**Airbnb Seattle Dataset Analysis**

**Analysis using Map Reduce**

**Listings per Host**

Some Airbnb hosts have multiple listings. A host may list separate rooms in the same apartment, or multiple apartments or homes available in their entirity. Hosts with multiple listings are more likely to be running a business, are unlikely to be living in the property, and in violation of most short term rental laws designed to protect residential housing.

**Activity**

Airbnb guests may leave a review after their stay, and these can be used as an indicator of airbnb activity. The minimum stay, price and number of reviews have been used to estimate the occupancy rate, the number of nights per year and the income per month for each listing.

**Room Type**

Airbnb hosts can list entire homes/apartments, private or shared rooms. Depending on the room type, [availability](http://insideairbnb.com/seattle/#availability), and [activity](http://insideairbnb.com/seattle/#activity), an airbnb listing could be more like a hotel, disruptive for neighbours, taking away housing, and [illegal](http://insideairbnb.com/seattle/).

**Availability**

An Airbnb host can setup a calendar for their listing so that it is only available for a few days or weeks a year. Other listings are available all year round (except for when it is already booked). Entire homes or apartments highly available year-round for tourists, probably don't have the owner present, could be illegal, and more importantly, are displacing residents.

**Top 50 Listings based on the Reviews Rating**

Based on the ratings provided: review\_scores\_rating, review\_scores\_accuracy, review\_scores\_cleanliness, review\_scores\_checkin, review\_scores\_communication, review\_scores\_location, review\_scores\_value we calculate the percentage of each reviews

**Indexing**

Index the listings based on the following criteria:

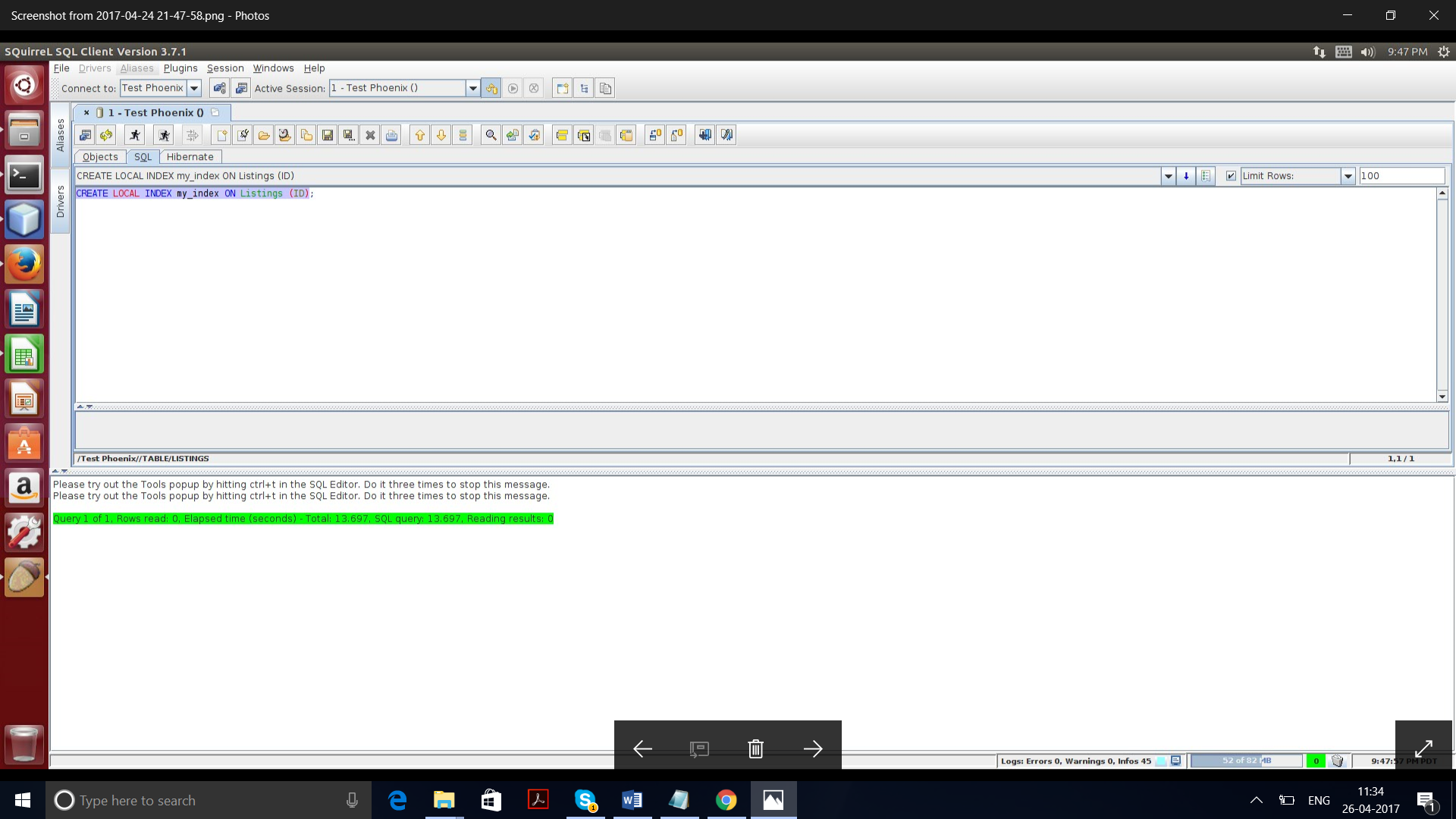
1. Price Range
2. Number of Beds
3. Number of People can be accommodated
4. Neighbourhood

**Partitioning**

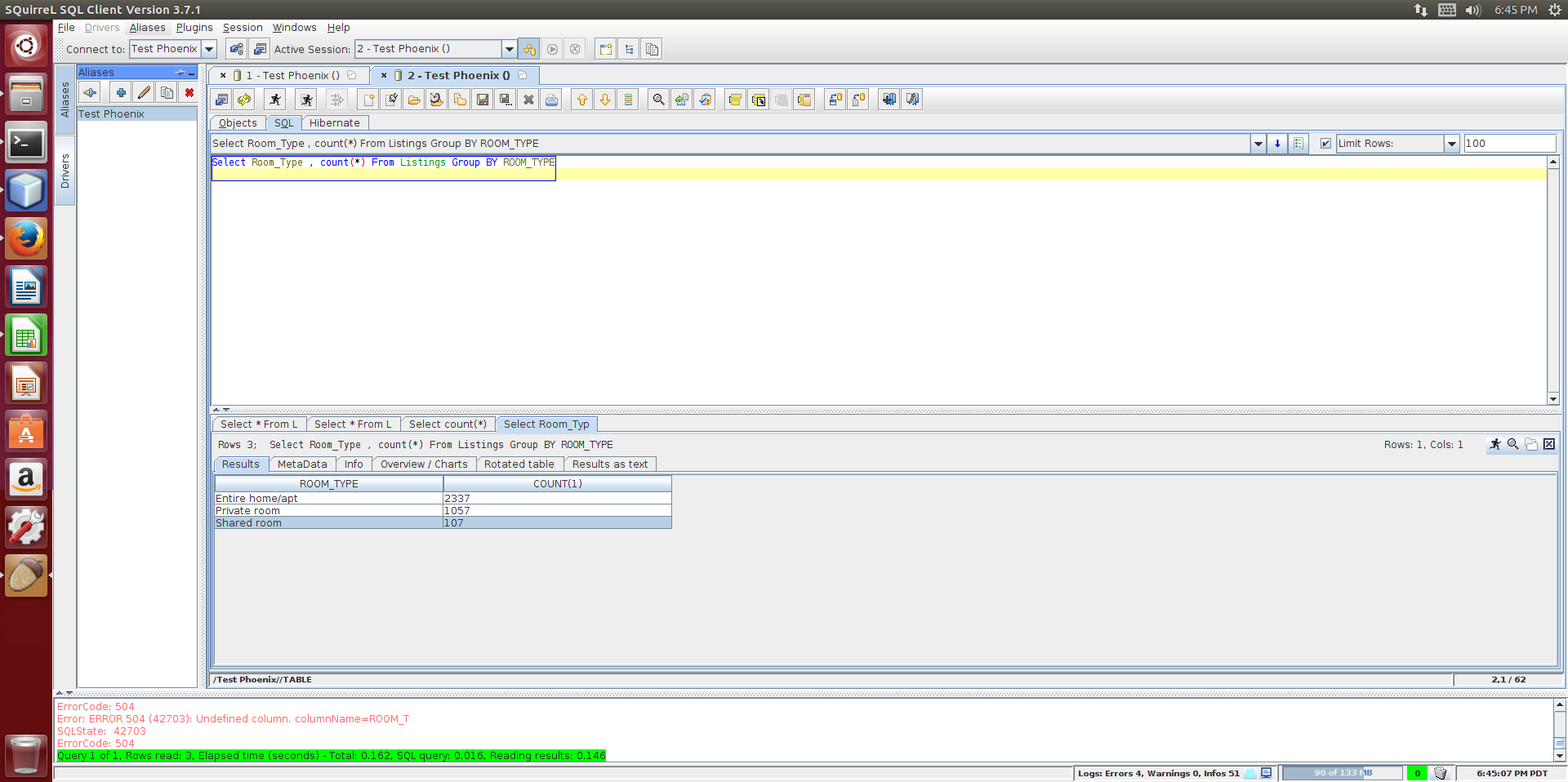
Partitions created on the data based on the room types available.

