PRINCIPALES CONCLUSIONES DEL PROYECTO BIG DATA PROCESSING – KEEPCODING

¿En qué consiste el procesamiento de datos dentro del mundo de Big Data y qué herramientas tenemos para poder llevar a cabo este trabajo?

- **Roberto Alagia**

En este artículo se presentan las conclusiones a las que se han llegado, al igual que la forma en que se ha desarrollado la ejecución del análisis de los datos planteados, referente a los países más felices del mundo (world-happiness-report.csv y world-happiness-report-2021.csv).

A pesar de las diferentes alternativas que se han planteado para el procesamiento de datos a lo largo del módulo, se ha decidido desarrollar la solución al problema planteado haciendo uso de Databricks: una plataforma de análisis y procesamiento de datos basada en la nube, diseñada para simplificar el desarrollo de aplicaciones de Big Data.

Algunas de los beneficios de usar esta plataforma han sido:

1. Apache Spark Integrado: Databricks está estrechamente integrado con Apache Spark, lo que permite ejecutar fácilmente trabajos de Spark y aprovechar todas las capacidades de Spark SQL.
2. Administración Automática de Clústeres: Databricks simplifica la administración de clústeres de Spark al proporcionar una gestión automática y escalable. Los clústeres se pueden configurar y aprovisionar fácilmente, y se pueden ajustar dinámicamente según la carga de trabajo (esta característica puede resultar más importante a medida que se impliquen más recursos y envergadura en los trabajos a desarrollar).
3. Escalabilidad: La plataforma ofrece escalabilidad automática para manejar grandes volúmenes de datos y aumentar o disminuir la capacidad de procesamiento según sea necesario.

Las bondades de la plataforma y su simplicidad para empezar a trabajar lo hacen una herramienta atractiva si se compara con una solución instalada en local en tu ordenador, la cual conlleva un tiempo en instalar y preparar el PC para poder empezar a trabajar.

Otro de los puntos importantes que considero una ventaja adicional para el uso, es la estructura del cuaderno y la simplicidad para seguir, sentencia a sentencia, lo que se está haciendo en cada momento (similar a Colab).

Otra de las alternativas que existían para desarrollar la práctica era haciendo uso de Python (PySpark) o Scala. En mi caso, he decidido realizarlo con Scala debido a que es un lenguaje de programación en el que tengo menos experiencia y suponía un esfuerzo extra.

Scala es un lenguaje de programación que combina características de la programación funcional y orientada a objetos. Es un lenguaje que se ejecuta en la máquina virtual de Java (JVM).

Las características principales de Scala son:

1. Orientado a Objetos: En Scala, todo es un objeto. Puedes definir clases, crear instancias de objetos y utilizar herencia.
2. Programación Funcional: Puedes pasar funciones como argumentos, devolver funciones como resultado y almacenarlas en variables.
3. Tipado Estático: Scala es un lenguaje de programación con tipado estático, lo que significa que las variables y expresiones tienen tipos que se verifican en tiempo de compilación.

RESULTADOS DEL PROYECTO

Habiendo explicado la estructura desde la cual se constituye el proyecto, se procede a explicar las conclusiones y los resultados obtenidos.

Se plantean 6 preguntas partiendo de los datasets mencionados al principio del artículo, las cuales son:

1. ¿Cuál es el país más “feliz” del 2021 según la data? (considerar que la columna “Ladder score” mayor número más feliz es el país)
2. ¿Cuál es el país más “feliz” del 2021 por continente según la data?
3. ¿Cuál es el país que más veces ocupó el primer lugar en todos los años?
4. ¿Qué puesto de Felicidad tiene el país con mayor GDP del 2020?
5. ¿En qué porcentaje ha variado a nivel mundial el GDP promedio del 2020 respecto al 2021? ¿Aumentó o disminuyó?
6. ¿Cuál es el país con mayor expectativa de vida (“Healthy life expectancy at birth”)? Y ¿Cuánto tenía en ese indicador en el 2019?

Para responder a ellas, lo primero que se ha hecho es importar la libería de SparkSession y crear la sesión de Spark:

Crear una sesión de spark

```
1  val spark = SparkSession.builder()
2        .appName("Happiest Country")
3        .master("local[*]")
4        .getOrCreate()
```

```
spark: org.apache.spark.sql.SparkSession = org.apache.spark.sql.SparkSession@2a2cfc3a
```

Luego, a partir de la ruta donde está almacenada la información se crea el Data Frame asociado al mismo:

Almacenar ruta de archivo en una variable para simplificar código

```
1  val csvFile = "dbfs:/FileStore/practica/world_happiness_report_2021.csv"
```

```
csvFile: String = dbfs:/FileStore/practica/world_happiness_report_2021.csv
```

```
Command took 1.83 seconds -- by r.alagia@ragautomation.com at 6/3/2024, 19:54:28 on Cluster_KC
```

Cmd 4

Crear el data frame (df) a partir del csv de 2021

```
1  import org.apache.spark.sql.{SparkSession, DataFrame}
2  val df: DataFrame = spark.read
3         .option("header", "true")
4         .option("inferSchema", "true")
5         .csv(csvFile)
```

Esto se replicó también para crear un Data Frame para el segundo archivo.

