrite').option('sep',',').save('file:///home/dns7665/Documents/analysis_PySpark_Days.csv')



Step 7)

Use put command in hdfs to move the output to the HDFS directory

/usr/local/bin/hadoop-2.10.0/bin/hadoop fs -put ~/Documents/analysis_PySpark_Days.csv /Airbnb_Analysis_Output




```
dns7665@ubuntu:/usr/local/bin/hadoop-2.10.0/bin$ ./hadoop fs -cat /Airbnb_Analysis_Output/analysis_PySpark_Days.csv/part-00000-7aa483cd-a7d5-42dc-968e-e
9ab9e371638-c000.csv
listingId,Number_of_available_Days
2843445,365
5652147,365
7879708,365
4452800,365
4736217,365
8827268,365
1106555,365
6939126,365
5834930,365
1425973,365
2915643,365
5455004,365
3673688,365
182049,365
238846,365
4917583,365
1303261,365
6488924,365
1529393,365
973695,365
8955473,365
4812786,365
5295491,365
8404805,365
6119918,365
4556374,365
1523772,365
```

Apache Mahout

Analysis 11: Recommendation Using Mahout

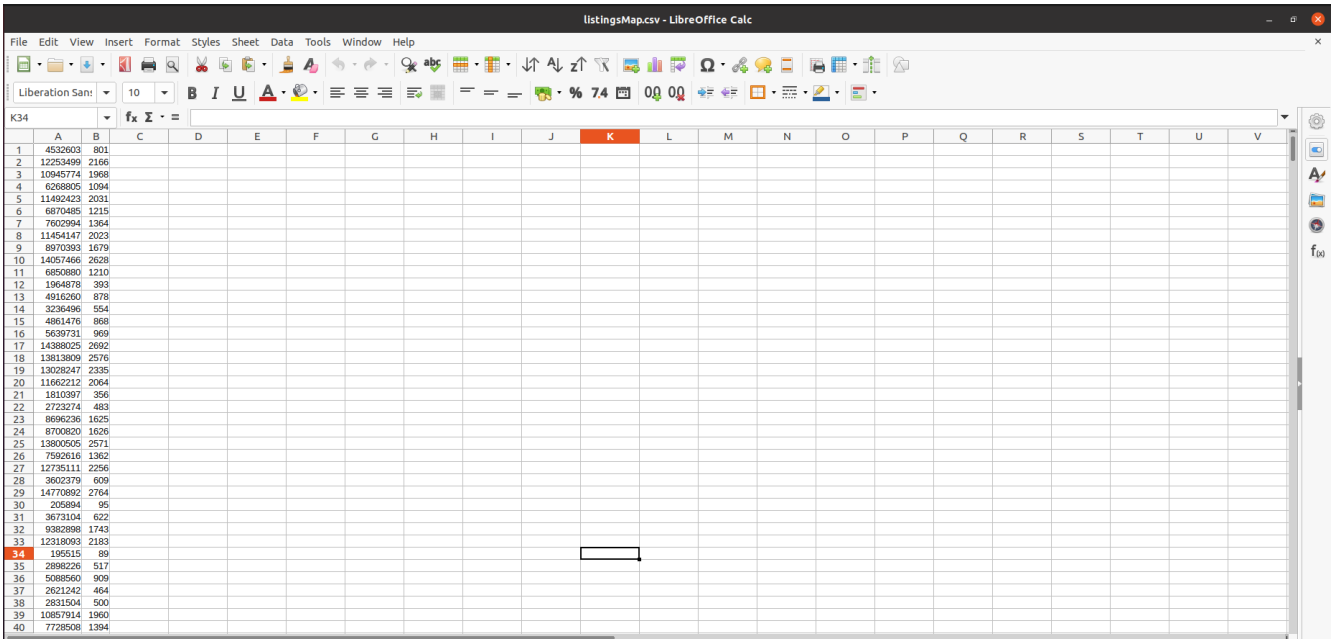
Step 1: Pig Commands to get the reviewerId, listingId, reviewRatingsScore

Open *Mahout ~/Documents Save

```
Mahout:


DataCleaning using pig

reviews = LOAD '/home/dns7665/Downloads/Airbnb_boston_Dataset/reviews.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'NO_MULTILINE', 'UNIX', 'SKIP_INPUT_HEADER');
 filterReview = FOREACH reviews GENERATE (int) $0 as listing_Id, (int)$3 as reviewer_Id;

limitFilter = LIMIT filterReview 5;

describe filterReview;

dump limitFilter;

testRev = filter filterReview by $0 is not null and  $1 is not null;

listings = LOAD '/home/dns7665/Downloads/Airbnb_boston_Dataset/listings.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'NO_MULTILINE', 'UNIX', 'SKIP_INPUT_HEADER');

filterList = FOREACH listings GENERATE (int) $0 as listing_Id, (int) $7 as review_ratings_score;

describe filterList;

test = filter filterList by $0 is not null and $1 is not null;

repJoin = Join testRev by listing_Id, test by listing_Id;

mahoutInput=FOREACH repJoin GENERATE (filterList = FOREACH listings GENERATE (int) $0 as listing_Id, (int) $1 as userId, (int) $3 as rating;

STORE mahoutInput INTO '/home/dns7665/Documents/MahoutInput' Using PigStorage(',');
```

