

**FINAL EXAM PROJECT  
BIG DATA: ANALYSIS OF “MUSICAL  
INSTRUMENTS RATINGS ON AMAZON”**



**Kelompok 3**

|                                |              |
|--------------------------------|--------------|
| Bintang Siahaan                | (2206024322) |
| Aliyah Rizky Al-Afifah Polanda | (2206024682) |
| Monica Vierin Pasman           | (2206029405) |
| Adhelia Putri Maylani          | (2206814816) |
| Yasmin Devina Sinuraya         | (2206817244) |

## DAFTAR ISI

|   |           |
|---|-----------|
| <b>DAFTAR ISI.....</b>  | <b>2</b>  |
| <b>BAB I</b>  |           |
| <b>PENDAHULUAN.....</b>   | <b>3</b>  |
| 1.1 Latar Belakang.....   | 3         |
| 1.2 Rumusan Masalah.....  | 3         |
| 1.3 Use Case.....   | 4         |
| 1.4 Tujuan.....   | 5         |
| <b>BAB II</b>   |           |
| <b>DASAR TEORI.....</b>   | <b>6</b>  |
| 2.1 Big Data.....   | 6         |
| 2.2 Hadoop Distributed File System (HDFS).....                    | 6         |
| 2.3 Apache Spark.....   | 7         |
| 2.4 NLTK (Natural Language ToolKit).....                          | 7         |
| 2.5 Grafana.....  | 7         |
| 2.6 Rating, Ulasan, dan Rekomendasi.....                          | 8         |
| <b>BAB III</b>  |           |
| <b>IMPLEMENTASI DAN PEMBAHASAN.....</b>                           | <b>9</b>  |
| 3.1 Pengumpulan dan Persiapan Dataset.....                        | 9         |
| 3.2 Penyimpanan Data di HDFS.....                                 | 11        |
| 3.3 Pemrosesan Data dengan Apache Spark.....                      | 11        |
| 3.3.1 Filtering Berdasarkan Rating.....                           | 12        |
| 3.3.2 Filtering Berdasarkan Waktu.....                            | 14        |
| 3.3.3 Filtering Berdasarkan Penurunan Rating pada Tahun 2018..... | 15        |
| 3.3.4 Analisis Sentimen Berdasarkan Review Pengguna.....          | 17        |
| 3.4 Visualisasi Hasil.....  | 23        |
| <b>BAB IV</b>   |           |
| <b>KESIMPULAN.....</b>  | <b>29</b> |
| <b>DAFTAR REFERENSI.....</b>                                      | <b>30</b> |

## BAB I

### PENDAHULUAN

#### 1.1 Latar Belakang

Dalam era digital yang semakin berkembang, data menjadi salah satu aset paling berharga bagi perusahaan, khususnya di sektor e-commerce. Platform seperti Amazon menghasilkan data dalam jumlah besar setiap harinya, termasuk informasi *rating*. Data ini menawarkan peluang besar untuk dianalisis, guna memahami pola perilaku pelanggan, tren pasar, serta performa produk.

Kategori produk *Musical Instrument*, yang mencakup alat-alat musik, perangkat ilmiah, dan alat teknis lainnya, memiliki karakteristik khusus yang membutuhkan pendekatan analisis yang mendalam. Konsumen dalam kategori ini sering kali membuat keputusan berdasarkan kualitas produk, yang tercermin dari *rating*. Dengan menganalisis data ini, dapat diperoleh wawasan penting seperti identifikasi produk dengan performa tinggi dan pola *rating* pelanggan. Kategori ini dipilih karena produk-produk di dalamnya sering kali digunakan oleh segmen pasar yang memiliki kebutuhan spesifik, seperti musisi, peneliti, atau teknisi, sehingga pemahaman terhadap pola *rating* di kategori ini dapat memberikan nilai tambah yang signifikan untuk memahami kepuasan pelanggan dan kebutuhan pasar lebih dalam.

Namun, volume data yang besar dan kompleks memerlukan teknologi canggih untuk pengelolaan dan analisis. Teknologi *big data* seperti *Hadoop Distributed File System* (HDFS) untuk mengelola dataset dalam skala besar seperti *rating* produk Amazon. Data yang tersimpan dalam format CSV di HDFS selanjutnya dapat diproses menggunakan Apache Spark sebagai kerangka kerja analisis data berkecepatan tinggi. Hasil dari analisis ini kemudian divisualisasikan menggunakan alat seperti Grafana untuk menghasilkan wawasan yang mudah dipahami oleh pengguna.

Melalui pendekatan ini, proyek ini bertujuan untuk menganalisis rating pelanggan pada kategori Musical Instruments di Amazon. Hasil analisis akan memberikan wawasan mendalam tentang pola *rating*, tren preferensi pelanggan, serta rekomendasi produk berbasis data.

#### 1.2 Rumusan Masalah

Seiring dengan meningkatnya volume data yang dihasilkan oleh platform e-commerce seperti Amazon, tantangan baru muncul dalam pengelolaan dan pemanfaatan data tersebut. Meskipun data rating memiliki potensi besar untuk menentukan kualitas dan kepuasan pelanggan, beberapa permasalahan utama muncul dalam proses analisisnya, khususnya untuk kategori produk Musical Instruments:

- Volume Data yang Besar

Dataset rating Amazon mencakup jutaan entri dari berbagai kategori produk. Pengelolaan data dalam skala besar ini memerlukan infrastruktur dan

teknologi yang efisien agar proses analisis dapat berjalan lancar tanpa memakan waktu yang lama.

➤ Pengelolaan Data yang Kompleks

Data *rating* yang tersedia biasanya tersebar dalam berbagai kategori dan tidak terstruktur. Untuk mendapatkan hasil yang relevan, data harus difilter dan dianalisis hanya untuk kategori Musical Instruments.

➤ Penyimpanan dan Pemrosesan Data

Data dalam volume besar membutuhkan solusi penyimpanan yang andal dan terdistribusi, seperti Hadoop Distributed File System (HDFS). Selain itu, diperlukan alat pemrosesan seperti Apache Spark untuk mengolah data dengan kecepatan tinggi tanpa mengorbankan akurasi hasil analisis.

➤ Kebutuhan Visualisasi yang Informatif

Setelah data diproses, hasil analisis seringkali sulit dipahami jika hanya disajikan dalam bentuk tabel atau teks. Oleh karena itu, diperlukan alat visualisasi seperti Grafana untuk menyajikan hasil analisis dalam bentuk grafik yang mudah dimengerti oleh pengambil keputusan.

➤ Kategori dengan Preferensi Spesifik

Produk pada kategori Musical Instruments memiliki konsumen dengan kebutuhan dan preferensi yang unik. Analisis rating untuk kategori ini harus mampu mengidentifikasi pola spesifik, seperti kualitas produk yang sering menjadi perhatian utama pelanggan.

Tanpa pemanfaatan teknologi yang tepat, tantangan ini dapat menyebabkan analisis data menjadi tidak efisien dan hasilnya kurang informatif. Oleh karena itu, proyek ini dirancang untuk mengatasi permasalah tersebut dengan memanfaatkan teknologi *big data* seperti HDFS untuk penyimpanan, Apache Spark untuk pemrosesan, dan alat visualisasi untuk penyajian hasil analisis.

### 1.3 Use Case

Proyek ini dirancang untuk memproses dan menganalisis data rating produk dari kategori *musical instruments* yang tersedia di platform Amazon. Dalam skenario nyata, proyek ini berguna untuk:

1. Menentukan distribusi rating dan mendapatkan wawasan tentang kualitas produk berdasarkan rating pengguna.
2. Menentukan distribusi sentimen dan mendapatkan wawasan mengenai kualitas produk berdasarkan ulasan pengguna.
3. Hal ini membantu untuk mengidentifikasi produk dengan performa terbaik atau yang paling populer.
4. Melihat apakah rating produk meningkat, menurun, atau tetap stabil dari waktu ke waktu, yang bisa memberikan informasi tentang perubahan persepsi pelanggan.
5. Memberikan rekomendasi berdasarkan data, seperti rating tertinggi, untuk membantu pelanggan atau bisnis dalam pengambilan keputusan.

#### **1.4 Tujuan**

Proyek ini bertujuan untuk membangun pipeline Big Data yang mampu memproses dataset besar dari rating produk di Amazon secara efisien. Berikut tujuan lebih spesifiknya:

1. Pipeline ini dirancang untuk menangani data batch secara efektif dengan memanfaatkan teknologi seperti HDFS dan Apache Spark.
2. Menganalisis data ulasan produk berdasarkan metrik yang relevan, seperti rating (1-5 bintang).
3. Laporan ini meliputi distribusi rating, distribusi sentimen, tren berdasarkan waktu, dan performa produk.
4. Menyediakan rekomendasi produk berdasarkan hasil analisis, yang dapat digunakan oleh pelanggan atau bisnis untuk meningkatkan pengalaman berbelanja atau strategi pemasaran.
5. Menyimpan hasil analisis secara fleksibel dalam format yang mudah diakses seperti file CSV atau JSON, sehingga hasilnya dapat digunakan untuk kebutuhan lebih lanjut.

## **BAB II**

### **DASAR TEORI**

#### **2.1 Big Data**

Big data merupakan istilah untuk data yang berukuran sangat besar, misalnya data yang berukuran petabyte ( $10^{15}$  byte). Data ini berupa gabungan dari data tidak terstruktur, semi-terstruktur, dan terstruktur yang dapat dikumpulkan dari berbagai sumber. Berikut adalah beberapa sumber big data:

- Media sosial: Misalnya Facebook, Instagram, atau LinkedIn yang menghasilkan data dalam jumlah yang besar setiap harinya, karena memiliki miliaran pengguna di seluruh dunia.
- E-Commerce: Misalnya Amazon atau Alibaba yang menghasilkan file log berukuran besar dan dapat digunakan untuk melacak tren pembelian pengguna.
- Pasar saham: Misalnya bursa saham internasional yang menghasilkan data dari transaksi harian yang mereka lakukan.
- Stasiun cuaca: Data berukuran sangat besar dihasilkan oleh stasiun cuaca atau satelit, digunakan untuk prakiraan cuaca.

