Project Report:

Short Summary:

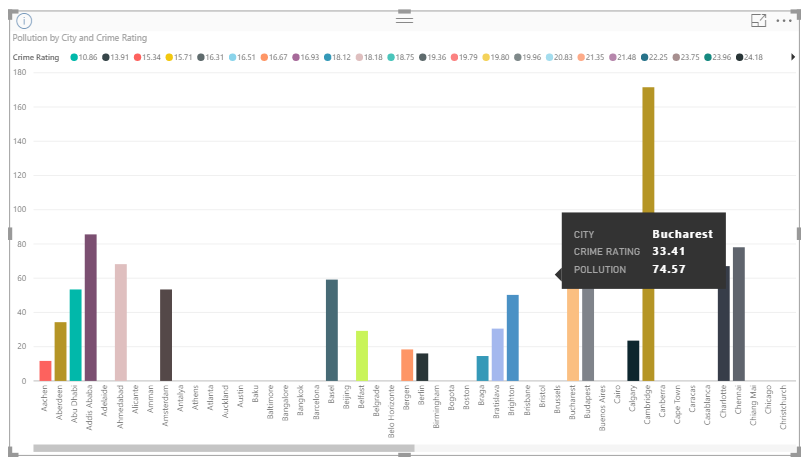
This project shows the analysis of the different cities all over the world. On the basis of quality of life, Rent, salary, gas etc

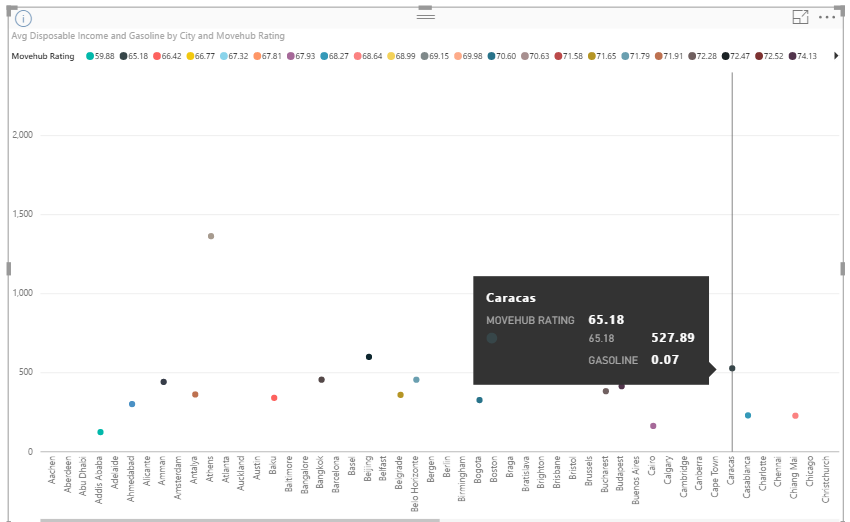
Link to the dataset:

<https://www.kaggle.com/blitzr/movehub-city-rankings>

Short Summary of the analysis:

I have used joins, Partition, Top ten filtering pattern, Hive and pig analysis, Power BI visualization.





/\*

\* 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 cityhub;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import java.util.ArrayList;

import java.util.logging.Level;

import java.util.logging.Logger;

import org.apache.hadoop.conf.Configuration;

import org.apache.hadoop.conf.Configured;

import org.apache.hadoop.fs.Path;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableUtils;

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.apache.hadoop.util.Tool;

import org.apache.hadoop.util.ToolRunner;

/\*\*

\*

\* @author pranali

\*/

public class CityHub extends Configured implements Tool {

@Override

public int run(String[] strings) throws Exception {

Configuration conf = new Configuration();

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

job.setJarByClass(CityHub.class);

MultipleInputs.addInputPath(job, new Path(strings[0]), TextInputFormat.class, JoinMapper1.class);

MultipleInputs.addInputPath(job, new Path(strings[1]), TextInputFormat.class, JoinMapper2.class);

job.getConfiguration().set("join.type", "innerjoin");

job.setReducerClass(JoinReducer.class);

job.setOutputFormatClass(TextOutputFormat.class);

