PUC Minas

Apache PIG

Diego Roberto Gonçalves de Pontes

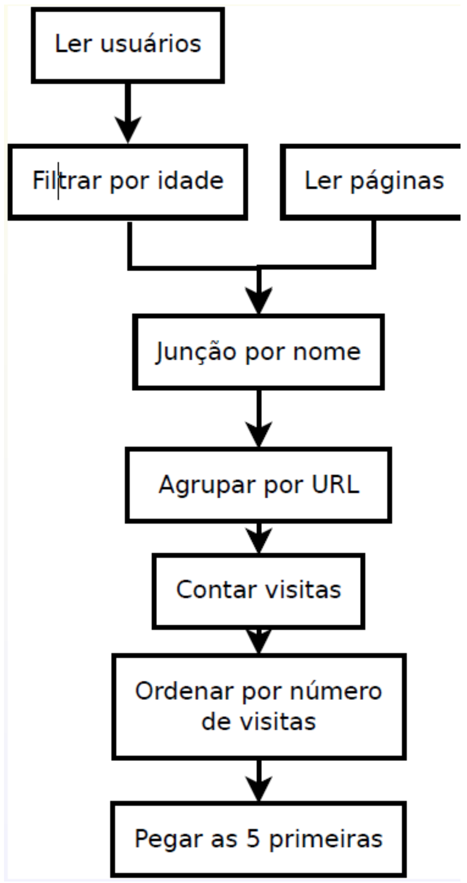
Apache PIG

- Plataforma para Análise de Grandes Volumes de Dados.
  - Linguagem de alto nível: Pig Latin
  - Infraestrutura de execução de programas
- Funcionalidades
  - Geração automática de programas
  - MapReduce
  - Otimização de execução
  - Compilação em tempo de execução