Sekumpulan data dapat dikatakan big data jika memiliki karakteristik 3V berikut:

- a. Volume: Jumlah data yang sangat besar dan dapat berasal dari berbagai sumber.
- b. Variety: Tipe data yang beragam.
- c. Velocity: Kecepatan yang tinggi dalam pengumpulan dan pemrosesan data.

#### **2.2 Hadoop Distributed File System (HDFS)**

HDFS merupakan sistem penyimpanan utama yang digunakan dalam aplikasi Hadoop. Cara kerjanya adalah dengan mengirimkan data berukuran besar secara cepat diantara node, sehingga sering digunakan untuk manajemen big data. Selain digunakan untuk mentransfer data, HDFS juga dapat digunakan untuk menganalisis data. Berikut adalah tiga tujuan utama dari HDFS:

- a. Mengelola dataset berukuran besar.  
HDFS secara khusus dirancang untuk menangani aplikasi yang bekerja dengan dataset berukuran besar. Untuk memungkinkan hal tersebut, HDFS memiliki ratusan node dalam satu kluster yang bertugas untuk menyimpan dan mengelola dataset tersebut secara efektif.
- b. Mendeteksi kesalahan.  
HDFS memiliki teknologi yang dapat digunakan untuk memindai dan mendeteksi kesalahan secara cepat dan efektif. Hal ini karena HDFS menggunakan banyak perangkat keras dalam pekerjaannya.
- c. Efisiensi perangkat keras.  
HDFS dapat mengurangi kepadatan lalu lintas jaringan dengan meningkatkan kecepatan pemrosesan data, sehingga penggunaan perangkat keras menjadi lebih efisien.

## **2.3 Apache Spark**

Apache Spark termasuk salah satu teknologi komputasi kluster yang dirancang untuk pemrosesan yang efisien dan bekerja pada memori, bukan disk. Spark didasarkan pada Hadoop MapReduce, di mana Spark memperluas model MapReduce dengan mendukung lebih banyak jenis komputasi, seperti pemrosesan data secara streaming. Apache Spark terdiri dari Spark Core dan Spark Library. Spark Core merupakan inti dari Apache Spark yang bertugas untuk transmisi tugas, penjadwalan, dan fungsi input/output. Sedangkan Spark Library berisi berbagai pustaka yang digunakan aplikasi untuk menjalankan Apache Spark. Fitur utama dari Apache Spark adalah sebagai berikut:

- a. Spark memungkinkan aplikasi untuk berjalan dengan 100 kali lebih cepat dalam kluster Hadoop. Hal ini dicapai dengan mengurangi operasi baca tulis ke disk.
- b. Apache Spark menyediakan API bawaan untuk Java, Scala, dan Python, sehingga aplikasi yang dikembangkan dapat ditulis dengan berbagai bahasa.
- c. Apache Spark tidak hanya mendukung fungsi MapReduce, tetapi juga mendukung kueri SQL, Machine Learning, dan algoritma grafik.

## **2.4 NLTK (Natural Language ToolKit)**

Natural Language Toolkit (NLTK) merupakan pustaka Python yang dirancang untuk pemrosesan bahasa alami (Natural Language Processing, NLP). Salah satu fitur utama dari NLTK adalah kemampuan untuk menganalisis sentimen dalam teks, yang sangat berguna untuk menilai opini dari ulasan produk, media sosial, atau sumber teks pendek lainnya.

Dalam analisis sentimen ulasan pengguna, lexicon memegang peran penting sebagai panduan untuk menilai opini dalam teks. Lexicon adalah kumpulan kata yang diberi label sentimen, seperti positif, negatif, atau netral, berdasarkan makna dan konteks penggunaannya. Salah satu lexicon populer yang tersedia di NLTK adalah VADER (Valence Aware Dictionary and sEntiment Reasoner). VADER dirancang secara khusus untuk analisis sentimen pada teks pendek, seperti ulasan produk atau komentar pelanggan. Lexicon tidak hanya mengenali kata-kata dengan makna sentimen, tetapi juga mempertimbangkan intensitas emosional berdasarkan elemen tambahan, seperti penggunaan huruf kapital, tanda seru, serta kata slang.

Keunggulan VADER dalam analisis ulasan pengguna terletak pada kemampuannya memberikan skor sentimen yang akurat, yang dapat digunakan untuk mengidentifikasi opini positif, negatif, atau netral terhadap suatu produk atau layanan. Fitur ini memungkinkan bisnis untuk memahami umpan balik pelanggan dengan lebih baik dan membuat keputusan berdasarkan data tersebut.

## **2.5 Grafana**

Grafana merupakan aplikasi web open-source yang dirancang untuk analitik dan visualisasi data. Dengan Grafana, pengguna dapat menarik data dari berbagai sumber, membuat kueri, dan menampilkan informasi dalam bentuk dashboard

interaktif dan mudah digunakan. Beberapa fitur utama Grafana adalah sebagai berikut:

- a. Visualisasi Fleksibel.

Grafana mendukung berbagai jenis grafik, seperti scatter plot, histogram, dan bar chart, yang disusun dalam unit bernama panel. Dengan kombinasi panel ini, pengguna dapat membangun dashboard yang komprehensif sesuai kebutuhan.

- b. Kustomisasi.

Setiap elemen visualisasi dalam Grafana dapat dikustomisasi. Pengguna dapat mengubah tampilan dan format grafik untuk menyesuaikan dengan preferensi atau konteks data yang dianalisis, seperti memilih warna, skala, atau tata letak.

- c. Integrasi Data.

Grafana mampu terhubung dengan berbagai sumber data, termasuk SQLite, dan menyediakan editor kueri yang dirancang khusus untuk setiap sumber data. Hal ini memungkinkan pengguna untuk memanfaatkan fitur optimal dari setiap sumber data secara efisien.

## **2.6 Rating, Ulasan, dan Rekomendasi**

Rating dan ulasan pengguna adalah elemen penting dalam platform e-commerce, seperti Amazon, untuk menilai kualitas produk atau layanan. Rating adalah nilai atau skor yang diberikan pengguna dalam skala tertentu, misalnya 1 hingga 5 bintang, yang mencerminkan kepuasan mereka terhadap produk. Produk dengan rating tinggi cenderung lebih disukai, karena dianggap memiliki kualitas yang lebih baik dan dapat meningkatkan kepercayaan konsumen.

Selain itu, ulasan pengguna memberikan wawasan lebih mendalam mengenai pengalaman mereka dalam menggunakan produk. Tidak hanya membantu calon pembeli memahami kelebihan dan kekurangan suatu produk, ulasan juga memperkuat nilai rating dengan informasi tambahan yang bersifat kualitatif, seperti deskripsi fitur, keluhan, atau rekomendasi.

Sistem rekomendasi di platform e-commerce memanfaatkan rating dan ulasan untuk menentukan produk yang relevan dan berkualitas untuk ditampilkan ke pengguna. Faktor ini memastikan bahwa rekomendasi produk tidak hanya populer, tetapi juga dapat dipercaya dan sesuai dengan kebutuhan pengguna. Dengan menampilkan produk berkualitas tinggi yang didukung oleh ulasan positif, platform e-commerce dapat meningkatkan kepuasan dan pengalaman pengguna secara keseluruhan.

## BAB III

### IMPLEMENTASI DAN PEMBAHASAN

#### 3.1 Pengumpulan dan Persiapan Dataset

- a) Kunjungi situs [berikut](#).

### Recommender Systems and Personalization Datasets

[Julian McAuley, UCSD](#)

#### Description

This page contains a collection of datasets that have been collected for research by our lab. Datasets contain the following features:

- user/item interactions
- star ratings
- timestamps

- b) Buka link Amazon Reviews 2018.

#### Download link

See the [Amazon Reviews 2023](#) page for download information.

You can also download data from previous versions of these datasets:

[Amazon Reviews 2018](#)

[Amazon Reviews 2014](#)

- c) Dapatkan dataset 5-core dan rating only untuk produk Amazon, dengan kategori Musical Instruments.

|                             |                             |                                   |
|-----------------------------|-----------------------------|-----------------------------------|
| Amazon Fashion              | 5-core (3,176 reviews)      | ratings only (883,636 ratings)    |
| All Beauty                  | 5-core (5,269 reviews)      | ratings only (371,345 ratings)    |
| Appliances                  | 5-core (2,277 reviews)      | ratings only (602,777 ratings)    |
| Arts, Crafts and Sewing     | 5-core (494,485 reviews)    | ratings only (2,875,917 ratings)  |
| Automotive                  | 5-core (1,711,519 reviews)  | ratings only (7,990,166 ratings)  |
| Books                       | 5-core (27,164,983 reviews) | ratings only (51,311,621 ratings) |
| CDs and Vinyl               | 5-core (1,443,755 reviews)  | ratings only (4,543,369 ratings)  |
| Cell Phones and Accessories | 5-core (1,128,437 reviews)  | ratings only (10,063,255 ratings) |
| Clothing, Shoes and Jewelry | 5-core (11,285,464 reviews) | ratings only (32,292,099 ratings) |
| Digital Music               | 5-core (169,781 reviews)    | ratings only (1,584,082 ratings)  |
| Electronics                 | 5-core (6,739,590 reviews)  | ratings only (20,994,353 ratings) |
| Gift Cards                  | 5-core (2,972 reviews)      | ratings only (147,194 ratings)    |
| Grocery and Gourmet Food    | 5-core (1,143,860 reviews)  | ratings only (5,074,160 ratings)  |
| Home and Kitchen            | 5-core (6,898,955 reviews)  | ratings only (21,928,568 ratings) |
| Industrial and Scientific   | 5-core (77,071 reviews)     | ratings only (1,758,333 ratings)  |
| Kindle Store                | 5-core (2,222,983 reviews)  | ratings only (5,722,988 ratings)  |
| Luxury Beauty               | 5-core (34,278 reviews)     | ratings only (574,628 ratings)    |
| Magazine Subscriptions      | 5-core (2,375 reviews)      | ratings only (89,689 ratings)     |
| Movies and TV               | 5-core (3,410,019 reviews)  | ratings only (8,765,568 ratings)  |
| Musical Instruments         | 5-core (231,392 reviews)    | ratings only (1,512,530 ratings)  |
| Office Products             | 5-core (800,357 reviews)    | ratings only (5,581,313 ratings)  |
| Patio, Lawn and Garden      | 5-core (798,415 reviews)    | ratings only (5,236,058 ratings)  |
| Pet Supplies                | 5-core (2,098,325 reviews)  | ratings only (6,542,483 ratings)  |
| Prime Pantry                | 5-core (137,788 reviews)    | ratings only (471,614 ratings)    |
| Software                    | 5-core (12,805 reviews)     | ratings only (459,436 ratings)    |
| Sports and Outdoors         | 5-core (2,839,940 reviews)  | ratings only (12,980,837 ratings) |
| Tools and Home Improvement  | 5-core (2,070,831 reviews)  | ratings only (9,015,203 ratings)  |
| Toys and Games              | 5-core (1,828,971 reviews)  | ratings only (8,201,231 ratings)  |
| Video Games                 | 5-core (497,577 reviews)    | ratings only (2,565,349 ratings)  |

