Assignment 9 (melting,pivot)

A. Subset: df_photo=df[df["Type"]=="Photo"]
B. Merge subsets:df_merged_ps=pd.concat([df_photo,df_link])
C. Sort by values: df_sort_like = df.sort_values('like',ascending**=False**)
D. df.transpose()
E. df_melted=pd.melt(df,id_vars=['Type','Category'],value_name='value',var_name='Metrics')
F. casting = pd.pivot_table(df,index=['Type','Category'],values='like')

Assignment 10()

pd.read_csv('./Iris.csv',names=['SepalLengthCm','SepalWidthCm','PetalLengthCm''])
pd.melt(df,id_vars=['sepal length in cm','petal length in cm'],value_name='value',var_name='Metrics')
casting =
pd.pivot_table(df,index=['SepalLengthCm','PetalLengthCm'],values=['SepalWidthCm','PetalWidthCm'])

Assignment 17,18,21,22,28

```
df.isnull().sum()

df = df.replace('?', np.nan)
df=df.dropna()

sns.histplot(data=df, x='age')

sns.stripplot(x='cp', y='trestbps', data=df)

sns.barplot(x=df['num'].value_counts().index, y=df['num'].value_counts())

sns.lineplot(x='thalach', y='cp', data=df)

sns.scatterplot(x='age', y='chol', data=df, hue='num')

plt.pie(df['cp'].value_counts(), labels=df['cp'].value_counts().index)

sns.boxplot(x="age",data=df)

sns.heatmap(df.corr())
```

Assignment 26

```
df.isna().sum()

numeric_columns = df.select_dtypes(include=np.number)
z_scores = (numeric_columns - numeric_columns.mean()) / numeric_columns.std()
df_no_outliers = df[(np.abs(z_scores) < 3).all(axis=1)]
```

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB,MultinomialNB
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import accuracy_score

x_train,x_test,y_train,y_test=train_test_split(X,y,test_size=0.2,random_state=46)
nvg=MultinomialNB()
nvg.fit(x_train,y_train)
ypred2 = nvg.predict(x_test)

accuracy_score(y_test,ypred3)
```

Assignment 27

```
duplicate_rows = df.duplicated()
df = df.drop_duplicates()

sns.heatmap(df.corr())   df = df.fillna(df.mean())
```

Webscraper

```
import requests
from bs4 import BeautifulSoup as bs
response = requests.get(url)
  soup = bs(response.text, 'html.parser')
  ratings = soup.findAll('div', {'class': '_3LWZlK _1BLPMq'})
 for rat in ratings:
    ratingls.append(rat.get_text())
import pandas as pd
df = pd.DataFrame()
df["Customer Name"] = customer_namels[:122]

df.to_excel('new1.xlsx')
```

Wordcount:

```
package wordcountpkg;

import java.io.IOException;
import java.text.ParseException;
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.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;
import org.apache.hadoop.util.GenericOptionsParser;

public class wordcountcls {
   public static void main(String [] args) throws Exception{
            Configuration c = new Configuration();
            String[] files = new GenericOptionsParser(c,args).getRemainingArgs();
            Path input=new Path(files[0]);
            Path output=new Path(files[1]);
            Job j=new Job(c,"WordCount");
            j.setJarByClass(wordcountcls.class);
            j.setMapperClass(MapForWordCount.class);
            j.setReducerClass(ReduceForWordCount.class);
            j.setOutputKeyClass(Text.class);
            j.setOutputValueClass(IntWritable.class);
            FileInputFormat.addInputPath(j, input);
            FileOutputFormat.setOutputPath(j, output);
            System.exit(j.waitForCompletion(true)?0:1);
      }

   public static class MapForWordCount extends Mapper<LongWritable, Text, Text, IntWritable>
{

      public void map(LongWritable key, Text value, Context con) throws IOException,
InterruptedException {
         String str = value.toString();
         String[] words = str.split(",");

         for(String word:words){
               Text output_key = new Text(word.toUpperCase().trim());
               IntWritable output_value = new IntWritable(1);

