

# Manual: MapReduce con Hadoop

## Índice

### Introducción

Diseño

Dataset

Descripción.

Formato:

Entorno de Trabajo: Cluster Hadoop

Estructura del Proyecto

Desarrollo

1. Implementación con `Context` `Reduce`

2. Implementación con `OutputCollect` `Reduce`

Código

Problemas Comunes y Soluciones

Consultas Implementadas

Ejecución en el Cluster

Repositorio Github.

Bibliografía.

## Introducción

Este manual está diseñado para guiar a los desarrolladores en la implementación de tareas de MapReduce utilizando un cluster de Hadoop. Se enfoca en resolver consultas específicas sobre datos de ventas almacenados en un dataset.

## Diseño

### Dataset

#### Descripción.

Order ID	Product	Quantity Ordered	Price Each	Order Date	Purchase Address
Un ID de pedido es el sistema numérico que Amazon utiliza exclusivamente para realizar un seguimiento de los pedidos. Cada pedido	El producto que se ha vendido.	La cantidad pedida es la cantidad total de artículos pedidos en el pedido inicial (sin cambios).	El precio de cada producto.	Es la fecha en la que el cliente solicita el envío del pedido.	La orden de compra es preparada por el comprador, a menudo a través de un departamento de compras. La orden de

<p>recibe su propio ID de pedido que no se duplicará. Este número puede ser útil para el vendedor cuando intente averiguar ciertos detalles sobre un pedido, como la fecha o el estado del envío.</p>			<p>compra, o PO, suele incluir un número de PO, que es útil para cotejar los envíos con las compras; una fecha de envío; la dirección de facturación; la dirección de envío; y los artículos solicitados, las cantidades y el precio.</p>
---	--	--	---

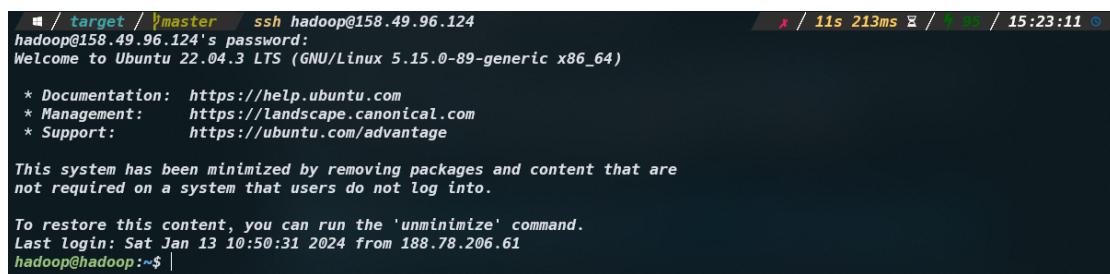
## Formato:

```
Order ID,Product,Quantity Ordered,Price Each,Order Date,Purchase Address
176558,USB-C Charging Cable,2,11.95,04/19/19 08:46,"917 1st St, Dallas, TX 75201, US
```

## Entorno de Trabajo: Cluster Hadoop

### 1. Acceso y uso del cluster Hadoop.

- Para la realización de esta práctica usaremos un nodo del CPD del Centro Universitario de Mérida donde se ha instalado previamente Hadoop. El nodo está provisto con las librerías necesarias para poder ejecutar la práctica bajo el paradigma MapReduce.
- Acceso al nodo.



```
# / target / master ssh hadoop@158.49.96.124
hadoop@158.49.96.124's password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-89-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:    https://landscape.canonical.com
 * Support:       https://ubuntu.com/advantage

This system has been minimized by removing packages and content that are
not required on a system that users do not log into.

To restore this content, you can run the 'unminimize' command.
Last login: Sat Jan 13 10:50:31 2024 from 188.78.206.61
hadoop@hadoop:~$ |
```

### 2. IDE: En este proyecto he hecho uso del IDE IntelliJ.

## Estructura del Proyecto

- Clases Mapper y Reducer:** Cada consulta requiere una clase Mapper y una clase Reducer. Los Mappers procesan los datos de entrada y los Reducers realizan operaciones de resumen.

- **Clase Main:** Coordina la ejecución, estableciendo las clases Mapper y Reducer, y configura las rutas de entrada y salida.

## 1. Diagrama Clases:



# Desarrollo

He desarrollado dos versiones de la práctica.

## 1. Implementación con `Context Reduce`

Además de emitir la salida clave-valores también proporciona funcionalidades adicionales como mostrar el producto que más ventas hay en una colección de datos.

## 2. Implementación con `OutputCollect Reduce`

La implementación con `OutputCollect Reduce` emite la salida clave-valores al igual que con la implementación `Context reduce` pero no proporciona funcionalidades adicionales como controlar el flujo de la tarea, ya sea para mostrar un mensaje antes del procesamiento del **Reduce** o al finalizar.

**NOTA:** Para mostrar mostrar los valores deseados he creado un script bash `max_value_row.sh`, el cual recibe un archivo como parámetro y muestra el mayor valor y su correspondiente clave. Los archivos constan de dos columnas separadas por un espacio en blanco, en todo caso para los archivos que tengan guiones o bien caracteres especiales, dichos caracteres son eliminados para evitar la incorrecta ejecución del script.

```

hadoop@hadoop:~$ hadoop fs -get /output3_Jloeln_2023/ .
hadoop@hadoop:~$ bash max_value_row.sh output3_Jloeln_2023/part*
Boston          22528

