LAPORAN QUIZ 2

BIG DATA

Oleh :

AHMAD FAZA ALFAN FASHLAH	**2241720186**
MOCHAMMAD ZAKARO AL FAJRI	**2241720175**
RIO BAGAS HERMAWAN	**2241720193**
SONY FEBRI HARI WIBOWO	**2241720202**

D-IV TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

POLITEKNIK NEGERI MALANG

2025

Dalam pengerjaan kuis ini kami menggunakan dua metode yaitu menggunakan GraphX via spark shell dan Graph Frames via jupyter notebook. berikut ini langkah-langkah dalam pengerjaan kuis 2 kami

Langkah - langkah penyelesaian:

1. **Memuat Data**
   - Data diperoleh melalui **"Amazon Product Co-purchasing Network"**

2. **Preprocessing**

   Preprocessing adalah langkah awal penting untuk membersihkan dan membentuk ulang data mentah agar siap dianalisis dengan algoritma graf atau metode lainnya. Tanpa preprocessing, hasil analisis bisa keliru atau gagal dijalankan.

   - Menggunakan GraphX

[1]:
```
# inisialisasi spark
from pyspark.sql import SparkSession
from pyspark import SparkContext
from pyspark.sql.functions import col

# Inisialisasi SparkSession dan SparkContext
spark = SparkSession.builder \
    .appName("Amazon PageRank") \
    .getOrCreate()

sc = spark.sparkContext
```

[3]:
```
# Load file .mtx sebagai RDD
raw_rdd = sc.textFile("com-Amazon.mtx")
```

[4]:
```
# Hapus baris komentar (dimulai dengan '%') dan header (baris pertama)
edges_rdd = raw_rdd \
    .filter(lambda line: not line.startswith('%')) \
    .zipWithIndex() \
    .filter(lambda x: x[1] > 0) \
    .map(lambda x: x[0])
```

[5]:
```
# Ubah string menjadi pasangan integer (source, destination)
edges = edges_rdd.map(lambda line: tuple(map(int, line.strip().split())))
```

[7]:
```
# Simpan ke file sebagai edge list bersih (untuk debugging atau analisis manual)
edges.map(lambda x: f"{x[0]}\t{x[1]}").saveAsTextFile("com-cleaned_amazon.mtx")
```

   - Menggunakan Graph Frames

[5]:
```
# 1 Baca file mtx sebagai RDD baris per baris
rdd_raw = spark.sparkContext.textFile("com-Amazon.mtx")

# Buang baris komentar (biasanya diawali '%')
rdd_data = rdd_raw.filter(lambda line: not line.startswith('%'))

# Tampilkan beberapa baris pertama
rdd_data.take(10)
```

[6]:
```python
# 2 Pisahkan baris jadi kolom
edges_rdd = rdd_data.map(lambda line: line.strip().split())

# Ambil src dan dst saja (abaikan kolom ke-3 kalau ada)
edges_rdd = edges_rdd.map(lambda cols: (int(cols[0]), int(cols[1])))

# Buat DataFrame edges
edges_df = edges_rdd.toDF(["src", "dst"])
edges_df.show(10)
```

[7]:
```python
# 3 Ambil semua id unik dari src dan dst
vertices_rdd = edges_rdd.flatMap(lambda x: [x[0], x[1]]).distinct()

# Konversi jadi DataFrame vertices
vertices_df = vertices_rdd.map(lambda x: (x,)).toDF(["id"])
vertices_df.show(10)
```