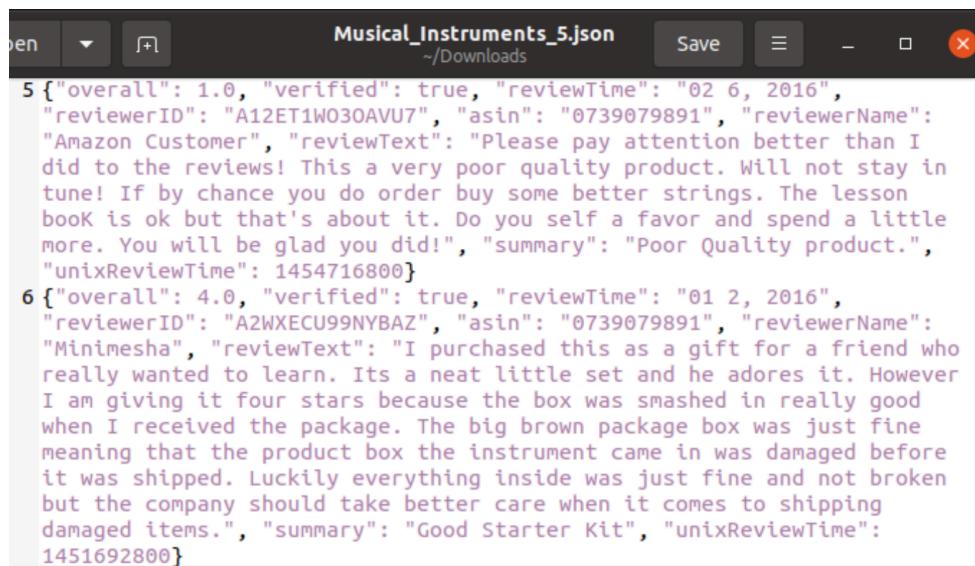
- d) Buka dataset yang sudah diunduh.

Dataset rating only:

| A   |
|---|
| 1 0470536454,AXHY24HWOF184,5.0,1092009600   |
| 2 0470536454,A29OWR79AM796H,4.0,1491436800  |
| 3 0470536454,AUPWU27A7X5F6,5.0,1489449600   |
| 4 0470536454,A1N69A47D4J06K,4.0,1487030400  |
| 5 0470536454,AHTIQUMVCGBFJ,5.0,1485648000   |
| 6 0470536454,A1J8LQ7HVLR9GU,5.0,1483488000  |
| 7 0470536454,ABVTZ63S6GOWF,5.0,1483315200   |
| 8 0470536454,A2HX9NF BXGSWRW,4.0,1482278400 |
| 9 0470536454,AP1TQR64HQRCI,4.0,1482192000   |
| 10 0470536454,A37FC9MED20AO,5.0,1481760000  |
| 11 0470536454,A26EU1X4VDNT4Z,5.0,1479772800 |
| 12 0470536454,A3II49ZWIOZF92,3.0,1477180800 |
| 13 0470536454,A347LQX3G500YN,5.0,1474848000 |
| 14 0470536454,A2FPOGMO1CCSUQ,4.0,1468972800 |
| 15 0470536454,AL8FRAYQS43LI,5.0,1467504000  |
| 16 0470536454,AHVLB4P706L86,4.0,1466812800  |
| 17 0470536454,AP609X3Z27XUH,4.0,1466726400  |
| 18 0470536454,A2JG74A5GTRKYM,3.0,1466553600 |
| 19 0470536454,A17B9D1PXJ3X6I,5.0,1465516800 |
| 20 0470536454,A2C7C71PD0ZCC8,5.0,1464739200 |
| 21 0470536454,A20TFBX9P653F2,5.0,1463184000 |
| 22 0470536454,A2H8PAAQZD5WTE,5.0,1461888000 |
| 23 0470536454,A2X0UAKARMY5KM,5.0,1458259200 |
| 24 0470536454,A2IWXMPJC3IGW1,4.0,1457740800 |
| 25 0470536454,ABYBPWL MGWC94,5.0,1456358400 |
| 26 0470536454,A3L93K90P67BM6,5.0,1456099200 |
| 27 0470536454,ARH82QI6F16HP,5.0,1456012800  |
| 28 0470536454,ASL14HDM0HNA9,5.0,1455926400  |

Pada dataset di atas, setiap baris menyatakan productID, reviewerID, rating, dan unixTime. Supaya lebih mudah dibaca, data unix time akan diubah menjadi bentuk time biasa melalui pemrosesan data di Apache Spark.

Dataset 5-core:



The screenshot shows a JSON viewer interface with the following content:

```
5 {"overall": 1.0, "verified": true, "reviewTime": "02 6, 2016", "reviewerID": "A12ET1W030AVU7", "asin": "0739079891", "reviewerName": "Amazon Customer", "reviewText": "Please pay attention better than I did to the reviews! This a very poor quality product. Will not stay in tune! If by chance you do order buy some better strings. The lesson book is ok but that's about it. Do you self a favor and spend a little more. You will be glad you did!", "summary": "Poor Quality product.", "unixReviewTime": 1454716800} 6 {"overall": 4.0, "verified": true, "reviewTime": "01 2, 2016", "reviewerID": "A2WXECU99NYBAZ", "asin": "0739079891", "reviewerName": "Minimesha", "reviewText": "I purchased this as a gift for a friend who really wanted to learn. Its a neat little set and he adores it. However I am giving it four stars because the box was smashed in really good when I received the package. The big brown package box was just fine meaning that the product box the instrument came in was damaged before it was shipped. Luckily everything inside was just fine and not broken but the company should take better care when it comes to shipping damaged items.", "summary": "Good Starter Kit", "unixReviewTime": 1451692800}
```

Dalam pemrosesan data ulasan produk, kolom yang akan digunakan hanya overall, reviewerID, asin, dan reviewText. Kolom overall berisi rating yang diberikan oleh reviewer, yang penting untuk analisis sentimen. reviewerID mengidentifikasi pengguna yang memberikan ulasan, sementara asin digunakan untuk mengelompokkan ulasan berdasarkan produk tertentu. reviewText berisi isi ulasan yang akan dianalisis untuk menilai sentimen positif, negatif, atau netral. Kolom lainnya seperti verified, reviewTime, reviewerName, summary, dan unixReviewTime tidak relevan untuk analisis sentimen yang fokus pada rating dan isi ulasan.

### 3.2 Penyimpanan Data di HDFS

- Jalankan HDFS.

```
> start-dfs.sh
```

```
vboxuser@Nireiv:~$ start-dfs.sh
Starting namenodes on [localhost]
Starting datanodes
Starting secondary namenodes [Nireiv]
```

- Cek proses Java yang sedang berjalan.

```
> jps
```

```
vboxuser@Nireiv:~$ jps
4624 DataNode
5010 Jps
4467 NameNode
4885 SecondaryNameNode
```

- Buat folder penyimpanan dataset Amazon di HDFS.

```
> hdfs dfs -mkdir /datasets
```

```
vboxuser@Nireiv:~$ hdfs dfs -mkdir /datasets
```

- Pindahkan dataset Amazon yang telah diunduh ke HDFS.

```
> hdfs dfs -put ~/Downloads/Musical_Instruments.csv /datasets/
```

```
aliyah@Ubuntu20:~$ hdfs dfs -put ~/Downloads/Musical_Instruments
.csv /datasets/
```

```
aliyah@Ubuntu20:~$ hdfs dfs -put ~/Downloads/Musical_Instruments
_5.json /datasets/
```

- Cek isi folder /datasets di HDFS untuk memastikan dataset telah berhasil dipindahkan.

```
> hdfs dfs -ls /datasets/
```

```
aliyah@Ubuntu20:~$ hdfs dfs -ls /datasets/
Found 2 items
-rw-r--r-- 1 aliyah supergroup          0 2024-12-14 17:49 /datasets/
Musical_Instruments.csv
-rw-r--r-- 1 aliyah supergroup 132000704 2024-12-14 17:46 /datasets/
Musical_Instruments_5.json
```

### 3.3 Pemrosesan Data dengan Apache Spark

- Jalankan Apache Spark.

```
> spark-shell
```

- b) Impor fungsi Apache Spark.

```
> import org.apache.spark.sql.functions._
```

```
scala> import org.apache.spark.sql.functions._  
import org.apache.spark.sql.functions._
```