```

## Código

<pre> ••• Ingresos2019Mapper package consultas;  import org.apache.hadoop.io.DoubleWritable; import org.apache.hadoop.io.LongWritable; import org.apache.hadoop.io.Text; import org.apache.hadoop.mapreduce.Mapper; import java.io.IOException;  /*  * @author Jose Luis Obiang Ela Nanguang  * @version 1.0  * {@link Mapper}  * @description Esta clase es un Mapper para MapReduce que procesa datos de ventas para calcular las ganancias totales en el año 2019. Esta clase extiende MapReduceBase e implementa la interfaz Mapper proporcionada por Hadoop.  * La clave de entrada es LongWritable, el valor de entrada es Text (una línea de datos), y produce un Text (año) y un DoubleWritable (ganancias totales)  */ public class Ingresos2019Mapper extends Mapper&lt;LongWritable, Text, Text, Double&gt;      @Override     public void map(LongWritable key, Text value, Context context) throws IOException {         // Convierte la linea de texto en una cadena de caracteres         String line = value.toString();         // Divide la linea en campos separados por comas         String[] fields = line.split(",");         // Verifica que la linea tenga suficientes campos         if (fields.length &gt; 5) {             // Extrae la fecha del pedido             String date = fields[4];             // Separa la fecha en sus componentes             String[] dateParts = date.split(" ")[0].split("/");             // Verifica que la fecha esté en el formato correcto             if (dateParts.length == 3) {                 // Extrae el año y lo convierte a formato '0xx'                 String year = "20" + dateParts[2];                 if (year.equals("2019")) { // Si es el año correcto                     try {                         // Calcula el total de ventas, multiplicando la cantidad                         int numVentas = Integer.parseInt(fields[2]); // Extraer                         double precio = Double.parseDouble(fields[3]); // Extraer                         double gananciaPorVenta = numVentas * precio; // Calcula                         // Emite el año y la ganancia por venta                         context.write(new Text(year), new DoubleWritable(gananciaPorVenta));                     } catch (NumberFormatException e) {                         context.setStatus("Error, precio no válido. ");                     } catch (InterruptedException e) {                         throw new RuntimeException(e);                     }                 }             }         }     } } </pre>	<pre> Ingresos2019Reducer package consultas;  import org.apache.hadoop.io.DoubleWritable; import org.apache.hadoop.io.Text; import org.apache.hadoop.mapreduce.Reducer; import java.io.IOException;  /**  * @Description Esta clase Reducer de MapReduce procesa los ingresos totales de ventas del año 2019.  * Extiende MapReduceBase e Implementa la interfaz Reducer de Hadoop,  * Toma Text como clave (año) y DoubleWritable como valor (monto de venta) y produce  * Text (año) y Text (ganancias totales formateadas) como salida.  */ public class Ingresos2019Reducer extends Reducer&lt;Text, DoubleWritable, Text, Text&gt; {      private double maxGanancias = 0.0;     private final Text maxKey = new Text();      @Override     public void reduce(Text key, Iterable&lt;DoubleWritable&gt; values, Context context) throws IOException, InterruptedException {         double gananciasTotales = 0;         // Suma todas las ganancias del año         for (DoubleWritable val : values) {             double ganancia = val.get();             gananciasTotales += ganancia;             if (gananciasTotales &gt; maxGanancias) {                 maxGanancias = gananciasTotales;                 maxKey.set("el año " + key + " se ganó: ");             }         }         // Emite el año y las ganancias totales formateadas         context.write(key, new Text(String.format("%.2f", gananciasTotales)));     }      /**      * Función que se invoca una vez al inicio de la tarea del Reducer,      * antes de procesar cualquier clave/valor.      */     @Override     protected void setup(Context context) throws IOException, InterruptedException {         context.write(new Text(""), new Text("1.Cuánto se ganó en 2019 "));     }      /**      * Se llama al final de la tarea del Reducer, después de procesar todas las claves/valores.      */     @Override     public void cleanup(Context context) throws IOException, InterruptedException {         context.write(maxKey, new Text(String.format("%.2f€", maxGanancias)));     } } </pre>
--	---



```

••• CiudadVentasMapper •••
package consultas;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapreduce.Mapper;

import java.io.IOException;

/**
 * @Description Esta clase Mapper de MapReduce analiza las ventas por ciudad.
 * Extiende MapReduceBase e implementa la interfaz Mapper de Hadoop.
 * Toma LongWritable y Text como entradas (clave y valor respectivamente)
 * produce Text (ciudad) e IntWritable (cantidad de ventas) como salida.
 * @author Jose Luis Obiang Ela Nanguang
 * @version 1.0
 * {@link Mapper}
 */
public class CiudadVentasMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

    private final Text ciudad = new Text();
    private final IntWritable cantidad = new IntWritable();

    @Override
    public void map(LongWritable key, Text value, Context context) throws IOException {
        // Convierte el valor de entrada (una línea de texto) en una cadena de caracteres
        String line = value.toString();
        // Divide la línea usando comas como separador para obtener los campos.
        String[] fields = line.split(",");
        try {
            if (fields.length > 6) { // Campo dirección <calle, ciudad, C-Postal>
                ciudad.set(fields[6].trim() + " "); // Extrae la ciudad
            } else if (fields.length == 6) { // Campo dirección <calle, ciudad;
                ciudad.set(fields[5].split(";")[1].trim()); // Extrae la ciudad
            }

            // Extrae la cantidad vendida (se asume que está en el tercer campo).
            cantidad.set(Integer.parseInt(fields[2]));
            if(ciudad.getLength() <= 7) {
                ciudad.set(ciudad + " ");
                context.write(ciudad, cantidad);
            }
        } catch (NumberFormatException e) {
            context.setStatus("Error al parsear la cantidad: " + line);
        } catch (ArrayIndexOutOfBoundsException e) {
            context.setStatus("Formato de linea incorrecto: " + line);
        }
    }
}

••• CiudadVentasReducer •••
package consultas;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import java.io.IOException;

/**
 * @Description Esta clase Reducer de MapReduce procesa las ventas por ciudad.
 * Extiende MapReduceBase e implementa la interfaz Reducer de Hadoop.
 * Toma Text (ciudad) y IntWritable (cantidad de ventas) como entradas
 * y produce Text (ciudad) e IntWritable (ventas totales) como salida.
 * Suma todas las ventas por ciudad.
 * @author Jose Luis Obiang Ela Nanguang
 * @version 1.0
 * {@link Reducer}
 */
public class CiudadVentasReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

    private int maxVentas = 0;
    private final Text maxKey = new Text();

    @Override
    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {
        int totalVentas = 0;
        // Suma las cantidades de ventas para cada ciudad. Cada valor representa la transacción.
        for (IntWritable val : values) {
            int ventas = val.get();
            totalVentas += ventas;
            if (totalVentas > maxVentas) {
                maxVentas = totalVentas;
                maxKey.set("La ciudad de " + key + " tiene el mayor número de ventas");
            }
        }
        // Emite la ciudad (key) y el total acumulado de ventas (totalVentas) como una tupla.
        context.write(key, new Text(totalVentas + ""));
    }

    public void setup(Context context) throws IOException, InterruptedException {
        context.write(new Text(""), new Text("3.¿Qué ciudad tuvo el mayor número de ventas?"));
    }

    @Override
    public void cleanup(Context context) throws IOException, InterruptedException {
        context.write(maxKey, new Text(maxVentas + " ventas"));
    }
}

```



```

●●● HoraPublicidadMapper
package consultas;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;

/**
 * @Description Esta clase Mapper de MapReduce analiza la hora de los pedidos para publicidad.
 * Extiende MapReduceBase e implementa la interfaz Mapper de Hadoop.
 * Toma LongWritable y Text como entradas (clave y valor respectivamente)
 * Text (hora) e IntWritable (contador) como salida.
 * @author Jose Luis Obiang Ela Nanguang
 * @version 1.0
 * {@link Mapper}
 */
public class HoraPublicidadMapper extends Mapper<LongWritable, Text, Text, IntWritable> {
    private final static IntWritable one = new IntWritable(1);
    private final Text hora = new Text();

    @Override
    public void map(LongWritable key, Text value, Context context){
        String[] parts = value.toString().split(",");
        if (parts.length > 5) {
            String fechaHora = parts[4]; // Extrae la fecha y hora
            try {
                // Extrae la hora
                String[] horaMinuto = fechaHora.split(" ")[1].split(":");
                if (!horaMinuto[0].equalsIgnoreCase("date")) { // Si no es un formato de hora.
                    int horaInt = Integer.parseInt(horaMinuto[0]);
                    int temp = horaInt;
                    String amPm = horaInt >= 12 ? "PM" : "AM";
                    // Convertir formato de 24 a 12 horas
                    horaInt = horaInt > 12 ? horaInt - 12 : (horaInt == 0 ? 12 : horaInt);
                    if(horaInt == 1 || horaInt == 2 || horaInt == 3 || horaInt == 4 || horaInt == 5 || horaInt == 6 || horaInt == 7 || horaInt == 8 || horaInt == 9){
                        hora.set(horaInt + "(" + temp + ")" + amPm + " ");
                    } else{
                        hora.set(horaInt + "(" + temp + ")" + amPm);
                    }
                    context.write(hora, one);
                }
            } catch (Exception e) {
                context.setStatus("Error al parsear la hora del pedido: " + fechaHora);
            }
        }
    }
}

●●● HoraPublicidadReducer
package consultas;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;

import java.io.IOException;
import java.util.Iterator;

/**
 * @Description Esta clase es un Reducer para MapReduce que procesa datos de pedidos para calcular la suma de pedidos realizados en cada hora. La clase extiende MapReduceBase e implementa la interfaz Reducer proporcionada por Hadoop.
 * La clave de entrada es Text (hora), y el valor de entrada es IntWritable (sumatoria de pedidos).
 * Produce un Text (hora) y un Text (suma de pedidos) como salida.
 * @author Jose Luis Obiang Ela Nanguang
 * @version 1.0
 * {@link Reducer}
 */
public class HoraPublicidadReducer extends Reducer<Text, IntWritable, Text, IntWritable> {
    private int maxPedidosHora = 0;
    private final Text maxKey = new Text();

    @Override
    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException {
        int totalPedidosPorHora = 0;
        for (IntWritable val : values) {
            int ventas = val.get();
            totalPedidosPorHora += ventas;
            if (totalPedidosPorHora > maxPedidosHora) {
                maxPedidosHora = totalPedidosPorHora;
                maxKey.set("La hora para maximizar la probabilidad de que el cliente " +
                           "\nnya que en esa hora se hizo un total de " + totalPedidosPorHora);
            }
        }
        // Emite la hora y el total de pedidos en esa hora
        context.write(key, new Text(totalPedidosPorHora + ""));
    }

    @Override
    public void setup(Context context) throws IOException, InterruptedException {
        context.write(new Text(""), new Text("4. ¿A qué hora debemos mostrar publicidad de que el cliente compra el producto?"));
    }

    @Override
    public void cleanup(Context context) throws IOException, InterruptedException {
        context.write(maxKey, new Text(maxPedidosHora + " pedidos."));
    }
}