               con.write(output_key, output_value);
         }
      }
   }
```

```
   public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text,
IntWritable> {

       public void reduce(Text key, Iterable<IntWritable> values, Context con) throws
IOException, InterruptedException {
          int count = 0;

          for(IntWritable num:values){
                    count+=num.get();
          }

          con.write(key, new IntWritable(count));
       }

   }
}
//gedit new.txt
//hadoop fs -put new.txt newhf
//hadoop jar wcjar.jar wordcountpkg.wordcountcls newhf wcdir
//hadoop fs -ls wcdir
// hadoop fs -cat wcdir/part-r-00000
```

2.Logfile

```
package logfilepkg;

import java.io.IOException;
import java.text.ParseException;

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.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;
import org.apache.hadoop.util.GenericOptionsParser;

public class logfilecls {
   public static void main(String [] args) throws Exception{
             Configuration c = new Configuration();
```

```
            String[] files = new GenericOptionsParser(c,args).getRemainingArgs();
            Path input=new Path(files[0]);
            Path output=new Path(files[1]);
            Job j=new Job(c,"WordCount");
            j.setJarByClass(logfilecls.class);
            j.setMapperClass(MapForWordCount.class);
            j.setReducerClass(ReduceForWordCount.class);
            j.setOutputKeyClass(Text.class);
            j.setOutputValueClass(IntWritable.class);
            FileInputFormat.addInputPath(j, input);
            FileOutputFormat.setOutputPath(j, output);
            System.exit(j.waitForCompletion(true)?0:1);
      }

   public static class MapForWordCount extends Mapper<LongWritable, Text, Text, IntWritable>
{

      public void map(LongWritable key, Text value, Context con) throws IOException,
InterruptedException {
         String alltext = value.toString();
         String[] words = alltext.split("\n");

         for(String word: words){
                  String[] lineele = word.split(",");
                           String ip = lineele[1];
                           String[] indatetime = lineele[5].split(" ");
                           String[] outdatetime = lineele[7].split(" ");
                           String intime = indatetime[1];
                           String outtime = outdatetime[1];
                           String[] intimearr = intime.split(":");
                           String[] outtimearr = outtime.split(":");
                           int inhr = Integer.parseInt(intimearr[0])*3600;
                           int inmin = Integer.parseInt(intimearr[1])*60;
                           int insec = Integer.parseInt(intimearr[2]);
                           int outhr = Integer.parseInt(outtimearr[0])*3600;
                           int outmin = Integer.parseInt(intimearr[1])*60;
                           int outsec = Integer.parseInt(intimearr[2]);
                           int totalin = inhr+inmin+insec;
                           int totalout = outhr+outmin+outsec;
                           int totallogin = totalout - totalin;

                  con.write(new Text(ip), new IntWritable(totallogin));
         }
      }
```

```
    }

    public static class ReduceForWordCount extends Reducer<Text, IntWritable, Text,
IntWritable> {
        int maxt=0;
        Text maxip=new Text();
      public void reduce(Text key, Iterable<IntWritable> values, Context con) throws
IOException, InterruptedException {
        int curr_time = 0;

        for(IntWritable value: values){
                curr_time+=value.get();
        }

        if(curr_time>maxt){
                maxt=curr_time;
                maxip=key;
        }

      }
      protected void cleanup(Context con) throws IOException, InterruptedException{
        con.write(maxip,new IntWritable(maxt));
      }

    }
}
```

3.Music

```
package PS2Package;

import java.io.*;
import java.util.*;

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.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;
```