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

job.setOutputKeyClass(Text.class);

job.setOutputValueClass(Text.class);

boolean complete = job.waitForCompletion(true);

Configuration conf1 = new Configuration();

Job job2 = Job.getInstance(conf1,"chaining");

if(complete){

job2.setJarByClass(CityHub.class);

MultipleInputs.addInputPath(job2, new Path(strings[2]), TextInputFormat.class, JoinMapper3.class);

MultipleInputs.addInputPath(job2, new Path(strings[3]), TextInputFormat.class, JoinMapper4.class);

job2.getConfiguration().set("join.type", "innerjoin");

job2.setReducerClass(JoinReducer1.class);

job2.setOutputFormatClass(TextOutputFormat.class);

job2.setOutputKeyClass(Text.class);

job2.setOutputValueClass(Text.class);

TextOutputFormat.setOutputPath(job2, new Path(strings[4]));

}

boolean success = job2.waitForCompletion(true);

return success ? 0 : 4;

}

public static class JoinMapper1 extends Mapper<Object, Text, Text, Text> {

private Text outKey = new Text();

private Text outValue = new Text();

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

try {

String[] cityData = value.toString().split(",");

String city = cityData[0].trim();

String str= city.substring(1, city.length()-1);

if (str == null) {

return;

}

System.out.println(str +"11");

outKey.set(str);

outValue.set("A" + value);

context.write(outKey, outValue);

} catch (IOException | InterruptedException ex) {

Logger.getLogger(CityHub.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

public static class JoinMapper2 extends Mapper<Object, Text, Text, Text> {

private Text outKey = new Text();

private Text outValue = new Text();

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

try {

String[] cityData = value.toString().split(",");

String city = cityData[0].trim();

System.out.println(city +"12");

if (city == null) {

return;

}

outKey.set(city);

outValue.set("B" + value);

context.write(outKey, outValue);

} catch (IOException | InterruptedException ex) {

Logger.getLogger(CityHub.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

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

private static final Text EMPTY\_TEXT = new Text();

private Text tmp = new Text();

private ArrayList<Text> listA = new ArrayList<Text>();

private ArrayList<Text> listB = new ArrayList<Text>();

// private ArrayList<Text> listC = new ArrayList<Text>();

private String joinType = null;

public void setup(Context context) {

joinType = context.getConfiguration().get("join.type");

}

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

listA.clear();

listB.clear();

while (values.iterator().hasNext()) {

tmp = values.iterator().next();

if (tmp.charAt(0) == 'A') {

listA.add(new Text(tmp.toString().substring(1)));

} else if (tmp.charAt(0) == 'B') {

listB.add(new Text(tmp.toString().substring(1)));

}

}

executeJoinLogic(context);

}

private void executeJoinLogic(Context context) throws InterruptedException, IOException {

if(joinType.equalsIgnoreCase("innerjoin"))

{

if(!listA.isEmpty() && !listB.isEmpty() )

{

for(Text A : listA)

{

for(Text B : listB)

{

context.write(A,B);

}

}

}

}

}

}

public static class JoinMapper3 extends Mapper<Object, Text, Text, Text> {

private Text outKey = new Text();

private Text outValue = new Text();

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

try {

String[] cityData = value.toString().split(",");

String city = cityData[0].trim();

String str= city.substring(1, city.length()-1);

if (str == null) {

return;

}

outKey.set(str);

outValue.set("C" + value);

context.write(outKey, outValue);

} catch (IOException | InterruptedException ex) {

Logger.getLogger(CityHub.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

public static class JoinMapper4 extends Mapper<Object, Text, Text, Text> {

private Text outKey = new Text();

private Text outValue = new Text();

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

try {

String[] cityData = value.toString().split(",");

String city = cityData[0].trim();

if (city == null) {

return;

}

outKey.set(city);

outValue.set("D" + value);

context.write(outKey, outValue);

} catch (IOException | InterruptedException ex) {

Logger.getLogger(CityHub.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

public static class JoinReducer1 extends Reducer<Text, Text, Text, Text> {

private static final Text EMPTY\_TEXT = new Text();

private Text tmp = new Text();

private ArrayList<Text> listC = new ArrayList<Text>();

private ArrayList<Text> listD = new ArrayList<Text>();