```



```

package consultas;  MejorMesReducer
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import java.io.IOException;

/**
 * @Description Clase Reducer para determinar el mejor mes en términos de ganancias. Es
 * MapReduceBase e implementa la interfaz Reducer de Hadoop. Su tarea es sumar las ganancias
 * totales de cada mes y emitir el mes junto con las ganancias totales formateadas.
 * @author Jose Luis Obiang Ela Nanguang
 * @version 1.0
 * {@link Reducer}
 */
public class MejorMesReducer extends Reducer<Text, DoubleWritable, Text, Text> {

    private double maxGananciasPorMes = 0.0;
    private final Text maxKey = new Text();

    @Override
    public void reduce(Text key, Iterable<DoubleWritable> values, Context context) throws
    InterruptedException {
        double gananciasTotalesPorMes = 0.0;
        // Suma las ganancias totales por mes
        for (DoubleWritable val : values) {
            double gananciasPorMes = val.get();
            gananciasTotalesPorMes += gananciasPorMes;
            if (gananciasTotalesPorMes > maxGananciasPorMes) {
                maxGananciasPorMes = gananciasTotalesPorMes;
                maxKey.set("El mes de " + key + " es el mejor mes ya que tiene el mayor
                total de:");
            }
        }
        // Formatea las ganancias totales a dos decimales
        String formattedSum = String.format("%.2f", gananciasTotalesPorMes);
        // Emite el mes y las ganancias totales
        context.write(key, new Text(formattedSum + "\n-----"));
    }

    @Override
    public void setup(Context context) throws IOException, InterruptedException {
        context.write(new Text(""), new Text("¿Cuál fue el mejor mes para las ventas"));
    }

    @Override
    public void cleanup(Context context) throws IOException, InterruptedException {
        context.write(maxKey, new Text(String.format("%.2f", maxGananciasPorMes)));
    }
}

● ● ●
package consultas;
import org.apache.hadoop.io.DoubleWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

/**
 * @description Clase Mapper para el análisis de ventas por mes. Esta clase extiende
 * MapReduceBase e implementa la interfaz Mapper de Hadoop. Su función principal es procesar
 * los títulos de texto (registros de ventas) y emitir el mes y el año como clave y el monto total
 * de ventas como valor.
 * @author Jose Luis Obiang Ela Nanguang
 * @version 1.0
 * {@link Mapper}
 */
public class VentasPorMesMapper extends Mapper<LongWritable, Text, Text, DoubleWritable> {

    private static final Map<String, String> meses = new HashMap<>();
    static {
        // Inicializa el mapa con los nombres de los meses.
        meses.put("01", "Enero");
        meses.put("02", "Febrero");
        meses.put("03", "Marzo");
        meses.put("04", "Abril");
        meses.put("05", "Mayo");
        meses.put("06", "Junio");
        meses.put("07", "Julio");
        meses.put("08", "Agosto");
        meses.put("09", "Septiembre");
        meses.put("10", "Octubre");
        meses.put("11", "Noviembre");
        meses.put("12", "Diciembre");
    }

    @Override
    public void map(LongWritable key, Text value, Context context) {
        // Divide la línea de entrada y verifica que tenga al menos 6 campos.
        String[] fields = value.toString().split(",");
        if (fields.length > 5) {
            // Extrae la fecha para obtener el mes y el año.
            String date = fields[4];
            String[] dateParts = date.split("-|");
            if (dateParts.length == 3) {
                // Formatea la fecha para el nombre del mes y el año.
                int numVentas = Integer.parseInt(fields[2]); // Extraer la cantidad de ventas
                double precio = Double.parseDouble(fields[3]); // Extraer el precio
                double gananciaPorVenta = numVentas * precio; // Calcula las ganancias por venta
                // Formatea el total de ventas multiplicando la cantidad de ventas por precio
                if ((dateParts[0].equals("04") || dateParts[0].equals("08") || dateParts[0].equals("05") || dateParts[0].equals("09")) && (dateParts[1].equals("02") || dateParts[1].equals("06") || dateParts[1].equals("07") || dateParts[1].equals("03")) && (dateParts[2].equals("01") || dateParts[2].equals("05") || dateParts[2].equals("08") || dateParts[2].equals("12")) {
                    context.writeValue(Text(monthYear + " " + ttt), new DoubleWritable(gananciaPorVenta));
                } else {
                    context.writeValue(Text(monthYear + " " + ttt), new DoubleWritable(gananciaPorVenta));
                }
            } catch (NumberFormatException e) {
                // Maneja errores si las ventas no son numéricas.
                context.setStatus("Error al procesar la linea: " + line + ". Asegúrate de que la cantidad y el
                precio sean numéricos.");
            } catch (InterruptedException | IOException e) {
                throw new RuntimeException(e);
            }
        }
    }
}

```



```

●●● ProductoMasVendidoMapper
●●● ProductoMasVendidoReducer

package consultas;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Mapper;
import java.io.IOException;

/**
 * @Description Clase Mapper para identificar el producto más vendido en el archivo de entrada y emitir el nombre del producto y la cantidad vendida.
 * @author Jose Luis Obiang Ela Nanguang
 * @version 1.0
 */
public class ProductoMasVendidoMapper extends Mapper<LongWritable, Text, Text, IntWritable> {

    @Override
    public void map(LongWritable key, Text value, Context context) throws IOException, InterruptedException {
        String line = value.toString();
        String[] fields = line.split(",");
        if (fields.length > 3) {
            try {
                // Extrae el nombre del producto y la cantidad vendida
                String nombreProducto = fields[1];
                int cantidadVentas = Integer.parseInt(fields[2]);
                // Emite el nombre del producto y la cantidad vendida
                if (nombreProducto.length() < 24) {
                    context.write(new Text(nombreProducto + " " + cantidadVentas));
                } else {
                    context.write(new Text(nombreProducto), new IntWritable(cantidadVentas));
                }
            } catch (NumberFormatException e) {
                context.setStatus("Error al parsear la cantidad");
            } catch (ArrayIndexOutOfBoundsException e) {
                context.setStatus("Formato de línea incorrecto");
            } catch (InterruptedException e) {
                throw new RuntimeException(e);
            }
        }
    }
}

package consultas;

import org.apache.hadoop.io.IntWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.mapreduce.Reducer;
import java.io.IOException;

/**
 * @Description Clase Reducer para la consulta del producto más vendido. Extiende MapReduceReducer y implementa la interfaz Reducer de Hadoop. Esta clase suma la cantidad total vendida para cada producto y emite el nombre del producto y la cantidad total vendida como un par clave-valor.
 * @author Jose Luis Obiang Ela Nanguang
 * @version 1.0
 * {@link Reducer}
 */
public class ProductoMasVendidoReducer extends Reducer<Text, IntWritable, Text, IntWritable> {

    private int maxVentas = 0;
    private final Text maxKey = new Text();

    @Override
    public void reduce(Text key, Iterable<IntWritable> values, Context context) throws IOException, InterruptedException {
        int totalOrden = 0;
        // Suma las cantidades de cada producto
        for (IntWritable val : values) {
            totalOrden += val.get();
        }
        if (totalOrden > maxVentas) {
            maxVentas = totalOrden;
            maxKey.set("El producto " + key + " tiene el mayor número de ventas, ya que se vendieron " + totalOrden + " unidades con un total de: " + maxVentas);
        }
        // Emite el nombre del producto y la cantidad total vendida
        context.write(key, new Text(totalOrden + ""));
    }

    @Override
    public void setup(Context context) throws IOException, InterruptedException {
        context.write(new Text(""), new Text("5. ¿Qué producto vendió más? Por qué crees que es así?"));
    }

    @Override
    public void cleanup(Context context) throws IOException, InterruptedException {
        context.write(maxKey, new Text(String.format("maxVentas = %d", maxVentas)));
    }
}

```



## Problemas Comunes y Soluciones

- Obtención de ciudad:** Todos los campos en los csv del dataset descargados del campus virtual están separados por ‘,’, pero en el cluster de Hadoop no solo están separados por comas sino también por ‘;’ como separador de los campos de la dirección, en donde obtenemos la ciudad.
- Tipos de Datos Incorrectos:** Verifica los tipos de datos al mapear y reducir. Usa `IntWritable`, `DoubleWritable`, etc., según sea necesario.