**Analysis using HBASE**

**Creating Index on HBASE**

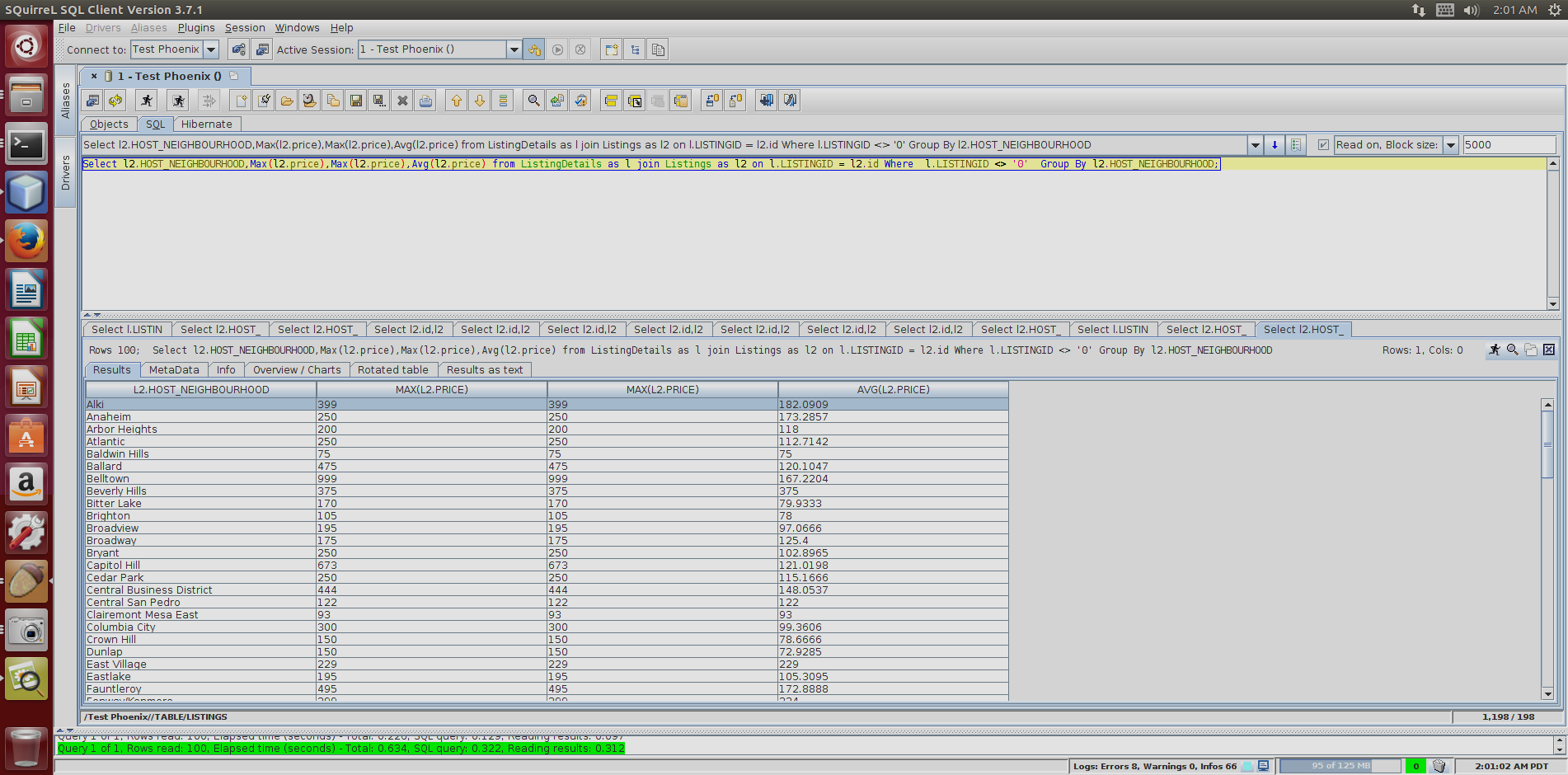


**Query 1:**

****

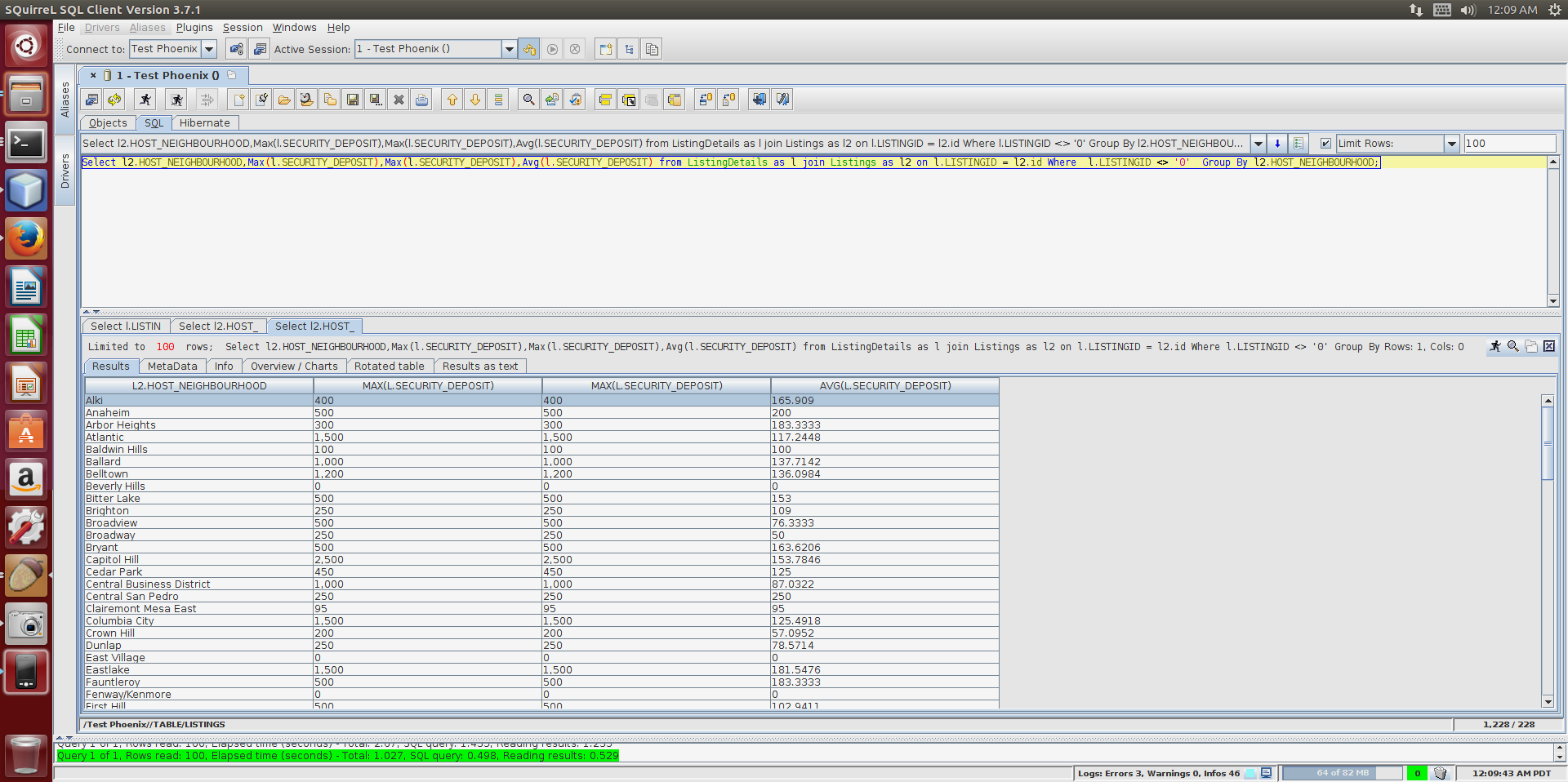
**Query 2:**

Select l2.HOST\_NEIGHBOURHOOD,Max(l2.price),Max(l2.price),Avg(l2.price) from ListingDetails as l join Listings as l2 on l.LISTINGID = l2.id Where l.LISTINGID <> '0' Group By l2.HOST\_NEIGHBOURHOOD;



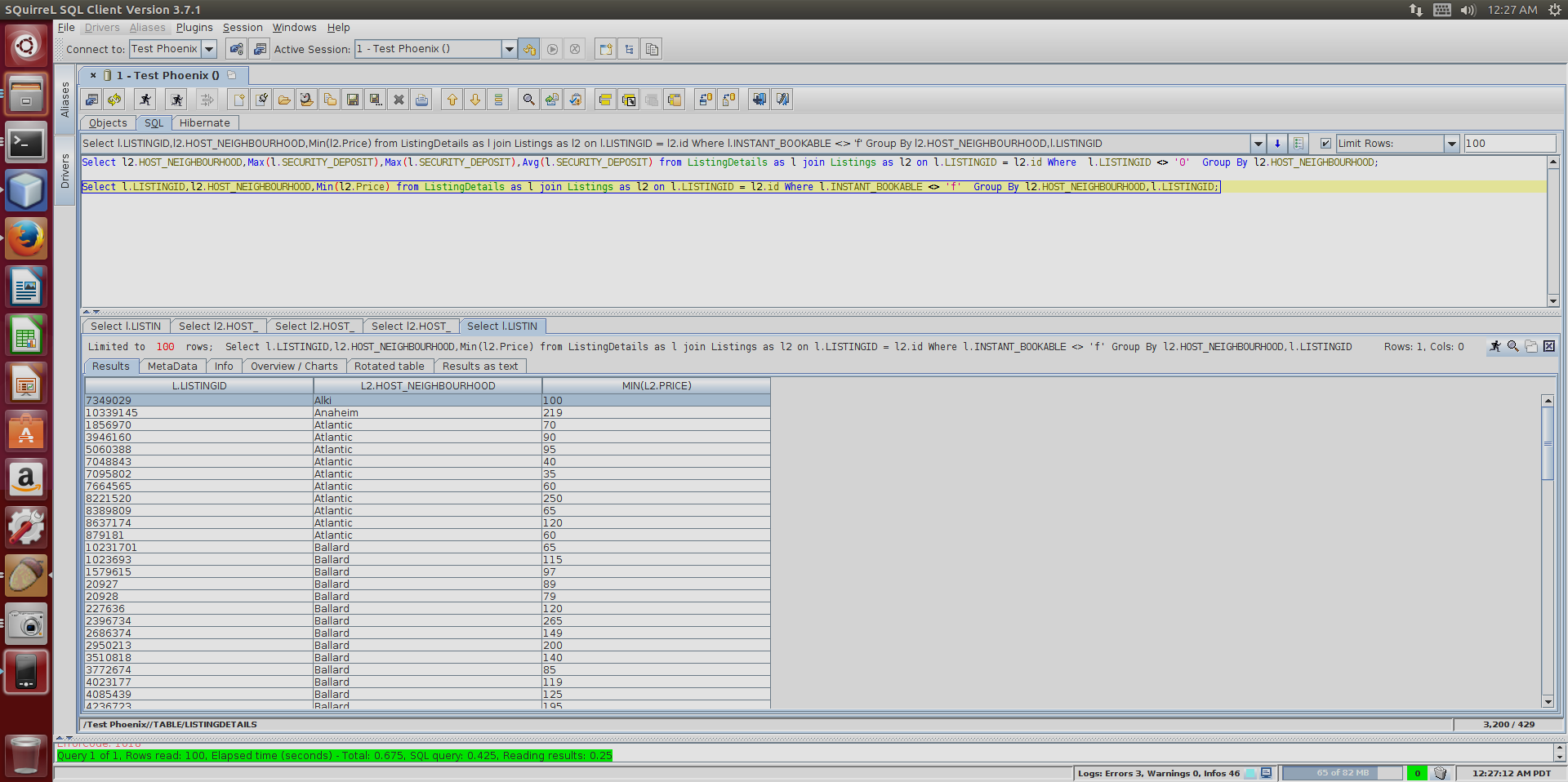
**Query 3:**

Select l2.HOST\_NEIGHBOURHOOD,Max(l.SECURITY\_DEPOSIT),Max(l.SECURITY\_DEPOSIT),Avg(l.SECURITY\_DEPOSIT) from ListingDetails as l join Listings as l2 on l.LISTINGID = l2.id Where l.LISTINGID <> '0' Group By l2.HOST\_NEIGHBOURHOOD;



