Big Data

"Visualisasi Data"

Oleh :

Rajendra Rakha Arya P

1941720080

TI 3H

JURUSAN TEKNOLOGI INFORMASI

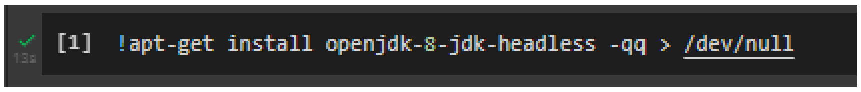
PROGRAM STUDI TEKNIK INFORMATIKA

POLITEKNIK NEGERI MALANG

2022

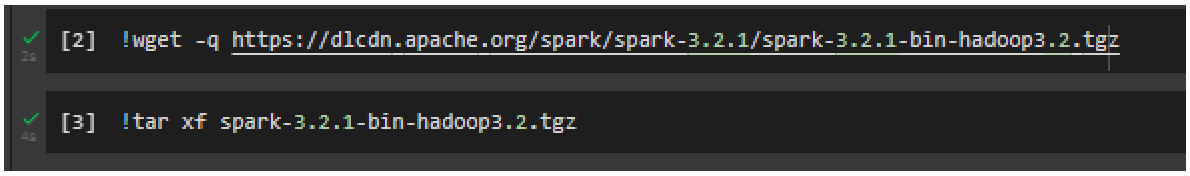
1. Access google colab
2. PySpark Setup in Colab

   Spark is written using the Scala programming language and requires a Java Virtual Machine (JVM) to run it. As a first step, perform a Java installation by writing the commands below.

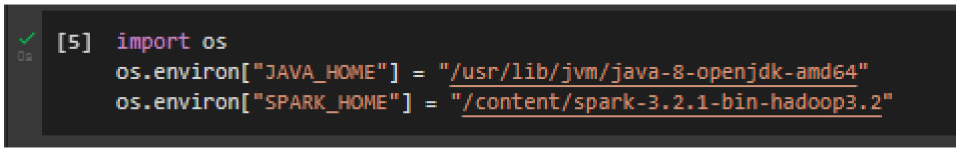
   ```
   [1]  !apt-get install openjdk-8-jdk-headless -qq > /dev/null
   ```

   For Apache Spark installations, download the file using the wget command then extract it with the tar command. Please copy the following command to perform the installation.

   ```
   [2]  !wget -q https://dlcdn.apache.org/spark/spark-3.2.1/spark-3.2.1-bin-hadoop3.2.tgz

   [3]  !tar xf spark-3.2.1-bin-hadoop3.2.tgz
   ```

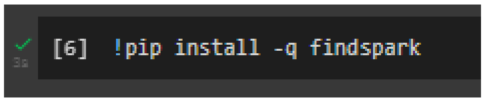
   As an advanced step, it takes settings related to Java and Spark Home. To doit, you can take advantage of the python script. Please enter the following code into the notebook.

   ```
   [5]  import os
        os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
        os.environ["SPARK_HOME"] = "/content/spark-3.2.1-bin-hadoop3.2"
   ```

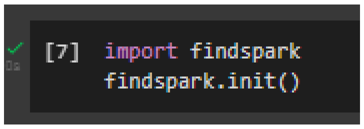
3. PySpark configuration

   PySpark configuration can be done by installing the findspark library which is used to find the location of Spark installed on the system. The installation process can utilize pip command, please pay attention to the command below.

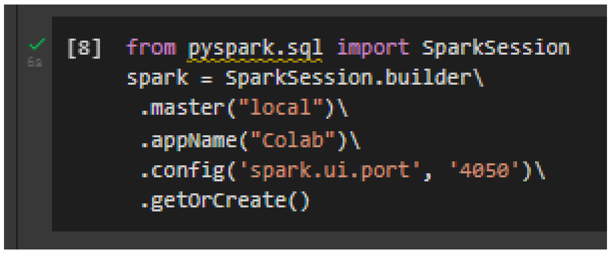
   ```
   [6]  !pip install -q findspark
   ```

   After the installation process is successful, import the library and initialize finds park. Please copy the following code into the notebook.