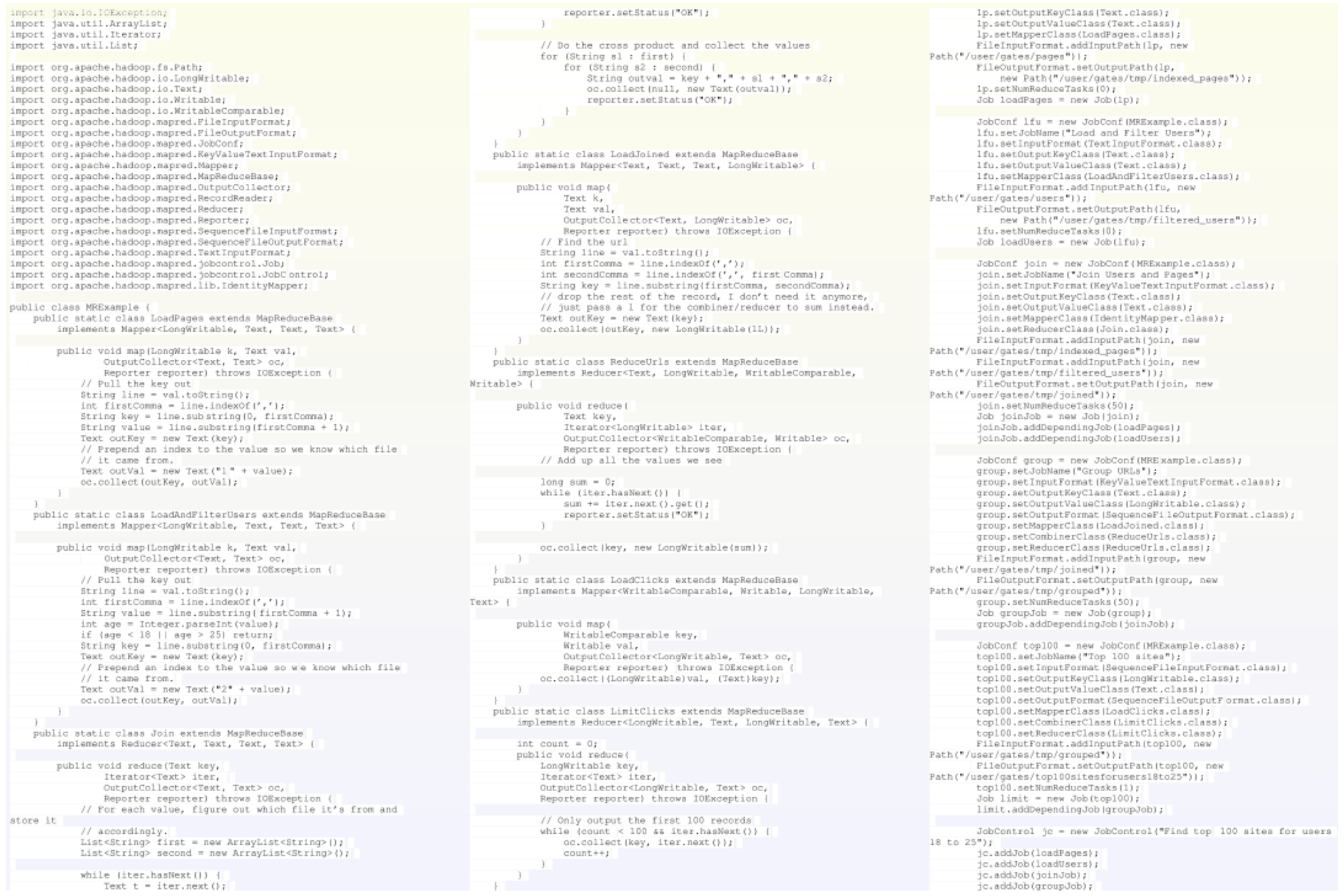
PUC Minas

Apache PIG

- “Problema: Suponha que você tenha dados dos seus usuários em um arquivo, logs de acesso a sites em outro, e você quer saber quais são os 5 sites mais visitados por usuários com idades entre 18 e 25 anos.”



