Fundamentos de proyectos Big Data

Programación básica en Java con Hadoop

José Manuel Bustos Muñoz

1. Dataset elegido

Para la realización de la práctica se han elegido varios dataset relativos a la NBA, la liga de baloncesto norteamericana.

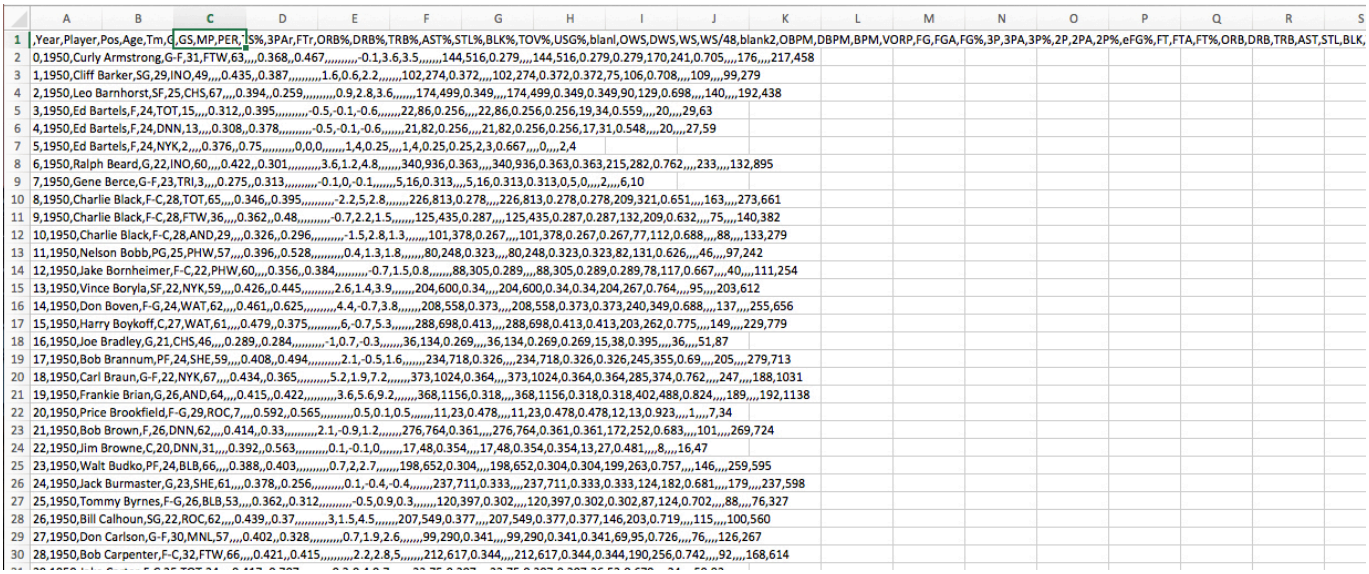
Se han descargado tres ficheros '.csv' desde algunos de los repositorios más importantes de la red.

Los tres archivos serían:

- **Player_stats.csv**: archivo con los jugadores que han participado en la NBA en la última temporada y las estadísticas que han tenido.

- **Players.csv**: archivo con la información personal de los jugadores que han jugado en la NBA, como altura, peso, lugar de nacimiento o universidad a la que asistieron antes de entrar en la liga.

- **Seasons_stats.csv**: archivo con las estadísticas de los jugadores que han jugado en la NBA, pero en lugar de ser por carrera del jugador está almacenada por temporada.