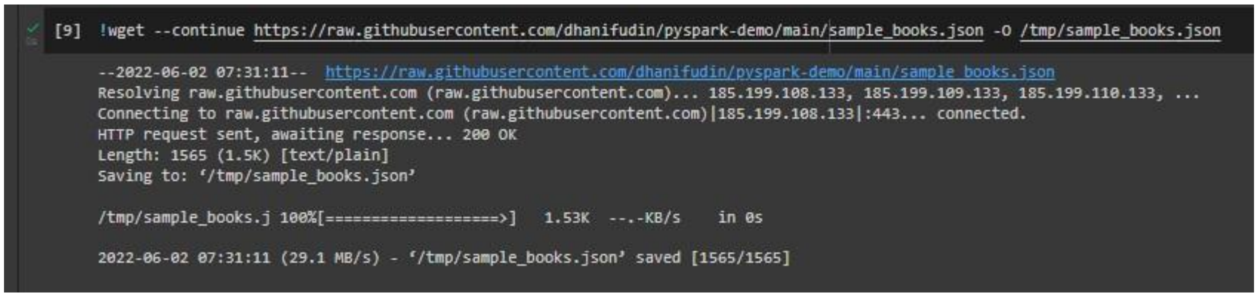
```
[7] import findspark
    findspark.init()
```

Connections into sparks can be made utilizing SparkSession. Copy the following code, where spark uses port 4050.



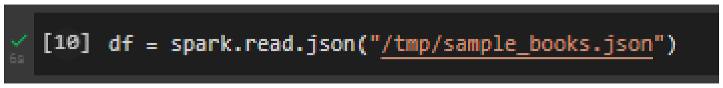
```
[8] from pyspark.sql import SparkSession
    spark = SparkSession.builder\
     .master("local")\
     .appName("Colab")\
     .config('spark.ui.port', '4050')\
     .getOrCreate()
```

4. Entering data into PySpark
   Spark has several modules for reading data in different formats. Spark will automatically specify each data type for each column. The data to be used as a dataset can be downloaded using the wget command.
   Please pay attention to the following command.



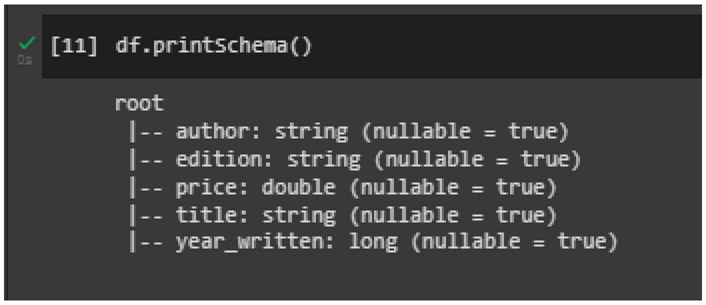
```
[9] !wget --continue https://raw.githubusercontent.com/dhanifudin/pyspark-demo/main/sample_books.json -O /tmp/sample_books.json

    --2022-06-02 07:31:11--  https://raw.githubusercontent.com/dhanifudin/pyspark-demo/main/sample_books.json
    Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
    Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
    HTTP request sent, awaiting response... 200 OK
    Length: 1565 (1.5K) [text/plain]
    Saving to: '/tmp/sample_books.json'

    /tmp/sample_books.j 100%[===================>]   1.53K  --.-KB/s    in 0s

    2022-06-02 07:31:11 (29.1 MB/s) - '/tmp/sample_books.json' saved [1565/1565]
```

The data that has been downloaded can be read by using the following code. Please copy the following code into the notebook.



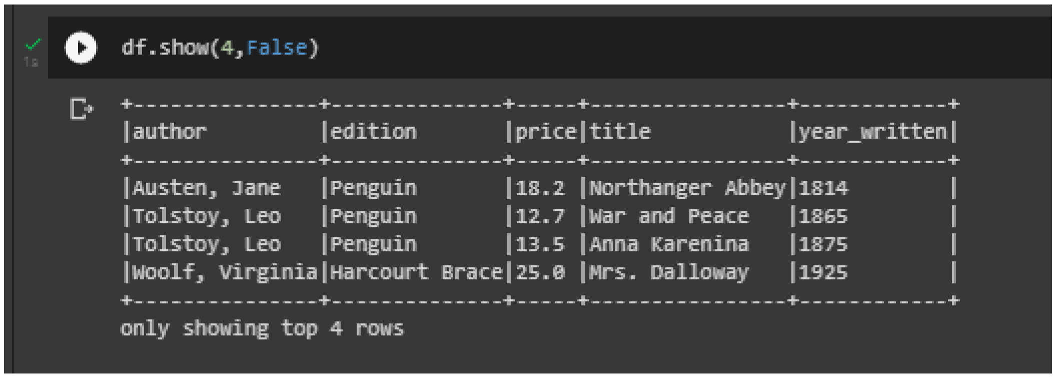
```
[10] df = spark.read.json("/tmp/sample_books.json")
```

5. Data analysis using PySpark
   Before you can analyze the dataset, it is necessary to know the schema of the data to be processed. The schema can be known by using the following code. This code leverages dataframes.