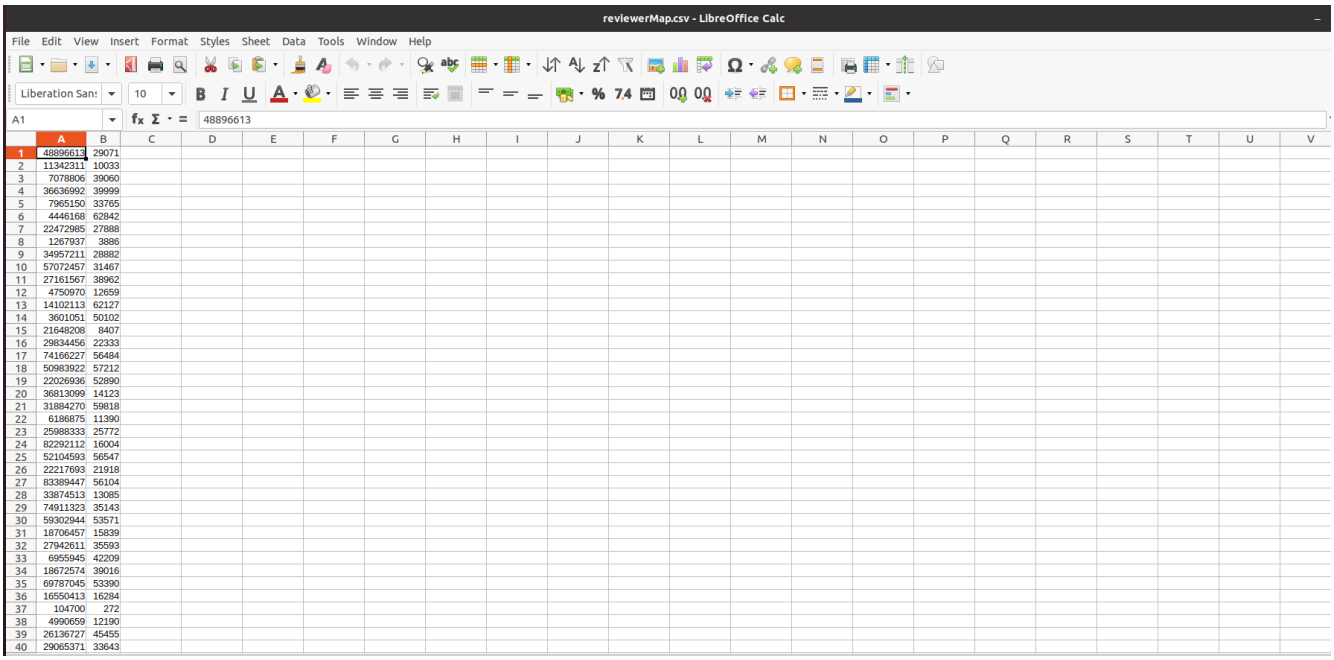
Step 2: To create a mapping file, since the output file is unstructured, and Mahout needs a proper structured file to perform recommendation so we are creating a hashmap for listingId and reviewerId and save their mapping

Listing Mapping



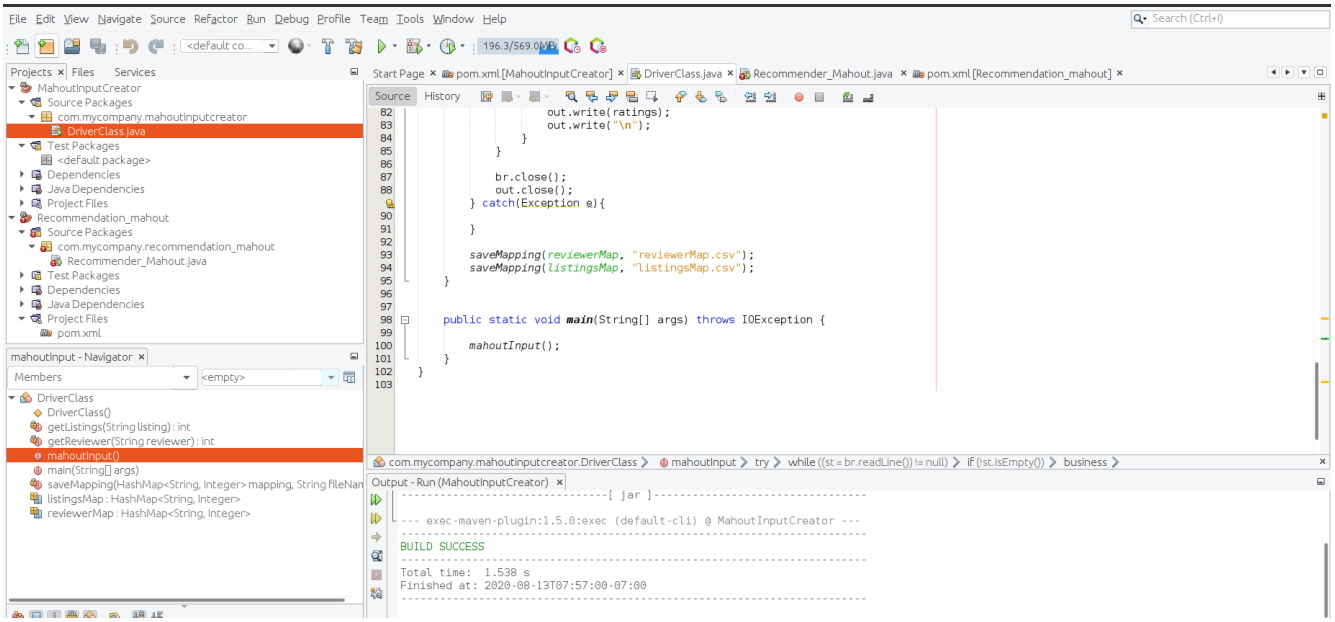
	A	B
1	4532603	801
2	12253499	2166
3	10945774	1968
4	6268805	1094
5	11492423	2031
6	6870485	1215
7	7602994	1364
8	11454147	2023
9	8970393	1679
10	14057466	2628
11	6850880	1210
12	1964878	393
13	4916260	878
14	3236496	554
15	4861476	868
16	5639731	969
17	14388025	2692
18	13813809	2576
19	13028247	2335
20	11662212	2064
21	1810397	356
22	2723274	483
23	8696236	1625
24	8700820	1626
25	13800505	2571
26	7592616	1362
27	12735111	2256
28	3602379	609
29	14770892	2764
30	205894	95
31	3673104	622
32	9382898	1743
33	12318093	2183
34	195515	89
35	2898226	517
36	5088560	909
37	2621242	464
38	2831504	500
39	10857914	1960
40	7728508	1394