- c) Baca dataset yang tersimpan di HDFS.

```
> val inputData = "hdfs://localhost:9000/datasets/Musical_Instruments.csv"
```

```
> val columns = Seq("productId", "reviewerId", "rating", "unixtime")
```

```
> val df = spark.read.option("header", "false").csv(inputData).toDF(columns:
```

\* )

```
scala> val inputData = "hdfs://localhost:9000/datasets/Musical_Instruments.csv"
inputData: String = hdfs://localhost:9000/datasets/Musical_Instruments.csv

scala> val columns = Seq("productId", "reviewerId", "rating", "unixtime")
columns: Seq[String] = List(productId, reviewerId, rating, unixtime)

scala> val df = spark.read.option("header", "false").csv(inputData).toDF(columns: _*)
df: org.apache.spark.sql.DataFrame = [productId: string, reviewerId: string ... 2 more fileds]
```

### 3.3.1 Filtering Berdasarkan Rating

- a) Lakukan data cleaning pada baris-baris yang terdapat nilai kosong dan ubah format waktu dari unixTime menjadi YYYY-MM-DD HH:MM:SS.

```
> val dfCleaned = df.withColumn("time", from_unixtime(col("time")))
```

```
scala> val dfCleaned = df.withColumn("time", from_unixtime(col("unixtime")))
dfCleaned: org.apache.spark.sql.DataFrame = [productId: string, reviewerId: string ... 3 more columns]
```

14. Sistemas de Información y la URGES

## Simpan hasil data cleaning

> Val cleanedOutputPath =

hdfs://localhost:9000/datasets/resultCleaned

```
>dfCleaned.coalesce(1).write.orOverwrite("hdfs://192.168.1.100:54310/dfCleaned")
```

```
> println("Cleaned data saved to $cleanedOutputPath")
```

```

scala> val cleanedOutputPath = "hdfs://localhost:9000/datasets/resultCleaned"
cleanedOutputPath: String = hdfs://localhost:9000/datasets/resultCleaned

scala> dfCleaned.coalesce(1).write.option("header", "true").csv(cleanedOutputPath)

scala> println(s"Cleaned data saved to $cleanedOutputPath")
Cleaned data saved to hdfs://localhost:9000/datasets/resultCleaned

```

- c) Lakukan data aggregation yang menghitung rata-rata rating dan jumlah rating per produk.

```

> val dfAggregated =
dfCleaned.groupBy("productId").agg(avg("rating"),
alias("average_rating"),count("reviewerId").alias("review_count"))

scala> val dfAggregated = dfCleaned.groupBy("productId").agg(avg("rating").alias("average_
rating"),count("reviewerId").alias("review_count"))
dfAggregated: org.apache.spark.sql.DataFrame = [productId: string, average_rating: double
... 1 more field]

```

- d) Simpan hasil data aggregation ke HDFS.

```

>           val           aggregatedOutputPath      =
"hdfs://localhost:9000/datasets/aggregated_data"
>dfAggregated.coalesce(1).write.option("header","true").csv(aggregat
e
dOutputPath)
> println(s"Aggregated data saved to $aggregatedOutputPath")

scala> val aggregatedOutputPath = "hdfs://localhost:9000/datasets/aggregated_data"
aggregatedOutputPath: String = hdfs://localhost:9000/datasets/aggregated_data

scala> dfAggregated.coalesce(1).write.option("header", "true").csv(aggregatedOutputPath)

scala> println(s"Aggregated data saved to $aggregatedOutputPath")
Aggregated data saved to hdfs://localhost:9000/datasets/aggregated_data

```

- e) Filter produk-produk yang populer, dengan rating di atas 4.5 dan jumlah pemberian rating di atas 50.

```

> val popularProducts = dfAggregated.filter(col("average_rating") > 4.5
&& col("review_count") > 50)

```

```

scala> val popularProducts = dfAggregated.filter(col("average_rating") > 4.5 && col("revie
w_count") > 50)
popularProducts: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [productId: stri
ng, average_rating: double ... 1 more field]

```

- f) Simpan hasil filtering ke HDFS.

```

>           val           popularOutputPath      =
"hdfs://localhost:9000/datasets/popular_products"
>popularProducts.coalesce(1).write.option("header","true").csv(popular
OutputPath)
> println(s"Popular products data saved to $popularOutputPath")

```

```

scala> val popularOutputPath = "hdfs://localhost:9000/datasets/popular_products"
popularOutputPath: String = hdfs://localhost:9000/datasets/popular_products

scala> popularProducts.coalesce(1).write.option("header", "true").csv(popularOutputPath)

scala> println(s"Popular products data saved to $popularOutputPath")
Popular products data saved to hdfs://localhost:9000/datasets/popular_products

```

- g) Distribusikan rating setiap produk.

```

>   val   ratingDistribution    =   dfCleaned.groupBy("productId",
"rating").count().withColumnRenamed("count",
"rating_count").orderBy("productId", "rating")

```

```

scala> val ratingDistribution = dfCleaned.groupBy("productId", "rating").count().withColumnRenamed("count", "rating_count").orderBy("productId", "rating")
ratingDistribution: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [productId: string, rating: string ... 1 more field]

```

- h) Simpan hasil distribusi ke HDFS.

```

>           val           distributionOutputPath      =
"dfs://localhost:9000/datasets/rating_distribution"
>ratingDistribution.coalesce(1).write.option("header","true").csv(distributionOutputPath)
> println(s"Rating distribution data saved to $distributionOutputPath")
scala> val distributionOutputPath = "dfs://localhost:9000/datasets/rating_distribution"
distributionOutputPath: String = dfs://localhost:9000/datasets/rating_distribution
scala> ratingDistribution.coalesce(1).write.option("header", "true").csv(distributionOutputPath)
scala> println(s"Rating distribution data saved to $distributionOutputPath")
Rating distribution data saved to dfs://localhost:9000/datasets/rating_distribution

```

### 3.3.2 Filtering Berdasarkan Waktu

- a) Membaca file CSV

```

- val inputData = "/home/bintangs/datasets/resultCleaned.csv"
- val columns = Seq("productId", "reviewId", "rating", "unixtime", "time")
-           val       df      =       spark.read.option("header",
"true").csv(inputData).toDF(columns:_*)

```

```

scala> val inputData = "/home/bintangs/datasets/resultCleaned.csv"
val inputData: String = /home/bintangs/datasets/resultCleaned.csv

scala> val columns = Seq("productId", "reviewId", "rating", "unixtime", "time")
val columns: Seq[String] = List(productId, reviewId, rating, unixtime, time)

scala> val df = spark.read.option("header", "true").csv(inputData).toDF(columns:_*)
[Stage 0:>                                         (0 + 1) / 1

val df: org.apache.spark.sql.DataFrame = [productId: string, reviewId: string .
.. 3 more fields]

```

- b) Konversi kolom time ke format timestamp

```

val dfWithTimestamp = df.withColumn("time", to_timestamp(col("time"),
"yyyy-MM-dd HH:mm:ss"))

```

```

scala> val dfWithTimestamp = df.withColumn("time", to_timestamp(col("time"), "yyyy-MM-dd HH:mm:ss"))
val dfWithTimestamp: org.apache.spark.sql.DataFrame = [productId: string, reviewId: string ... 3 more fields]

scala> dfWithTimestamp.printSchema()
root
|-- productId: string (nullable = true)
|-- reviewId: string (nullable = true)
|-- rating: string (nullable = true)
|-- unixtime: string (nullable = true)
|-- time: timestamp (nullable = true)

```

- c) Menambahkan kolom baru untuk hari, bulan, dan tahun (tetapi disini yang dipakai hanya tahun untuk menunjukkan perubahan paling mencolok)

```

scala> val dfWithDatePart = dfWithTimestamp.withColumn("day", dayofmonth(col("time"))).withColumn("month", month(col("time"))).withColumn("year", year(col("time")))
val dfWithDatePart: org.apache.spark.sql.DataFrame = [productId: string, reviewId: string ... 6 more fields]

```

- d) Filter berdasarkan tahun untuk menunjukkan tren perubahan rata-rata rating dan jumlah ulasan yang signifikan setiap tahunnya.

```

> val yearlyPattern = dfWithDateParts.groupBy("year").agg(avg("rating").alias("average_rating"), count("rating").alias("rating_count")).orderBy("year")

```

```

scala> val yearlyPattern = dfWithDatePart.groupBy("year").agg(avg("rating").alias("average_rating"), count("rating").alias("rating_count")).orderBy("year")
val yearlyPattern: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [year: int, average_rating: double ... 1 more field]

```

- e) Simpan hasil filtering ke HDFS

```

> yearlyPattern.coalesce(1).write.option("header", "true").csv("/home/bintangs/yearly_pattern.csv")

```

```

scala> yearlyPattern.coalesce(1).write.option("header", "true").csv("/home/bintangs/datasets/yearly_pattern.csv")
[Stage 28:>                                         (0 + 3) / 3
[Stage 28:=====]>                               (2 + 1) / 3

```

### 3.3.3 Filtering Berdasarkan Penurunan Rating pada Tahun 2018

- a) Mengonversi kolom unixtime dari format UNIX timestamp menjadi kolom date dengan format tanggal yang dapat dibaca manusia.

```

> val dfWithDate = df.withColumn("date", to_date(col("unixtime").cast("timestamp")))

```

```

scala> val dfWithDate = df.withColumn("date", to_date(col("unixtime").cast("timestamp")))
dfWithDate: org.apache.spark.sql.DataFrame = [productId: string, reviewerId: string ... 4 more fields]

```

- b) Memfilter data untuk hanya menyertakan baris-baris di mana tahun pada kolom date adalah 2018.

```

> val df2018 = dfWithDate.filter(year(col("date")) === 2018)

```

```

scala> val df2018 = dfWithDate.filter(year(col("date")) === 2018)
df2018: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [productId: string, reviewerId: string ... 4 more fields]