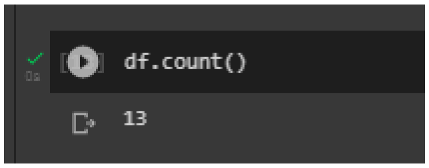
```
[11] df.printSchema()

root
 |-- author: string (nullable = true)
 |-- edition: string (nullable = true)
 |-- price: double (nullable = true)
 |-- title: string (nullable = true)
 |-- year_written: long (nullable = true)
```

6. View data
   The dataset can be displayed with the show command from the dataframe. The show command has parameters in the form of the number of records displayed as well as the truncate option. Please copy the following code into the notebook.



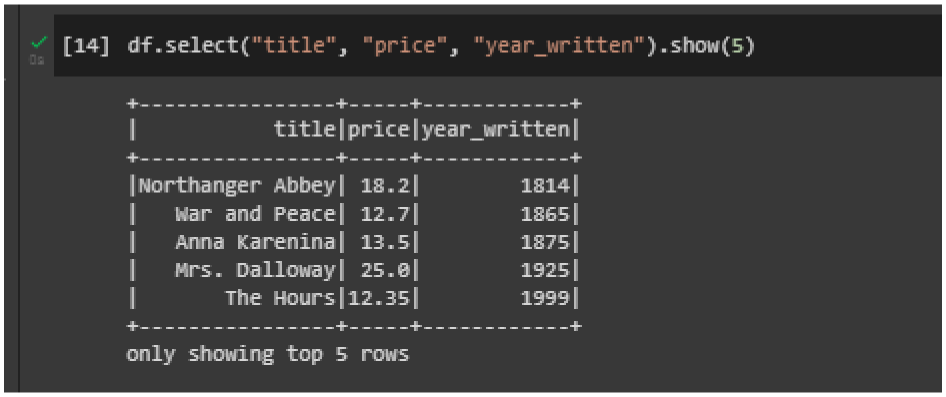
```
df.show(4,False)

+---------------+-------------+-----+----------------+------------+
|author         |edition      |price|title           |year_written|
+---------------+-------------+-----+----------------+------------+
|Austen, Jane   |Penguin      |18.2 |Northanger Abbey|1814        |
|Tolstoy, Leo   |Penguin      |12.7 |War and Peace   |1865        |
|Tolstoy, Leo   |Penguin      |13.5 |Anna Karenina   |1875        |
|Woolf, Virginia|Harcourt Brace|25.0 |Mrs. Dalloway   |1925        |
+---------------+-------------+-----+----------------+------------+
only showing top 4 rows
```

7. Counting the number of datasets
   The dataset can be calculated using the count function, please copy the following code into to find out the number of datasets owned.



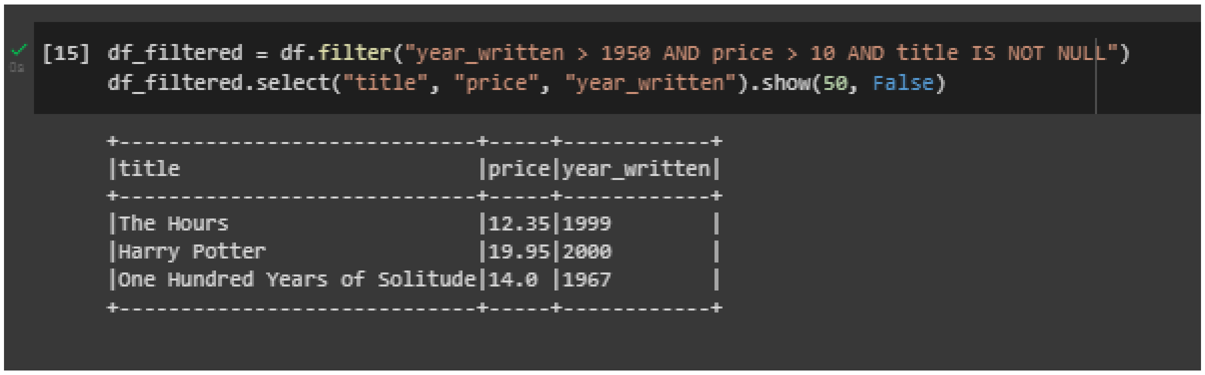
```
df.count()

13
```

8. Display the desired column
   Datasets can be selected to display data with specific columns with the select function. To display only the title, price and year_written fields, please copy the following code into the colab.