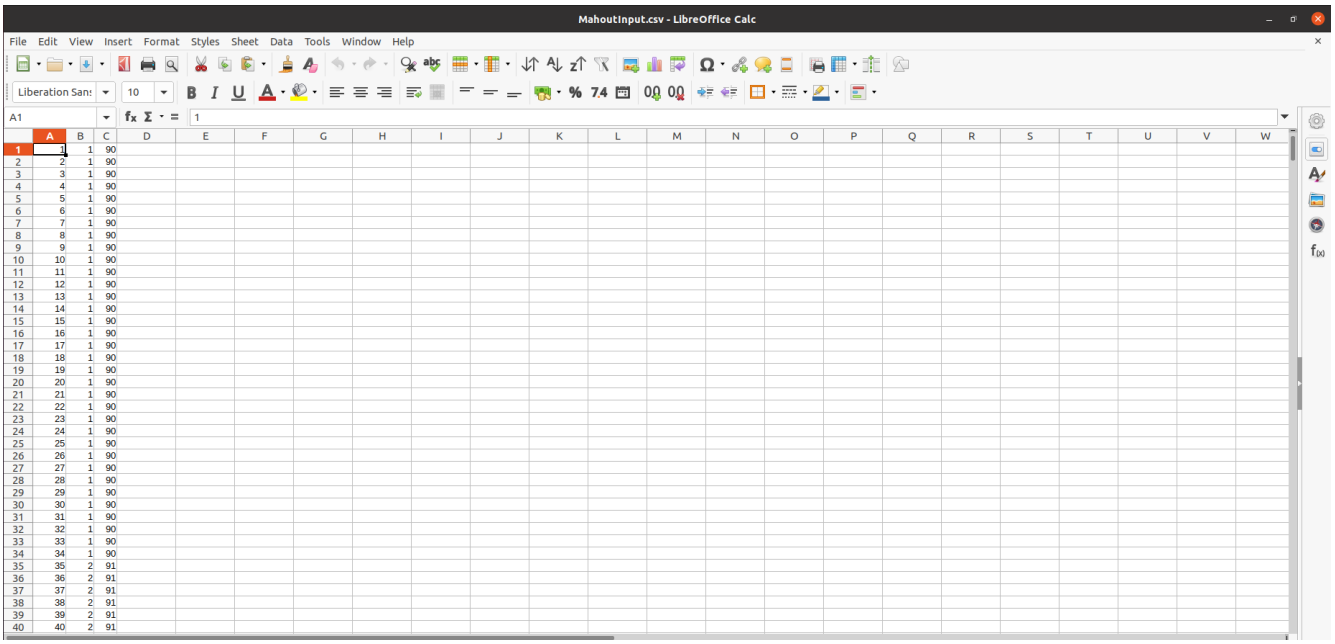
Reviewer Mapping



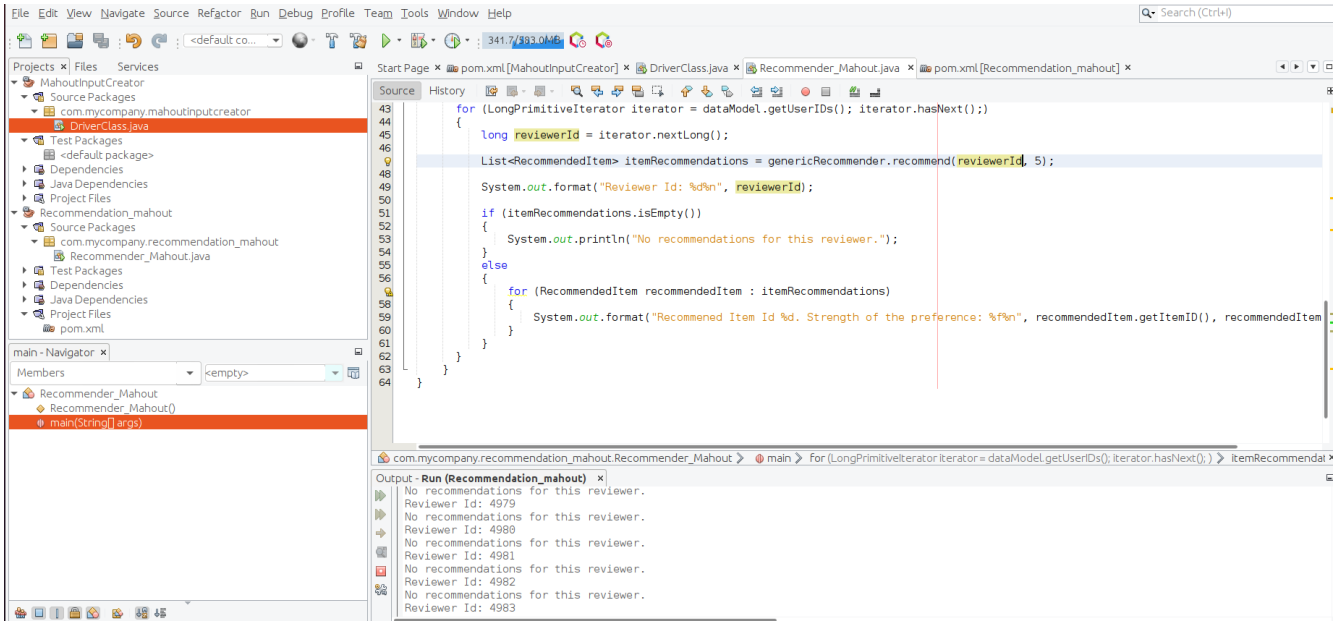
	A	B
1	48896613	29071
2	11342311	10033
3	7078806	39060
4	36636992	39999
5	7965150	33765
6	4446168	62842
7	22472985	27888
8	1267937	3886
9	34957211	28882
10	57072457	31467
11	27161567	38962
12	4750970	12659
13	14102113	62127
14	3601051	50102
15	21648208	8407
16	29834456	22333
17	74166227	56484
18	50983922	57212
19	22026936	52890
20	36813099	14123
21	31884270	59818
22	6186875	11390
23	25988333	25772
24	82292112	16004
25	52104593	56547
26	22217693	21918
27	83389447	56104
28	33874513	13085
29	74911323	35143
30	59302944	53571
31	18706457	15839
32	27942611	35593
33	6955945	42209
34	18672574	39016
35	69787045	53390
36	16550413	16284
37	104700	272
38	4990659	12190
39	26136727	45455
40	29065371	33643

Mahout Input Passed to Mahout Recommendation Engine





Step 3: Mahout Recommendation



Appendix:

Analysis 1:

MongoDb commands:

Import file

```
mongoimport --db FinalProjDB --collection AirBnBListings --type csv --file
C:\Users\shahd\Documents\BigData\Airbnb_FinalProject\BostonAirbnb_Dataset\l
istings.csv --headerline;
```

Map:

```
function (){
      emit({neighborhood:
this.neighbourhood_cleansed},{Number_of_listings:1})
}
```

Reduce:

```
function(key,countObjVals){

reducedVal = { Number_of_listings:0 };
 for (var idx = 0; idx < countObjVals.length; idx++) {