A continuación, se muestran extractos de las soluciones:

1. ¿Cuál es el país más “feliz” del 2021 según la data? (considerar que la columna “Ladder score” mayor número más feliz es el país)

1. ¿Cuál es el país más “feliz” del 2021 según la data?

```
1  import org.apache.spark.sql.functions.{desc,col, sum, sum_distinct, avg, count}
2
3  val dfHappiestCountry21 = df.groupBy("Country name","Ladder score")
4      .count
5      .orderBy(desc("Ladder score"))
6      .show(1)
```

▸ (2) Spark Jobs

```
+------------+------------+-----+
|Country name|Ladder score|count|
+------------+------------+-----+
|     Finland|       7.842|    1|
+------------+------------+-----+
only showing top 1 row
```

2. ¿Cuál es el país más “feliz” del 2021 por continente según la data?

```
import org.apache.spark.sql.expressions.Window
import org.apache.spark.sql.functions.{row_number,desc,col, sum, sum_distinct, avg, count}

val dfWithRowNumber = df.withColumn("row_number", row_number().over(Window.partitionBy("Regional indicator").orderBy(desc("Ladder score"))))

val result = dfWithRowNumber
.select("Country name","Regional indicator","Ladder score")
.orderBy(desc("Ladder score"))
.filter(col("row_number") === 1)
.show()
```

(2) Spark Jobs

dfWithRowNumber: org.apache.spark.sql.DataFrame = [Country name: string, Regional indicator: string ... 19 more fields]

Country name	Regional indicator	Ladder score
Finland	Western Europe	7.842
New Zealand	North America and...	7.277
Israel	Middle East and N...	7.157
Costa Rica	Latin America and...	7.069
Czech Republic	Central and Easte...	6.965
Taiwan Province o...	East Asia	6.584
Singapore	Southeast Asia	6.377
Uzbekistan	Commonwealth of I...	6.179
Mauritius	Sub-Saharan Africa	6.049
Nepal	South Asia	5.269

3. ¿Cuál es el país que más veces ocupó el primer lugar en todos los años?

3. ¿Cuál es el país que más veces ocupó el primer lugar en todos los años?

```
val happiestCountryName = mostFrequentCountry.select("Country name","Championships").show(1)
```

(3) Spark Jobs

Country name	Championships
Denmark	7

only showing top 1 row

Para llegar a estas conclusiones, se hizo un filtrado de los campeones de cada año:

Crear resultado de ganador en cada año

```
import org.apache.spark.sql.expressions.Window
import org.apache.spark.sql.functions.{row_number,desc,col, sum, sum_distinct, avg, count}

val dfWithRowNumber = df.withColumn("row_number", row_number().over(Window.partitionBy("year").orderBy(desc("Life Ladder"))))

val result = dfWithRowNumber
.select("Country name","year","Life Ladder")
.orderBy(desc("Life Ladder"))
.filter(col("row_number") === 1)

val dfResult: DataFrame = result.toDF()
dfResult.show()
```

Y posteriormente, se creó el ranking de los países campeones:

Crear ranking de paises más felices (histórico)

```
val mostFrequentCountry = dfResult
                          .groupBy("Country name")
                          .agg(count("*").alias("Championships"))
                          .orderBy(col("Championships").desc)

val happiestCountryNameRank = mostFrequentCountry.select("Country name","Championships").show()
```

(3) Spark Jobs

mostFrequentCountry: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Country name: string, Championships: long]

```
+------------+-------------+
|Country name|Championships|
+------------+-------------+
|     Denmark|            7|
|     Finland|            6|
|      Norway|            1|
| Switzerland|            1|
|      Canada|            1|
+------------+-------------+
```

4. ¿Qué puesto de Felicidad tiene el país con mayor GDP del 2020?

El país con mayor GDP en el 2020 es Irlanda y está en el puesto 13 de países más felices según la data.