```

- c) Menghitung perbedaan rating berdasarkan waktu menggunakan fungsi lag dengan window. Membuat window berdasarkan productId dan mengurutkan berdasarkan tanggal.

```

> import org.apache.spark.sql.expressions.Window

```

```

> val windowSpec = Window.partitionBy("productId").orderBy("date")

```

```

scala> import org.apache.spark.sql.expressions.Window
import org.apache.spark.sql.expressions.Window

```

```

scala> val windowSpec = Window.partitionBy("productId").orderBy("date")
windowSpec: org.apache.spark.sql.expressions.WindowSpec = org.apache.spark.sql.expressions.WindowSpec@683c907a

```

- d) Menambahkan kolom baru prev\_rating yang berisi nilai rating dari baris sebelumnya dalam urutan waktu berdasarkan spesifikasi window (windowSpec).

```
> val dfWithPrevRating = df2018.withColumn("prev_rating", lag("rating", 1).over(windowSpec))
```

```
scala> val dfWithPrevRating = df2018.withColumn("prev_rating", lag("rating", 1).over(windowSpec))
dfWithPrevRating: org.apache.spark.sql.DataFrame = [productId: string, reviewerId: string ... 5 more fields]
```

- e) Menghitung perubahan rating dengan menambahkan kolom rating\_change, yang merupakan selisih antara nilai rating saat ini (rating) dan rating sebelumnya (prev\_rating).

```
> val dfWithRatingChange = dfWithPrevRating.withColumn("rating_change", col("rating") - col("prev_rating"))
```

```
scala> val dfWithRatingChange = dfWithPrevRating.withColumn("rating_change", col("rating") - col("prev_rating"))
dfWithRatingChange: org.apache.spark.sql.DataFrame = [productId: string, reviewerId: string ... 6 more fields]
```

- f) Menambahkan kolom baru month yang berisi angka bulan yang diekstrak dari kolom date untuk mempermudah pengelompokan berdasarkan semester.

```
> val dfWithMonth = dfWithRatingChange.withColumn("month", month(col("date")))
```

```
scala> val dfWithMonth = dfWithRatingChange.withColumn("month", month(col("date")))
dfWithMonth: org.apache.spark.sql.DataFrame = [productId: string, reviewerId: string ... 7 more fields]
```

- g) Memisahkan menjadi dua DataFrame: Januari-Juni dan Juli-Desember. Filter untuk Januari-Juni

```
> val janJun = dfWithMonth.filter(col("month") >= 1 && col("month") <= 6)
```

```
> val avgJanJun = janJun.select(avg(col("rating_change")).alias("avg_rating_change_JanJun"))
```

```
scala> val janJun = dfWithMonth.filter(col("month") >= 1 && col("month") <= 6)
janJun: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [productId: string, reviewerId: string ... 7 more fields]
```

```
scala> val avgJanJun = janJun.select(avg(col("rating_change")).alias("avg_rating_change_JanJun"))
avgJanJun: org.apache.spark.sql.DataFrame = [avg_rating_change_JanJun: double]
```

Filter untuk Juli-Desember

```
> julDec = dfWithMonth.filter(col("month") >= 7 && col("month") <= 12)
```

```
> val avgJulDec = julDec.select(avg(col("rating_change")).alias("avg_rating_change_JulDec"))
```

```
scala> val julDec = dfWithMonth.filter(col("month") >= 7 && col("month") <= 12)
julDec: org.apache.spark.sql.Dataset[org.apache.spark.sql.Row] = [productId: string, reviewerId: string ... 7 more fields]
```

```
scala> val avgJulDec = julDec.select(avg(col("rating_change")).alias("avg_rating_change_JulDec"))
avgJulDec: org.apache.spark.sql.DataFrame = [avg_rating_change_JulDec: double]
```

- h) Menghitung rata-rata untuk masing-masing semester.

```
> val avgJanJunValue = avgJanJun.collect()(0).getAs[Double]("avg_rating_change_JanJun")
```

```

> val avgJulDecValue = avgJulDec.collect()(0).getAs[Double]("avg_rating_change_JulDec")
scala> val avgJanJunValue = avgJanJun.collect()(0).getAs[Double]("avg_rating_change_JanJun")
avgJanJunValue: Double = -0.0070013357079252

scala> val avgJulDecValue = avgJulDec.collect()(0).getAs[Double]("avg_rating_change_JulDec")
avgJulDecValue: Double = -0.009957817578314087

```

- i) Menghitung perubahan rata-rata.

```
> val changeRating = avgJulDecValue - avgJanJunValue
```

```
scala> val changeRating = avgJulDecValue - avgJanJunValue
changeRating: Double = -0.0029564818703888864
```

- j) Membuat DataFrame hasil akhir.

```
> val result = Seq(("All Products", avgJanJunValue, avgJulDecValue,
changeRating)).toDF("ProductId", "Avg_Rating_Change_JanJun",
"Avg_Rating_Change_JulDec", "Change_Rating")
```

```
scala> val result = Seq(("All Products", avgJanJunValue, avgJulDecValue, changeRating)).toDF("ProductId", "Avg_Rating_Change_JanJun", "Avg_Rating_Change_JulDec", "Change_Rating")
result: org.apache.spark.sql.DataFrame = [ProductId: string, Avg_Rating_Change_JanJun: double ... 2 more fields]
```

- k) Hasil akhir menunjukkan bahwa rata-rata penurunan rating pada Januari-Juni adalah -0.007, sedangkan pada Juli-Desember adalah -0.01, dengan perubahan rata-rata sebesar -0.003. Ini mengindikasikan bahwa penurunan rating lebih signifikan pada Juli-Desember dibandingkan Januari-Juni.

```
scala> result.show()
+-----+-----+-----+
| ProductId|Avg_Rating_Change_JanJun|Avg_Rating_Change_JulDec|    Change_Rating|
+-----+-----+-----+
|All Products|      -0.0070013357079252|      -0.00995781757831...|-0.00295648187038...|
+-----+-----+-----+
```

### 3.3.4 Analisis Sentimen Berdasarkan Review Pengguna

- a) Instalasi Pyspark dan NLTK.

```
aliyah@Ubuntu20:~$ pip3 install pyspark nltk
```

- b) Melakukan data cleaning.

Berikut adalah kode yang digunakan:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import regexp_replace
from pyspark.sql.types import StructType, StructField,
StringType, DoubleType

# Membuat sesi Spark
spark = SparkSession.builder.appName("JSON
Cleaning").getOrCreate()

# Mendefinisikan skema untuk file JSON
schema = StructType([
    StructField("id", StringType(), True),
    StructField("asin", StringType(), True),
    StructField("reviewerID", StringType(), True),
    StructField("helpful", DoubleType(), True),
    StructField("rating", DoubleType(), True),
    StructField("summary", StringType(), True),
    StructField("text", StringType(), True)
])
```

```

        StructField("overall", DoubleType(), True),
        StructField("reviewerID", StringType(), True),
        StructField("asin", StringType(), True),
        StructField("reviewText", StringType(), True),
    ]
)

# Membaca file JSON dengan skema yang ditentukan dari HDFS
df =
spark.read.schema(schema).json("hdfs://localhost:9000/datasets/Musical_Instruments_5.json")

# Mengganti newline di reviewText dengan spasi
df_cleaned = df.withColumn("reviewText",
regexp_replace("reviewText", "\n", " "))

# Mengganti nama kolom 'asin' menjadi 'productID'
df_cleaned = df_cleaned.withColumnRenamed("overall",
"rating").withColumnRenamed("asin", "productID")

# Menampilkan 5 baris dari hasil cleaning
df_cleaned.show(5)

# Menulis data ke file CSV di HDFS
df_cleaned.coalesce(1).write.option("header",
"true").mode("overwrite").csv("hdfs://localhost:9000/datasets/cleaned_output/")

```

Menjalankan program:

```
aliyah@Ubuntu20:~/Downloads$ python3 cleaning.py
```

Hasil:

| rating | reviewerID     | productID  | reviewText           |
|--------|----------------|------------|----------------------|
| 5.0    | A3F05AKVTFRCRJ | 0739079891 | It's good for beg... |
| 5.0    | A3UCGC1DHFMBC  | 0739079891 | I recommend this ... |
| 5.0    | A2S9SLRYLPGYZB | 0739079891 | G'daughter receiv... |
| 4.0    | A15RTJWPG8OKOE | 0739079891 | According to my o... |
| 1.0    | A12ET1W030AVU7 | 0739079891 | Please pay attent... |

only showing top 5 rows

```
aliyah@Ubuntu20:~/Downloads$ hdfs dfs -ls /datasets/cleaned_output/
Found 2 items
-rw-r--r-- 3 aliyah supergroup          0 2024-12-14 18:09 /datasets/cleaned_output/_SUCCESS
-rw-r--r-- 3 aliyah supergroup 78433060 2024-12-14 18:09 /datasets/cleaned_output/part-00000-bc32e7dd-02b0-4ded-8652-a7312728033a-c000.csv
```

- c) Melakukan analisis sentimen terhadap ulasan pengguna menggunakan NLTK.

Berikut adalah kode yang digunakan:

```
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, udf
from pyspark.sql.types import StringType
import nltk
from nltk.sentiment import SentimentIntensityAnalyzer

# Inisialisasi Spark session
spark =
SparkSession.builder.appName("SentimentAnalysis").getOrCreate()

# Memuat data ulasan dari file CSV pada HDFS
data_path =
"hdfs://localhost:9000/datasets/cleaned_output/part-00000-bc
32e7dd-02b0-4ded-8652-a7312728033a-c000.csv"
df = spark.read.option("header",
"true").option("inferSchema", "true").csv(data_path)