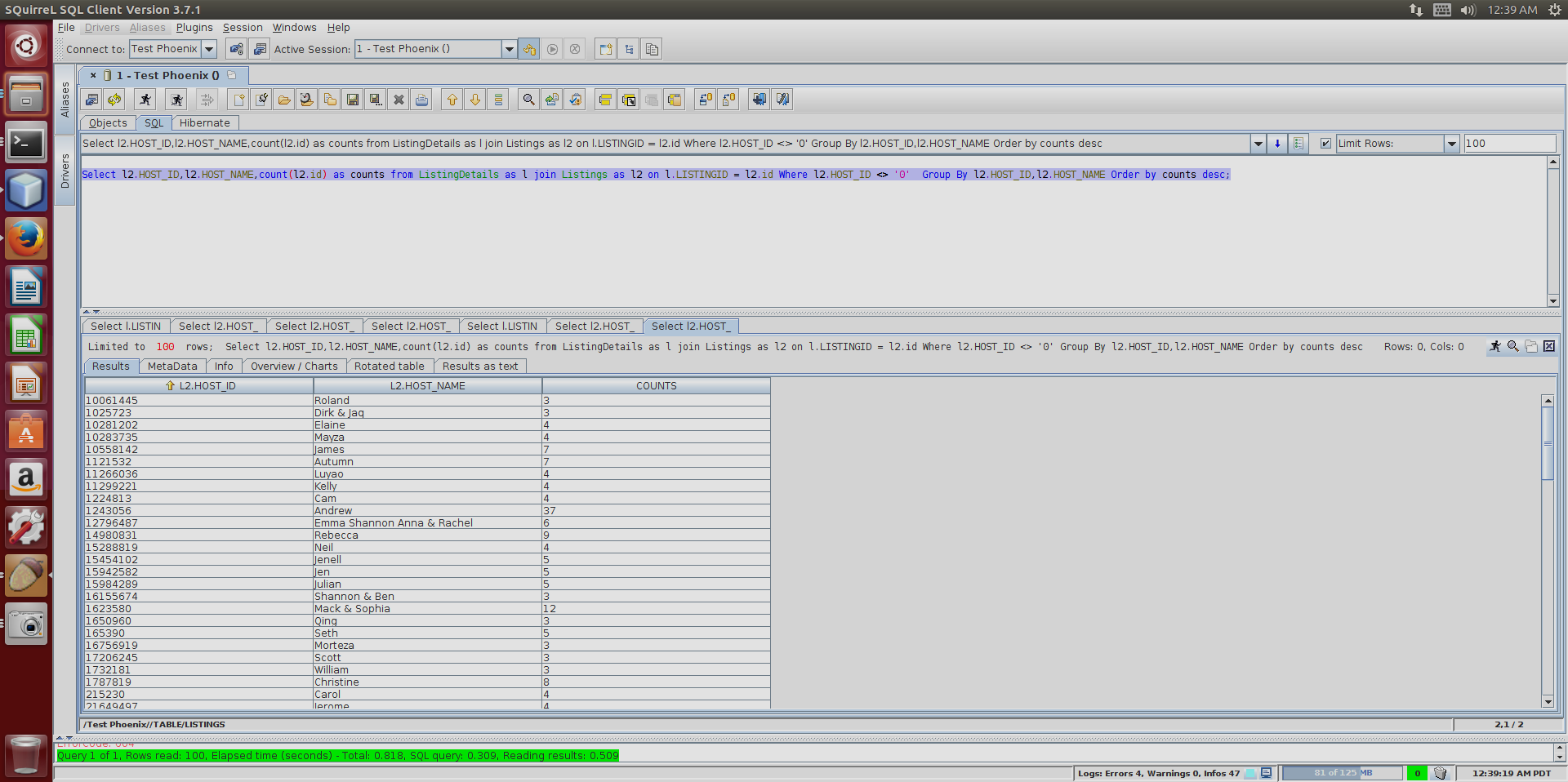
**Query 4:**

Select l.LISTINGID,l2.HOST\_NEIGHBOURHOOD,Min(l2.Price) from ListingDetails as l join Listings as l2 on l.LISTINGID = l2.id Where l.INSTANT\_BOOKABLE <> 'f' Group By l2.HOST\_NEIGHBOURHOOD,l.LISTINGID;



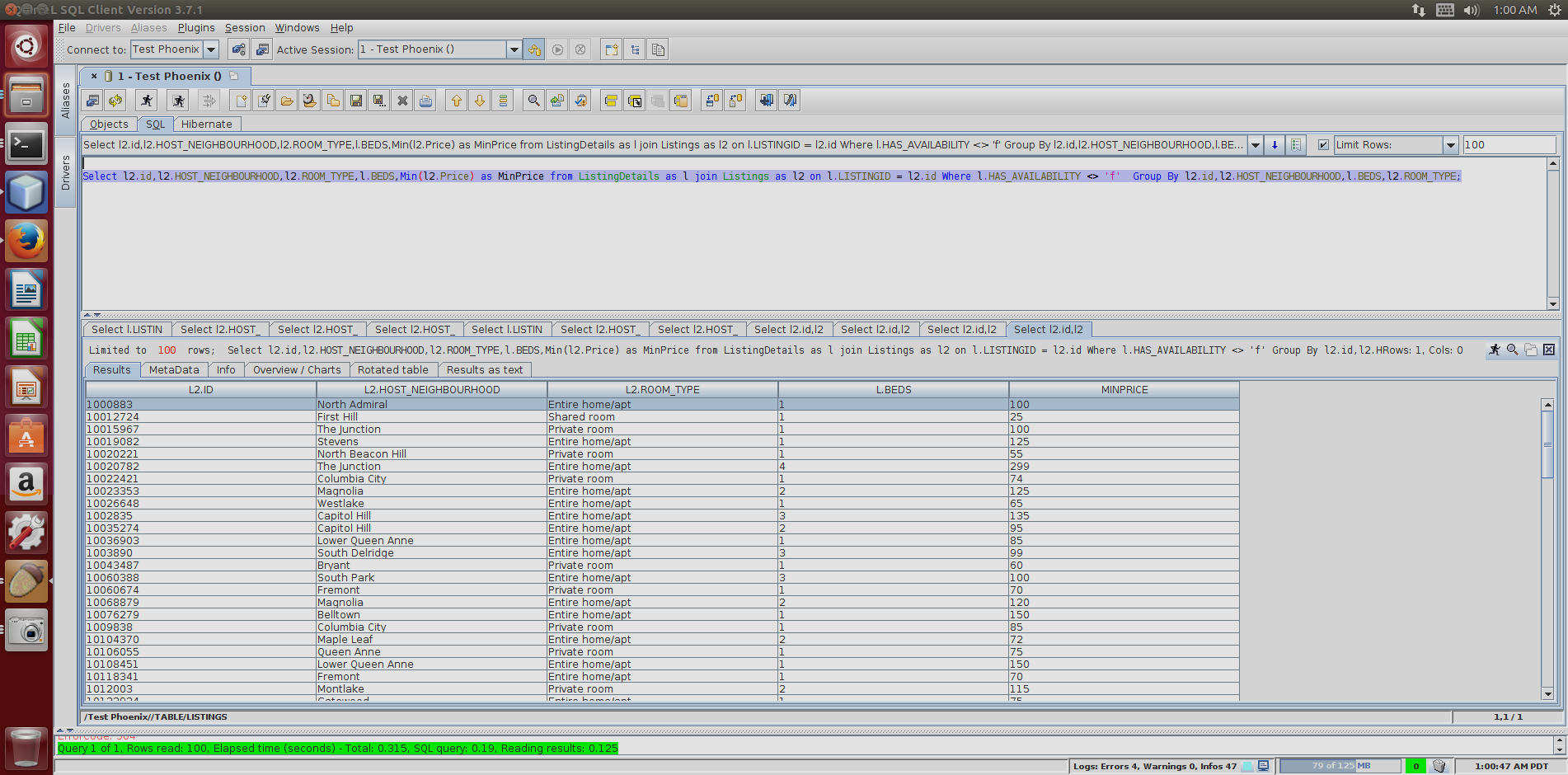
**Query 5:**

Select l2.HOST\_ID,l2.HOST\_NAME,count(l2.id) as counts from ListingDetails as l join Listings as l2 on l.LISTINGID = l2.id Where l2.HOST\_ID <> '0' Group By l2.HOST\_ID,l2.HOST\_NAME Order by counts desc;

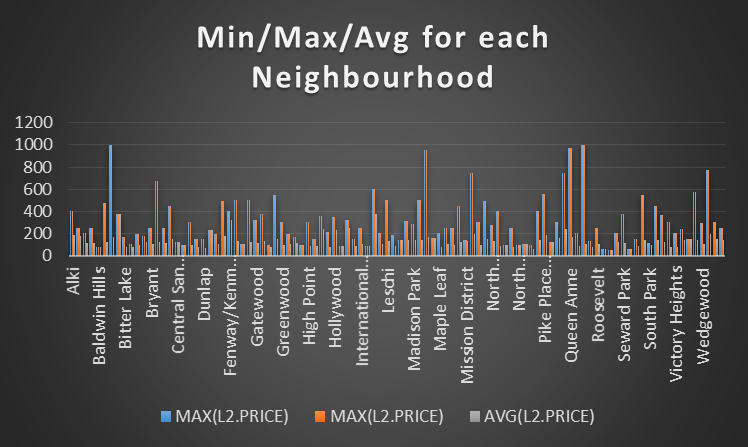


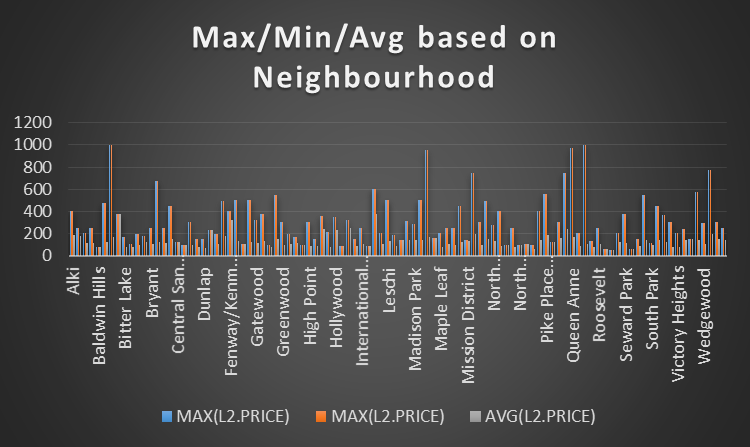
**Query 6:**

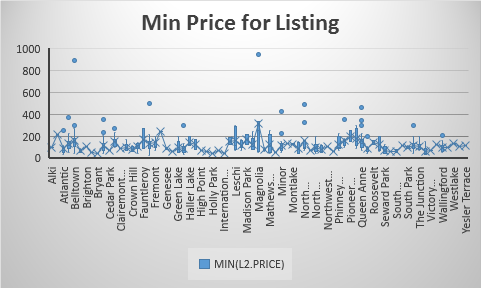
Select l2.id,l2.HOST\_NEIGHBOURHOOD,l2.ROOM\_TYPE,l.BEDS,Min(l2.Price) as MinPrice from ListingDetails as l join Listings as l2 on l.LISTINGID = l2.id Where l.HAS\_AVAILABILITY <> 'f' Group By l2.id,l2.HOST\_NEIGHBOURHOOD,l.BEDS,l2.ROOM\_TYPE;