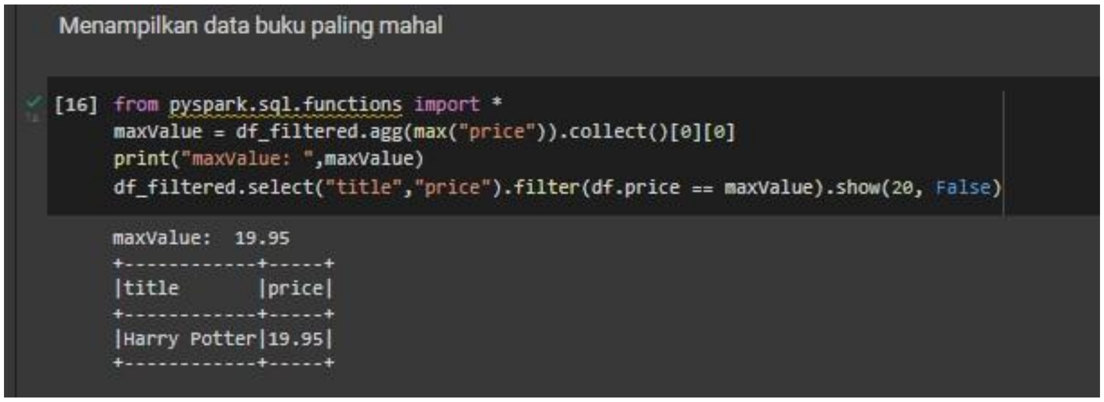
```
[14] df.select("title", "price", "year_written").show(5)

+---------------+-----+------------+
|          title|price|year_written|
+---------------+-----+------------+
|Northanger Abbey| 18.2|        1814|
|   War and Peace| 12.7|        1865|
|   Anna Karenina| 13.5|        1875|
|   Mrs. Dalloway| 25.0|        1925|
|       The Hours|12.35|        1999|
+---------------+-----+------------+
only showing top 5 rows
```

9. Filtering dataset
   PySpark can also filter a dataset based on the conditions needed. For example: the dataset you want to display is a book written after 1950 and costs more than $10. Filtering can be done by writing down the code along.



```
[15] df_filtered = df.filter("year_written > 1950 AND price > 10 AND title IS NOT NULL")
     df_filtered.select("title", "price", "year_written").show(50, False)

+----------------------------+-----+------------+
|title                       |price|year_written|
+----------------------------+-----+------------+
|The Hours                   |12.35|1999        |
|Harry Potter                |19.95|2000        |
|One Hundred Years of Solitude|14.0 |1967        |
+----------------------------+-----+------------+
```

10. Use of PySpark SQL functions
    PySpark has other SQL functions, for example max, aggregate function (groupBy, sum, count etc.). Example: displaying the most expensive book data, can use the max function