      if(typeof countObjVals[idx] != "undefined")
            reducedVal.Number_of_listings =
reducedVal.Number_of_listings+countObjVals[idx].Number_of_listings;
      }
```

```
 return{Number_of_listings:reducedVal.Number_of_listings};
}
```

Analysis 2: Pig Replicated Join

- reviews = LOAD '/home/dns7665/Downloads/Airbnb_boston_Dataset/reviews.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'NO_MULTILINE', 'UNIX', 'SKIP_INPUT_HEADER');
- filterReview = FOREACH reviews GENERATE (chararray) $0 as listing_Id, (chararray) $1 as review_Id, (chararray) $2 as date, (chararray) $3 as reviewer_Id, (chararray) $4 as reviewer_name, (chararray) $5 as ReviewContent;
- limitFilter = LIMIT filterReview 5;
- describe filterReview;
- dump limitFilter;
- listings = LOAD '/home/dns7665/Downloads/Airbnb_boston_Dataset/listings.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'NO_MULTILINE', 'UNIX', 'SKIP_INPUT_HEADER');
- filterList = FOREACH listings GENERATE (chararray) $0 as listing_Id, (chararray) $1 as listing_Url, (chararray) $4 as listing_name, (chararray) $7 as listing_desc;
- describe filterList;
- limitList = LIMIT filterList 5;
- dump limitList
- repJoin = Join filterReview by listing_Id, filterList by listing_Id USING 'replicated';
- limitJoin = LIMIT repJoin 5;
- dump limitJoin
- STORE repJoin INTO '/home/dns7665/Documents/ReplicatedJoin_Pig.txt' Using PigStorage(',');