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
1	,Year,Player,Pos,Age,Tm,G,GS,MP,PER,TS%,3PAr,FTr,ORB%,DRB%,TRB%,AST%,STL%,BLK%,TOV%,USG%,blanl,OWS,DWS,WS,WS/48,blank2,OBPM,DBPM,BPM,VORP,FG,FGA,FG%,3P,3PA,3P%,2P,2PA,2P%,eFG%,FT,FTA,FT%,ORB,DRB,TRB,AST,STL,BLK,																		
2	0,1950,Curly Armstrong,G-F,31,FTW,63,,,,0.368,,0.467,,,,,,,,,-0.1,3.6,3.5,,,,,,144,516,0.279,,,,144,516,0.279,0.279,170,241,0.705,,,,176,,,,217,458																		
3	1,1950,Cliff Barker,SG,29,INO,49,,,,0.435,,0.387,,,,,,,,,1.6,0.6,2.2,,,,,,102,274,0.372,,,,102,274,0.372,0.372,75,106,0.708,,,,109,,,,99,279																		
4	2,1950,Leo Barnhorst,SF,25,CHS,67,,,,0.394,,0.259,,,,,,,,,0.9,2.8,3.6,,,,,,174,499,0.349,,,,174,499,0.349,0.349,90,129,0.698,,,,140,,,,192,438																		
5	3,1950,Ed Bartels,F,24,TOT,15,,,,0.312,,0.395,,,,,,,,,-0.5,-0.1,-0.6,,,,,,22,86,0.256,,,,22,86,0.256,0.256,19,34,0.559,,,,20,,,,29,63																		
6	4,1950,Ed Bartels,F,24,DNN,13,,,,0.308,,0.378,,,,,,,,,-0.5,-0.1,-0.6,,,,,,21,82,0.256,,,,21,82,0.256,0.256,17,31,0.548,,,,20,,,,27,59																		
7	5,1950,Ed Bartels,F,24,NYK,2,,,,0.376,,0.75,,,,,,,,,0,0,0,,,,,,1,4,0.25,,,,1,4,0.25,0.25,2,3,0.667,,,,0,,,,2,4																		
8	6,1950,Ralph Beard,G,22,INO,60,,,,0.422,,0.301,,,,,,,,,3.6,1.2,4.8,,,,,,340,936,0.363,,,,340,936,0.363,0.363,215,282,0.762,,,,233,,,,132,895																		
9	7,1950,Gene Berce,G-F,23,TRI,3,,,,0.275,,0.313,,,,,,,,,-0.1,0,-0.1,,,,,,5,16,0.313,,,,5,16,0.313,0.313,0,5,0,,,,2,,,,6,10																		
10	8,1950,Charlie Black,F-C,28,TOT,65,,,,0.346,,0.395,,,,,,,,,-2.2,5,2.8,,,,,,226,813,0.278,,,,226,813,0.278,0.278,209,321,0.651,,,,163,,,,273,661																		
11	9,1950,Charlie Black,F-C,28,FTW,36,,,,0.362,,0.48,,,,,,,,,-0.7,2.2,1.5,,,,,,125,435,0.287,,,,125,435,0.287,0.287,132,209,0.632,,,,75,,,,140,382																		
12	10,1950,Charlie Black,F-C,28,AND,29,,,,0.326,,0.296,,,,,,,,,-1.5,2.8,1.3,,,,,,101,378,0.267,,,,101,378,0.267,0.267,77,112,0.688,,,,88,,,,133,279																		
13	11,1950,Nelson Bobb,PG,25,PHW,57,,,,0.396,,0.528,,,,,,,,,0.4,1.3,1.8,,,,,,80,248,0.323,,,,80,248,0.323,0.323,82,131,0.626,,,,46,,,,97,242																		
14	12,1950,Jake Bornheimer,F-C,22,PHW,60,,,,0.356,,0.384,,,,,,,,,-0.7,1.5,0.8,,,,,,88,305,0.289,,,,88,305,0.289,0.289,78,117,0.667,,,,40,,,,111,254																		
15	13,1950,Vince Boryla,SF,22,NYK,59,,,,0.426,,0.445,,,,,,,,,2.6,1.4,3.9,,,,,,204,600,0.34,,,,204,600,0.34,0.34,204,267,0.764,,,,95,,,,203,612																		