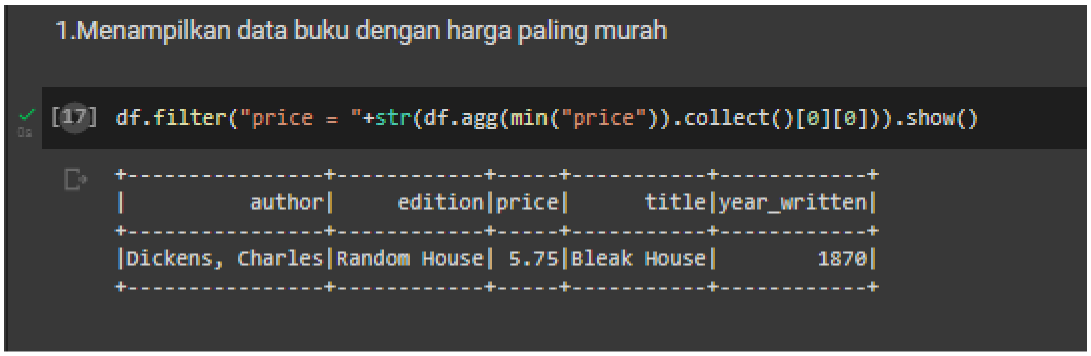


Menampilkan data buku paling mahal

```
[16] from pyspark.sql.functions import *
     maxValue = df_filtered.agg(max("price")).collect()[0][0]
     print("maxValue: ",maxValue)
     df_filtered.select("title","price").filter(df.price == maxValue).show(20, False)

maxValue:  19.95
+------------+-----+
|title       |price|
+------------+-----+
|Harry Potter|19.95|
+------------+-----+
```

ASSIGNMENT

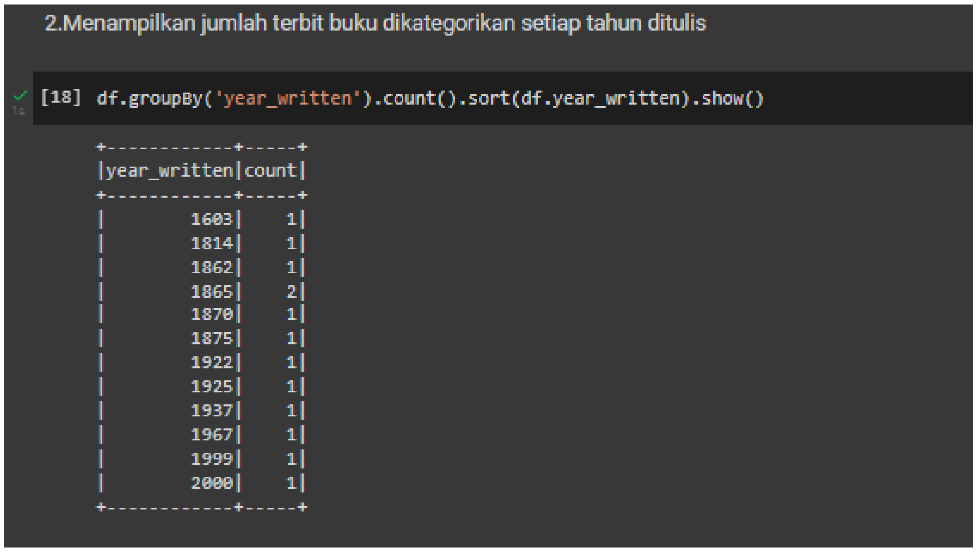
1. Show book data at the lowest price!



1.Menampilkan data buku dengan harga paling murah

```
[17] df.filter("price = "+str(df.agg(min("price")).collect()[0][0])).show()

+----------------+------------+-----+-----------+------------+
|          author|     edition|price|      title|year_written|
+----------------+------------+-----+-----------+------------+
|Dickens, Charles|Random House| 5.75|Bleak House|        1870|
+----------------+------------+-----+-----------+------------+
```

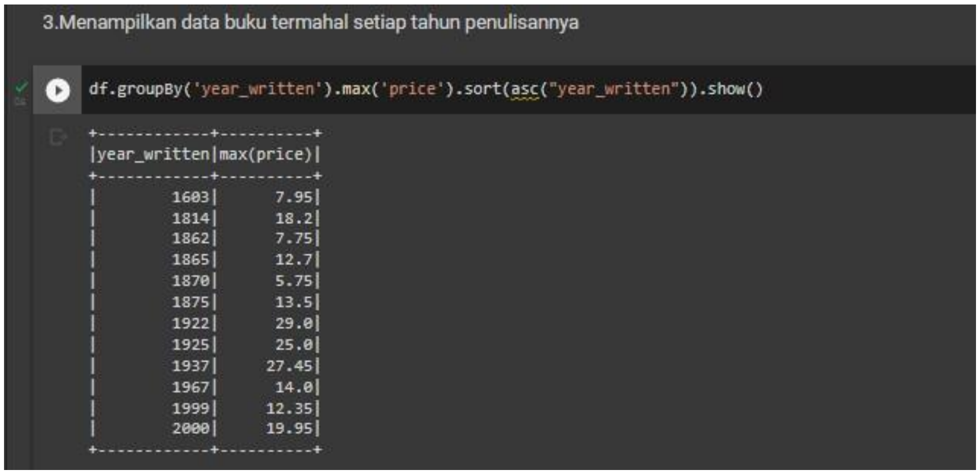
2. Show the number of published books categorized each year written!