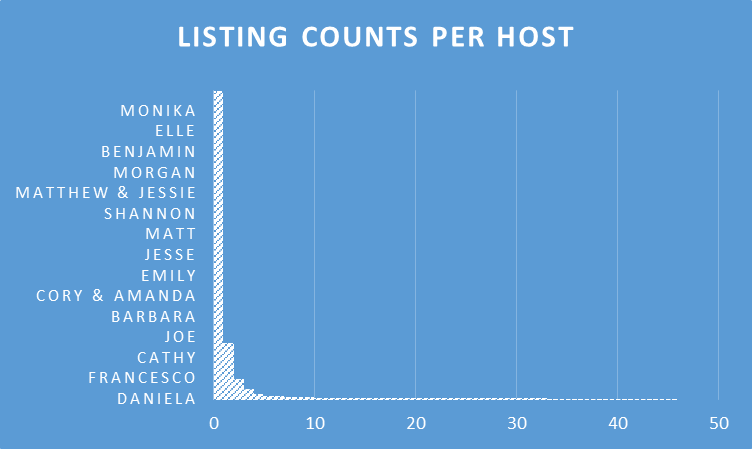


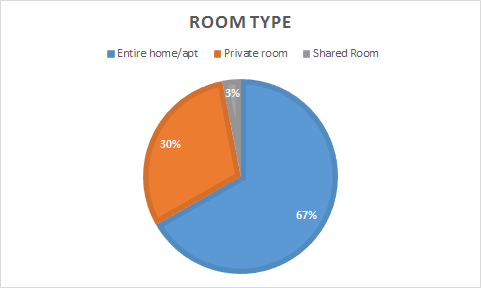
**Visualizations**











**Sentimental Analysis Using Hive**

**Step 1: Create table**

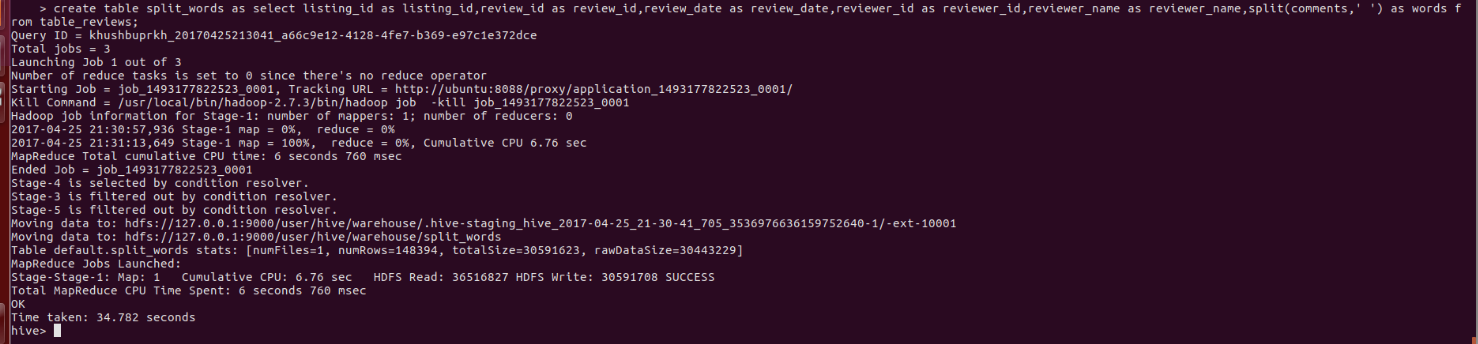
create table table\_reviews(listing\_id BIGINT,review\_id BIGINT,review\_date DATE,reviewer\_id BIGINT,reviewer\_name STRING,comments STRING) row format delimited fields terminated by ‘,’;

**Step 2: Load data**

LOAD DATA INPATH '/home/khushbuprkh/Desktop/reviews\_new.csv' OVERWRITE INTO TABLE table\_reviews;

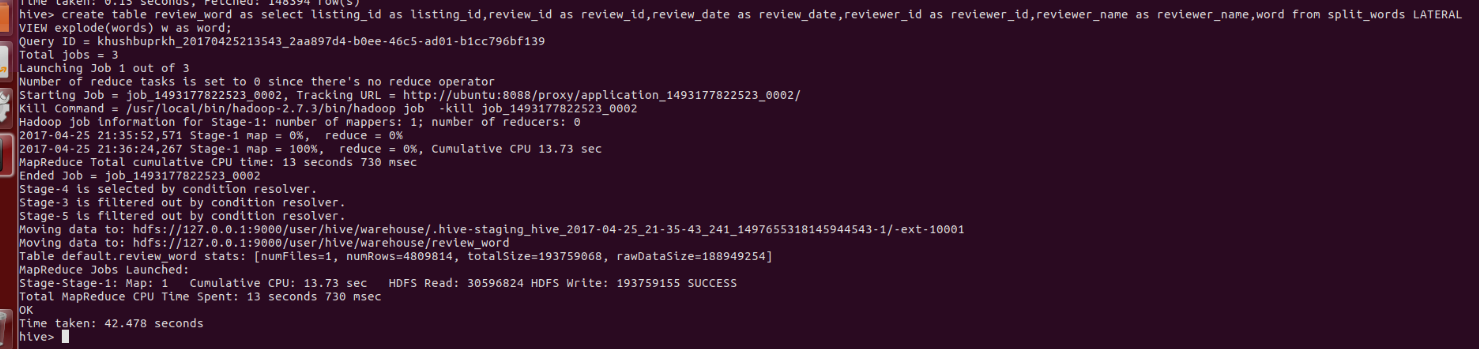
**Step 3: Create a new table with comments data being split and saved in an array form**

create table split\_words as select listing\_id as listing\_id,review\_id as review\_id,review\_date as review\_date,reviewer\_id as reviewer\_id,reviewer\_name as reviewer\_name,split(comments,' ') as words from table\_reviews;



**Step 4: Create a new table with comments Column data being tokenized**

create table review\_word as select listing\_id as listing\_id,review\_id as review\_id,review\_date as review\_date,reviewer\_id as reviewer\_id,reviewer\_name as reviewer\_name,word from split\_words LATERAL VIEW explode(words) w as word;



**Step 5: Create table for dictionary**

create table dictionary(word string,rating int) ROW FORMAT DELIMITED FIELDS TERMINATED BY '\t';



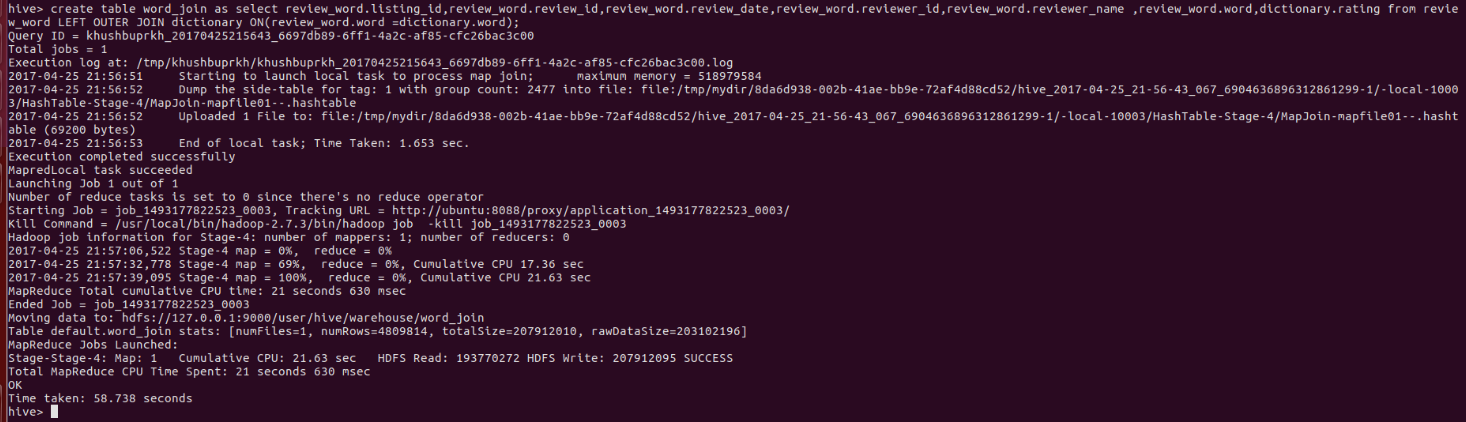
**Step 6: Load data into the dictionary table**

LOAD DATA LOCAL INPATH '/home/khushbuprkh/Desktop/AFINN.txt' into TABLE dictionary;



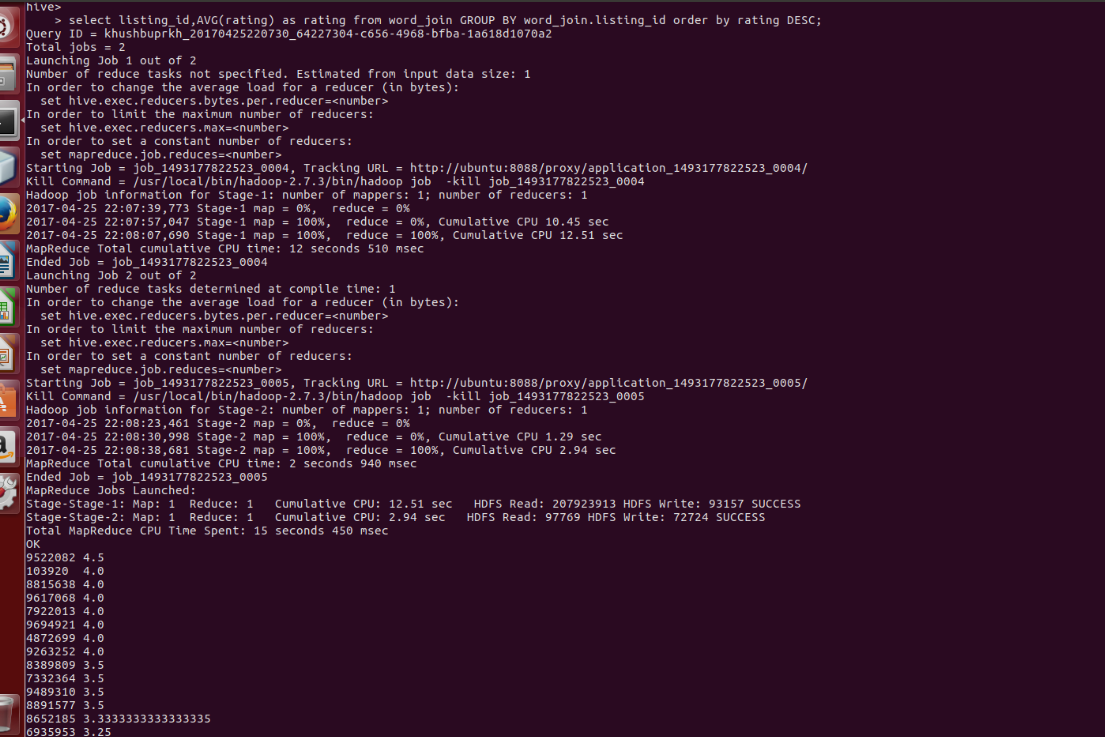
**Step 7: Create a table with values from both the tables**

create table word\_join as select review\_word.listing\_id, review\_word.review\_id, review\_word.review\_date, review\_word.reviewer\_id, review\_word.reviewer\_name, review\_word.word,dictionary.rating from review\_word LEFT OUTER JOIN dictionary ON(review\_word.word =dictionary.word);