16	14,1950,Don Boven,F-G,24,WAT,62,,,,0.461,,0.625,,,,,,,,,4.4,-0.7,3.8,,,,,,208,558,0.373,,,,208,558,0.373,0.373,240,349,0.688,,,,137,,,,255,656																		
17	15,1950,Harry Boykoff,C,27,WAT,61,,,,0.479,,0.375,,,,,,,,,6,-0.7,5.3,,,,,,288,698,0.413,,,,288,698,0.413,0.413,203,262,0.775,,,,149,,,,229,779																		
18	16,1950,Joe Bradley,G,21,CHS,46,,,,0.289,,0.284,,,,,,,,,-1,0.7,-0.3,,,,,,36,134,0.269,,,,36,134,0.269,0.269,15,38,0.395,,,,36,,,,51,87																		
19	17,1950,Bob Brannum,PF,24,SHE,59,,,,0.408,,0.494,,,,,,,,,2.1,-0.5,1.6,,,,,,234,718,0.326,,,,234,718,0.326,0.326,245,355,0.69,,,,205,,,,279,713																		
20	18,1950,Carl Braun,G-F,22,NYK,67,,,,0.434,,0.365,,,,,,,,,5.2,1.9,7.2,,,,,,373,1024,0.364,,,,373,1024,0.364,0.364,285,374,0.762,,,,247,,,,188,1031																		
21	19,1950,Frankie Brian,G,26,AND,64,,,,0.415,,0.422,,,,,,,,,3.6,5.6,9.2,,,,,,368,1156,0.318,,,,368,1156,0.318,0.318,402,488,0.824,,,,189,,,,192,1138																		
22	20,1950,Price Brookfield,F-G,29,ROC,7,,,,0.592,,0.565,,,,,,,,,0.5,0.1,0.5,,,,,,11,23,0.478,,,,11,23,0.478,0.478,12,13,0.923,,,,1,,,,7,34																		
23	21,1950,Bob Brown,F,26,DNN,62,,,,0.414,,0.33,,,,,,,,,2.1,-0.9,1.2,,,,,,276,764,0.361,,,,276,764,0.361,0.361,172,252,0.683,,,,101,,,,269,724																		
24	22,1950,Jim Browne,C,20,DNN,31,,,,0.392,,0.563,,,,,,,,,0.1,-0.1,0,,,,,,17,48,0.354,,,,17,48,0.354,0.354,13,27,0.481,,,,8,,,,16,47																		
25	23,1950,Walt Budko,PF,24,BLB,66,,,,0.388,,0.403,,,,,,,,,0.7,2,2.7,,,,,,198,652,0.304,,,,198,652,0.304,0.304,199,263,0.757,,,,146,,,,259,595																		
26	24,1950,Jack Burmaster,G,23,SHE,61,,,,0.378,,0.256,,,,,,,,,0.1,-0.4,-0.4,,,,,,237,711,0.333,,,,237,711,0.333,0.333,124,182,0.681,,,,179,,,,237,598																		
27	25,1950,Tommy Byrnes,F-G,26,BLB,53,,,,0.362,,0.312,,,,,,,,,-0.5,0.9,0.3,,,,,,120,397,0.302,,,,120,397,0.302,0.302,87,124,0.702,,,,88,,,,76,327																		
28	26,1950,Bill Calhoun,SG,22,ROC,62,,,,0.439,,0.37,,,,,,,,,3,1.5,4.5,,,,,,207,549,0.377,,,,207,549,0.377,0.377,146,203,0.719,,,,115,,,,100,560																		
29	27,1950,Don Carlson,G-F,30,MNL,57,,,,0.402,,0.328,,,,,,,,,0.7,1.9,2.6,,,,,,99,290,0.341,,,,99,290,0.341,0.341,69,95,0.726,,,,76,,,,126,267																		
30	28,1950,Bob Carpenter,F-C,32,FTW,66,,,,0.421,,0.415,,,,,,,,,2.2,2.8,5,,,,,,212,617,0.344,,,,212,617,0.344,0.344,190,256,0.742,,,,92,,,,168,614																		