# Menampilkan schema dan beberapa baris data
df.printSchema()
df.show(5)

# Inisialisasi SentimentIntensityAnalyzer dari NLTK
nltk.download('vader_lexicon')
sia = SentimentIntensityAnalyzer()

# Fungsi untuk analisis sentimen menggunakan NLTK
def get_sentiment(rating, text):
    if text is None or text.strip() == "":
        return "Neutral"

    score = sia.polarity_scores(text)

    if score['compound'] > 0.2:
        return "Positive"
    elif score['compound'] < -0.2:
        return "Negative"
    else:
        return "Neutral"
```

```

# UDF untuk Spark
sentiment_udf = udf(get_sentiment, StringType())

# Menerapkan UDF untuk kolom 'reviewText' dan menambah kolom
'sentiment'
df = df.withColumn("sentiment", sentiment_udf(col("rating"),
col("reviewText")))

# Menyimpan hasil distribusi sentimen ke HDFS
df.groupBy("sentiment").count().coalesce(1).write.option("he
ader",
"true").mode("overwrite").csv("hdfs://localhost:9000/dataset
s/sentiment_distribution/")

# Menampilkan hasil analisis sentimen
df.select("productID", "rating", "reviewText",
"sentiment").show(10)

```

Kode di atas bertujuan untuk melakukan analisis sentimen terhadap ulasan produk yang terdapat dalam file input. Pertama, data dibaca menggunakan PySpark dan diolah untuk menambah kolom baru bernama "sentiment" yang berisi hasil analisis sentimen dari teks ulasan menggunakan SentimentIntensityAnalyzer milik NLTK. Fungsi get\_sentiment menganalisis teks ulasan dan rating produk untuk menentukan apakah sentimen ulasan tersebut positif, negatif, atau netral. Ulasan yang memiliki rating di bawah 2.5 akan langsung dikategorikan sebagai negatif.

Berikut adalah hasil dari analisis sentimen untuk setiap ulasan yang diberikan pengguna terhadap produk tertentu:

| productID  | rating                   | reviewText | sentiment |
|------------|--------------------------|------------|-----------|
| 0739079891 | 5.0 It's good for beg... | Positive   |           |
| 0739079891 | 5.0 I recommend this ... | Positive   |           |
| 0739079891 | 5.0 G'daughter receiv... | Positive   |           |
| 0739079891 | 4.0 According to my o... | Positive   |           |
| 0739079891 | 1.0 Please pay attent... | Negative   |           |
| 0739079891 | 4.0 I purchased this ... | Positive   |           |
| 0739079891 | 5.0  thanx, b            | Neutral    |           |
| 0739079891 | 4.0 Good cheap ukulel... | Positive   |           |
| 0739079891 | 5.0 My grandson is ve... | Positive   |           |
| 0739079891 | 3.0  Good basic guide.   | Positive   |           |

only showing top 10 rows

```
aliyah@Ubuntu20:~/Downloads$ hdfs dfs -ls /datasets/sentiment_distribution/
Found 2 items
-rw-r--r-- 3 aliyah supergroup          0 2024-12-14 18:15 /datasets/sentimen
t_distribution/_SUCCESS
-rw-r--r-- 3 aliyah supergroup      61 2024-12-14 18:15 /datasets/sentimen
t_distribution/part-00000-ab485838-6b6a-4f2e-aae2-db2846e9c0c3-c000.csv
```

Selanjutnya akan ditampilkan distribusi sentimen, yaitu menghitung total ulasan yang termasuk dalam kategori sentimen positif, netral, dan negatif:

```
# Distribusi Sentimen
df.groupBy("sentiment").count().show()
```

| sentiment | count  |
|-----------|--------|
| Positive  | 183314 |
| Neutral   | 23031  |
| Negative  | 25047  |

Kode di bawah ini bertujuan untuk mengelompokkan data ulasan berdasarkan kolom productID dan sentiment, kemudian menghitung jumlah ulasan untuk setiap kombinasi productID dan sentiment. Setelah pengelompokan dan perhitungan jumlah ulasan, hasilnya diurutkan berdasarkan kolom productID secara menaik. Hasilnya dapat digunakan untuk menganalisis distribusi sentimen ulasan berdasarkan produk.

```
# Mengelompokkan berdasarkan kolom 'productID' dan
'sentiment', kemudian menghitung jumlah ulasan
df_grouped_by_product_sentiment = df.groupBy("productID",
"sentiment").count()
df_grouped_by_product_sentiment =
df_grouped_by_product_sentiment.orderBy(col("productID").asc
())
df_grouped_by_product_sentiment.coalesce(1).write.option("he
ader",
"true").mode("overwrite").csv("hdfs://localhost:9000/dataset
s/product_sentiment/")
df_grouped_by_product_sentiment.show(10)
```

```
+-----+-----+-----+
| productID|sentiment|count|
+-----+-----+-----+
|0739079891| Positive| 26|
|0739079891| Neutral| 2|
|0739079891| Negative| 7|
|0786615206| Negative| 1|
|0786615206| Neutral| 1|
|0786615206| Positive| 3|
|1480360295| Positive| 19|
|1480360295| Neutral| 5|
|1480360295| Negative| 10|
|1928571018| Positive| 6|
+-----+-----+-----+
only showing top 10 rows
```

```
aliyah@Ubuntu20:~/Downloads$ hdfs dfs -ls /datasets/product_sentiment/
Found 2 items
-rw-r--r-- 3 aliyah supergroup      0 2024-12-14 18:17 /datasets/
product_sentiment/_SUCCESS
-rw-r--r-- 3 aliyah supergroup 527775 2024-12-14 18:17 /datasets/
product_sentiment/part-00000-85c2a21c-9ca1-46de-8b26-65d33e3263e1-c000.
csv
```

Kode di bawah ini bertujuan untuk menghasilkan daftar 20 produk dengan jumlah ulasan positif terbanyak. Langkah pertama adalah memfilter data untuk hanya menyertakan ulasan dengan sentimen "Positive". Kemudian, data tersebut dikelompokkan berdasarkan productID, dan jumlah ulasan untuk setiap produk akan dihitung menggunakan fungsi count(). Data akan diurutkan berdasarkan jumlah ulasan positif secara menurun, dan dibatasi hanya pada 20 produk teratas.

```
# 20 produk dengan sentimen positif terbanyak
df_positive_sentiment = df.filter(col("sentiment") == "Positive")
df_positive_top_20 = df_positive_sentiment.groupBy("productID").count().withColumnRenamed("count", "positive_review_count").orderBy("positive_review_count", ascending=False).limit(20)
df_positive_top_20.coalesce(1).write.option("header", "true").mode("overwrite").csv("hdfs://localhost:9000/datasets/top20_sentiment")
df_positive_top_20.show(5)
```

```
+-----+-----+
| productID|positive_review_count|
+-----+-----+
|B0006LOBA8|          1569|
|B0002E3CK4|          1569|
|B0002H03YY|          1569|
|B0002H05BA|          1569|
|B0002E1NWI|          1280|
+-----+-----+
only showing top 5 rows
```

```
aliyah@Ubuntu20:~/Downloads$ hdfs dfs -ls /datasets/top20_sentiment/
Found 2 items
-rw-r--r--  3 aliyah supergroup          0 2024-12-14 18:18 /datasets/top20_se
ntiment/_SUCCESS
-rw-r--r--  3 aliyah supergroup      340 2024-12-14 18:18 /datasets/top20_se
ntiment/part-00000-00ee804a-4dc2-41d8-b9a8-3ad89ef8d71f-c000.csv
```

### 3.4 Visualisasi Hasil

- a) Simpan data dari HDFS ke penyimpanan lokal.

```
> hdfs dfs -get /datasets/resultCleaned/part-00000-* .csv ./resultCleaned.csv
> hdfs dfs -get /datasets/aggregated_data/part-00000-* .csv ./aggregated_data.csv
> hdfs dfs -get /datasets/popular_products/part-00000-* .csv ./popular_products.csv
> hdfs dfs -get /datasets/rating_distribution/part-00000-* .csv ./rating_distribution.csv
```

```
vboxuser@Nireiv:~$ hdfs dfs -get /datasets/resultCleaned/part-00000-* .csv ./resu
ltCleaned.csv
vboxuser@Nireiv:~$ hdfs dfs -get /datasets/resultCleaned/part-00000-* .csv ./resu
ltCleaned.csv
vboxuser@Nireiv:~$ hdfs dfs -get /datasets/aggregated_data/part-00000-* .csv ./ag
gregated_data.csv
vboxuser@Nireiv:~$ hdfs dfs -get /datasets/popular_products/part-00000-* .csv ./p
opular_products.csv
vboxuser@Nireiv:~$ hdfs dfs -get /datasets/rating_distribution/part-00000-* .csv
./rating_distribution.csv
```

```
aliyah@Ubuntu20:~/Downloads$ hdfs dfs -get /datasets/product_sentiment/part-000
00-* .csv ./product_sentiment.csv
```

- b) Ubah format file dari CSV ke SQLite.