Código MapReduce



```java
import java.io.IOException;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;

import org.apache.hadoop.fs.Path;
import org.apache.hadoop.io.LongWritable;
import org.apache.hadoop.io.Text;
import org.apache.hadoop.io.Writable;
import org.apache.hadoop.io.WritableComparable;
import org.apache.hadoop.mapred.FileInputFormat;
import org.apache.hadoop.mapred.FileOutputFormat;
import org.apache.hadoop.mapred.JobConf;
import org.apache.hadoop.mapred.KeyValueTextInputFormat;
import org.apache.hadoop.mapred.Mapper;
import org.apache.hadoop.mapred.MapReduceBase;
import org.apache.hadoop.mapred.OutputCollector;
import org.apache.hadoop.mapred.RecordReader;
import org.apache.hadoop.mapred.Reducer;
import org.apache.hadoop.mapred.Reporter;
import org.apache.hadoop.mapred.SequenceFileInputFormat;
import org.apache.hadoop.mapred.SequenceFileOutputFormat;
import org.apache.hadoop.mapred.TextInputFormat;
import org.apache.hadoop.mapred.jobcontrol.Job;
import org.apache.hadoop.mapred.jobcontrol.JobC ontrol;
import org.apache.hadoop.mapred.lib.IdentityMapper;

public class MRExample {
    public static class LoadPages extends MapReduceBase
        implements Mapper<LongWritable, Text, Text, Text> {

        public void map(LongWritable k, Text val,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // Pull the key out
            String line = val.toString();
            int firstComma = line.indexOf(’,’);
            String key = line.sub string(0, firstComma);
            String value = line.substring(firstComma + 1);
            Text outKey = new Text(key);
            // Prepend an index to the value so we know which file
            // it came from.
            Text outVal = new Text("1" + value);
            oc.collect(outKey, outVal);
        }
    }
    public static class LoadAndFilterUsers extends MapReduceBase
        implements Mapper<LongWritable, Text, Text, Text> {

        public void map(LongWritable k, Text val,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // Pull the key out
            String line = val.toString();
            int firstComma = line.indexOf(’,’);
            String value = line.substring( firstComma + 1);
            int age = Integer.parseInt(value);
            if (age < 18 || age > 25) return;
            String key = line.substring(0, firstComma);
            Text outKey = new Text(key);
            // Prepend an index to the value so w e know which file
            // it came from.
            Text outVal = new Text("2" + value);
            oc.collect(outKey, outVal);
        }
    }
    public static class Join extends MapReduceBase
        implements Reducer<Text, Text, Text, Text> {

        public void reduce(Text key,
                Iterator<Text> iter,
                OutputCollector<Text, Text> oc,
                Reporter reporter) throws IOException {
            // For each value, figure out which file it’s from and
store it
            // accordingly.
            List<String> first = new ArrayList<String>();
            List<String> second = new ArrayList<String>();

            while (iter.hasNext()) {
                Text t = iter.next();
```

```java
                reporter.setStatus("OK");
            }

            // Do the cross product and collect the values
            for (String s1 : first) {
                for (String s2 : second) {
                    String outval = key + "," + s1 + "," + s2;
                    oc.collect(null, new Text(outval));
                    reporter.setStatus("OK");
                }
            }
        }
    }
    public static class LoadJoined extends MapReduceBase
        implements Mapper<Text, Text, Text, LongWritable> {

        public void map(
                Text k,
                Text val,
                OutputColle ctor<Text, LongWritable> oc,
                Reporter reporter) throws IOException {
            // Find the url
            String line = val.toString();
            int firstComma = line.indexOf(’,’);
            int secondComma = line.indexOf(’,’, first Comma);
            String key = line.substring(firstComma, secondComma);
            // drop the rest of the record, I don’t need it anymore,
            // just pass a 1 for the combiner/reducer to sum instead.
            Text outKey = new Text(key);
            oc.collect(outKey, new LongWritable(1L));
        }
    }
    public static class ReduceUrls extends MapReduceBase
        implements Reducer<Text, LongWritable, WritableComparable,
Writable> {

        public void reduce(
                Text key,
                Iterator<LongWritable> iter,
                OutputCollector<WritableComparable, Writable> oc,