Con estos tres archivos se cubren bastantes aspectos de la liga, y se pueden analizar tanto la información de los jugadores como sus estadísticas desde diferentes puntos de vista.

2. Objetivo de la aplicación

Al igual que ocurre en muchos ámbitos de la vida actual, el deporte es un área donde el Big Data, el mundo del Data Science y todo lo relacionado está en pleno auge y expansión. El baloncesto y más concretamente la NBA llevan bastantes años siendo grandes partícipes en este área aplicada al deporte. En la NBA es muy habitual que dentro del organigrama de cada franquicia de la liga haya especialistas en la estadística avanzada utilizando las últimas técnicas de análisis de datos para intentar mejorar el rendimiento individual y colectivo del equipo en pos de conseguir el máximo de ayuda de cara a ganar partidos.

Se puede jugar con todos los datos almacenados a lo largo de la historia de la liga, o de todas las jugadas e información que se guarda de cada partido para analizar diferentes estadísticas de jugadores o equipos en el plano histórico, o analizando las jugadas de cierto jugador ser capaz de medir de distintas maneras su impacto positivo o negativo en el equipo.

En el ejemplo vamos a calcular alguna estadística general, relativamente sencilla, basándonos en el ejemplo del wordcount visto en clase.

Por ejemplo, una vez hemos subido los ficheros con los datos vistos en el primer punto, podríamos aplicar sobre alguno de los ficheros el ejemplo del WordCount. El funcionamiento que utilizaremos para el ejemplo será similar:

Primero compilamos el fichero java: "**./compilar.bash WordCount**".

Una vez compilado el archivo java utilizamos "yarn" para lanzar el programa pasándole el fichero .csv donde están los datos a los que queremos aplicar los algoritmos para analizar los datos y obtener un resultado.
"**yarn jar WordCount.jar uam.WordCount Players.csv WordCount**"

Cuando se termina la ejecución de los programas y se ha obtenido un resultado, se puede navegar por el sistema de ficheros HDFS para comprobar que se creo un directorio con la salida del programa Hadoop, y que se ejecutó correctamente.
"hdfs dfs -ls WordCount"

Accedemos al directorio y mostramos por pantalla el archivo resultado generado, viendo que efectivamente se realizó un wordcount en el fichero pasado a la aplicación.
"hdfs dfs -cat WordCount/part-r-00000"

Al realizar el recuento del WordCount directamente sobre el fichero csv no parece que se obtenga un resultado coherente, ya que el código a lo mejor no estaba preparado para este fichero o uno similar.
El fichero .csv tiene los datos en cada fila separados por "," así que haciendo algunos cambios en el código basado en el ejemplo del WordCount, podríamos por ejemplo hacer un recuento por las universidades que aparecen en el fichero y tener el listado de todas las universidades y el número de veces que aparecen, que sería el número de jugadores que cada universidad ha aportado a lo largo de la historia a la liga.

Modificamos el código de la función Map para leer del fichero y tener en cuenta que el carácter que delimita cada dato es la “,”. Además se indica la posición del dato por el que queremos hacer el recuento.

```
public void map(Object key, Text value, Context context) throws IOException, InterruptedException {
    StringTokenizer itrLines = new StringTokenizer(value.toString(), "\n");
    while (itrLines.hasMoreTokens()){
        StringTokenizer itrFields = new StringTokenizer(itrLines.nextToken(), ",");
        int i = 0;

        while(itrFields.hasMoreTokens()){
            if(i==4){
                word.set(itrFields.nextToken());
                context.write(word, one);
            }
            else{
                itrFields.nextToken();
            }
            i++;
        }
    }
}
```

Al compilar, ejecutar y posteriormente ver el resultado observamos que se ha realizado el recuento por el campo que se quería:

```
Appalachian State University    1
Arizona State University        21
Assumption College      2
Auburn University       16
Auburn University at Montgomery 2
Augsburg College        3
Augusta State University        1
Augustana College (SD)  2
Aurora University       1
Austin Peay State University    5
Averett University      1
Ball State University   2
Barton County Community College 1
Baylor University       11
Belmont Abbey College   1
Belmont University      1
Beloit College  1
Bemidji State University        2
Bethel College  1
Blinn College   1
Boise State University  3
Boston College  18
Boston University       3
Bowling Green State University  15
Bradley University      14
Brigham Young University        19
Brigham Young University Hawaii 1
Brown University        1
Bucknell University     1
Butler University       4
```