Por ejemplo: El error

```
java.io.IOException: Type mismatch in key from map: expected org.apache.hadoop.io.LongWritable,
received org.apache.hadoop.io.Text
```

indica un problema con los tipos de datos que se están pasando entre el Mapper y el Reducer en tu trabajo MapReduce. Este error suele ocurrir cuando hay una discrepancia entre los tipos de salida definidos en el Mapper y los tipos de entrada esperados en el Reducer.

- Rutas de Entrada/Salida en HDFS:** Verifica que las rutas de entrada y salida estén configuradas correctamente en HDFS.

- Verifica la Ruta de Entrada:** Asegúrate de que la ruta de entrada que proporcionas (`/input/practica2/`) exista en HDFS. Puedes verificar esto con el comando `hadoop fs -ls /input/practica2/`.
  - Corrige la Ruta de Salida:** En la ejecución del jar no debe existir la ruta. La ruta se crea durante la ejecución del jar.
  - Verifica los Datos:** Asegúrate de que los datos que deseas procesar están correctamente subidos a la ruta de entrada en HDFS.
- Uso de Context vs OutputCollector:** `Context` se usa en las versiones más recientes de Hadoop y nos permite añadir un mensaje al inicio de la tarea de Reducer antes de procesar cualquier clave/valor o al finalizar la tarea Reducer después de procesar todas las claves/valores mientras que `OutputCollector` se utiliza en versiones más antiguas únicamente para procesar clave/valores.
  - Etiqueta build (`pom.xml`):**

**NOTA:** En el plugin `compiler` si no se añade la versión aparece el mensaje de error `It is highly recommended to fix these problems because they threaten the stability of your build. For this reason, future Maven versions might no longer support building such malformed projects.`, lo cual sugiere que falta la versión del plugin `maven-compiler-plugin` en el archivo `pom.xml`. Para resolver esto, se debe especificar la versión del plugin en la sección de plugins de el archivo `pom.xml`.

En IntelliJ no hace falta añadir el plugin `compiler`.

```
<build>
  <plugins>
    <plugin>
      <groupId>org.apache.maven.plugins</groupId>
      <artifactId>maven-compiler-plugin</artifactId>
      <version>3.8.1</version> <!-- Puedes cambiarla -->
      <configuration>
        <encoding>${project.build.sourceEncoding}</encoding>
        <source>1.7</source>
        <target>1.7</target>
      </configuration>
    </plugin>

    <!-- Plugin para generar el jar con todas las dependencias -->
    <plugin>
      <artifactId>maven-assembly-plugin</artifactId>
      <executions>
        <execution>
```

```

<phase>package</phase>
<goals>
    <goal>single</goal>
</goals>
</execution>
</executions>
<configuration>
    <descriptorRefs>
        <descriptorRef>jar-with-dependencies</descriptorRef>
    </descriptorRefs>
</configuration>
</plugin>

</plugins>

</build>

```

No es necesario añadir el plugin `assembly` si no se ha añadido nuevas librerías.

## Dependencias.

```

<dependencies>
    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-common</artifactId>
        <version>3.3.6</version>
    </dependency>

    <dependency>
        <groupId>org.apache.hadoop</groupId>
        <artifactId>hadoop-client</artifactId>
        <version>3.3.6</version>
    </dependency>

    <!--Para los colores en consola -->
    <dependency>
        <groupId>org.fusesource.jansi</groupId>
        <artifactId>jansi</artifactId>
        <version>2.4.0</version>
    </dependency>
</dependencies>

```

- Para los colores en consola he hecho uso del plugin porque sino se produce el siguiente error:

```
hadoop@hadoop:~$ hadoop jar mapReduce-1.0-SNAPSHOT_Jloeln_2023V2.jar Main /input/practica2 /output
Exception in thread "main" java.lang.ClassNotFoundException: Main
        at java.base/java.net.URLClassLoader.findClass(URLClassLoader.java:476)
        at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:594)
        at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:527)
        at java.base/java.lang.Class.forName0(Native Method)
        at java.base/java.lang.Class.forName(Class.java:398)
        at org.apache.hadoop.util.RunJar.run(RunJar.java:316)
        at org.apache.hadoop.util.RunJar.main(RunJar.java:236)
```

## Consultas Implementadas

### 1. ¿Cuánto se ganó en 2019?

**Ganancias Totales en 2019:** Usa un Mapper para filtrar y calcular ganancias por línea, y un Reducer para sumarlas.

### 2. ¿Cuál fue el mejor mes para las ventas? ¿Cuánto se ganó ese mes?

**Mejor Mes para Ventas:** Implementa un Mapper para calcular las ventas por mes y un Reducer para sumarlas.

### 3. ¿Qué ciudad tuvo el mayor número de ventas?

**Ciudad con Mayor Número de Ventas:** Usa un Mapper para contar ventas por ciudad y un Reducer para sumarlas.

### 4. ¿A qué hora debemos mostrar publicidad para maximizar la probabilidad de que el cliente compre el producto?

#### Hora Óptima para Publicidad

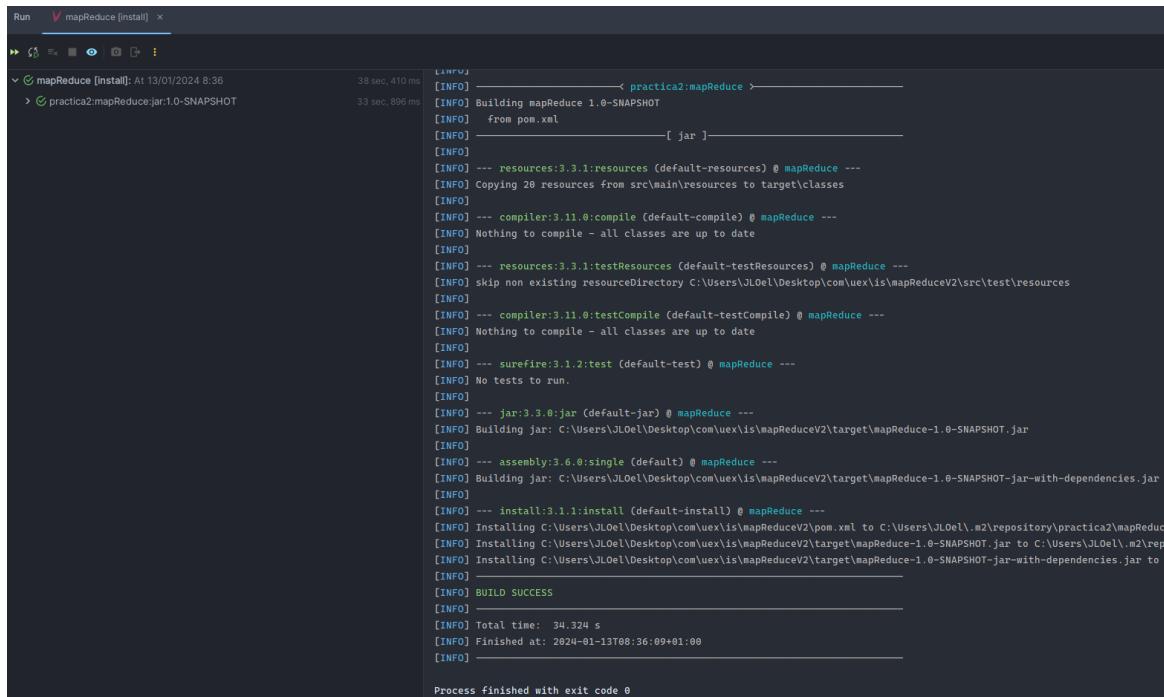
: Esta consulta se enfoca en determinar la hora del día con mayor número de ventas. Se utiliza un Mapper para extraer y contar pedidos por hora y un Reducer para sumar estas cantidades, identificando así el período más activo para las ventas.