                Reporter reporter) throws IOException {
            // Add up all the values we see

            long sum = 0;
            while (iter.hasNext()) {
                sum += iter.next().get();
                reporter.setStatus("OK");
            }

            oc.collect(key, new LongWritable(sum));
        }
    }
    public static class LoadClicks extends MapReduceBase
        implements Mapper<WritableComparable, Writable, LongWritable,
Text> {

        public void map(
                WritableComparable key,
                Writable val,
                OutputCollector<LongWritable, Text> oc,
                Reporter reporter)  throws IOException {
            oc.collect((LongWritable)val, (Text)key);
        }
    }
    public static class LimitClicks extends MapReduceBase
        implements Reducer<LongWritable, Text, LongWritable, Text> {

        int count = 0;
        public void reduce(
            LongWritable key,
            Iterator<Text> iter,
            OutputCollector<LongWritable, Text> oc,
            Reporter reporter) throws IOException {

            // Only output the first 100 records
            while (count < 100 && iter.hasNext()) {
                oc.collect(key, iter.next());
                count++;
            }
        }
    }
```

```java
        lp.setOutputKeyClass(Text.class);
        lp.setOutputValueClass(Text.class);
        lp.setMapperClass(LoadPages.class);
        FileInputFormat.addInputPath(lp, new
Path("/user/gates/pages"));
        FileOutputFormat.setOutputPath(lp,
            new Path("/user/gates/tmp/indexed_pages"));
        lp.setNumReduceTasks(0);
        Job loadPages = new Job(lp);

        JobConf lfu = new JobConf(MRExample.class);
        lfu.setJobName("Load and Filter Users");
        lfu.setInputFormat(TextInputFormat.class);
        lfu.setOutputKeyClass(Text.class);
        lfu.setOutputValueClass(Text.class);
        lfu.setMapperClass(LoadAndFilterUsers.class);
        FileInputFormat.add InputPath(lfu, new
Path("/user/gates/users"));
        FileOutputFormat.setOutputPath(lfu,
            new Path("/user/gates/tmp/filtered_users"));
        lfu.setNumReduceTasks(0);
        Job loadUsers = new Job(lfu);

        JobConf join = new JobConf( MRExample.class);
        join.setJobName("Join Users and Pages");
        join.setInputFormat(KeyValueTextInputFormat.class);
        join.setOutputKeyClass(Text.class);
        join.setOutputValueClass(Text.class);
        join.setMapperClass(IdentityMap per.class);
        join.setReducerClass(Join.class);
        FileInputFormat.addInputPath(join, new
Path("/user/gates/tmp/indexed_pages"));
        FileInputFormat.addInputPath(join, new
Path("/user/gates/tmp/filtered_users"));
        FileOutputFormat.setOutputPath(join, new
Path("/user/gates/tmp/joined"));
        join.setNumReduceTasks(50);
        Job joinJob = new Job(join);
        joinJob.addDependingJob(loadPages);
        joinJob.addDependingJob(loadUsers);

        JobConf group = new JobConf(MRE xample.class);
        group.setJobName("Group URLs");
        group.setInputFormat(KeyValueTextInputFormat.class);
        group.setOutputKeyClass(Text.class);
        group.setOutputValueClass(LongWritable.class);
        group.setOutputFormat(SequenceFi leOutputFormat.class);
        group.setMapperClass(LoadJoined.class);
        group.setCombinerClass(ReduceUrls.class);
        group.setReducerClass(ReduceUrls.class);
        FileInputFormat.addInputPath(group, new
Path("/user/gates/tmp/joined"));
        FileOutputFormat.setOutputPath(group, new
Path("/user/gates/tmp/grouped"));
        group.setNumReduceTasks(50);
        Job groupJob = new Job(group);
        groupJob.addDependingJob(joinJob);

        JobConf top100 = new JobConf(MRExample.class);
        top100.setJobName("Top 100 sites");
        top100.setInputFormat(SequenceFileInputFormat.class);
        top100.setOutputKeyClass(LongWritable.class);
        top100.setOutputValueClass(Text.class);
        top100.setOutputFormat(SequenceFileOutputF ormat.class);
        top100.setMapperClass(LoadClicks.class);
        top100.setCombinerClass(LimitClicks.class);
        top100.setReducerClass(LimitClicks.class);
        FileInputFormat.addInputPath(top100, new
Path("/user/gates/tmp/grouped"));
        FileOutputFormat.setOutputPath(top100, new
Path("/user/gates/top100sitesforusers18to25"));
        top100.setNumReduceTasks(1);
        Job limit = new Job(top100);
        limit.addDependingJob(groupJob);

        JobControl jc = new JobControl("Find top 100 sites for users
18 to 25");
        jc.addJob(loadPages);
        jc.addJob(loadUsers);
        jc.addJob(joinJob);
        jc.addJob(groupJob);
```