```
convertSQL.py
Open ▾ Save ⌂ ×
1 import pandas as pd
2 import sqlite3
3
4 # Baca data dari CSV
5 df_cleaned = pd.read_csv('resultCleaned.csv')
6 df_aggregated = pd.read_csv('aggregated_data.csv')
7 df_popular = pd.read_csv('popular_products.csv')
8 df_distribution = pd.read_csv('rating_distribution.csv')
9
10 # Simpan ke SQLite
11 conn = sqlite3.connect("spark_results.db")
12 df_cleaned.to_sql("cleaned_data", conn, if_exists="replace", index=False)
13 df_aggregated.to_sql("aggregated_data", conn, if_exists="replace", index=False)
14 df_popular.to_sql("popular_products", conn, if_exists="replace", index=False)
15 df_distribution.to_sql("rating_distribution", conn, if_exists="replace", index=False)
16 conn.close()
```

- c) Navigasi ke Home > Connection > Add new connection > SQLite > Add new data source untuk menghubungkan Grafana dengan database SQLite.

Name: Amazon Musical Instrument Product

Path: /var/lib/grafana/spark\_results.db

Path Prefix: file:

Path Options: mode=ro&\_ignore\_check\_constraints=1

Secure Path:

Options:

Attach Limit: 0

**File System Permissions**

The plugin runs with the same permissions as the Grafana user. Any file that can be opened with the Grafana user can be opened with the SQLite plugin.  
Beware that by enabling attaching databases (setting an "attach limit" above 0) you enable any user with access to the plugin to attach any database that the Grafana user has access to.  
It is the most secure (and recommended) approach to set the "attach limit" to 0.

**Data source is working**

Next, you can start to visualize data by [building a dashboard](#), or by querying data in the [Explore view](#).

- d) Navigasi ke Home > Dashboards > New > New dashboard > Add visualization > Select data source > Amazon Musical Instrument Product untuk membuat tampilan dashboard baru.

Panel Title

No data

Queries 1 Transformations 0 Alert 0

Data source: Amazon Musical Instrument Product

```
SELECT CAST(strftime('%s', 'now', '-1 minute') as INTEGER) as time, 4 as value
WHERE time >= $_from / 1000 and time < $_to / 1000
```

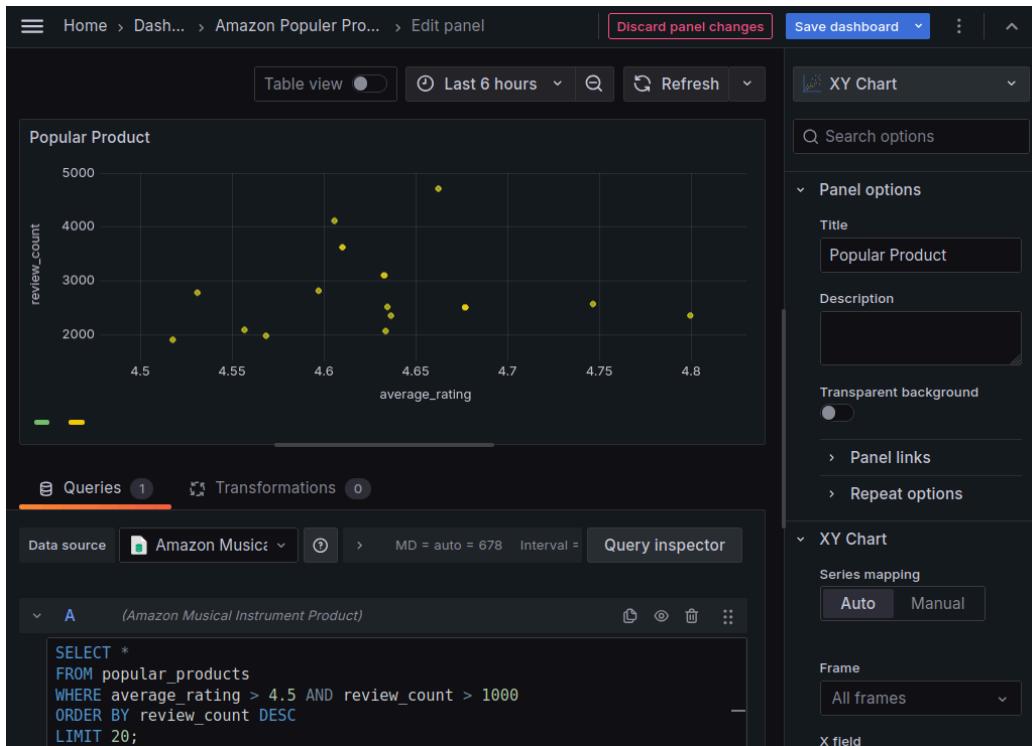
Time series

- e) Buat panel baru dengan menambahkan query SQL dan memilih jenis visualisasi data yang sesuai.  
> SELECT \*

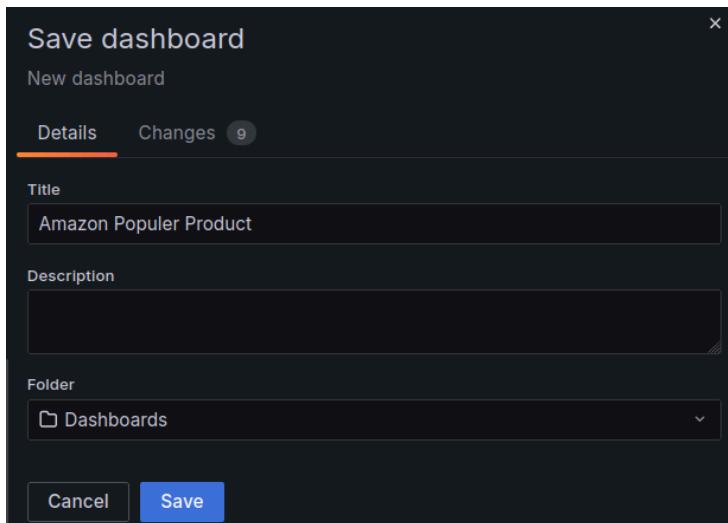
```

FROM popular_products
WHERE average_rating > 4.5 AND review_count > 1000
ORDER BY review_count DESC
LIMIT 20;

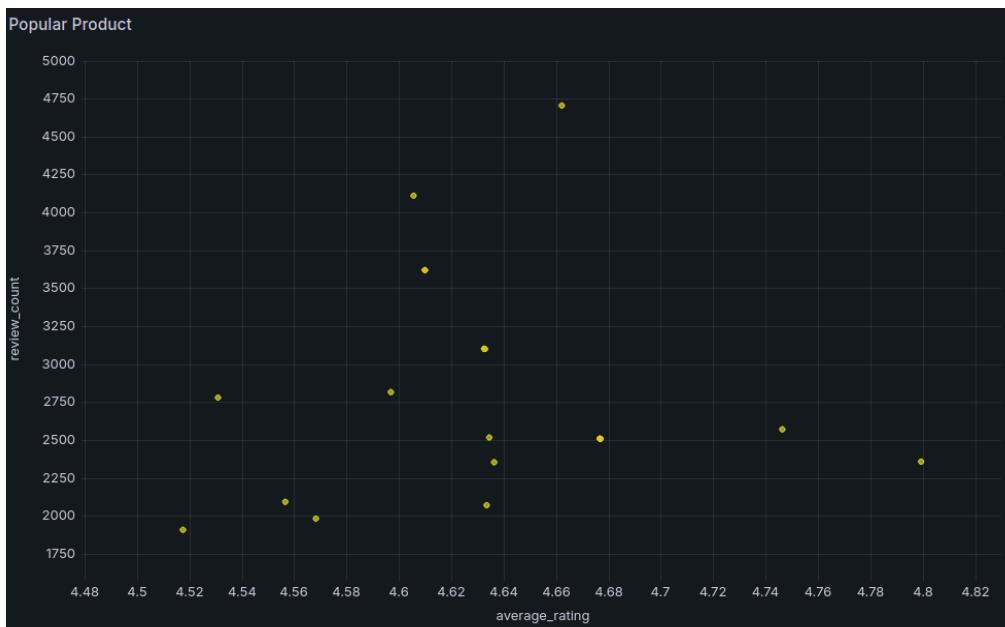
```



- f) Simpan dashboard yang telah dibuat.



- g) Lihat kembali panel yang telah dibuat.



- h) Versi tabel dari daftar produk terpopuler.

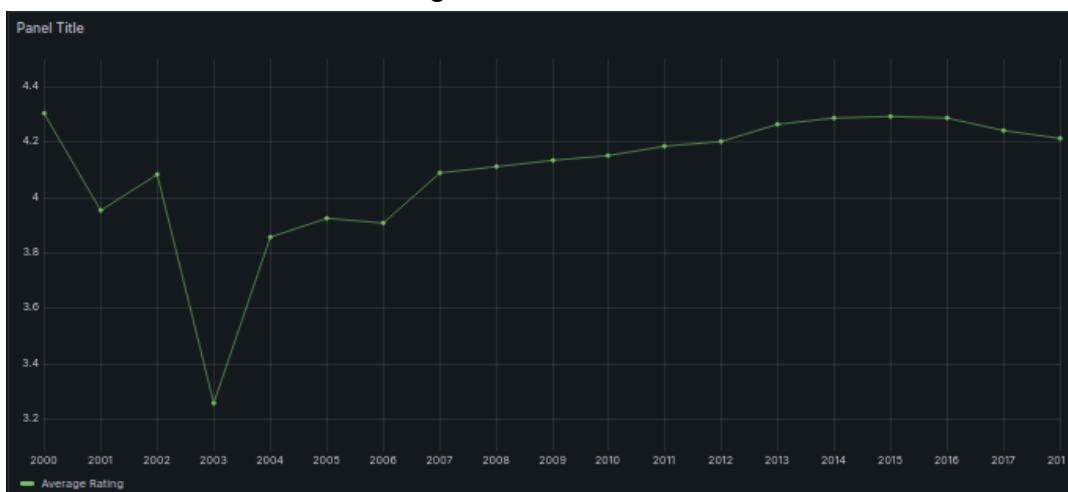
| productId  | average_rating | review_count |
|------------|----------------|--------------|
| B004XNK7AI | 4.66           | 4704         |
| B00IFOTSJW | 4.61           | 4110         |
| B0002E1NWI | 4.61           | 3620         |
| B0002E1NNC | 4.61           | 3620         |
| B0007Y09VO | 4.63           | 3102         |
| B0002H0A3S | 4.63           | 3100         |
| B00IFHCOAE | 4.60           | 2816         |
| B00063678K | 4.53           | 2780         |
| B0004L3F9E | 4.75           | 2571         |
| B001PGXHXA | 4.63           | 2517         |
| B0002H03YY | 4.68           | 2509         |
| B0006LOBA8 | 4.68           | 2509         |
| B0002E3CK4 | 4.68           | 2509         |

- i) Visualisasi data dari distribusi rating setiap produk dalam bentuk tabel.