3. Estructura de la solución propuesta

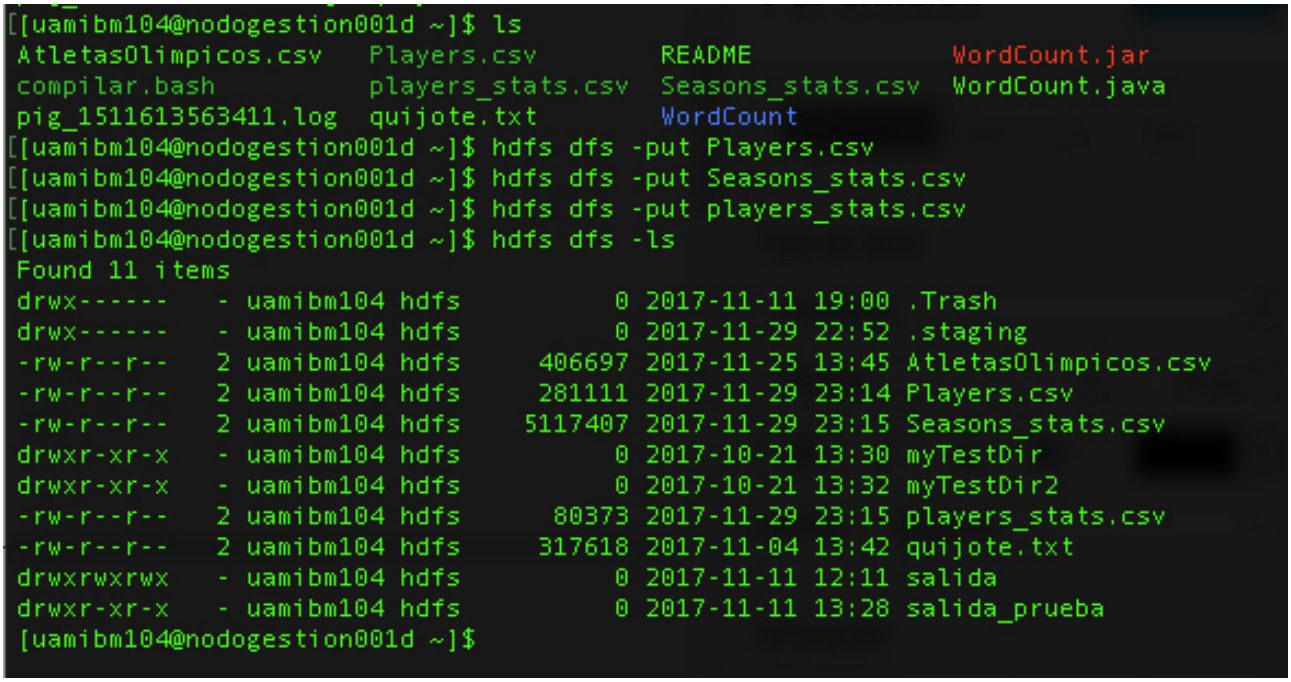
• Cómo subir los datos

Los archivos '.csv' con los datos los copiamos a nuestro home del cluster para trabajar con ellos, mediante el comando 'scp'.

"scp Players.csv umaibm104@150.244.65.33:/home/uamibm104"

Una vez copiados al cluster, listando con 'ls' podemos ver que efectivamente ya están los ficheros en la carpeta del cluster.

Ahora queda copiar los ficheros al sistema de ficheros HDFS de Hadoop, para poder trabajar con ellos en la práctica que vamos a realizar con Hadoop.
Para ello utilizamos con cada uno la sentencia: "**hdfs dfs -put fichero.csv**". Una vez realizada la operación con cada fichero que se requiera, se puede hacer un 'ls' en el hdfs y se podrán ver los ficheros cargados anteriormente.



- **Cómo repartir el trabajo a realizar en Mappers/Reducers**

Vamos a utilizar el fichero “Seasons_stats.csv” donde están las estadísticas de los jugadores de la historia de la liga por cada temporada.
Lo que queremos es obtener la media de puntos de los jugadores de un equipo, en este ejemplo del equipo de Boston.
Para ello se deberá filtrar por el acrónimo de dicho equipo en la columna del fichero que identifica a los equipos en cada registro.

El código de la función mapper sería el de la siguiente imagen. Los datos se especifica que vienen separados por “,”. Se tiene en cuenta que el registro no tenga valores vacíos que puedan hacer fallar el programa o desvirtuar los resultados, y entonces se filtra por equipo para quedarse con los registros que nos interesan.

```
public static class PuntosMapper extends Mapper<LongWritable, Text, Text, DoubleWritable> {
    private Text equipoBos = new Text();

    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String siguienteCampo = value.toString();
        String campos[] = siguienteCampo.split(",");
        String equipo = "BOS";

        if(campos.length==53){
            if(campos[5].equals(equipo)){
                equipoBos.set(campos[5]);
                double stock_volumen = Double.parseDouble(campos[52]);
                context.write(equipoBos, new DoubleWritable(stock_volumen));
            }
        }
    }
}
```