### 5. ¿Qué producto vendió más? ¿Por qué crees que vendió más?

**Producto Más Vendido:** Se implementa un Mapper para identificar y contar cada producto vendido, y un Reducer para sumar las ventas de cada producto. Esta consulta ayuda a determinar cuál producto ha sido el más popular en términos de unidades vendidas.

## Ejecución en el Cluster

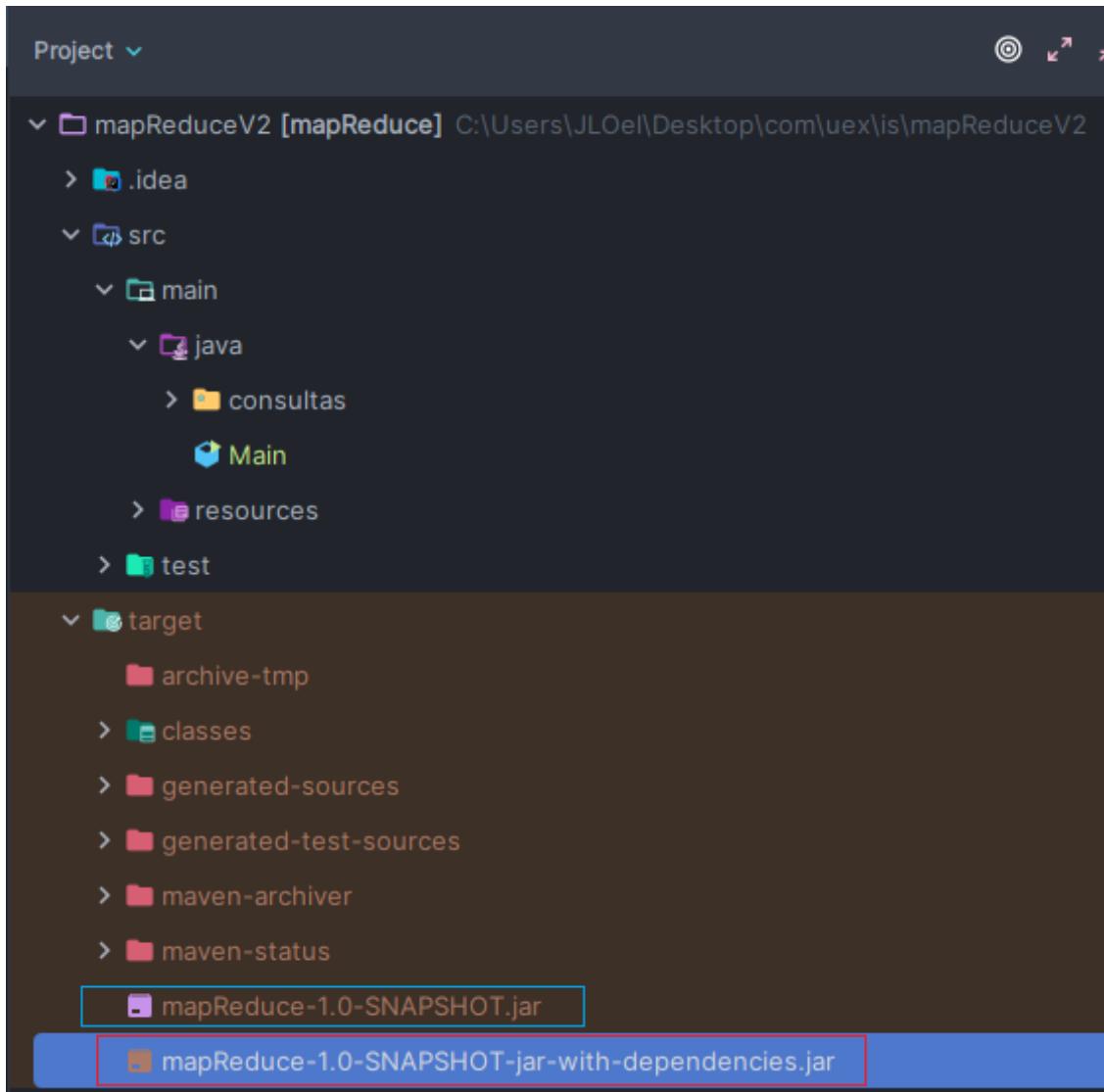
### 1. Compilación y creación del Jar.



```
[INFO] _____ < practica2:mapReduce > _____
[INFO] Building mapReduce 1.0-SNAPSHOT
[INFO] from pom.xml
[INFO] [ jar ]
[INFO]
[INFO] --- resources:3.3.1:resources (default-resources) @ mapReduce ---
[INFO] Copying 20 resources from src\main\resources to target\classes
[INFO]
[INFO] --- compiler:3.11.0:compile (default-compile) @ mapReduce ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- resources:3.3.1:testResources (default-testResources) @ mapReduce ---
[INFO] skip non existing resourceDirectory C:\Users\JLoel\Desktop\com\uex\is\mapReduceV2\src\test\resources
[INFO]
[INFO] --- compiler:3.11.0:testCompile (default-testCompile) @ mapReduce ---
[INFO] Nothing to compile - all classes are up to date
[INFO]
[INFO] --- surefire:3.1.2:test (default-test) @ mapReduce ---
[INFO] No tests to run.
[INFO]
[INFO] --- jar:3.3.0:jar (default-jar) @ mapReduce ---
[INFO] Building jar: C:\Users\JLoel\Desktop\com\uex\is\mapReduceV2\target\mapReduce-1.0-SNAPSHOT.jar
[INFO]
[INFO] --- assembly:3.6.0:single (default) @ mapReduce ---
[INFO] Building jar: C:\Users\JLoel\Desktop\com\uex\is\mapReduceV2\target\mapReduce-1.0-SNAPSHOT-jar-with-dependencies.jar
[INFO]
[INFO] --- install:3.1.1:install (default-install) @ mapReduce ---
[INFO] Installing C:\Users\JLoel\Desktop\com\uex\is\mapReduceV2\pom.xml to C:\Users\JLoel\.m2\repository\practica2\mapReduce\1.0-SNAPSHOT\mapReduce-1.0-SNAPSHOT.pom
[INFO] Installing C:\Users\JLoel\Desktop\com\uex\is\mapReduceV2\target\mapReduce-1.0-SNAPSHOT.jar to C:\Users\JLoel\.m2\repository\practica2\mapReduce\1.0-SNAPSHOT\mapReduce-1.0-SNAPSHOT.jar
[INFO] Installing C:\Users\JLoel\Desktop\com\uex\is\mapReduceV2\target\mapReduce-1.0-SNAPSHOT-jar-with-dependencies.jar to C:\Users\JLoel\.m2\repository\practica2\mapReduce\1.0-SNAPSHOT\mapReduce-1.0-SNAPSHOT-jar-with-dependencies.jar
[INFO]
[INFO] BUILD SUCCESS
[INFO]
[INFO] Total time:  34.324 s
[INFO] Finished at: 2024-01-13T08:36:09+01:00
[INFO]
```

Process finished with exit code 0

- Jar Generado:



2. **Ejecución:** Sube el paquete JAR a tu cluster Hadoop y ejecútalo utilizando el comando `hadoop jar`.