**Step 8:Write a select query to get the rating for each review**

select listing\_id,AVG(rating) as rating from word\_join GROUP BY word\_join.review\_id order by rating DESC;



**MapReduce Codes**

**Code 1**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/\*\*

\*

\* @author khushbuprkh

\*/

public class AirBnB {

/\*\*

\* @param args the command line arguments

\*/

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

Text outKey = new Text();

IntWritable outValue = new IntWritable();

@Override

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

StringTokenizer itr = new StringTokenizer(value.toString(), ",");

while (itr.hasMoreTokens()) {

int i = 0;

while (itr.hasMoreTokens()) {

String str = itr.nextToken();

if (i == 5) {

outKey.set(str);

outValue.set(1);

}

i++;

}

context.write(outKey, outValue);

}

}

}

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

@Override

protected void reduce(Text key, Iterable<IntWritable> values,

Context context) throws IOException, InterruptedException {

Integer count = 0;

IntWritable outValue = new IntWritable();

for (IntWritable val : values) {

count += val.get();

}

outValue.set(count);

context.write(key, outValue);

}

}

/\*\*

\* @param args the command line arguments

\*/

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

// TODO code application logic here

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Green Taxi");

job.setJarByClass(AirBnB.class);

job.setMapperClass(MapperClass.class);

job.setCombinerClass(ReducerClass.class);

job.setReducerClass(ReducerClass.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

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

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

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

}

}

**Code 2:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_2;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/\*\*

\*

\* @author khushbuprkh

\*/

public class AirBnB\_2 {

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

Text outKey = new Text();

IntWritable outValue = new IntWritable();

@Override

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

StringTokenizer itr = new StringTokenizer(value.toString(), ",");

while (itr.hasMoreTokens()) {

int i = 0;

while (itr.hasMoreTokens()) {

String str = itr.nextToken();

if (i == 4) {

outKey.set(str);

outValue.set(1);

}

i++;

}

context.write(outKey, outValue);

}

}

}

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

@Override

protected void reduce(Text key, Iterable<IntWritable> values,

Context context) throws IOException, InterruptedException {

Integer count = 0;

IntWritable outValue = new IntWritable();

for (IntWritable val : values) {

count += val.get();

}

outValue.set(count);

context.write(key, outValue);

}

}

/\*\*

\* @param args the command line arguments

\*/

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

// TODO code application logic here

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Green Taxi");

job.setJarByClass(AirBnB\_2.class);

job.setMapperClass(MapperClass.class);

job.setCombinerClass(ReducerClass.class);

job.setReducerClass(ReducerClass.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(IntWritable.class);

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

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

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

}

}

**Code 3:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_3;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Partitioner;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/\*\*

\*

\* @author khushbuprkh

\*/

public class AirBnB\_3 {

public static class MapperClass extends Mapper<LongWritable, Text, Text, Text> {

private Text roomType = new Text();

@Override

protected void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString(), ",");

while (itr.hasMoreTokens()) {

int i = 0;

while (itr.hasMoreTokens()) {

String str = itr.nextToken();

if (i == 4) {

roomType = new Text(str);

}

i++;

}

context.write(roomType, value);

}

}

}

public static class PartitionerClass extends Partitioner<Text, Text> {

@Override

public int getPartition(Text key, Text value, int i) {

String k = key.toString();

if (k.equals("Entire home/apt")) {

return 0;

} else if (k.equals("Private room")) {

return 1;

} else if (k.equals("Shared room")) {

return 2;

} else {

return 3;

}

}

}

public static class ReducerClass extends Reducer<Text, Text, Text, NullWritable> {

@Override

protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

for (Text val : values) {

context.write(val, NullWritable.get());

}

}

}

/\*\*

\* @param args the command line arguments

\*/

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

// TODO code application logic here

Configuration config = new Configuration();

Job job = Job.getInstance(config, "Partitioning Pattern");

job.setJarByClass(AirBnB\_3.class);

job.setMapperClass(MapperClass.class);

job.setPartitionerClass(PartitionerClass.class);

job.setNumReduceTasks(4);

job.setReducerClass(ReducerClass.class);

job.setInputFormatClass(TextInputFormat.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setOutputKeyClass(NullWritable.class);

job.setOutputValueClass(Text.class);

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

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

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

}

}

**Code 4:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_4;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/\*\*

\*

\* @author khushbuprkh

\*/

public class AirBnb\_4 {

public static class MapperClass extends Mapper<Object, Text, Text, FloatWritable> {

Text outKey = new Text();

FloatWritable outValue = new FloatWritable();

Float price = 0.F;

@Override

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

StringTokenizer itr = new StringTokenizer(value.toString(), ",");

while (itr.hasMoreTokens()) {

int i = 0;

while (itr.hasMoreTokens()) {

String str = itr.nextToken();

if (i == 1) {

outKey.set(str);

} else if (i == 2) {

if(!str.equals("price")){

price = Float.parseFloat(str);

outValue.set(price);

}

}

i++;

}

context.write(outKey, outValue);

}

}

}

public static class ReducerClass extends Reducer<Text, FloatWritable, Text, FloatWritable> {

Float avg=0.F;

Float sum = 0.F;

Long count = 0L;

@Override

protected void reduce(Text key, Iterable<FloatWritable> values, Context context) throws IOException, InterruptedException {

for (FloatWritable val : values) {

sum += val.get();

count++;

}

avg = (Float)sum/count;

context.write(key, new FloatWritable(avg));

}

}

/\*\*

\* @param args the command line arguments

\*/

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

// TODO code application logic here

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Green Taxi");

job.setJarByClass(AirBnb\_4.class);

job.setMapperClass(MapperClass.class);

job.setCombinerClass(ReducerClass.class);

job.setReducerClass(ReducerClass.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(FloatWritable.class);

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

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

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

}

}

**Code 5:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_5;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

/\*\*

\*

\* @author khushbuprkh

\*/

public class AirBnB\_5 {

public static class InvertedMapper extends Mapper<Object, Text, Text, Text> {

Text outKey = new Text();

Text outValue = new Text();

Float price = 0.F;

@Override

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString(), ",");

while (itr.hasMoreTokens()) {

int i = 0;

while (itr.hasMoreTokens()) {

String str = itr.nextToken();

if(i==0){

if(!str.equals("id")){

outValue.set(str);

}

}

if (i == 2) {

if(!str.equals("price")){

price = Float.parseFloat(str);

if (price >= 0.00 && price <= 50.00) {

outKey.set("0-50");

}else if(price >= 51.00 && price <= 100.00) {

outKey.set("51-100");

}else if(price >= 101.00 && price <= 150.00) {

outKey.set("101-150");

}else if(price >= 151.00 && price <= 200.00) {

outKey.set("151-200");

}else if(price >= 201.00 && price <= 250.00) {

outKey.set("201-250");

}else if(price >= 251.00 && price <= 300.00) {

outKey.set("251-300");

}else if(price >= 301.00 && price <= 350.00) {

outKey.set("301-350");

}else if(price >= 351.00 && price <= 400.00) {

outKey.set("351-400");

}else if(price >= 401.00 && price <= 450.00) {

outKey.set("401-450");

}else if(price >= 451.00 && price <= 500.00) {

outKey.set("451-500");

}else if(price >= 501.00){

outKey.set("Greater than 500");

}

}

}

i++;

}

context.write(outKey, outValue);

}

}

}

public static class InvertedReducer extends Reducer<Text, Text, Text, Text> {

private Text result = new Text();

@Override

public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

StringBuilder sb = new StringBuilder();

boolean first = true;

for (Text id : values) {

if (first) {

first = false;

} else {

sb.append(" ");

}

sb.append(id.toString());

}

result.set(sb.toString());

context.write(key, result);

}

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

try {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Inverted Index");

job.setJarByClass(AirBnB\_5.class);

job.setMapperClass(InvertedMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setReducerClass(InvertedReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

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

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

job.waitForCompletion(true);

} catch (IOException | InterruptedException | ClassNotFoundException ex) {

System.out.println("Error in Main" + ex.getMessage());

}

}

}

**Code 6:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_6;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

/\*\*

\*

\* @author khushbuprkh

\*/

public class AirBnB\_6 {

public static class InvertedMapper extends Mapper<Object, Text, Text, Text> {

Text outKey = new Text();

Text outValue = new Text();

Integer bedroom = 0;

@Override

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString(), ",");

while (itr.hasMoreTokens()) {

int i = 0;

while (itr.hasMoreTokens()) {

String str = itr.nextToken();

if (i == 0) {

if (!str.equals("id")) {

outValue.set(str);

}

}

if (i == 26) {

if (!str.equals("bedrooms")) {

bedroom = Integer.parseInt(str);

outKey.set(bedroom + "-Bedrooms");

}

}

i++;

}

context.write(outKey, outValue);

}

}

}

public static class InvertedReducer extends Reducer<Text, Text, Text, Text> {

private Text result = new Text();

@Override

public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

StringBuilder sb = new StringBuilder();

boolean first = true;

for (Text id : values) {

if (first) {

first = false;

} else {

sb.append(" ");

}

sb.append(id.toString());

}

result.set(sb.toString());

context.write(key, result);

}

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

try {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Inverted Index");

job.setJarByClass(AirBnB\_6.class);

job.setMapperClass(InvertedMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setReducerClass(InvertedReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

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

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

job.waitForCompletion(true);

} catch (IOException | InterruptedException | ClassNotFoundException ex) {

System.out.println("Error in Main" + ex.getMessage());

}

}

}

**Code 7:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_7;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

/\*\*

\*

\* @author khushbuprkh

\*/

public class AirBnB\_7 {

public static class InvertedMapper extends Mapper<Object, Text, Text, Text> {

Text outKey = new Text();

Text outValue = new Text();

Integer accommodates = 0;

@Override

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString(), ",");

while (itr.hasMoreTokens()) {

int i = 0;

while (itr.hasMoreTokens()) {

String str = itr.nextToken();

if (i == 0) {

if (!str.equals("id")) {

outValue.set(str);

}

}

if (i == 24) {

if (!str.equals("accommodates")) {

accommodates = Integer.parseInt(str);

outKey.set(accommodates + "- People can accommodated");

}

}

i++;

}

context.write(outKey, outValue);

}

}

}

public static class InvertedReducer extends Reducer<Text, Text, Text, Text> {

private Text result = new Text();

@Override

public void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

StringBuilder sb = new StringBuilder();

boolean first = true;

for (Text id : values) {

if (first) {

first = false;

} else {

sb.append(" ");

}

sb.append(id.toString());

}

result.set(sb.toString());

context.write(key, result);

}

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

try {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Inverted Index");

job.setJarByClass(AirBnB\_7.class);

job.setMapperClass(InvertedMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setReducerClass(InvertedReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

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

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

job.waitForCompletion(true);

} catch (IOException | InterruptedException | ClassNotFoundException ex) {

System.out.println("Error in Main" + ex.getMessage());

}

}

}

**Code 8:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_8;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

/\*\*

\*

\* @author khushbuprkh

\*/

public class AirBnB\_8 {

public static class InvertedMapper extends Mapper<Object, Text, Text, Text> {

Text outKey = new Text();

Text outValue = new Text();

String host\_neighbourhood = null;

@Override

public void map(Object key, Text value,Context context) throws IOException, InterruptedException {