4. ¿Qué puesto de Felicidad tiene el país con mayor GDP del 2020?

```
val dfWithRowNumber = dfRanking2020.withColumn("ranking_number", row_number().over(Window.orderBy(desc("Life Ladder"))))

val result = dfWithRowNumber
.select("ranking_number","Country name","Life Ladder","Log GDP per capita","year")
.orderBy(desc("Log GDP per capita"))

val dfResult: DataFrame = result.toDF()
dfResult.show(1)
```

(4) Spark Jobs

dfWithRowNumber: org.apache.spark.sql.DataFrame = [Country name: string, Life Ladder: double ... 3 more fields]

result: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [ranking_number: integer, Country name: string ... 3 more fields]

dfResult: org.apache.spark.sql.DataFrame = [ranking_number: integer, Country name: string ... 3 more fields]

```
+--------------+------------+-----------+------------------+----+
|ranking_number|Country name|Life Ladder|Log GDP per capita|year|
+--------------+------------+-----------+------------------+----+
|            13|     Ireland|      7.035|            11.323|2020|
+--------------+------------+-----------+------------------+----+
```

5. ¿En qué porcentaje ha variado a nivel mundial el GDP promedio del 2020 respecto al 2021? ¿Aumentó o disminuyó?

El GDP promedio disminuyó con respecto al 2020.

```
+----+-----------------+
|year|     GDP per year|
+----+-----------------+
|2020|9.751329545454546|
+----+-----------------+

+----------------+
|    GDP per year|
+----------------+
|9.36845269210662|
+----------------+
```

Para llegar a estas conclusiones, primero se calculó el promedio de cada año, de la siguiente manera:

5.1 GDP en el año 2020

```scala
import org.apache.spark.sql.expressions.Window
import org.apache.spark.sql.functions.{row_number,desc,col, sum, sum_distinct, avg, count}

val ranking2020_2021 = df
                       .groupBy("year")
                       .agg(avg(col("Log GDP per capita")).as("GDP per year"))
                       .orderBy(desc("GDP per year"))
                       .where(col("year") === "2020")

val dfRanking2020_2021: DataFrame = ranking2020_2021.toDF()
dfRanking2020_2021.show()
```

▸ (2) Spark Jobs

▸ ranking2020_2021: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [year: integer, GDP per year: double]

▸ dfRanking2020_2021: org.apache.spark.sql.DataFrame = [year: integer, GDP per year: double]

```
+----+-----------------+
|year|     GDP per year|
+----+-----------------+
|2020|9.751329545454546|
+----+-----------------+
```

6. ¿Cuál es el país con mayor expectativa de vida (“Healthy life expectancy at birth”)? Y ¿Cuánto tenía en ese indicador en el 2019?

El país con mayor expectativa de vida para el año más actualizado dentro del dataset histórico (2020) fue Japón:

6.1 ¿Cuál es el país con mayor expectativa de vide (“Healthy life expectancy at birth”)?

```
import org.apache.spark.sql.expressions.Window
import org.apache.spark.sql.functions.{row_number,desc,col, sum, sum_distinct, avg, count}

val rankingLife2020 = df
                      .select("Country name","Healthy life expectancy at birth","year")
                      .orderBy(desc("Healthy life expectancy at birth"))
                      .where(col("year") === "2020")

val dfRankingLife2020: DataFrame = rankingLife2020.toDF()
dfRankingLife2020.show(1)
```

```
▸ (1) Spark Jobs
▸ rankingLife2020:  org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Country name: string, Healthy life expectancy at birth: double ... 1 more field]
▸ dfRankingLife2020:  org.apache.spark.sql.DataFrame = [Country name: string, Healthy life expectancy at birth: double ... 1 more field]
+------------+--------------------------------+----+
|Country name|Healthy life expectancy at birth|year|
+------------+--------------------------------+----+
|       Japan|                            75.2|2020|
+------------+--------------------------------+----+
```

Sabiendo esta información, se calcula para el 2019 su valor:

6.2 Y ¿Cuánto tenia en ese indicador en el 2019?

```
import org.apache.spark.sql.expressions.Window
import org.apache.spark.sql.functions.{row_number,desc,col, sum, sum_distinct, avg, count}

val japanLife2019 = df
                    .select("Country name","Healthy life expectancy at birth","year")
                    .orderBy(desc("Healthy life expectancy at birth"))
                    .where(col("year") === "2019" && col("Country name") === "Japan")

val dfJapanLife2019: DataFrame = japanLife2019.toDF()
dfJapanLife2019.show()
```

```
▸ (1) Spark Jobs
▸ japanLife2019:  org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [Country name: string, Healthy life expectancy at birth: double ... 1 more field]
▸ dfJapanLife2019:  org.apache.spark.sql.DataFrame = [Country name: string, Healthy life expectancy at birth: double ... 1 more field]
+------------+--------------------------------+----+
|Country name|Healthy life expectancy at birth|year|
+------------+--------------------------------+----+
|       Japan|                            75.1|2019|
+------------+--------------------------------+----+
```

En caso de querer profundizar en el código de la solución y acceder, se facilita el enlace a Github con la solución completa:

https://github.com/robertoalagia/kpcd_bigdataproc_ra.git

Gracias!