// private ArrayList<Text> listC = new ArrayList<Text>();

private String joinType = null;

public void setup(Context context) {

joinType = context.getConfiguration().get("join.type");

}

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

listC.clear();

listD.clear();

while (values.iterator().hasNext()) {

tmp = values.iterator().next();

if (tmp.charAt(0) == 'C') {

listC.add(new Text(tmp.toString().substring(1)));

} else if (tmp.charAt(0) == 'D') {

listD.add(new Text(tmp.toString().substring(1)));

}

}

executeJoinLogic(context);

}

private void executeJoinLogic(Context context) throws InterruptedException, IOException

{

if(joinType.equalsIgnoreCase("innerjoin"))

{

if(!listC.isEmpty() && !listD.isEmpty())

{

for(Text C : listC)

{

for(Text D : listD)

{

context.write(C,D);

}

}

}

}

}

}

public static void main(String[] args) throws Exception {

try {

int res = ToolRunner.run(new Configuration(), new CityHub(), args);

} catch (IOException ex) {

Logger.getLogger(CityHub.class.getName()).log(Level.SEVERE, null, ex);

}

}

}

/\*

\* 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 cityhubpartitioningcountry;

import java.io.IOException;

import org.apache.hadoop.conf.Configurable;

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.Partitioner;

import org.apache.hadoop.mapreduce.Reducer;

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

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

/\*\*

\*

\* @author pranali

\*/

public class CityHubPartitioning {

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

private Text country= new Text();

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

String[] tokens = value.toString().split(",");

String mn = tokens[1].trim();

country.set(mn);

context.write(country, value);

}

}

public static class PartitionCountryPartitioner extends Partitioner<Text,Text> implements Configurable{

private Configuration conf= null;

private int ret=0;

@Override

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

if(key.toString().startsWith("A")){

return 0;

}

else if(key.toString().startsWith("B")){

return 1;

}

else if(key.toString().startsWith("C")){

return 2;

}

else if(key.toString().startsWith("D")){

return 3;

}

else if(key.toString().startsWith("E")){

return 4;

}

else if(key.toString().startsWith("F")){

return 5;

}

else if(key.toString().startsWith("G")){

return 6;

}

else if(key.toString().startsWith("H")){

return 7;

}

else if(key.toString().startsWith("I")){

return 8;

}

else if(key.toString().startsWith("J")){

return 9;

}

else if(key.toString().startsWith("K")){

return 10;

}

else if(key.toString().startsWith("L")){

return 11;

}

else if(key.toString().startsWith("M")){

return 12;

}

else if(key.toString().startsWith("N")){

return 13;

}

else if(key.toString().startsWith("O")){

return 14;

}

else if(key.toString().startsWith("P")){

return 15;

}

else if(key.toString().startsWith("Q")){

return 16;

}

else if(key.toString().startsWith("R")){

return 17;

}

else if(key.toString().startsWith("S")){

return 18;

}

else if(key.toString().startsWith("T")){

return 19;

}

else if(key.toString().startsWith("U")){

return 20;

}

else if(key.toString().startsWith("V")){

return 21;

}

else if(key.toString().startsWith("W")){

return 22;

}

else if(key.toString().startsWith("X")){

return 23;

}

else if(key.toString().startsWith("Y")){

return 24;

}

else if(key.toString().startsWith("Z")){

return 25;

}

else

{return 26;}

}

@Override

public void setConf(Configuration conf) {

this.conf=conf;

}

@Override

public Configuration getConf() {

return conf;

}

}

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

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

for (Text t: values){

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

}

}

}

/\*\*

\* @param args the command line arguments

\*/

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

Configuration conf = new Configuration();

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

job.setJarByClass(CityHubPartitioning.class);

job.setMapperClass(PartitionMonthMapper.class);

job.setReducerClass(countryReducer.class);

job.setMapOutputKeyClass(Text.class);

job.setMapOutputValueClass(Text.class);

job.setPartitionerClass(PartitionCountryPartitioner.class);

job.setNumReduceTasks(27);

job.setOutputKeyClass(Text.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);

}

}

/\*

\* 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 cityhubtopten;

import java.io.IOException;

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.lib.input.FileInputFormat;