StringTokenizer itr = new StringTokenizer(value.toString(), ",");

while (itr.hasMoreTokens()) {

int i = 0;

while (itr.hasMoreTokens()) {

String str = itr.nextToken();

if (i == 0) {

if (!str.equals("id")) {

outValue.set(str);

}

}

if (i == 1) {

if (!str.equals("host\_neighbourhood")) {

host\_neighbourhood = str;

outKey.set(host\_neighbourhood);

}

}

i++;

}

context.write(outKey, outValue);

}

}

}

public static class InvertedReducer extends Reducer<Text, Text, Text, Text> {

private Text result = new Text();

@Override

public void reduce(Text key, Iterable<Text> values,Context context) throws IOException, InterruptedException {

StringBuilder sb = new StringBuilder();

boolean first = true;

for (Text id : values) {

if (first) {

first = false;

} else {

sb.append(" ");

}

sb.append(id.toString());

}

result.set(sb.toString());

context.write(key, result);

}

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

try {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Inverted Index");

job.setJarByClass(AirBnB\_8.class);

job.setMapperClass(InvertedMapper.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setReducerClass(InvertedReducer.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

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

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

job.waitForCompletion(true);

} catch (IOException | InterruptedException | ClassNotFoundException ex) {

System.out.println("Error in Main" + ex.getMessage());

}

}

}

**Code 9:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_10;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/\*\*

\*

\* @author khushbuprkh

\*/

public class AirBnB\_10 {

public static class MapperClass extends Mapper<Object, Text, Text, CustomWritableClass> {

@Override

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

Text host\_neighbourhood = new Text();

Integer min\_Bedrooms = 0;

Integer max\_Bedrooms = 0;

Float average\_Price = 0.F;

CustomWritableClass cw = new CustomWritableClass();

StringTokenizer itr = new StringTokenizer(value.toString(), ",");

while (itr.hasMoreTokens()) {

int i = 0;

while (itr.hasMoreTokens()) {

String str = itr.nextToken();

if (i == 1) {

if (!str.equals("host\_neighbourhood")) {

host\_neighbourhood.set(str);

}

}

if (i == 26) {

if (!str.equals("bedrooms")) {

min\_Bedrooms = Integer.parseInt(str);

max\_Bedrooms = Integer.parseInt(str);

}

}

if (i == 2) {

if (!str.equals("price")) {

average\_Price = Float.parseFloat(str);

}

}

i++;

}

}

cw.setMax\_Bedrooms(min\_Bedrooms);

cw.setMin\_Bedrooms(max\_Bedrooms);

cw.setAverage\_Price(average\_Price);

context.write(host\_neighbourhood, cw);

}

}

public static class ReducerClass extends Reducer<Text, CustomWritableClass, Text, CustomWritableClass> {

@Override

public void reduce(Text key, Iterable<CustomWritableClass> values, Context context) throws IOException, InterruptedException {

Float averagePrice = 0F;

Long count = 0L;

Integer min\_Bedrooms = 0;

Integer max\_Bedrooms = 0;

Integer max = Integer.MIN\_VALUE;

Integer min = Integer.MAX\_VALUE;

CustomWritableClass cw = new CustomWritableClass();

for (CustomWritableClass val : values) {

if (val.getMin\_Bedrooms() > max) {

max=val.getMin\_Bedrooms();

max\_Bedrooms = val.getMin\_Bedrooms();

}

if (val.getMin\_Bedrooms() < min) {

min=val.getMin\_Bedrooms();

min\_Bedrooms = val.getMin\_Bedrooms();

}

averagePrice += val.getAverage\_Price();

count++;

}

averagePrice = (Float) averagePrice / count;

cw.setMin\_Bedrooms(min\_Bedrooms);

cw.setMax\_Bedrooms(max\_Bedrooms);

cw.setAverage\_Price(averagePrice);

context.write(key, cw);

}

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

try {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "MinMax");

job.setJarByClass(AirBnB\_10.class);

job.setMapperClass(MapperClass.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(CustomWritableClass.class);

job.setReducerClass(ReducerClass.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(CustomWritableClass.class);

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

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

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

} catch (IOException | InterruptedException | ClassNotFoundException ex) {

System.out.println("Erorr Message" + ex.getMessage());

}

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_10;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableUtils;

/\*\*

\*

\* @author khushbuprkh

\*/

public class CustomWritableClass implements Writable {

private int min\_Bedrooms;

private int max\_Bedrooms;

private float average\_Price;

public CustomWritableClass() {

}

public int getMin\_Bedrooms() {

return min\_Bedrooms;

}

public void setMin\_Bedrooms(int min\_Bedrooms) {

this.min\_Bedrooms = min\_Bedrooms;

}

public int getMax\_Bedrooms() {

return max\_Bedrooms;

}

public void setMax\_Bedrooms(int max\_Bedrooms) {

this.max\_Bedrooms = max\_Bedrooms;

}

public float getAverage\_Price() {

return average\_Price;

}

public void setAverage\_Price(float average\_Price) {

this.average\_Price = average\_Price;

}

@Override

public void write(DataOutput d) throws IOException {

d.writeInt(min\_Bedrooms);

d.writeInt(max\_Bedrooms);

d.writeFloat(average\_Price);

}

@Override

public void readFields(DataInput di) throws IOException {

min\_Bedrooms = di.readInt();

max\_Bedrooms = di.readInt();

average\_Price = di.readFloat();

}

public String toString() {

return (new StringBuilder().append(min\_Bedrooms).append("\t").append(max\_Bedrooms).append("\t").append(average\_Price)).toString();

}

}

**Code 10:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_11;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Partitioner;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/\*\*

\*

\* @author khushbuprkh

\*/

public class AirBnB\_11 {

public static class MapperClass extends Mapper<Object, Text, Text, CustomWritableClass> {

@Override

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

Text host\_neighbourhood = new Text();

Integer bedroom = 0;

Float min\_Price = 0F;

Float max\_Price = 0F;

CompositeKeyWritable ckw = new CompositeKeyWritable();

CustomWritableClass cw = new CustomWritableClass();

StringTokenizer itr = new StringTokenizer(value.toString(), ",");

while (itr.hasMoreTokens()) {

int i = 0;

while (itr.hasMoreTokens()) {

String str = itr.nextToken();

if (i == 1) {

if (!str.equals("host\_neighbourhood") && !str.equals(" ") && !str.equals("null") && !str.equals(null)) {

ckw.setHost\_neighbourhood(str);

host\_neighbourhood.set(str);

}

}

if (i == 2) {

if (!str.equals("price") && !str.equals(" ") && !str.equals("null") && !str.equals(null) && str != "0") {

min\_Price=Float.parseFloat(str);

if(min\_Price != 0.0){

cw.setMin\_Price(Float.parseFloat(str));

cw.setMax\_Price(Float.parseFloat(str));

}

}

}

if (i == 26) {

if (!str.equals("bedrooms") && !str.equals("null") ) {

ckw.setBedroom(str);

}

}

i++;

}

}

context.write(host\_neighbourhood, cw);

}

}

public static class ReducerClass extends Reducer<Text, CustomWritableClass, Text, CustomWritableClass> {

@Override

public void reduce(Text key, Iterable<CustomWritableClass> values, Context context) throws IOException, InterruptedException {

Float min\_Price = 0F;

Float max\_Price = 0F;

Float max = Float.MIN\_VALUE;

Float min = Float.MAX\_VALUE;

CustomWritableClass cw = new CustomWritableClass();

for (CustomWritableClass val : values) {

if (val.getMin\_Price() > max) {

max = val.getMin\_Price();

max\_Price = val.getMin\_Price();

}

if (val.getMax\_Price() < min) {

min=val.getMin\_Price();

min\_Price = val.getMin\_Price();

}

}

cw.setMin\_Price(min\_Price);

cw.setMax\_Price(max\_Price);

context.write(key, cw);

}

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

try {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "SecondarySort");

job.setJarByClass(AirBnB\_11.class);

job.setMapperClass(MapperClass.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(CustomWritableClass.class);

// job.setGroupingComparatorClass(GroupComparator.class);

// job.setPartitionerClass(PartitionerClass.class);

job.setReducerClass(ReducerClass.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(CustomWritableClass.class);

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

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

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

} catch (IOException | InterruptedException | ClassNotFoundException ex) {

System.out.println("Erorr Message"+ ex.getMessage());

}

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_11;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.io.WritableUtils;

/\*\*

\*

\* @author khushbu

\*/

public class CompositeKeyWritable implements Writable, WritableComparable<CompositeKeyWritable> {

private String host\_neighbourhood;

private String bedroom;

public CompositeKeyWritable() {

}

public CompositeKeyWritable(String host\_neighbourhood, String bedroom) {

this.host\_neighbourhood = host\_neighbourhood;

this.bedroom = bedroom;

}

public String getHost\_neighbourhood() {

return host\_neighbourhood;

}

public void setHost\_neighbourhood(String host\_neighbourhood) {

this.host\_neighbourhood = host\_neighbourhood;

}

public String getBedroom() {

return bedroom;

}

public void setBedroom(String bedroom) {

this.bedroom = bedroom;

}

@Override

public void write(DataOutput d) throws IOException {

WritableUtils.writeString(d, host\_neighbourhood);

WritableUtils.writeString(d, bedroom);

}

@Override

public void readFields(DataInput di) throws IOException {

host\_neighbourhood = WritableUtils.readString(di);

bedroom = WritableUtils.readString(di);

}

public String toString() {

return (new StringBuilder().append(host\_neighbourhood).append("\t").append(bedroom)).toString();

}

@Override

public int compareTo(CompositeKeyWritable o) {

int result = bedroom.compareTo(o.bedroom);

if (result == 0) {

result = host\_neighbourhood.compareTo(o.host\_neighbourhood);

}

return result;

}

}

**Code 11:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_11;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableUtils;

/\*\*

\*

\* @author khushbuprkh

\*/

public class CustomWritableClass implements Writable {

private float min\_Price;

private float max\_Price;

public CustomWritableClass() {

}

public float getMin\_Price() {

return min\_Price;

}

public void setMin\_Price(float min\_Price) {

this.min\_Price = min\_Price;

}

public float getMax\_Price() {

return max\_Price;

}

public void setMax\_Price(float max\_Price) {

this.max\_Price = max\_Price;

}

@Override

public void write(DataOutput d) throws IOException {

d.writeFloat(min\_Price);

d.writeFloat(max\_Price);

}

@Override

public void readFields(DataInput di) throws IOException {

min\_Price = di.readFloat();

max\_Price = di.readFloat();

}

public String toString() {

return (new StringBuilder().append(min\_Price).append("\t").append(max\_Price)).toString();

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_11;

import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.io.WritableComparator;

/\*\*

\*

\* @author pooja

\*/

public class GroupComparator extends WritableComparator {

protected GroupComparator()

{

super(CompositeKeyWritable.class, true);

}

@Override

public int compare(WritableComparable w1, WritableComparable w2)

{

CompositeKeyWritable cw1 = (CompositeKeyWritable) w1;

CompositeKeyWritable cw2 = (CompositeKeyWritable) w2;

return (cw1.getBedroom().compareTo(cw2.getBedroom()));

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_11;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.mapreduce.Partitioner;