- Envío de datos de nuestro equipo al nodo Hadoop.

```
psh::target * + ~
/ target / master scp .\mapReduce-1.0-SNAPSHOT-jar-with-dependencies_Jloeln_2023V2.jar hadoop@158.49.96.124:/home/hadoop
hadoop@158.49.96.124's password:
mapReduce-1.0-SNAPSHOT-jar-with-dependencies_Jloeln_2023V2.jar
21% 13MB 4.8MB/s 00:09 ETA
```

- Comprobar que se ha hecho correctamente el envío del jar al nodo Hadoop.

```
hadoop@hadoop:~$ ls -l mapReduce-1.0-SNAPSHOT_Jloeln_2023V2.jar
-rw-rw-r-- 1 hadoop hadoop 58056341 Jan 13 07:27 mapReduce-1.0-SNAPSHOT_Jloeln_2023V2.jar
```

- Como podemos ver se lanza el trabajo a través de archivo `.jar` que hemos subido al nodo, indicando el paquete que contiene la clase principal y la propia clase. Como argumentos del programa tenemos la ruta donde se encuentran los ficheros a procesar y la ruta donde dejaremos la salida del procesamiento.

NOTA: Los archivos .jar generados en un proyecto Maven en el IDE IntelliJ permiten ejecutarlos en el cluster de Hadoop sin necesidad de indicar el paquete de la clase, únicamente el nombre de la clase ejecutable.

```

hadoop@hadoop:~$ hadoop jar mapReduce-1.0-SNAPSHOT-jar-with-dependencies_Jloeln_2023V2.jar Main /input/practica2 /output
000      oooooo          oooooooooo.     .08
'88.    .888'           '888     '888.     "888
888b   d'888  .0000.  00.0000.  888     .d88'  .0000.  .00000.  0000  0000  .00000.  .00000.
8 Y88.  .P 888  '88b  888' '88b  888oo888P' d88' '88b d88' '888  888 '888 d88' "Y8 d88' '88b
8 `888' 888  .o888888 888 888 88888888 888`88b.  888oo888 888 888 888 888 888 888 888 888 888 888 888 888 888 888
8  Y   888 d( 888 888 888 888 888 888 888 888 888 888 888 888 888 888 888 888 888 888 888 888 888 888 888 888
o8o    o888o `Y888***8o 888bod8P'      o888o o888o `Y8bod8P' `Y8bod88P' `V88V"V8P' `Y8bod8P' `Y8bod8P'
               888
               o888o

```

Alumno: Jose Luis Obiang Ela Nanguang  
 Profesor: Francisco Chávez de la O  
 Asignatura: Sistema de Información (Sinf)  
 Fecha de entrega: 15-01-2024 23:55

```

=====
[ ] MENÚ DE CONSULTAS:
[ ]
[ ] ==>1. ¿Cuánto se ganó en 2019?
[ ] ==>2. ¿Cuál fue el mejor mes para las ventas? ¿Cuánto se ganó ese mes?
[ ] ==>3. ¿Qué ciudad tuvo el mayor número de ventas?
[ ] ==>4. ¿A qué hora debemos mostrar publicidad para maximizar la probabilidad de que el cliente compre el producto?
[ ] ==>5. ¿Qué producto vendió más? ¿Por qué crees que vendió más?
[ ] ==>6. Salir
[ ]
[ ] ==>Seleccione una opción: |

```

- Ejecución de cada una de las consultas.

- Archivos generados:

```

hadoop@hadoop:~$ hadoop fs -ls /output*Jloeln_2023/
Found 2 items
-rw-r--r-- 1 hadoop supergroup          0 2024-01-13 14:49 /output1_Jloeln_2023/_SUCCESS
-rw-r--r-- 1 hadoop supergroup          91 2024-01-13 14:49 /output1_Jloeln_2023/part-r-00000
Found 2 items
-rw-r--r-- 1 hadoop supergroup          0 2024-01-13 14:52 /output2_Jloeln_2023/_SUCCESS
-rw-r--r-- 1 hadoop supergroup         1275 2024-01-13 14:52 /output2_Jloeln_2023/part-r-00000
Found 2 items
-rw-r--r-- 1 hadoop supergroup          0 2024-01-13 14:54 /output3_Jloeln_2023/_SUCCESS
-rw-r--r-- 1 hadoop supergroup         225 2024-01-13 14:54 /output3_Jloeln_2023/part-r-00000
Found 2 items
-rw-r--r-- 1 hadoop supergroup          0 2024-01-13 14:58 /output4_Jloeln_2023/_SUCCESS
-rw-r--r-- 1 hadoop supergroup         637 2024-01-13 14:58 /output4_Jloeln_2023/part-r-00000
Found 2 items
-rw-r--r-- 1 hadoop supergroup          0 2024-01-13 15:01 /output5_Jloeln_2023/_SUCCESS
-rw-r--r-- 1 hadoop supergroup         810 2024-01-13 15:01 /output5_Jloeln_2023/part-r-00000

```

- Si quisiéramos eliminar todos los archivos.

```

hadoop@hadoop:~$ hadoop fs -rm -r /out*Jloeln*
Deleted /output1_Jloeln_2023
Deleted /output2_Jloeln_2023
Deleted /output3_Jloeln_2023
Deleted /output4_Jloeln_2023
Deleted /output5_Jloeln_2023

```

### 1. ¿Cuánto se ganó en 2019?

```

hadoop@hadoop:~ x + ~
=>Seleccione una opción: 1
Creando ruta de salida /output1_Jloeln_2023...
2024-01-13 14:48:12,014 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-01-13 14:48:12,544 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2024-01-13 14:48:12,573 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.staging/job_1701455301118_0449
2024-01-13 14:48:13,513 INFO input.FileInputFormat: Total input files to process : 12
2024-01-13 14:48:13,593 INFO mapreduce.JobSubmitter: number of splits:12
2024-01-13 14:48:13,994 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1701455301118_0449
2024-01-13 14:48:13,995 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-01-13 14:48:14,297 INFO conf.Configuration: resource-types.xml not found
2024-01-13 14:48:14,297 INFO resource.ResourceUtils: Unable to find 'resource-types.xml'.
2024-01-13 14:48:14,403 INFO impl.YarnClientImpl: Submitted application application_1701455301118_0449
2024-01-13 14:48:14,464 INFO mapreduce.Job: The url to track the job: http://hadoop:8088/proxy/application_1701455301118_0449/
2024-01-13 14:48:14,464 INFO mapreduce.Job: Running job: job_1701455301118_0449
2024-01-13 14:48:14,464 INFO mapreduce.Job: Job job_1701455301118_0449 running in uber mode : false
2024-01-13 14:48:14,510 INFO mapreduce.Job: Counters: 55
File System Counters
FILE: Number of bytes read=2788746
FILE: Number of bytes written=9161683
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=16230961
HDFS: Number of bytes written=91
HDFS: Number of read operations=41

```

- Salida:

```

hadoop@hadoop:~ x hadoop@hadoop:~ x + ~
hadoop@hadoop:~$ hadoop fs -cat /output1_Jloeln_2023/part*
1.¿Cuánto se ganó en 2019?
2019 34483365.68
En el año 2019 se ganó: 34483365.68€