Analysis 3: PIG-Get listings with more than 50 reviews

- reviews = LOAD '/home/dns7665/Downloads/Airbnb_boston_Dataset/reviews.csv' USING org.apache.pig.piggybank.storage.CSVExcelStorage(',', 'NO_MULTILINE', 'UNIX', 'SKIP_INPUT_HEADER');
- filterReview = FOREACH reviews GENERATE (chararray) $0 as listing_Id, (chararray) $1 as review_Id;
- limitR = LIMIT filterReview 5;
- limitR = LIMIT filterReview 50;
- filterReview = FOREACH reviews GENERATE (int) $0 as listing_Id, (int) $1 as review_Id;
- limitR = LIMIT filterReview 50;
- distinct_data = DISTINCT limitR;
- distinct_data = DISTINCT filterReview;
- reviewGroup = Group distinct_data by listing_Id;
- reviewCount = FOREACH reviewGroup GENERATE group as listing_Id, COUNT(distinct_data) as reviews;
- resultGenerator = FOREACH reviewCount GENERATE listing_Id, reviews;
- limitres = LIMIT resultGenerator 10;
- finalResult = FILTER resultGenerator by reviews > 50;
- STORE finalResult INTO '/home/dns7665/Documents/ListingsWithMoreThan50Rev.txt' Using PigStorage(',');

Analysis 4: PIG-Get listings with more than 50 reviews

- CREATE TABLE listings (listingId int, host_since String, neighbourhood String, listing_url String, scrape_Id String, last_Scrape String, price String, review_scores_ratings String) ROW FORMAT DELIMITED FIELDS TERMINATED BY ',';
- Load Data inpath 'hdfs://localhost:9000/Airbnb/Dataset/Airbnb_boston_Dataset/Hive' into table listings;

- Create Table assortedListings as Select listingid, neighbourhood, Cast(review_scores_ratings as int) AS rating_Data from listings where listingid is not NULL AND length(review_scores_ratings)==2 ;
- select * from(Select listingid, neighbourhood, rating_Data, rank() over (partition by neighbourhood order by rating_Data desc) as rank from assortedListings) r where rank <7;
- Insert Overwrite Directory 'hdfs://localhost:9000/Airbnb_Analysis_Output/analysis_Hive_Top7ListingsForEachNeighbourhood/' select * from(Select listingid, neighbourhood, rating_Data, rank() over (partition by neighbourhood order by rating_Data desc) as rank from assortedListings) r where rank <7;

Analysis 5: MapReduce Partitioning data by year

DriverClass

```
public class DriverClass {

  public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {
    Configuration conf = new Configuration();
    Job job = new Job(conf, "DataOrg");
    job.setJarByClass(DriverClass.class);

    job.setPartitionerClass(PartitionerClass.class);
    PartitionerClass.setMinLastAccessDate(job, 2008);

    job.setNumReduceTasks(9);

    FileInputFormat.addInputPath(job, new Path(args[0]));
    FileOutputFormat.setOutputPath(job, new Path(args[1]));

    job.setMapOutputKeyClass(IntWritable.class);
    job.setMapOutputValueClass(Text.class);

    job.setMapperClass(MapperClass.class);
    job.setReducerClass(ReducerClass.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(NullWritable.class);
```

```
    System.exit(job.waitForCompletion(true)? 0:1);
  }

}
```

MapperClass

```
public class MapperClass extends Mapper<Object, Text, IntWritable, Text>{

  private final static SimpleDateFormat dateFormat = new SimpleDateFormat("MM/DD/yyyy");

  private IntWritable outputKey = new IntWritable();

  @Override
  protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {

    String values[] = value.toString().split(",", -1);
    if (values.length>2 && values[1]!=null && values[0] != null && !values[1].equals("host_since")) {

      Pattern p = Pattern.compile("\\d+");
      Matcher m = p.matcher(values[0]);
      if (m.matches()) {
        String strDate = "";
        if (!values[1].equals("host_since")) {
          strDate = values[1];
        }

        String strDateSplit[] = strDate.split("/");

        if (strDateSplit.length == 3 && strDateSplit[2] != null) {
          outputKey.set(Integer.parseInt(strDateSplit[2]));
        }
        context.write(outputKey, value);
      }
    }
  }
}
```

PartitionerClass

```
public class PartitionerClass extends Partitioner<IntWritable, Text> implements Configurable {

  private static final String MIN_LAST_ACCESS_DATE_YEAR = "min.last.access.date.year";

  private Configuration conf = null;
  private int minLastAccessDateYear = 0;

  public int getPartition(IntWritable key, Text value, int numPartitions) {
```

```
    return Math.abs(key.get() - minLastAccessDateYear);
  }

  public Configuration getConf() {
    return conf;
  }

  public void setConf(Configuration conf) {
    this.conf = conf;
    minLastAccessDateYear = conf.getInt(MIN_LAST_ACCESS_DATE_YEAR, 0);
  }

  public static void setMinLastAccessDate(Job job, int minLastAccessDateYear) {
    job.getConfiguration().setInt(MIN_LAST_ACCESS_DATE_YEAR, minLastAccessDateYear);
  }
}
```