```
import org.apache.hadoop.util.GenericOptionsParser;


public class Music1 {
	public static void main(String[] args) throws Exception{
		Configuration c=new Configuration();
		String[] files=new GenericOptionsParser(c,args).getRemainingArgs();
		Path inputfile=new Path(files[0]),outputfile=new Path(files[1]);
		Job j1=new Job(c,"getlisteners");
		j1.setJarByClass(Music1.class);
		j1.setMapperClass(Music1Mapper.class);
		j1.setReducerClass(Music1Reducer.class);
		j1.setOutputKeyClass(Text.class);
		j1.setOutputValueClass(IntWritable.class);
		FileOutputFormat.setOutputPath(j1,outputfile);
		FileInputFormat.addInputPath(j1,inputfile);
		System.exit(j1.waitForCompletion(true)?0:1);
	}
	public static class Music1Mapper extends Mapper<LongWritable,Text,Text,IntWritable> {
		public void map(LongWritable key,Text textfile,Context con) throws IOException,
InterruptedException{
			String textlines=textfile.toString().toLowerCase().trim();
			String[] lines=textlines.split("/n");
			for( String line: lines){
				String[] values=line.split(",");
				Text songid=new Text(values[1]);
				IntWritable shared=new IntWritable(Integer.parseInt(values[2]));
				con.write(songid, shared);
			}
		}
	}
	public static class Music1Reducer extends Reducer<Text,IntWritable,Text,IntWritable> {
		//int count=0;
		public void reduce(Text key, Iterable<IntWritable> args,Context con) throws
IOException, InterruptedException{
			//++count;
			int count=0;
			int sharedcount=0;
			for(IntWritable value: args){
				++count;
				sharedcount+=value.get();
			}
```

```
                        String uniquelistener="Number of unique listeners for track id
"+key.toString()+" :";
                        //uniquelistener.concat(key.toString());
                        //uniquelistener.concat(": ");
                        String shares="Number of times track id "+key.toString()+" was shared: ";
                        //shares.concat(key.toString());
                        con.write(new Text(uniquelistener),new IntWritable(count));
                        con.write(new Text(shares),new IntWritable(sharedcount));
                }
        }
}
```

4. Radio

```
package radiopkg;

import java.io.IOException;
import java.text.ParseException;
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.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;
import org.apache.hadoop.util.GenericOptionsParser;

public class radiocls {
   public static void main(String [] args) throws Exception{
            Configuration c = new Configuration();
            String[] files = new GenericOptionsParser(c,args).getRemainingArgs();
            Path input=new Path(files[0]);

            Path output=new Path(files[1]);
            Job j=new Job(c,"WordCount");
            j.setJarByClass(radiocls.class);
            j.setMapperClass(MapForWordCount.class);
            j.setReducerClass(ReduceForWordCount.class);
            j.setOutputKeyClass(Text.class);
            j.setOutputValueClass(IntWritable.class);
```

```
            FileInputFormat.addInputPath(j, input);
            FileOutputFormat.setOutputPath(j, output);

            Path output1=new Path(files[2]);
            Job j1=new Job(c,"WordCount1");
            j1.setJarByClass(radiocls.class);
            j1.setMapperClass(MapForWordCount1.class);
            j1.setReducerClass(ReduceForWordCount1.class);
            j1.setOutputKeyClass(Text.class);
            j1.setOutputValueClass(IntWritable.class);
            FileInputFormat.addInputPath(j1, input);
            FileOutputFormat.setOutputPath(j1, output1);

            System.exit(j.waitForCompletion(true) && j1.waitForCompletion(true)?0:1);
      }

   public static class MapForWordCount extends Mapper<LongWritable, Text, Text, IntWritable>
{

      public void map(LongWritable key, Text value, Context con) throws IOException,
InterruptedException {
         String alltext = value.toString();
         String[] lines = alltext.split("\n");

         for(String line:lines){

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

                  Text trackid = new Text(row[1].toString());
                  IntWritable output_value = new IntWritable(Integer.parseInt(row[3].toString()));


                  con.write(trackid, output_value);
         }
      }
   }

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

      public void reduce(Text key, Iterable<IntWritable> values, Context con) throws
IOException, InterruptedException {
         int count = 0;
```

```
        for(IntWritable num:values){
                count+=num.get();
        }