/\*\*

\*

\* @author pooja

\*/

public class PartitionerClass extends Partitioner<CompositeKeyWritable, NullWritable>{

@Override

public int getPartition(CompositeKeyWritable key, NullWritable value, int numOfPartitions) {

return (key.getBedroom().hashCode() % numOfPartitions);

}

}

**Code 12:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_13;

import java.io.IOException;

import java.io.StringReader;

import java.io.StringWriter;

import java.util.ArrayList;

import java.util.List;

import java.util.StringTokenizer;

import java.util.logging.Level;

import java.util.logging.Logger;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;

import javax.xml.transform.OutputKeys;

import javax.xml.transform.Transformer;

import javax.xml.transform.TransformerConfigurationException;

import javax.xml.transform.TransformerException;

import javax.xml.transform.TransformerFactory;

import javax.xml.transform.dom.DOMSource;

import javax.xml.transform.stream.StreamResult;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.MultipleInputs;

import org.apache.hadoop.mapreduce.lib.input.TextInputFormat;

import org.apache.hadoop.mapreduce.lib.output.TextOutputFormat;

import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.xml.sax.InputSource;

import org.xml.sax.SAXException;

/\*\*

\*

\* @author khushbuprkh

\*/

public class AirBnB\_13 {

public static class ListingMapper extends Mapper<Object, Text, Text, Text> {

private Text outputKey = new Text();

private Text outputValue = new Text();

@Override

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

StringTokenizer itr = new StringTokenizer(value.toString(), ",");

while (itr.hasMoreTokens()) {

int i = 0;

while (itr.hasMoreTokens()) {

String str = itr.nextToken();

if (i == 0) {

if (!str.equals("id")) {

outputKey.set(str);

}

}

i++;

}

}

outputValue.set("L" + value.toString());

context.write(outputKey, outputValue);

}

}

public static class ReviewMapper extends Mapper<Object, Text, Text, Text> {

private Text outputKey = new Text();

private Text outputValue = new Text();

@Override

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

StringTokenizer itr = new StringTokenizer(value.toString(), ",");

while (itr.hasMoreTokens()) {

int i = 0;

while (itr.hasMoreTokens()) {

String str = itr.nextToken();

if (i == 0) {

if (!str.equals("listingId")) {

outputKey.set(str);

}

}

i++;

}

}

outputValue.set("R" + value.toString());

context.write(outputKey, outputValue);

}

}

public static class ReducerClass extends Reducer<Text, Text, Text, NullWritable> {

private ArrayList<String> review = new ArrayList<String>();

private DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();

private String listing = null;

@Override

protected void reduce(Text key, Iterable<Text> values, Context context) throws IOException, InterruptedException {

listing = null;

review.clear();

for (Text val : values) {

if (val.charAt(0) == 'L') {

listing = val.toString().substring(1, val.toString().length()).trim();

} else {

review.add(val.toString().substring(1, val.toString().length()).trim());

}

}

if (listing != null) {

try {

String movieTitleWithTags = nestElements(listing, review);

context.write(new Text(movieTitleWithTags), NullWritable.get());

} catch (TransformerException ex) {

Logger.getLogger(AirBnB\_13.class.getName()).log(Level.SEVERE, null, ex);

} catch (ParserConfigurationException ex) {

Logger.getLogger(AirBnB\_13.class.getName()).log(Level.SEVERE, null, ex);

} catch (SAXException ex) {

Logger.getLogger(AirBnB\_13.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

private String nestElements(String listing, List<String> reviews) throws TransformerException, ParserConfigurationException, SAXException, IOException {

// Create the new document to build the XML

DocumentBuilder bldr = dbf.newDocumentBuilder();

Document doc = bldr.newDocument();

// Copy parent node to document

String listingEl = null;

StringTokenizer itr = new StringTokenizer(listing.toString(), ",");

while (itr.hasMoreTokens()) {

int i = 0;

while (itr.hasMoreTokens()) {

String str = itr.nextToken();

if (i == 0) {

if (!str.equals("id")) {

listingEl=str;

}

}

i++;

}

}

Element toAddlistingEl = doc.createElement("Listing");

// Copy the attributes of the original post element to the new one

copyAttributesToElement("Listing", listingEl, toAddlistingEl);

// For each comment, copy it to the "post" node

for (String tagsXml : reviews) {

String reviewsEl = null;

StringTokenizer iitr = new StringTokenizer(tagsXml.toString(), ",");

while (iitr.hasMoreTokens()) {

int i = 0;

while (iitr.hasMoreTokens()) {

String str = iitr.nextToken();

if (i == 5) {

if (!str.equals("comments")) {

reviewsEl=str;

}

}

i++;

}

}

Element toAddreviewsEl = doc.createElement("reviews");

// Copy the attributes of the original comment element to

// the new one

copyAttributesToElement("reviews", reviewsEl, toAddreviewsEl);

// Add the copied comment to the post element

toAddlistingEl.appendChild(toAddreviewsEl);

}

// Add the post element to the document

doc.appendChild(toAddlistingEl);

// Transform the document into a String of XML and return

return transformDocumentToString(doc);

}

private Element getXmlElementFromString(String xml) throws ParserConfigurationException, SAXException, IOException {

// Create a new document builder

DocumentBuilder bldr = dbf.newDocumentBuilder();

return bldr.parse(new InputSource(new StringReader(xml)))

.getDocumentElement();

}

private void copyAttributesToElement(String key, String value,

Element element) {

// For each attribute, copy it to the element

element.setAttribute(key, value);

}

private String transformDocumentToString(Document doc) throws TransformerConfigurationException, TransformerException {

TransformerFactory tf = TransformerFactory.newInstance();

Transformer transformer = tf.newTransformer();

transformer.setOutputProperty(OutputKeys.OMIT\_XML\_DECLARATION,

"yes");

StringWriter writer = new StringWriter();

transformer.transform(new DOMSource(doc), new StreamResult(

writer));

// Replace all new line characters with an empty string to have

// one record per line.

return writer.getBuffer().toString().replaceAll("\n|\r", "");

}

}

/\*\*

\* @param args the command line arguments

\*/

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

// TODO code application logic here

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "PostCommentHierarchy");

job.setJarByClass(AirBnB\_13.class);

MultipleInputs.addInputPath(job, new Path(args[0]),

TextInputFormat.class, ListingMapper.class);

MultipleInputs.addInputPath(job, new Path(args[1]),

TextInputFormat.class, ReviewMapper.class);

job.setReducerClass(ReducerClass.class);

job.setOutputFormatClass(TextOutputFormat.class);

TextOutputFormat.setOutputPath(job, new Path(args[2]));

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

System.exit(job.waitForCompletion(true) ? 0 : 2);

}

}

**Code 13:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_14;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/\*\*

\*

\* @author khushbuprkh

\*/

public class AirBnB\_14 {

public static class MapperClass extends Mapper<Object, Text, Text, CustomWritableClass> {

@Override

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

Text listing\_Id = new Text();

Integer minimum\_nights = 0;

Integer number\_of\_reviews = 0;

Float price = 0.F;

CustomWritableClass cw = new CustomWritableClass();

StringTokenizer itr = new StringTokenizer(value.toString(), ",");

while (itr.hasMoreTokens()) {

int i = 0;

while (itr.hasMoreTokens()) {

String str = itr.nextToken();

if (i == 0) {

if (!str.equals("id")) {

listing\_Id.set(str);

}

}

if (i == 37) {

if (!str.equals("minimum\_nights")) {

minimum\_nights = Integer.parseInt(str);

}

}

if (i == 45) {

if (!str.equals("number\_of\_reviews")) {

number\_of\_reviews = Integer.parseInt(str);

}

}

if (i == 2) {

if (!str.equals("price")) {

price = Float.parseFloat(str);

}

}

i++;

}

}

cw.setMinimum\_nights(minimum\_nights);

cw.setNumber\_of\_reviews(number\_of\_reviews);

cw.setPrice(price);

context.write(listing\_Id, cw);

}

}

public static class ReducerClass extends Reducer<Text, CustomWritableClass, Text, CustomWritableOutputClass> {

@Override

public void reduce(Text key, Iterable<CustomWritableClass> values, Context context) throws IOException, InterruptedException {

Float costBasedOnMinDays = 0F;

Float incomePerYear = 0F;

Float incomePerMonth = 0F;

Integer nightsPerYear = 0;

Long count = 0L;

Integer minimum\_nights = 0;

Float occupancyRate = 0F;

CustomWritableOutputClass cow = new CustomWritableOutputClass();

for (CustomWritableClass val : values) {

costBasedOnMinDays = val.getMinimum\_nights() \* val.getPrice();

incomePerYear = val.getNumber\_of\_reviews() \* costBasedOnMinDays;

incomePerMonth = incomePerYear / 12;

nightsPerYear = val.getNumber\_of\_reviews() \* val.getMinimum\_nights();

occupancyRate = (nightsPerYear / 365.0F) \* 100;

}

cow.setIncomePerMonth(incomePerMonth);

cow.setNightsPerYear(nightsPerYear);

cow.setOccupancyRate(occupancyRate);

context.write(key, cow);

}

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

try {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Occupancy Rate");

job.setJarByClass(AirBnB\_14.class);

job.setMapperClass(MapperClass.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(CustomWritableClass.class);

job.setReducerClass(ReducerClass.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(CustomWritableOutputClass.class);

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

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

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

} catch (IOException | InterruptedException | ClassNotFoundException ex) {

System.out.println("Erorr Message" + ex.getMessage());

}

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_14;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableUtils;

/\*\*

\*

\* @author khushbuprkh

\*/

public class CustomWritableClass implements Writable {

private int minimum\_nights;

private int number\_of\_reviews;

private float price;

public CustomWritableClass() {

}

public int getMinimum\_nights() {

return minimum\_nights;

}

public void setMinimum\_nights(int minimum\_nights) {

this.minimum\_nights = minimum\_nights;

}

public int getNumber\_of\_reviews() {

return number\_of\_reviews;

}

public void setNumber\_of\_reviews(int number\_of\_reviews) {

this.number\_of\_reviews = number\_of\_reviews;

}

public float getPrice() {

return price;

}

public void setPrice(float price) {

this.price = price;

}

@Override

public void write(DataOutput d) throws IOException {

d.writeInt(minimum\_nights);

d.writeInt(number\_of\_reviews);

d.writeFloat(price);

}

@Override

public void readFields(DataInput di) throws IOException {

minimum\_nights = di.readInt();

number\_of\_reviews = di.readInt();

price = di.readFloat();

}

public String toString() {

return (new StringBuilder().append(minimum\_nights).append("\t").append(number\_of\_reviews).append("\t").append(price)).toString();

}

}

**Code 14:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_14;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

/\*\*

\*

\* @author khushbuprkh

\*/

public class CustomWritableOutputClass implements Writable {

private float incomePerMonth;

private int nightsPerYear;

private float occupancyRate;

public CustomWritableOutputClass() {

}

public float getIncomePerMonth() {

return incomePerMonth;

}

public void setIncomePerMonth(float incomePerMonth) {

this.incomePerMonth = incomePerMonth;

}

public int getNightsPerYear() {

return nightsPerYear;

}

public void setNightsPerYear(int nightsPerYear) {

this.nightsPerYear = nightsPerYear;