3. Membuat Graph dan Menjalankan Algoritma PageRank

- Menggunakan Spark Shell
  1. Inisialisasi fungsi :

```
def runPageRank(): Unit = {
  import org.apache.spark.graphx._
  import org.apache.spark.rdd.RDD

  val raw = sc.textFile("/opt/spark_data/com-cleaned_amazon.mtx")

  val edgesRDD = raw
    .filter(line => !line.startsWith("%"))
    .zipWithIndex()
    .filter { case (_, idx) => idx > 0 }
    .map { case (line, _) => line }

  val edges: RDD[Edge[Int]] = edgesRDD.map { line =>
    val fields = line.split("\\s+")
    Edge(fields(0).toLong, fields(1).toLong, 1)
  }

  val vertices: RDD[(VertexId, String)] = edges
    .flatMap(e => Seq(e.srcId, e.dstId))
    .distinct()
    .map(id => (id, s"Product-$id"))

  val graph = Graph(vertices, edges)

  val ranks = graph.pageRank(0.001).vertices

  val ranksByProduct = vertices.join(ranks).map {
    case (id, (name, rank)) => (name, rank)
```

```
  }

  val top10 = ranksByProduct.sortBy(-_._2).take(10)

  println("Top 10 produk berdasarkan PageRank:")
  top10.zipWithIndex.foreach { case ((name, rank), idx) =>
    println(s"""Rank ${idx + 1} : "$name", PageRank : "$rank"""")
  }
}
```

2. Pemanggilan fungsi :

```
runPageRank()
```

- Menggunakan Graph Frames

```
from graphframes import GraphFrame

# Buat Graph
graph = GraphFrame(vertices_df, edges_df)

# Jalankan PageRank
result = graph.pageRank(resetProbability=0.15, maxIter=10)

# Tampilkan hasil PageRank
print("Daftar produk yang diurutkan berdasarkan skor PageRank dari
tertinggi ke terendah: ")
result.vertices.select("id", "pagerank").orderBy("pagerank",
ascending=False).show(10)
```

4. Menganalisa Hasil

- Menggunakan GraphX

```
scala> runPageRank()
Top 10 produk berdasarkan PageRank:
Rank 1 : "Product-27287", PageRank : "60.525230395118705"
Rank 2 : "Product-12994", PageRank : "48.43773625341467"
Rank 3 : "Product-6410", PageRank : "46.13702530038872"
Rank 4 : "Product-3958", PageRank : "42.14429936240926"
Rank 5 : "Product-1991", PageRank : "41.75744576962717"
Rank 6 : "Product-832", PageRank : "41.471319032443674"
Rank 7 : "Product-33532", PageRank : "41.18802376941552"
Rank 8 : "Product-4493", PageRank : "40.4135582156341"
Rank 9 : "Product-11097", PageRank : "40.12641661181258"
Rank 10 : "Product-23200", PageRank : "39.717353664059196"
```

- Menggunakan GraphFrames

```
Daftar produk yang diurutkan berdasarkan skor PageRank dari tertinggi ke terendah:
+-----+------------------+
|   id|          pagerank|
+-----+------------------+
|27287| 60.52333936844408|
|12994| 48.50092854672143|
| 6410| 46.19725016853048|
| 3958|42.230366050738134|
| 1991|41.796636413874964|
|  832| 41.66108216095725|
|33532|41.206145538503925|
| 4493| 40.56672444690634|
|11097|40.157690738311125|
|23200|39.742303647024144|
+-----+------------------+
only showing top 10 rows
```

5. Interpretasi

Berdasarkan hasil dari metode yang kami gunakan, Produk ID 27287 memiliki skor PageRank tertinggi (60.52), mengindikasikan bahwa produk ini memiliki peran paling sentral dan berpengaruh dalam jaringan hubungan antar produk. Produk lain seperti 12994, 6410, dan 3958 juga menempati posisi penting dengan skor di atas 40.

Distribusi skor menurun bertahap dari 60 ke 39, menandakan seluruh produk dalam daftar cukup terhubung, meski ada beberapa yang lebih dominan.

Produk-produk dengan skor tinggi ini bisa dimanfaatkan untuk:

1. Rekomendasi produk: karena sering muncul bersama produk lain.
2. Penempatan atau promosi utama: karena potensinya menarik pembelian produk lain.
3. Evaluasi pengaruh produk: misalnya dalam pengelolaan stok atau analisis performa produk.

Secara umum, analisis ini membantu melihat produk mana yang paling berpengaruh dalam ekosistem e-commerce berdasarkan pola pembelian pelanggan.