```
> SELECT *
  FROM rating_distribution
  ORDER BY productId ASC;
```

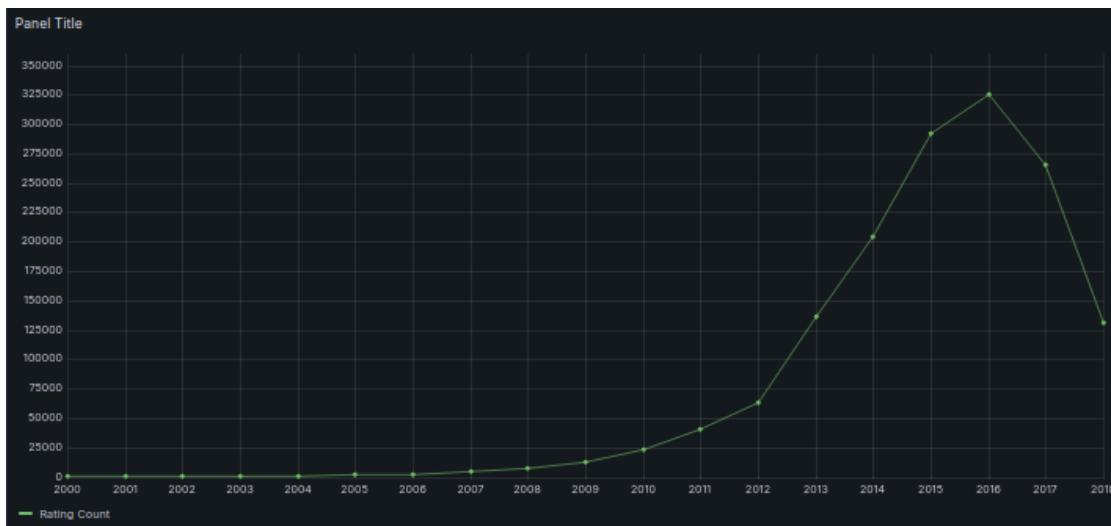
| Rating Distribution |        |              |  |
|---------------------|--------|--------------|--|
| productId           | rating | rating_count |  |
| 0000098906          | 1      | 1            |  |
| 0000098906          | 4      | 2            |  |
| 0000098906          | 5      | 7            |  |
| 0000989983          | 5      | 2            |  |
| 0041291905          | 5      | 1            |  |
| 0060015500          | 5      | 1            |  |
| 0193757710          | 5      | 1            |  |
| 0470536454          | 1      | 5            |  |
| 0470536454          | 2      | 10           |  |
| 0470536454          | 3      | 12           |  |
| 0470536454          | 4      | 46           |  |
| 0470536454          | 5      | 168          |  |
| 0594478944          | 4      | 1            |  |
| 0594478944          | 5      | 1            |  |

- j) Visualisasi data dari pola tahunan jumlah ulasan dan rata-rata rating produk.
- Berdasarkan Rata-rata Rating Produk



Grafik di atas menunjukkan tren rata-rata rating (Average Rating) yang fluktuatif pada awal periode, dengan penurunan tajam sekitar tahun 2003, namun stabil dan meningkat secara perlahan sejak 2006 hingga mencapai puncak stabilitas pada sekitar 2014, sebelum sedikit menurun menjelang 2018.

- Berdasarkan jumlah ulasan



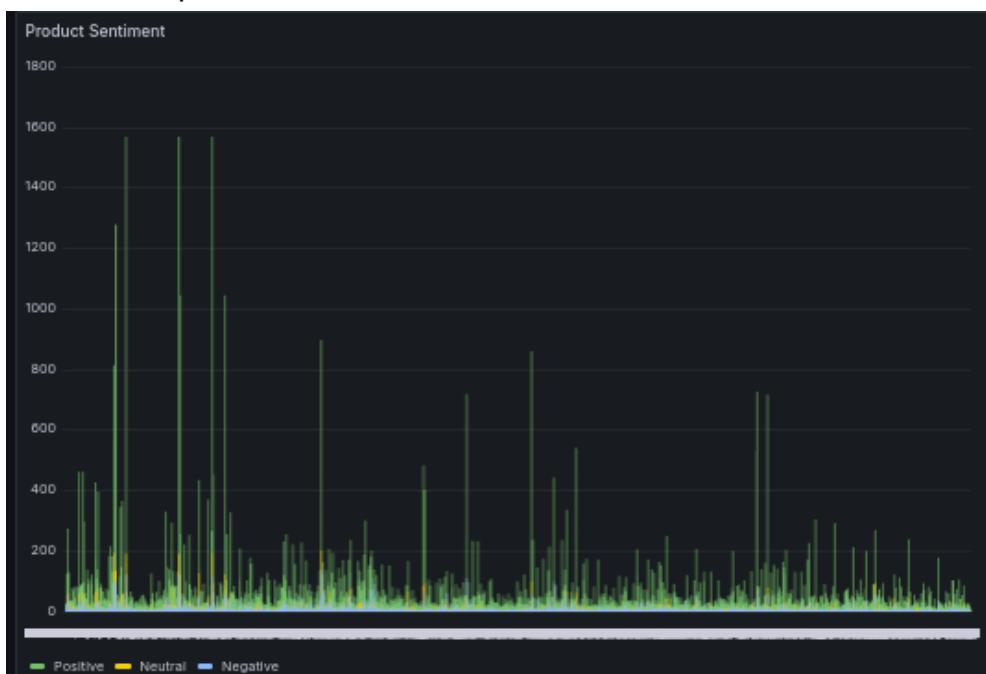
Grafik di atas menunjukkan tren jumlah ulasan yang meningkat tajam hingga mencapai puncak pada tahun 2016, kemudian menurun secara signifikan hingga tahun 2018.

- k) Visualisasi sentimen berdasarkan produk.

> SELECT

```

productID,
SUM(CASE WHEN sentiment = 'Positive' THEN count ELSE 0 END)
AS Positive,
SUM(CASE WHEN sentiment = 'Neutral' THEN count ELSE 0 END)
AS Neutral,
SUM(CASE WHEN sentiment = 'Negative' THEN count ELSE 0 END)
AS Negative
FROM product_sentiment
GROUP BY productID
ORDER BY productID;
```



## **BAB IV**

### **KESIMPULAN**

Proyek ini berhasil menganalisis data rating dan ulasan pelanggan Amazon pada kategori Musical Instruments dengan memanfaatkan teknologi Big Data, seperti HDFS dan Apache Spark. Dengan HDFS, data dalam jumlah besar dapat disimpan secara terdistribusi, sehingga mempermudah pengelolaan dan memastikan data tetap aman untuk proses analisis. Teknologi ini memungkinkan penanganan dataset yang kompleks dan beragam format secara efisien.

Pemrosesan data menggunakan Apache Spark memberikan kecepatan dan akurasi tinggi, terutama dalam memfilter data untuk kategori yang relevan. Proses ini menghasilkan informasi penting seperti performa produk berdasarkan rating pelanggan. Teknologi ini membantu meringkas data besar menjadi data yang mudah dimengerti untuk kebutuhan analisis lebih lanjut.

Hasil analisis divisualisasikan menggunakan Grafana, yang menyajikan data dalam bentuk grafik informatif untuk memudahkan pengambil keputusan memahami pola rating dan sentimen pelanggan. Analisis ini mampu mengidentifikasi produk dengan rating tertinggi dan sentimen positif terbanyak, memberikan manfaat bagi pelanggan untuk memilih produk terbaik, sekaligus mendukung pelaku bisnis dalam menentukan strategi pasar yang lebih efektif. Informasi ini membantu bisnis meningkatkan kualitas produk dan merancang langkah pemasaran berbasis data yang lebih terarah.

Analisis pada kategori Musical Instruments menunjukkan bahwa produk-produk dalam kategori ini memiliki pola konsumsi unik. Pemahaman yang lebih mendalam tentang preferensi konsumen di kategori ini memberikan nilai tambah dalam meningkatkan pengalaman pelanggan dan menyusun strategi pemasaran yang lebih efektif. Pendekatan ini dapat diadaptasi untuk analisis data pada kategori produk lain di masa depan guna mendukung pengambilan keputusan berbasis data.

## DAFTAR REFERENSI

"What is Big Data," javatpoint.com. [Online]. Available: <https://www.javatpoint.com/what-is-big-data>. [Accessed: Dec. 6, 2024].

C.H Pour, B. Botelho, S.J Bigelow, "big data," techtarget.com. [Online]. Available: <https://www.techtarget.com/searchdatamanagement/definition/big-data>. [Accessed: Dec. 6, 2024].

"Hadoop Distributed File System (HDFS)" databricks.com. [Online]. Available: <https://www.databricks.com/glossary/hadoop-distributed-file-system-hdfs>. [Accessed: Dec. 6, 2024].

"Hadoop - MapReduce," tutorialspoint.com. [Online]. Available: [https://www.tutorialspoint.com/hadoop/hadoop\\_mapreduce.htm](https://www.tutorialspoint.com/hadoop/hadoop_mapreduce.htm). [Accessed: Dec. 6, 2024].

"Apache Spark - Introduction," tutorialspoint.com. [Online]. Available: [https://www.tutorialspoint.com/apache\\_spark/apache\\_spark\\_introduction.htm](https://www.tutorialspoint.com/apache_spark/apache_spark_introduction.htm). [Accessed: Dec. 6, 2024].

"Apache Spark," databricks.com. [Online]. Available: <https://www.databricks.com/glossary/what-is-apache-spark>. [Accessed: Dec. 6, 2024].

MetricFire, "What is Grafana?" medium.com, Aug. 2023. [Online]. Available: <https://medium.com/@MetricFire/what-is-grafana-8de44d241765>. [Accessed: Dec. 15, 2024].

"Natural Language ToolKit (NLTK)" javatpoint.com. [Online]. Available: <https://www.javatpoint.com/natural-language-toolkit>. [Accessed: Dec. 15, 2024].