        Text op = new Text("Track listned: "+key.toString());

        con.write(op, new IntWritable(count));
      }

   }

   public static class MapForWordCount1 extends Mapper<LongWritable, Text, Text,
IntWritable> {

      public void map(LongWritable key, Text value, Context con) throws IOException,
InterruptedException {
        String alltext = value.toString();
        String[] lines = alltext.split("\n");

        for(String line:lines){

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

                Text trackid = new Text(row[1].toString());
                IntWritable output_value = new IntWritable(Integer.parseInt(row[4].toString()));


                con.write(trackid, output_value);
        }
      }
   }

   public static class ReduceForWordCount1 extends Reducer<Text, IntWritable, Text,
IntWritable> {

      public void reduce(Text key, Iterable<IntWritable> values, Context con) throws
IOException, InterruptedException {
        int count = 0;

        for(IntWritable num:values){
                count+=num.get();
        }

        Text op = new Text("Track skipped: "+key.toString());
```

```
            con.write(op, new IntWritable(count));
        }

    }
}
```

5.Movie

```
package moviepkg;

import java.io.IOException;
import java.text.ParseException;

import org.apache.hadoop.conf.Configuration;
import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
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;
import org.apache.hadoop.util.GenericOptionsParser;

public class moviecls {
    public static void main(String [] args) throws Exception{
            Configuration c = new Configuration();
            String[] files = new GenericOptionsParser(c,args).getRemainingArgs();
            Path input=new Path(files[0]);
            Path output=new Path(files[1]);
            Job j=new Job(c,"WordCount");
            j.setJarByClass(moviecls.class);
            j.setMapperClass(MapForWordCount.class);
            j.setReducerClass(ReduceForWordCount.class);
            j.setOutputKeyClass(Text.class);
            j.setOutputValueClass(DoubleWritable.class);
            FileInputFormat.addInputPath(j, input);
            FileOutputFormat.setOutputPath(j, output);
            System.exit(j.waitForCompletion(true)?0:1);
        }
```

```
   public static class MapForWordCount extends Mapper<LongWritable, Text, Text,
DoubleWritable> {

      public void map(LongWritable key, Text value, Context con) throws IOException,
InterruptedException {
         String alltext = value.toString();
         String[] lines = alltext.split("\n");

         for(String line:lines){

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

                  Text movieid = new Text(row[1].toString());
                  DoubleWritable output_value = new
DoubleWritable(Double.parseDouble(row[2].toString()));

                  con.write(movieid, output_value);
         }
      }
   }

   public static class ReduceForWordCount extends Reducer<Text, DoubleWritable, Text,
DoubleWritable> {

      public void reduce(Text key, Iterable<DoubleWritable> values, Context con) throws
IOException, InterruptedException {
         double count = 0;
         double sum=0;

         for(DoubleWritable num:values){
                  double curr_rat = num.get();
                  count++;
                  sum+=curr_rat;
         }
         double avg = sum/(count*1.0000);

         if(avg>=4.0)
                  con.write(new Text("Movie id  "+key.toString()), new DoubleWritable(avg));
      }

   }
}
```

6.Flight

```
create '32flight','32finfo','32fsch'
put '32flight',1,'32finfo:source','pune'
put '32flight',1,'32finfo:dest','mumbai'
put '32flight',1,'32fsch:at','10:25a.m'
put '32flight',1,'32fsch:dt','11.25a.m'
put '32flight',1,'32fsch:delay',5

alter '32flight',NAME=>'revenue'
put '32flight',4,'revenue:rs','45000'
get '32flight',4
alter '32flight',NAME=>'revenue',METHOD=>'delete'
scan '32flight', {LIMIT => 10}

drop 'deltable'

hive> CREATE external TABLE 32hbase_flight_new(fno int, fsource string,fdest string,fsh_at
string,fsh_dt string,fsch_delay string) STORED BY
'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES
("hbase.columns.mapping"=":key,32finfo:source,32finfo:dest,32fsch:at,32fsch:dt,32fsch:delay")
TBLPROPERTIES ("hbase.table.name" = "32flight");

select sum(fsch_delay) from 32hbase_flight_new;

select avg(fsch_delay) from 32hbase_flight_new;

CREATE INDEX idx_invoice_no ON TABLE retail4 (InvoiceNo) AS 'COMPACT' WITH
DEFERRED REBUILD;