Mesmo código em Pig Latin

```
Users = load `users' as (name, age);
Fltrd = filter Users by age >= 18 and age <= 25;
Pages = load `pages' as (user, url);
Jnd = join Fltrd by name, Pages by user;
Grpd = group Jnd by url;
Smmd = foreach Grpd generate group,
COUNT(Jnd) as clicks;
Srtd = order Smmd by clicks desc;
Top5 = limit Srtd 5;
store Top5 into `top5sites';
```

Execução

- Ao executar o script Pig, a plataforma se encarrega de:
- fazer o parse do arquivo
- verificar erros de sintaxe
- otimizar o código do script
- criar um plano de execução — quais tarefas Map e Reduce serão necessárias e qual a melhor ordem para executá-las?
- enviar todos os arquivos necessários para o HDFS
- monitorar os processos em execução

Onde é usado:

- Processamento de logs de servidores web
- Construção de modelos de predição de comportamento de usuários
- Processamento de imagens
- Construção de índices de páginas da web
- Pesquisa em conjuntos de dados “brutos”

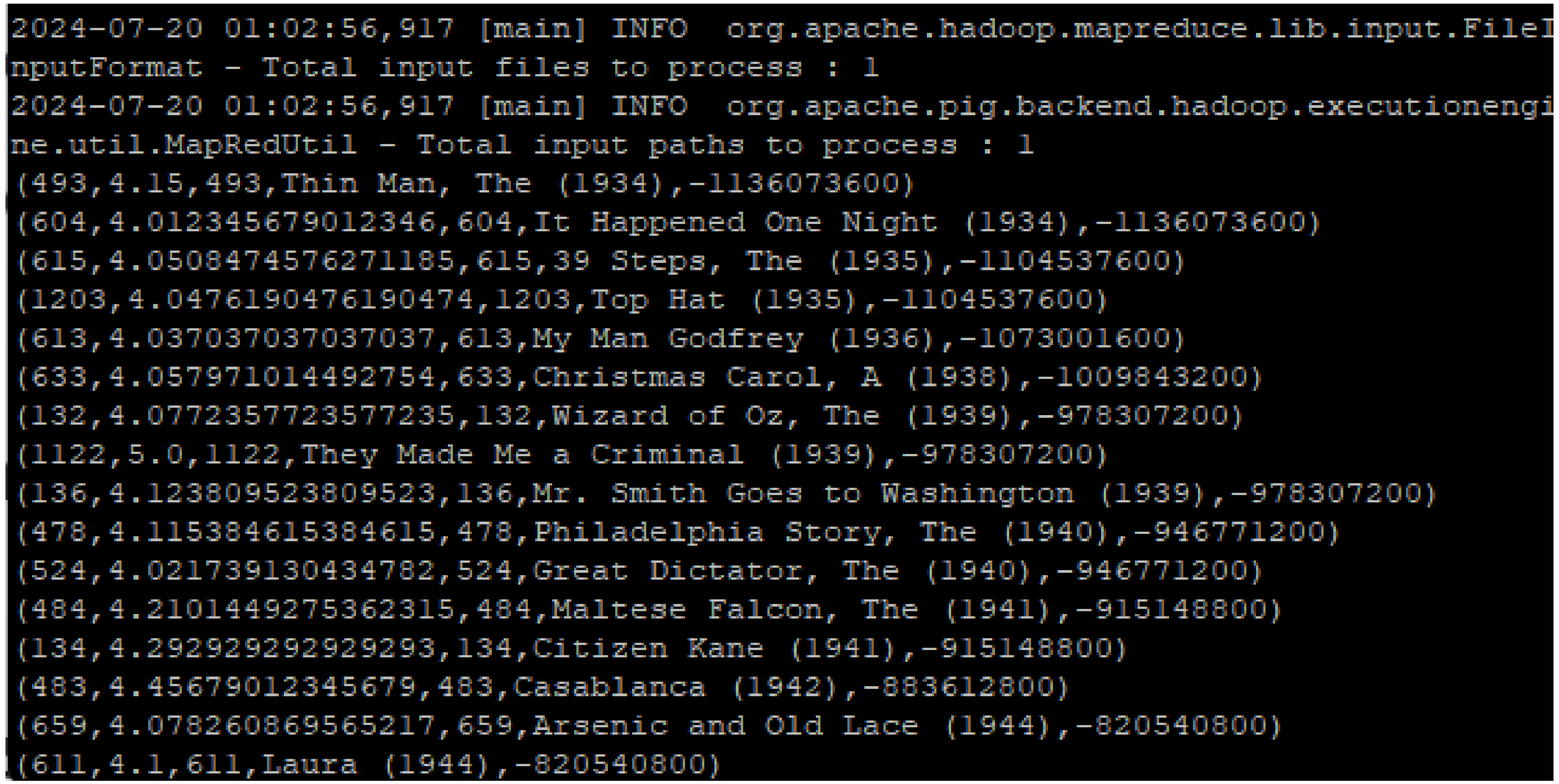
PUC Minas

Código PIG – Filmes 5 estrelas por ordem de lançamento

```
ratings = LOAD '/user/maria_dev/ml-100k/u.data' AS (userID:int, movieID:int, rating:int, ratingTime:int);

metadata = LOAD '/user/maria_dev/ml-100k/u.item' USING PigStorage('|')
    AS (movieID:int, movieTitle:chararray, releaseDate:chararray, videoRealese:chararray, imdblink:chararray);

nameLookup = FOREACH metadata GENERATE movieID, movieTitle,
    ToUnixTime(ToDate(releaseDate, 'dd-MMM-yyyy')) AS releaseTime;

ratingsByMovie = GROUP ratings BY movieID;
avgRatings = FOREACH ratingsByMovie GENERATE group as movieID, AVG(ratings.rating) as avgRating;
fiveStarMovies = FILTER avgRatings BY avgRating > 4.0;
fiveStarsWithData = JOIN fiveStarMovies BY movieID, nameLookup BY movieID;
oldestFiveStarMovies = ORDER fiveStarsWithData BY nameLookup::releaseTime;
DUMP oldestFiveStarMovies;
```

PUC Minas

Saída



```
2024-07-20 01:02:56,917 [main] INFO  org.apache.hadoop.mapreduce.lib.input.FileI
nputFormat - Total input files to process : 1
2024-07-20 01:02:56,917 [main] INFO  org.apache.pig.backend.hadoop.executionengi
ne.util.MapRedUtil - Total input paths to process : 1
(493,4.15,493,Thin Man, The (1934),-1136073600)
(604,4.012345679012346,604,It Happened One Night (1934),-1136073600)
(615,4.0508474576271185,615,39 Steps, The (1935),-1104537600)