2.Menampilkan jumlah terbit buku dikategorikan setiap tahun ditulis

```
[18] df.groupBy('year_written').count().sort(df.year_written).show()

+------------+-----+
|year_written|count|
+------------+-----+
|        1603|    1|
|        1814|    1|
|        1862|    1|
|        1865|    2|
|        1870|    1|
|        1875|    1|
|        1922|    1|
|        1925|    1|
|        1937|    1|
|        1967|    1|
|        1999|    1|
|        2000|    1|
+------------+-----+
```

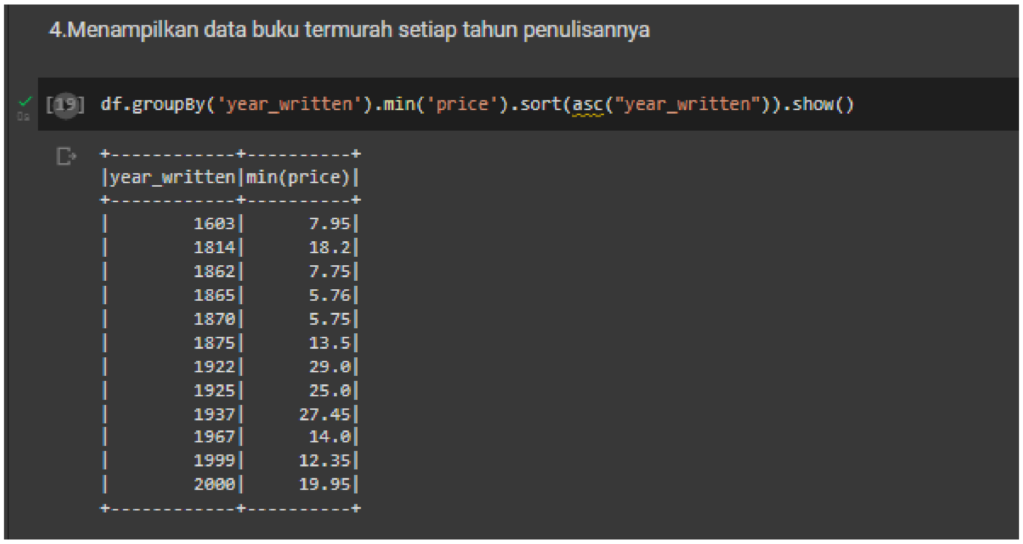
3. Show the most expensive book data every year it's written!



3.Menampilkan data buku termahal setiap tahun penulisannya

```
df.groupBy('year_written').max('price').sort(asc("year_written")).show()

+------------+----------+
|year_written|max(price)|
+------------+----------+
|        1603|      7.95|
|        1814|      18.2|
|        1862|      7.75|
|        1865|      12.7|
|        1870|      5.75|
|        1875|      13.5|
|        1922|      29.0|
|        1925|      25.0|
|        1937|     27.45|
|        1967|      14.0|
|        1999|     12.35|
|        2000|     19.95|
+------------+----------+
```

4. Show the cheapest book data every year it's written!

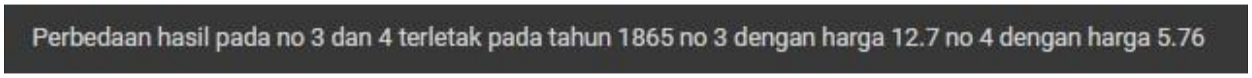


4.Menampilkan data buku termurah setiap tahun penulisannya

```
[19] df.groupBy('year_written').min('price').sort(asc("year_written")).show()
```

```
+------------+----------+
|year_written|min(price)|
+------------+----------+
|        1603|      7.95|
|        1814|      18.2|
|        1862|      7.75|
|        1865|      5.76|
|        1870|      5.75|
|        1875|      13.5|
|        1922|      29.0|
|        1925|      25.0|
|        1937|     27.45|
|        1967|      14.0|
|        1999|     12.35|
|        2000|     19.95|
+------------+----------+
```

Conclusion



Perbedaan hasil pada no 3 dan 4 terletak pada tahun 1865 no 3 dengan harga 12.7 no 4 dengan harga 5.76

Link GitHub & Collab :

https://github.com/Rjndrkha/JS-BigData_Visualisasi