SHOW INDEX ON 32hbase_flight_new;
```

7.Hive

```
create table cust_info(Cust_id INT,Cust_name STRING,orderID INT)ROW FORMAT
DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

create table order_info(orderID INT,ItemID INT,Quantity INT,ItemPrice DOUBLE)ROW FORMAT
DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

create table item_info(ItemID INT,Item_Name STRING,ItemPrice DOUBLE)ROW FORMAT
DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH '/path/to/customer_info.csv' INTO TABLE cust_info;
```

```
SELECT * FROM cust_info JOIN order_info ON cust_info.orderID = order_info.orderID JOIN
item_info ON order_info.ItemID = item_info.ItemID;

 CREATE INDEX cust_idx on TABLE cust_info(Cust_id) AS 'COMPACT' WITH DEFERRED
REBUILD;

SELECT SUM(ItemPrice * Quantity) AS total_sales, AVG(ItemPrice * Quantity) AS avg_sales
FROM order_info;

SELECT * FROM order_info WHERE (ItemPrice * Quantity) = (SELECT MAX(ItemPrice *
Quantity) FROM order_info);

SELECT oi.* FROM order_info oi JOIN (SELECT MAX(ItemPrice * Quantity) AS max_value
FROM order_info) subq ON (oi.ItemPrice * oi.Quantity) = subq.max_value;

HBASE=> CREATE 'CustomerInfo','Customer'

CREATE EXTERNAL TABLE CustomerInfoHBase (Cust_ID INT,Cust_Name STRING,OrderID
INT)STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler'
WITH SERDEPROPERTIES ("hbase.columns.mapping" =
":key,Customer:cust_name,Customer:order_id")TBLPROPERTIES ("hbase.table.name" =
"CustomerInfo");

scan 'CustomerInfo';
```

8.Hive

```
create table retail4(InvoiceNo INT,StockCode STRING,Description STRING,Quantity
INT,InvoiceDate STRING,UnitPrice FLOAT,CustomerID STRING,Country STRING)ROW
FORMAT DELIMITED FIELDS TERMINATED BY ',' STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH '/home/cloudera/Downloads/Retail.csv' INTO TABLE retail4;

SELECT * FROM retail4 LIMIT 5;

CREATE INDEX idx_invoice_no ON TABLE retail4 (InvoiceNo) AS 'COMPACT' WITH
DEFERRED REBUILD;

 SELECT SUM(Quantity * UnitPrice) AS total_sales FROM retail4;

 SELECT AVG(Quantity * UnitPrice) AS total_sales FROM retail4;

SELECT InvoiceNo, SUM(Quantity * UnitPrice) AS order_cost FROM retail4 GROUP BY
InvoiceNo ORDER BY order_cost DESC LIMIT 1;
```

```
SELECT CustomerID, SUM(Quantity * UnitPrice) AS total_order_amount FROM retail4 GROUP BY CustomerID ORDER BY total_order_amount DESC LIMIT 1;

SELECT Country, SUM(Quantity * UnitPrice) AS total_sales FROM retail4 GROUP BY Country ORDER BY total_sales DESC LIMIT 1;

SELECT Country, SUM(Quantity * UnitPrice) AS total_sales FROM retail4 GROUP BY Country ORDER BY total_sales LIMIT 1;

IN HBASE
Create 'order1','order'

CREATE EXTERNAL TABLE EXTTBL (InvoiceNo INT,StockCode STRING,Description STRING,Quantity INT,InvoiceDate STRING,UnitPrice INT,CustomerID INT,Country STRING) STORED BY 'org.apache.hadoop.hive.hbase.HBaseStorageHandler' WITH SERDEPROPERTIES("hbase.columns.mapping" = ":key,order:StockCode,order:Description,order:Quantity,order:InvoiceDate,order:UnitPrice,order:CustomerID,order:Country") TBLPROPERTIES("hbase.table.name"="order1");

INSERT INTO EXTTBL SELECT * FROM retail4;

scan 'order', {LIMIT => 10}
```