ReducerClass:

```
public class ReducerClass extends Reducer<IntWritable, Text, Text, NullWritable> {

  protected void reduce(IntWritable key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {
    for (Text t : values) {
      context.write(t, NullWritable.get());
    }
  }

}
```

Analysis 6: MapReduce Partitioning data by year and Chaining to find number of hosts joining Airbnb in each year

DriverClass:

```
public class DriverClass {

  public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {
    Configuration conf = new Configuration();
    Job job = new Job(conf, "DataOrg");
    job.setJarByClass(DriverClass.class);

    job.setPartitionerClass(PartitionerClass.class);
    PartitionerClass.setMinLastAccessDate(job, 2008);

    job.setNumReduceTasks(9);
```

```
    FileInputFormat.addInputPath(job, new Path(args[0]));
    Path outDir = new Path(args[1]);
    FileOutputFormat.setOutputPath(job, outDir);

    FileSystem fs = FileSystem.get(job.getConfiguration());
    if(fs.exists(outDir)){
       fs.delete(outDir, true);
    }
    job.setMapOutputKeyClass(IntWritable.class);
    job.setMapOutputValueClass(Text.class);

    job.setMapperClass(MapperClass.class);
    job.setReducerClass(ReducerClass.class);

    job.setOutputKeyClass(Text.class);
    job.setOutputValueClass(NullWritable.class);

    boolean result = job.waitForCompletion(true);
    if(result) {
       Job job1 = Job.getInstance(conf, "SecondMR");
       job1.setJarByClass(DriverClass.class);

       job1.setMapperClass(MapperClassTwo.class);
       job1.setCombinerClass(ReducerClassTwo.class);
       job1.setReducerClass(ReducerClassTwo.class);

       job1.setOutputKeyClass(Text.class);
       job1.setOutputValueClass(IntWritable.class);

       FileInputFormat.addInputPath(job1, new Path(args[1]));
       Path outDir2 = new Path(args[2]);
       FileOutputFormat.setOutputPath(job1, outDir2);

        FileSystem fs2 = FileSystem.get(job1.getConfiguration());
       if(fs2.exists(outDir2)){
          fs2.delete(outDir2, true);
       }


       result = job1.waitForCompletion(true);
    }

  }

}
```

MapperClass:

```
public class MapperClass extends Mapper<Object, Text, IntWritable, Text>{
```

```
    private IntWritable outputKey = new IntWritable();

    @Override
    protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {

        String values[] = value.toString().split(",", -1);
        if (values.length>2 && values[1]!=null && values[0] != null && !values[1].equals("host_since")) {

            Pattern p = Pattern.compile("\\d+");
            Matcher m = p.matcher(values[0]);
            if (m.matches()) {
                String strDate = "";
                if (!values[1].equals("host_since")) {
                    strDate = values[1];
                }

                String strDateSplit[] = strDate.split("/");

                if (strDateSplit.length == 3 && strDateSplit[2] != null) {
                    outputKey.set(Integer.parseInt(strDateSplit[2]));
                }
                context.write(outputKey, value);
            }
        }
    }
}
```

MapperClassTwo:

```
public class MapperClassTwo extends Mapper<Object, Text, Text, IntWritable>{

    private IntWritable MapOutVal = new IntWritable();
    private Text MapOutKey = new Text();


    @Override
    protected void map(Object key, Text value, Context context) throws IOException, InterruptedException {

        String values[] = value.toString().split(",", -1);
        if (values.length>2 && values[1]!=null && values[0] != null && !values[1].equals("host_since")) {

            Pattern p = Pattern.compile("\\d+");
            Matcher m = p.matcher(values[0]);
            if (m.matches()) {
                String strDate = "";
                if (!values[1].equals("host_since")) {
                    strDate = values[1];
                }

                String strDateSplit[] = strDate.split("/");
```

```
            if (strDateSplit.length == 3 && strDateSplit[2] != null) {
               MapOutKey.set(strDateSplit[2]);
            }
            MapOutVal.set(1);
            context.write(MapOutKey, MapOutVal);
         }
      }
   }
}
```

PartitionerClass:

```
public class PartitionerClass extends Partitioner<IntWritable, Text> implements Configurable {

   private static final String MIN_LAST_ACCESS_DATE_YEAR = "min.last.access.date.year";

   private Configuration conf = null;
   private int minLastAccessDateYear = 0;

   public int getPartition(IntWritable key, Text value, int numPartitions) {
      return Math.abs(key.get() - minLastAccessDateYear);
   }

   public Configuration getConf() {
      return conf;
   }

   public void setConf(Configuration conf) {
      this.conf = conf;
      minLastAccessDateYear = conf.getInt(MIN_LAST_ACCESS_DATE_YEAR, 0);
   }

   public static void setMinLastAccessDate(Job job, int minLastAccessDateYear) {
      job.getConfiguration().setInt(MIN_LAST_ACCESS_DATE_YEAR, minLastAccessDateYear);
   }
}
```

ReducerClass:

```
public class ReducerClass extends Reducer<IntWritable, Text, Text, NullWritable> {

 protected void reduce(IntWritable key, Iterable<Text> values, Context context) throws IOException,
InterruptedException {
      for (Text t : values) {
         context.write(t, NullWritable.get());
```

```
    }
  }

}
```