```

## 2. ¿Cuál fue el mejor mes para las ventas? ¿Cuánto se ganó ese mes?

```

hadoop@hadoop:~ x + ~
=>Seleccione una opción: 2
La ruta /output2_Jloeln_2023 ya existe
Creando nueva ruta de salida: /output2_Jloeln_2023...
2024-01-13 14:51:24,174 INFO client.DefaultNoHARMFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-01-13 14:51:24,204 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2024-01-13 14:51:24,206 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.staging/job_1701455301118_0450
2024-01-13 14:51:24,888 INFO input.FileInputFormat: Total input files to process : 12
2024-01-13 14:51:24,928 INFO mapreduce.JobSubmitter: number of splits:12
2024-01-13 14:51:24,967 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1701455301118_0450
2024-01-13 14:51:24,993 INFO impl.YarnClientImpl: Submitted application application_1701455301118_0450
2024-01-13 14:51:24,998 INFO mapreduce.Job: The url to track the job: http://hadoop:8088/proxy/application_1701455301118_0450/
2024-01-13 14:51:24,998 INFO mapreduce.Job: Running job: job_1701455301118_0450
2024-01-13 14:51:39,180 INFO mapreduce.Job: Job job_1701455301118_0450 running in uber mode : false
2024-01-13 14:51:39,183 INFO mapreduce.Job: map 0% reduce 0%
2024-01-13 14:52:01,480 INFO mapreduce.Job: map 25% reduce 0%
2024-01-13 14:52:02,489 INFO mapreduce.Job: map 50% reduce 0%
2024-01-13 14:52:20,673 INFO mapreduce.Job: map 67% reduce 0%
2024-01-13 14:52:21,679 INFO mapreduce.Job: map 92% reduce 0%
2024-01-13 14:52:22,686 INFO mapreduce.Job: map 100% reduce 0%
2024-01-13 14:52:27,726 INFO mapreduce.Job: map 100% reduce 100%
2024-01-13 14:52:27,743 INFO mapreduce.Job: Job job_1701455301118_0450 completed successfully
2024-01-13 14:52:27,825 INFO mapreduce.Job: Counters: 55
File System Counters
FILE: Number of bytes read=5977838
FILE: Number of bytes written=15539815
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=16230961
HDFS: Number of bytes written=1275
HDFS: Number of read operations=41

```

- Salida:

```

hadoop@hadoop:~ $ hadoop fs -cat /output2_Jloeln_2023/part*
2.¿Cuál fue el mejor mes para las ventas? ¿Cuánto se ganó ese mes?
Abril(04)/2019 3390670.24
-----
Agosto(08)/2019 2244467.88
Diciembre(12)/2019 4613443.34
Enero(01)/2019 1813586.44
Enero(01)/2020 8670.29
Febrero(02)/2019 2202022.42
Julio(07)/2019 2647775.76
Junio(06)/2019 2577802.26
Marzo(03)/2019 2807100.38
Mayo(05)/2019 3152606.75
Noviembre(11)/2019 3199603.20
Octubre(10)/2019 3736726.88
Septiembre(09)/2019 2097560.13
-----
El mes de Diciembre(12)/2019 es el mejor mes ya que tiene el mayor total de ganancias, con un total de: 4613443.34€

```

### 3. ¿Qué ciudad tuvo el mayor número de ventas?

```

==> Seleccione una opción: 3
La ruta /output2_Jloeln_2023 ya existe

Creando nueva ruta de salida: /output3_Jloeln_2023...
2024-01-13 14:53:27,499 INFO client.DefaultHttpAgnFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-01-13 14:53:27,522 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2024-01-13 14:53:27,530 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.staging/job_1701455301118_0451
2024-01-13 14:53:28,000 INFO input.FileInputFormat: Total input files to process : 12
2024-01-13 14:53:28,054 INFO mapreduce.JobSubmitter: number of splits:12
2024-01-13 14:53:28,104 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1701455301118_0451
2024-01-13 14:53:28,105 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-01-13 14:53:28,144 INFO impl.YarnClientImpl: Submitted application application_1701455301118_0451
2024-01-13 14:53:28,152 INFO mapreduce.Job: The url to track the job: http://hadoop:8088/proxy/application_1701455301118_0451/
2024-01-13 14:53:28,152 INFO mapreduce.Job: Running job: job_1701455301118_0451
2024-01-13 14:53:43,301 INFO mapreduce.Job: Job job_1701455301118_0451 running in uber mode : false
2024-01-13 14:53:43,302 INFO mapreduce.Job: map% reduce 0%
2024-01-13 14:54:05,557 INFO mapreduce.Job: map 50% reduce 0%
2024-01-13 14:54:23,713 INFO mapreduce.Job: map 58% reduce 0%
2024-01-13 14:54:24,779 INFO mapreduce.Job: map 75% reduce 0%
2024-01-13 14:54:25,785 INFO mapreduce.Job: map 92% reduce 0%
2024-01-13 14:54:26,791 INFO mapreduce.Job: map 100% reduce 0%
2024-01-13 14:54:30,820 INFO mapreduce.Job: Job job_1701455301118_0451 completed successfully
2024-01-13 14:54:30,916 INFO mapreduce.Job: Counters:
File System Counters
FILE: Number of bytes read=1292243
FILE: Number of bytes written=6168638
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=16230961
HDFS: Number of bytes written=225
HDFS: Number of read operations=41

```

- Salida:

```

hadoop@hadoop:~ $ hadoop fs -cat /output3_Jloeln_2023/part*
3.¿Qué ciudad tuvo el mayor número de ventas?
Atlanta 16602
Austin 11153
Boston 22528
Dallas 16730
Seattle 16553
La ciudad de Boston tiene el mayor número de ventas con un total de: 22528 ventas

```

### 4. ¿A qué hora debemos mostrar publicidad para maximizar la probabilidad de que el cliente compre el producto?

```

hadoop@hadoop:~$ hadoop fs -cat /output4_Jloeln_2023/part-r-00000
4.¿A qué hora debemos mostrar publicidad para maximizar la probabilidad de que el cliente compre el producto?
1(1) AM 2350
1(13) PM 12129
10(10) AM 10944
10(22) PM 8822
11(11) AM 12411
11(23) PM 6275
12(0) AM 3910
12(12) PM 12587
2(14) PM 10984
2(2) AM 1243
3(15) PM 10175
3(3) AM 831
4(16) PM 10384
4(4) AM 854
5(17) PM 10899
5(5) AM 1321
6(18) PM 12280
6(6) AM 2482
7(19) PM 12905
7(7) AM 4011
8(20) PM 12228
8(8) AM 6256
9(21) PM 10921
9(9) AM 8748
La hora para maximizar la probabilidad de que el cliente compre el producto es: 7(19) PM
ya que en esa hora se hizo un total de 12905 pedidos.

```

- Salida:

```

hadoop@hadoop:~$ hadoop fs -cat /output4_Jloeln_2023/part-r-00000
4.¿A qué hora debemos mostrar publicidad para maximizar la probabilidad de que el cliente compre el producto?
1(1) AM 2350
1(13) PM 12129
10(10) AM 10944
10(22) PM 8822
11(11) AM 12411
11(23) PM 6275
12(0) AM 3910
12(12) PM 12587
2(14) PM 10984
2(2) AM 1243
3(15) PM 10175
3(3) AM 831
4(16) PM 10384
4(4) AM 854
5(17) PM 10899
5(5) AM 1321
6(18) PM 12280
6(6) AM 2482
7(19) PM 12905
7(7) AM 4011
8(20) PM 12228
8(8) AM 6256
9(21) PM 10921
9(9) AM 8748
La hora para maximizar la probabilidad de que el cliente compre el producto es: 7(19) PM
ya que en esa hora se hizo un total de 12905 pedidos.