El código de la función reducer sería el de la siguiente imagen. Donde se van acumulando los datos correspondientes a los puntos anotados por cada jugador del equipo de Boston. Por último se realiza la media del recuento.

```
public static class PuntosReducer extends Reducer<Text, DoubleWritable, Text, DoubleWritable> {
    public void reduce(Text key, Iterable<DoubleWritable> values, Context context) throws IOException, Interrupt
        double sum = 0.0;
        int count = 0;

        for (DoubleWritable value : values) {
            sum += value.get();
            count++;
        }

        context.write(key, new DoubleWritable(sum/count));
    }
}
```

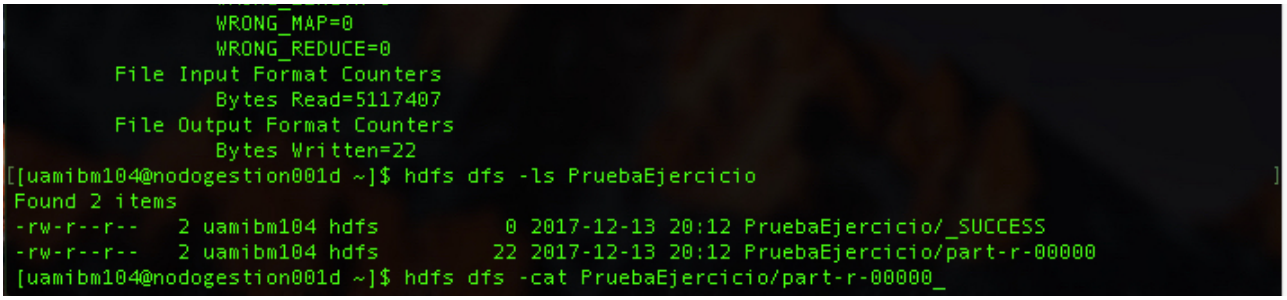
En la función main se realizan las llamadas correspondientes a las funciones y clases para realizar el trabajo map-reduce.

Una vez el código Java está preparado, se copia el archivo al cluster. Se compila para generar el archivo .jar y los class correspondientes:
"**./compilar.bash AnalisisPuntos**"

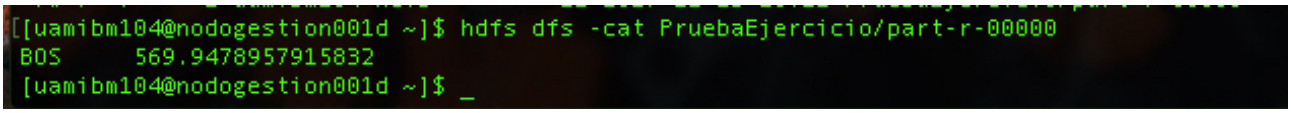
Una vez compilado, con yarn lanzamos el programa:
"**yarn jar AnalisisPuntos.jar uam.AnalisisPuntos Seasons_stats.csv PruebaEjercicio**"

Si la ejecución ha sido correcta, en el sistema HDFS de Hadoop se habrá creado el directorio PruebaEjercicio, y en su interior está el fichero con la salida obtenida con el programa.

Con la sentencia "**hdfs dfs -ls PruebaEjercicio**" obtendremos la salida por pantalla.



La salida será la media del total de puntos obtenidos en una temporada por los jugadores del equipo de Boston.



4. Conclusión y lecciones aprendidas

Con la realización de la práctica se han adquirido o afianzado los siguientes conceptos:

- Se han puesto en práctica los conceptos teóricos vistos en las distintas clases en las que hemos visto el ecosistema Hadoop.

- Se han afianzado los conceptos teóricos y prácticos aprendidos en las clases prácticas de Hadoop con el ejemplo de wordCount.

- Se han aprendido conceptos del desarrollo con Java, tanto en Hadoop como en general.

- Se ha aprendido donde y como conseguir buenos datasets con multitud de datos y de distinta índole, como cargarlos y como trabajar con ellos, lo que se supone que es un aprendizaje vital para el trabajo del día a día.

- Se han cogido conocimientos para ver que se puede realizar con Hadoop sobre unos datos, aunque sea a un nivel bajo, pero es un primer contacto que puede dar pie a profundizar en el tema y elaborar programas más complejos que sean capaces de obtener mayores resultados sobre un grupo de datos.