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

/\*\*

\*

\* @author pranali

\*/

public class CityHubTopTen {

/\*\*

\* @param args the command line arguments

\*/

public static void main(String[] args) {

try {

Configuration conf = new Configuration();

Job job = Job.getInstance(conf, "Top10");

job.setJarByClass(CityHubTopTen.class);

job.setMapperClass(Top10Mapper.class);

job.setReducerClass(Top10Reducer.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);

} 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 cityhubtopten;

import java.io.DataInput;

import java.io.DataOutput;

import java.io.IOException;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.FloatWritable;

import org.apache.hadoop.io.Writable;

import org.apache.hadoop.io.WritableComparable;

import org.apache.hadoop.io.WritableUtils;

/\*\*

\*

\* @author pranali

\*/

public class QualityOfLife implements Writable, WritableComparable<QualityOfLife>{

private double qualityOfLife;

public double getQualityOfLife() {

return qualityOfLife;

}

public void setQualityOfLife(double qualityOfLife) {

this.qualityOfLife = qualityOfLife;

}

public QualityOfLife(double Ql) {

super();

this.qualityOfLife=Ql;

}

@Override

public void write(DataOutput d) throws IOException {

d.writeDouble(qualityOfLife);

}

@Override

public void readFields(DataInput di) throws IOException {

qualityOfLife = di.readDouble();

}

@Override

public int compareTo(QualityOfLife e1) {

if(qualityOfLife > e1.getQualityOfLife()){

return 1;

} else {

return -1;

}

}

public String toString()

{

return (new StringBuilder().append(qualityOfLife).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 cityhubtopten;

import java.io.IOException;

import java.util.Iterator;

import java.util.Map.Entry;

import java.util.TreeMap;

import org.apache.hadoop.io.DoubleWritable;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Mapper;

/\*\*

\*

\* @author pranali

\*/

public class Top10Mapper extends Mapper<LongWritable, Text, NullWritable, Text> {

public static TreeMap<QualityOfLife, Text> ToRecordMap = new TreeMap<QualityOfLife , Text>();

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

String line=value.toString();

String[] tokens=line.split(",");

if(line.contains("Quality of Life")){

return;

}

double quality=Double.parseDouble(tokens[6]);

ToRecordMap.put(new QualityOfLife (quality), new Text(value));

Iterator<Entry<QualityOfLife , Text>> iter = ToRecordMap.entrySet().iterator();

Entry<QualityOfLife , Text> entry = null;

while(ToRecordMap.size()>10){

entry = iter.next();

iter.remove();

}

}

protected void cleanup(Context context) throws IOException, InterruptedException {

for (Text t:ToRecordMap.values()) {

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

}

}

}

/\*

\* 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 cityhubtopten;

/\*\*

\*

\* @author pranali

\*/

import java.io.IOException;

import java.util.Iterator;

import java.util.TreeMap;

import java.util.Map.Entry;

import org.apache.hadoop.io.LongWritable;

import org.apache.hadoop.io.NullWritable;

import org.apache.hadoop.io.Text;

import org.apache.hadoop.mapreduce.Reducer;

import org.apache.hadoop.mapreduce.Reducer.Context;

public class Top10Reducer extends Reducer<NullWritable, Text, NullWritable, Text> {

public static TreeMap<QualityOfLife , Text> ToRecordMap = new TreeMap<QualityOfLife , Text>();

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

for (Text value : values) {

String line=value.toString();

if(line.length()>0){

String[] tokens=line.split(",");

//split the data and fetch salary

double quality=Double.parseDouble(tokens[6]);

//insert salary as key and entire row as value

//tree map sort the records based on salary

ToRecordMap.put(new QualityOfLife (quality), new Text(value));

}

}

// If we have more than ten records, remove the one with the lowest sal

// As this tree map is sorted in descending order, the user with

// the lowest sal is the last key.

Iterator<Entry<QualityOfLife , Text>> iter = ToRecordMap.entrySet().iterator();

Entry<QualityOfLife , Text> entry = null;

while(ToRecordMap.size()>10){

entry = iter.next();

iter.remove();

}

for (Text t : ToRecordMap.descendingMap().values()) {

// Output our ten records to the file system with a null key

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

}

}

}