ReducerClassTwo:

```
public class ReducerClassTwo extends Reducer<Text, IntWritable, Text, IntWritable> {

  private IntWritable no_of_hosts = new IntWritable();
  protected void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException,
InterruptedException {

        int hostSum = 0;
        for (IntWritable val : values)
        {
           hostSum += val.get();
        }
        no_of_hosts.set(hostSum);
        context.write(key, no_of_hosts);

  }

}
```

Analysis 7: Secondary Sort-To Sort the Listings according to reviews received per year

CompositeKeyWritable Class:

```
public class CompositeKeyWritable implements WritableComparable<CompositeKeyWritable> {

  String listingId;
  String month;

  public CompositeKeyWritable(){
  }

  public CompositeKeyWritable(String listingId, String month) {
    this.listingId = listingId;
    this.month = month;
  }

  public String getListingId() {
    return listingId;
  }
```

```
  public void setListingId(String listingId) {
    this.listingId = listingId;
  }

  public String getMonth() {
    return month;
  }

  public void setMonth(String month) {
    this.month = month;
  }

  public void readFields(DataInput in) throws IOException {
    listingId = in.readUTF();
    month = in.readUTF();
  }

  public void write(DataOutput out) throws IOException {
    out.writeUTF(listingId);
    out.writeUTF(month);
  }

  public int compareTo(CompositeKeyWritable o) {
    int result = this.month.compareTo(o.getMonth());
    return (result < 0 ? -1 : (result == 0 ? 0 : 1));
  }

  @Override
  public String toString() {
    return listingId + "," + month;
  }
}
```

CompositeKeyComparator:

```
  public CompositeKeyComparator(){
    super(CompositeKeyWritable.class, true);
  }

  @Override
  public int compare(WritableComparable a, WritableComparable b) {
    CompositeKeyWritable ckw1 = (CompositeKeyWritable) a;
    CompositeKeyWritable ckw2 = (CompositeKeyWritable) b;

    int result=0;

      result = -1 * ckw1.getMonth().compareTo(ckw2.getMonth());
```

```
        return result;
    }
}
```

DriverClass:

```
public class DriverClass {

    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {
        Configuration config = new Configuration();
        Job job = Job.getInstance(config, "SecondarySort");

        job.setJarByClass(DriverClass.class);

        job.setGroupingComparatorClass(NaturalGroupingKeyComparator.class);
        job.setSortComparatorClass(CompositeKeyComparator.class);
        job.setPartitionerClass(NaturalKeyPartitioner.class);

        job.setMapOutputKeyClass(CompositeKeyWritable.class);
        job.setMapOutputValueClass(IntWritable.class);

        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);

        job.setMapperClass(MapperClass.class);
        job.setReducerClass(ReducerClass.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
        Path outDir = new Path(args[1]);
        FileOutputFormat.setOutputPath(job, outDir);

        job.setNumReduceTasks(1);

        FileSystem fs = FileSystem.get(job.getConfiguration());
        if(fs.exists(outDir)){
            fs.delete(outDir, true);
        }

        job.waitForCompletion(true);

    }
}
```

MapperClass:

```
public class MapperClass extends Mapper<LongWritable, Text, CompositeKeyWritable, IntWritable> {

   IntWritable one = new IntWritable(1);

   @Override
   protected void map(LongWritable key, Text value, Context context) throws IOException,
InterruptedException {
      String line = value.toString();
      String[] tokens = line.split(",");

      String listingId = null;
      String reviewDate = null;

         if ( tokens.length>4 && tokens[0] != null && !tokens[0].equals("listing_id") && tokens[0].length() > 0 &&
tokens[2] != null && tokens[2].length() > 0) {
            Pattern p = Pattern.compile("\\d+");
            Matcher m = p.matcher(tokens[0]);
            if (m.matches()) {
               listingId = tokens[0];
                reviewDate = tokens[2];

                String month[] = reviewDate.split("-");
                String monthRev = month[0];


                  CompositeKeyWritable obj = new CompositeKeyWritable(listingId, monthRev);

                  context.write(obj, one);

               }
            }
         }

   }
```

NaturalKeyParitioner:

```
public class NaturalKeyPartitioner extends Partitioner<CompositeKeyWritable, IntWritable> {

   @Override
   public int getPartition(CompositeKeyWritable key, IntWritable value, int noOfPartitions) {
      return key.getListingId().hashCode() % noOfPartitions;
   }
}
```

NaturalGroupingKeyComparator:

```
public class NaturalGroupingKeyComparator extends WritableComparator {
```

```
    public NaturalGroupingKeyComparator(){
        super(CompositeKeyWritable.class, true);
    }

    @Override
    public int compare(WritableComparable a, WritableComparable b) {
        CompositeKeyWritable ckw1 = (CompositeKeyWritable) a;
        CompositeKeyWritable ckw2 = (CompositeKeyWritable) b;

        int result = ckw1.getListingId().compareTo(ckw2.getListingId());
        return result;
    }
}
```