}

public float getOccupancyRate() {

return occupancyRate;

}

public void setOccupancyRate(float occupancyRate) {

this.occupancyRate = occupancyRate;

}

@Override

public void write(DataOutput d) throws IOException {

d.writeFloat(incomePerMonth);

d.writeInt(nightsPerYear);

d.writeFloat(occupancyRate);

}

@Override

public void readFields(DataInput di) throws IOException {

incomePerMonth = di.readFloat();

nightsPerYear = di.readInt();

occupancyRate = di.readFloat();

}

public String toString() {

return (new StringBuilder().append(incomePerMonth).append("\t").append(nightsPerYear).append("\t").append(occupancyRate)).toString();

}

}

**Code 15:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_15;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.IntWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapred.lib.IdentityReducer;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/\*\*

\*

\* @author khushbuprkh

\*/

public class AirBnB\_15 {

public static class MapperClass extends Mapper<Object, Text, CompositeKeyWritable, NullWritable> {

@Override

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

Text listing\_Id = new Text();

Integer review\_scores\_rating = 0;

Integer review\_scores\_accuracy = 0;

Integer review\_scores\_cleanliness = 0;

Integer review\_scores\_checkin = 0;

Integer review\_scores\_communication = 0;

Integer review\_scores\_location = 0;

Integer review\_scores\_value = 0;

Integer totalRating = 0;

Float percentage = 0F;

CompositeKeyWritable ckw = new CompositeKeyWritable();

StringTokenizer itr = new StringTokenizer(value.toString(), ",");

while (itr.hasMoreTokens()) {

int i = 0;

while (itr.hasMoreTokens()) {

String str = itr.nextToken();

if (i == 0) {

if (!str.equals("id")) {

listing\_Id.set(str);

}

}

if (i == 48) {

if (!str.equals("review\_scores\_rating")) {

review\_scores\_rating = Integer.parseInt(str);

}

}

if (i == 49) {

if (!str.equals("review\_scores\_accuracy")) {

review\_scores\_accuracy = Integer.parseInt(str);

}

}

if (i == 50) {

if (!str.equals("review\_scores\_cleanliness")) {

review\_scores\_cleanliness = Integer.parseInt(str);

}

}

if (i == 51) {

if (!str.equals("review\_scores\_checkin")) {

review\_scores\_checkin = Integer.parseInt(str);

}

}

if (i == 52) {

if (!str.equals("review\_scores\_communication")) {

review\_scores\_communication = Integer.parseInt(str);

}

}

if (i == 53) {

if (!str.equals("review\_scores\_location")) {

review\_scores\_location = Integer.parseInt(str);

}

}

if (i == 54) {

if (!str.equals("review\_scores\_value")) {

review\_scores\_value = Integer.parseInt(str);

}

}

i++;

}

}

totalRating = review\_scores\_rating + review\_scores\_accuracy + review\_scores\_cleanliness + review\_scores\_checkin + review\_scores\_communication + review\_scores\_location

+ review\_scores\_value;

percentage = (totalRating / 160.0F) \* 100;

ckw.setListing\_Id(listing\_Id.toString());

ckw.setPercentage(percentage);

context.write(ckw, NullWritable.get());

}

}

public static class ReducerClass extends Reducer<CompositeKeyWritable, NullWritable, CompositeKeyWritable, NullWritable> {

public static int a = 0;

@Override

public void reduce(CompositeKeyWritable key, Iterable<NullWritable> values, Context context) throws IOException, InterruptedException {

for (NullWritable val : values) {

if (a == 50) {

break;

} else {

context.write(key, NullWritable.get());

a++;

}

}

}

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

try {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Occupancy Rate");

job.setJarByClass(AirBnB\_15.class);

job.setMapperClass(MapperClass.class);

job.setMapOutputKeyClass(CompositeKeyWritable.class);

job.setMapOutputValueClass(NullWritable.class);

job.setReducerClass(ReducerClass.class);

job.setGroupingComparatorClass(GroupComparator.class);

job.setOutputKeyClass(CompositeKeyWritable.class);

job.setOutputValueClass(NullWritable.class);

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

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

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

} catch (IOException | InterruptedException | ClassNotFoundException ex) {

System.out.println("Erorr Message" + ex.getMessage());

}

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_15;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.io.WritableUtils;

/\*\*

\*

\* @author khushbu

\*/

public class CompositeKeyWritable implements Writable, WritableComparable<CompositeKeyWritable> {

private String listing\_Id;

private Float percentage;

public CompositeKeyWritable() {

}

public CompositeKeyWritable(String listing\_Id, Float percentage) {

this.listing\_Id = listing\_Id;

this.percentage = percentage;

}

public String getListing\_Id() {

return listing\_Id;

}

public void setListing\_Id(String listing\_Id) {

this.listing\_Id = listing\_Id;

}

public Float getPercentage() {

return percentage;

}

public void setPercentage(Float percentage) {

this.percentage = percentage;

}

@Override

public void write(DataOutput d) throws IOException {

WritableUtils.writeString(d, listing\_Id);

WritableUtils.writeString(d, String.valueOf(percentage));

}

@Override

public void readFields(DataInput di) throws IOException {

listing\_Id = WritableUtils.readString(di);

percentage =Float.parseFloat(WritableUtils.readString(di));

}

public String toString() {

return (new StringBuilder().append(listing\_Id).append("\t").append(percentage)).toString();

}

@Override

public int compareTo(CompositeKeyWritable o) {

int result = percentage.compareTo(o.percentage);

if (result == 0) {

result = listing\_Id.compareTo(o.listing\_Id);

}

return -result;

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_15;

import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.io.WritableComparator;

/\*\*

\*

\* @author pooja

\*/

public class GroupComparator extends WritableComparator {

protected GroupComparator()

{

super(CompositeKeyWritable.class, true);

}

@Override

public int compare(WritableComparable w1, WritableComparable w2)

{

CompositeKeyWritable cw1 = (CompositeKeyWritable) w1;

CompositeKeyWritable cw2 = (CompositeKeyWritable) w2;

return (cw1.getPercentage().compareTo(cw2.getPercentage()));

}

}

**Code 16:**

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_16;

import java.io.IOException;

import java.util.StringTokenizer;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Job;

import org.apache.hadoop.mapreduce.Mapper;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.lib.input.FileInputFormat;

import org.apache.hadoop.mapreduce.lib.output.FileOutputFormat;

/\*\*

\*

\* @author khushbuprkh

\*/

public class AirBnB\_16 {

public static class MapperClass extends Mapper<Object, Text, Text, CustomWritableClass> {

@Override

public void map(Object key, Text value, Context context) throws IOException, InterruptedException {

Text host\_neighbourhood = new Text();

Float min\_Security\_Deposit = 0F;

Float max\_Security\_Deposit = 0F;

Float average\_Security\_Deposit = 0.F;

CustomWritableClass cw = new CustomWritableClass();

StringTokenizer itr = new StringTokenizer(value.toString(), ",");

while (itr.hasMoreTokens()) {

int i = 0;

while (itr.hasMoreTokens()) {

String str = itr.nextToken();

if (i == 1) {

if (!str.equals("host\_neighbourhood")) {

host\_neighbourhood.set(str);

}

}

if (i == 33) {

if (!str.equals("security\_deposit")) {

min\_Security\_Deposit = Float.parseFloat(str);

max\_Security\_Deposit = Float.parseFloat(str);

average\_Security\_Deposit = Float.parseFloat(str);

}

}

i++;

}

}

cw.setMin\_Security\_Deposit(min\_Security\_Deposit);

cw.setMax\_Security\_Deposit(max\_Security\_Deposit);

cw.setAverage\_Security\_Deposit(average\_Security\_Deposit);

context.write(host\_neighbourhood, cw);

}

}

public static class ReducerClass extends Reducer<Text, CustomWritableClass, Text, CustomWritableClass> {

@Override

public void reduce(Text key, Iterable<CustomWritableClass> values, Context context) throws IOException, InterruptedException {

Float average\_Security\_Deposit = 0F;

Long count = 0L;

Float min\_Security\_Deposit = 0F;

Float max\_Security\_Deposit = 0F;

Float max = Float.MIN\_VALUE;

Float min = Float.MAX\_VALUE;

CustomWritableClass cw = new CustomWritableClass();

for (CustomWritableClass val : values) {

if (val.getMin\_Security\_Deposit()> max) {

max=val.getMin\_Security\_Deposit();

max\_Security\_Deposit = val.getMin\_Security\_Deposit();

}

if (val.getMin\_Security\_Deposit()< min) {

min=val.getMin\_Security\_Deposit();

min\_Security\_Deposit = val.getMin\_Security\_Deposit();

}

average\_Security\_Deposit += val.getAverage\_Security\_Deposit();

count++;

}

average\_Security\_Deposit = (Float) average\_Security\_Deposit / count;

cw.setMin\_Security\_Deposit(min\_Security\_Deposit);

cw.setMax\_Security\_Deposit(max\_Security\_Deposit);

cw.setAverage\_Security\_Deposit(average\_Security\_Deposit);

context.write(key, cw);

}

}

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

// TODO code application logic here

try {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "MinMax");

job.setJarByClass(AirBnB\_16.class);

job.setMapperClass(MapperClass.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(CustomWritableClass.class);

job.setReducerClass(ReducerClass.class);

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(CustomWritableClass.class);

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

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

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

} catch (IOException | InterruptedException | ClassNotFoundException ex) {

System.out.println("Erorr Message" + ex.getMessage());

}

}

}

/\*

\* To change this license header, choose License Headers in Project Properties.

\* To change this template file, choose Tools | Templates

\* and open the template in the editor.

\*/

package airbnb\_16;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableUtils;

/\*\*

\*

\* @author khushbuprkh

\*/

public class CustomWritableClass implements Writable {

private float min\_Security\_Deposit;

private float max\_Security\_Deposit;

private float average\_Security\_Deposit;

public CustomWritableClass() {

}

public float getMin\_Security\_Deposit() {

return min\_Security\_Deposit;

}

public void setMin\_Security\_Deposit(float min\_Security\_Deposit) {

this.min\_Security\_Deposit = min\_Security\_Deposit;

}

public float getMax\_Security\_Deposit() {

return max\_Security\_Deposit;

}

public void setMax\_Security\_Deposit(float max\_Security\_Deposit) {

this.max\_Security\_Deposit = max\_Security\_Deposit;

}

public float getAverage\_Security\_Deposit() {

return average\_Security\_Deposit;

}

public void setAverage\_Security\_Deposit(float average\_Security\_Deposit) {

this.average\_Security\_Deposit = average\_Security\_Deposit;

}

@Override

public void write(DataOutput d) throws IOException {

d.writeFloat(min\_Security\_Deposit);

d.writeFloat(max\_Security\_Deposit);

d.writeFloat(average\_Security\_Deposit);

}

@Override

public void readFields(DataInput di) throws IOException {

min\_Security\_Deposit = di.readFloat();

max\_Security\_Deposit = di.readFloat();

average\_Security\_Deposit = di.readFloat();

}

public String toString() {

return (new StringBuilder().append(min\_Security\_Deposit).append("\t").append(max\_Security\_Deposit).append("\t").append(average\_Security\_Deposit)).toString();

}

}