```

## 5. ¿Qué producto vendió más? ¿Por qué crees que vendió más?

```

hadoop@hadoop:~$ hadoop fs -cat /output5_Jloeln_2023/part-r-00000
5.¿Qué producto vendió más? ¿Por qué crees que vendió más?
La ruta /output5_Jloeln_2023 ya existe

Creando nueva ruta de salida: /output5_Jloeln_2023...
2024-01-13 14:59:57,792 INFO client.DefaultHttpHARFFailoverProxyProvider: Connecting to ResourceManager at /0.0.0.0:8032
2024-01-13 14:59:57,816 WARN mapreduce.JobResourceUploader: Hadoop command-line option parsing not performed. Implement the Tool interface and execute your application with ToolRunner to remedy this.
2024-01-13 14:59:57,822 INFO mapreduce.JobResourceUploader: Disabling Erasure Coding for path: /tmp/hadoop-yarn/staging/hadoop/.staging/job_1701455301118_0453
2024-01-13 14:59:58,337 INFO input.FileInputFormat: Total input files to process : 12
2024-01-13 14:59:58,382 INFO mapreduce.JobSubmitter: number of splits:12
2024-01-13 14:59:58,480 INFO mapreduce.JobSubmitter: Submitting tokens for job: job_1701455301118_0453
2024-01-13 14:59:58,482 INFO mapreduce.JobSubmitter: Executing with tokens: []
2024-01-13 14:59:58,522 INFO impl.YarnClientImpl: Submitted application application_1701455301118_0453
2024-01-13 14:59:58,528 INFO mapreduce.Job: The url to track the job: http://hadoop:8088/proxy/application_1701455301118_0453/
2024-01-13 14:59:58,529 INFO mapreduce.Job: Running job: job_1701455301118_0453
2024-01-13 15:00:12,669 INFO mapreduce.Job: Job job_1701455301118_0453 running in uber mode : false
2024-01-13 15:00:12,671 INFO mapreduce.Job: map 0% reduce 0%
2024-01-13 15:00:35,894 INFO mapreduce.Job: map 17% reduce 0%
2024-01-13 15:00:36,902 INFO mapreduce.Job: map 50% reduce 0%
2024-01-13 15:00:54,030 INFO mapreduce.Job: map 58% reduce 0%
2024-01-13 15:00:55,040 INFO mapreduce.Job: map 67% reduce 0%
2024-01-13 15:00:56,040 INFO mapreduce.Job: map 92% reduce 0%
2024-01-13 15:01:00,068 INFO mapreduce.Job: map 100% reduce 0%
2024-01-13 15:01:02,088 INFO mapreduce.Job: map 100% reduce 100%
2024-01-13 15:01:02,098 INFO mapreduce.Job: Job job_1701455301118_0453 completed successfully
2024-01-13 15:01:02,178 INFO mapreduce.Job: Counters: 55
File System Counters
FILE: Number of bytes read=6193428
FILE: Number of bytes written=15971164
FILE: Number of read operations=0
FILE: Number of large read operations=0
FILE: Number of write operations=0
HDFS: Number of bytes read=16230961
HDFS: Number of bytes written=810

```

- Salida:

```

hadoop@hadoop:~$ hadoop fs -cat /output5_Jloeln_2023/part*
      5.¿Qué producto vendió más? ¿Por qué crees que vendió más?
20in Monitor          4129
27in 4K Gaming Monitor    6244
27in FHD Monitor        7550
34in Ultrawide Monitor   6199
AA Batteries (4-pack)    27635
AAA Batteries (4-pack)   31017
Apple Airpods Headphones 15661
Bose SoundSport Headphones 13457
Flatscreen TV           4819
Google Phone             5532
LG Dryer                 646
LG Washing Machine       666
Lightning Charging Cable 23217
Macbook Pro Laptop        4728
ThinkPad Laptop           4130
USB-C Charging Cable     23975
Vareebadd Phone           2068
Wired Headphones          20557
iPhone                   6849
El producto AAA Batteries (4-pack) tiene el mayor número de ventas, ya que se ha vendido más que los demás con un total de: 3 1017 ventas

```

- Fin de ejecución.



- Mover datos del nodo hadoop a nuestro equipo

- Para ello primero copiar los archivos de las salida del procesamiento del HDFS al directorio /home/hadoop.

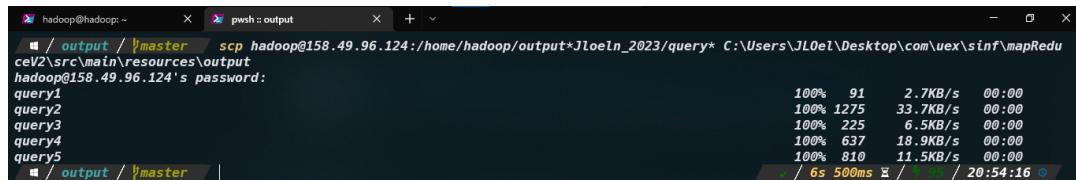
```

hadoop@hadoop:~$ hadoop fs -get /output*Jloel*/ .|
```

- Modificar los nombres de los ficheros part\*



- Copiar los archivos a la máquina local.



- Mostrar nuevamente el contenido de los archivos.

```

hadoop@hadoop:~          psh :: output          + 
■ / output / \master      cat query1           ✓ / 6s 500ms ✅ / 20:54:16 ✎
    1.¿Cuánto se ganó en 2019?
    2019   34483365.68
En el año 2019 se ganó: 34483365.68€
■ / output / \master      cat query2           ✓ / 5s ✅ / 20:55:41 ✎
    2.¿Cuál fue el mejor mes para las ventas? ¿Cuánto se ganó ese mes?
Abril(04)/2019           3390670.24
Agosto(08)/2019          2244467.88
Diciembre(12)/2019       4613443.34
Enero(01)/2019           1813586.44
Enero(01)/2020           8670.29
Febrero(02)/2019         2202022.42
Julio(07)/2019            2647775.76
Junio(06)/2019            2577802.26
Marzo(03)/2019            2807100.38
Mayo(05)/2019             3152606.75
Noviembre(11)/2019        3199603.20
Octubre(10)/2019          3736726.88
Septiembre(09)/2019        2097560.13
El mes de Diciembre(12)/2019 es el mejor mes ya que tiene el mayor total de ganancias, con un total de: 4613443.34€

hadoop@hadoop:~          psh :: output          + 
■ / output / \master      cat query3           ✓ / 4s ✅ / 20:55:44 ✎
    3.¿Qué ciudad tuvo el mayor número de ventas?
Atlanta      16602
Austin       11153
Boston       22528
Dallas       16730
Seattle      16553
La ciudad de Boston tiene el mayor número de ventas con un total de: 22528 ventas
■ / output / \master      cat query4           ✓ / 4s ✅ / 20:55:55 ✎
    4.¿A qué hora debemos mostrar publicidad para maximizar la probabilidad de que el cliente compre el producto?
1(1) AM      2350
1(13) PM     12129
10(10) AM     10944
10(22) PM     8822
11(11) AM     12411
11(23) PM     6275
12(0) AM      3910
12(12) PM     12587
2(14) PM      10984
2(2) AM       1243
3(15) PM      10175
3(3) AM       831
4(16) PM      10384
4(4) AM       854
5(17) PM      10899
5(5) AM       1321
6(18) PM      12280
6(6) AM       2482
7(19) PM      12905
7(7) AM       4011
8(20) PM      12228
8(8) AM       6256
9(21) PM      10921
9(9) AM       8748
La hora para maximizar la probabilidad de que el cliente compre el producto es: 7(19) PM

ya que en esa hora se hizo un total de 12905 pedidos.
■ / output / \master      cat query5           ✓ / 4s ✅ / 20:55:58 ✎
    5.¿Qué producto vendió más? ¿Por qué crees que vendió más?
20in Monitor      4129
27in 4K Gaming Monitor  6244
27in FHD Monitor    7550
34in Ultrawide Monitor 6199
AA Batteries (4-pack) 27635
AAA Batteries (4-pack) 31017
Apple Airpods Headphones 15661
Bose Soundsport Headphones 13457
Flatscreen TV        4819
Google Phone          5532
LG Dryer              646
LG Washing Machine    666
Lightning Charging Cable 23217
Macbook Pro Laptop    4728
ThinkPad Laptop        4130
USB-C Charging Cable  23975
Vareebadd Phone        2068
Wired Headphones       20557
iPhone                6849
El producto AAA Batteries (4-pack) tiene el mayor número de ventas, ya que se ha vendido más que los demás con un total de: 31017 ventas
■ / output / \master      |

```

## Repositorio Github.

Para comenzar a utilizar el programa, siga estos pasos para su descarga:

### 1. Acceso al Repositorio:

- Versión 1 del programa → <https://github.com/Jloen1999/mapReduceJloenV1>
- Versión 2 del programa → <https://github.com/Jloen1999/mapReduceJloenV2>

## 2. Descarga del Código Fuente:

- **Clonación del Repositorio:**

Para clonar el repositorio a su máquina local, use el siguiente comando en su terminal o línea de comandos:

```
Version 1:  
git clone https://github.com/Jloen1999/mapReduceJloenV1.git  
Version 2:  
git clone https://github.com/Jloen1999/mapReduceJloenV2.git
```

## Bibliografía.

- Apuntes de la asignatura en el Campus Virtual.
- ChatGPT