ReducerClass:

```
public class ReducerClass extends Reducer<CompositeKeyWritable, IntWritable, Text, IntWritable> {

    IntWritable result = new IntWritable();
    Text text = new Text();
    @Override
    protected void reduce(CompositeKeyWritable key, Iterable<IntWritable> values, Context context) throws
IOException, InterruptedException {
        int count=0;
        for(IntWritable val : values){
            count +=val.get();
        }
        text.set(key.getListingId() + "," + key.getMonth());
        result.set(count);
        context.write(text, result);
    }
}
```

Analysis 8: To find the top 20 listings according to ratings given by guests

DriverClass:

```
public class DriverClass {

    public static void main(String[] args) throws IOException, ClassNotFoundException, InterruptedException {

        Configuration conf = new Configuration();
        Job job = Job.getInstance(conf, "Top20Listings");
        job.setJarByClass(DriverClass.class);

        FileInputFormat.addInputPath(job, new Path(args[0]));
```

```
        FileOutputFormat.setOutputPath(job, new Path(args[1]));


        job.setInputFormatClass(TextInputFormat.class);
        job.setOutputFormatClass(TextOutputFormat.class);
        job.setMapOutputKeyClass(Text.class);
        job.setMapOutputValueClass(IntWritable.class);


        job.setMapperClass(MapperClass.class);
        job.setReducerClass(ReducerClass.class);

        job.setOutputKeyClass(Text.class);
        job.setOutputValueClass(IntWritable.class);


        boolean result = job.waitForCompletion(true);

        if(result) {
           Job job1 = Job.getInstance(conf, "Top20Listings");
           job1.setJarByClass(DriverClass.class);

           job1.setInputFormatClass(TextInputFormat.class);
           job1.setOutputFormatClass(TextOutputFormat.class);

           job1.setMapperClass(MapperClassTwo.class);
           job1.setReducerClass(ReducerClassTwo.class);

           job1.setOutputKeyClass(NullWritable.class);
           job1.setOutputValueClass(Text.class);

           FileInputFormat.addInputPath(job1, new Path(args[1]));
           FileOutputFormat.setOutputPath(job1, new Path(args[2]));

           result = job1.waitForCompletion(true);
        }
     }
}
```

MapperClass:

```
public class MapperClass extends Mapper<Object, Text, Text, IntWritable>{

   Text MapOutputKey = new Text();
   IntWritable MapOutputValue = new IntWritable();

   @Override
   protected void map(Object key, Text value, Mapper<Object, Text, Text, IntWritable>.Context context)
         throws IOException, InterruptedException {
```

```
      try {
         String values[] = value.toString().split(",");
         MapOutputKey.set(values[0]);
         MapOutputValue.set(1);
         context.write(MapOutputKey, MapOutputValue);
      } catch(Exception e) {

      }
   }
}
```

MapperClassTwo:

```
public class MapperClassTwo extends Mapper<Object, Text, NullWritable, Text> {

   private TreeMap<Integer, Text> topTree = new TreeMap<Integer, Text>();

   @Override
   protected void map(Object key, Text value, Mapper<Object, Text, NullWritable, Text>.Context context)
         throws IOException, InterruptedException {

      String values[] = value.toString().split("\t");
      try {
         String listings = values[0];
         int TotalReviews = Integer.parseInt(values[1]);
         topTree.put(TotalReviews, new Text(value));

         if (topTree.size() > 20) {
            topTree.remove(topTree.firstKey());
         }

      } catch (Exception e) {
      }
   }


         @Override
         protected void cleanup (Mapper < Object, Text, NullWritable, Text >.Context context)
                              throws IOException, InterruptedException {

            for (Text t : topTree.values()) {
               context.write(NullWritable.get(), t);
            }
         }

      }
```

ReducerClass:

```
public class ReducerClass extends Reducer<Text, IntWritable, Text, IntWritable> {

   IntWritable ReduceOutputValue = new IntWritable();

   @Override
   protected void reduce(Text key, Iterable<IntWritable> values,
                  Reducer<Text, IntWritable, Text, IntWritable>.Context context) throws IOException,
InterruptedException {

      int totalReviews = 0;
      for(IntWritable val : values) {
         totalReviews += val.get();
      }
      ReduceOutputValue.set(totalReviews);
      context.write(key, ReduceOutputValue);
   }
}
```

ReducerClassTwo:

```
public class ReducerClassTwo extends Reducer<NullWritable, Text, NullWritable, Text>{

   private TreeMap<Integer, Text> topTree = new TreeMap<Integer, Text>();

   @Override
   protected void reduce(NullWritable key, Iterable<Text> values, Reducer<NullWritable, Text, NullWritable,
Text>.Context context) throws IOException, InterruptedException {


      for(Text value : values) {
         String val[] = value.toString().split("\t");
         int count = Integer.parseInt(val[1]);
         topTree.put(count, new Text(value));
         if(topTree.size()> 20) {
            topTree.remove(topTree.firstKey());
         }
      }
   }

   @Override
   protected void cleanup(Reducer<NullWritable, Text, NullWritable, Text>.Context context)
         throws IOException, InterruptedException {
      for(Text t : topTree.descendingMap().values()) {
         context.write(NullWritable.get(), t);
      }
```

```
    }
}
```