(1203,4.0476190476190474,1203,Top Hat (1935),-1104537600)
(613,4.037037037037037,613,My Man Godfrey (1936),-1073001600)
(633,4.057971014492754,633,Christmas Carol, A (1938),-1009843200)
(132,4.0772357723577235,132,Wizard of Oz, The (1939),-978307200)
(1122,5.0,1122,They Made Me a Criminal (1939),-978307200)
(136,4.123809523809523,136,Mr. Smith Goes to Washington (1939),-978307200)
(478,4.115384615384615,478,Philadelphia Story, The (1940),-946771200)
(524,4.021739130434782,524,Great Dictator, The (1940),-946771200)
(484,4.2101449275362315,484,Maltese Falcon, The (1941),-915148800)
(134,4.292929292929293,134,Citizen Kane (1941),-915148800)
(483,4.45679012345679,483,Casablanca (1942),-883612800)
(659,4.078260869565217,659,Arsenic and Old Lace (1944),-820540800)
(611,4.1,611,Laura (1944),-820540800)
```

Exemplo Filmes100Mais

```
ratings = LOAD '/user/maria_dev/ml-100k/u.data' AS (userID:int, movieID:int, rating:int, ratingTime:int);

metadata = LOAD '/user/maria_dev/ml-100k/u.item' USING PigStorage('|')
           AS (movieID:int, movieTitle:chararray, releaseDate:chararray, videoRelease:chararray, imdbLink:chararray);

-- Contar o número de avaliações por filme
ratingsByMovie = GROUP ratings BY movieID;
movieRatingCount = FOREACH ratingsByMovie GENERATE group AS movieID, COUNT(ratings) AS ratingCount;

-- Filtrar filmes com mais de 100 avaliações
popularMovies = FILTER movieRatingCount BY ratingCount > 100;

-- Juntar com metadados para obter títulos
popularMoviesWithData = JOIN popularMovies BY movieID, metadata BY movieID;

-- Ordenar por contagem de avaliações
orderedPopularMovies = ORDER popularMoviesWithData BY ratingCount DESC;

-- Exibir resultado
DUMP orderedPopularMovies;
```

Exemplo FilmesRecentes

```
ratings = LOAD '/user/maria_dev/u.data' AS (userID:int, movieID:int, rating:int, ratingTime:int);

metadata = LOAD '/user/maria_dev/u.item' USING PigStorage('|')
           AS (movieID:int, movieTitle:chararray, releaseDate:chararray, videoRelease:chararray, imdbLink:chararray);

-- Converter datas e filtrar lançamentos dos últimos 5 anos
recentMovies = FOREACH metadata GENERATE movieID, movieTitle,
                ToUnixTime(ToDate(releaseDate, 'dd-MMM-yyyy')) AS releaseTime;
recentMoviesFiltered = FILTER recentMovies BY releaseTime >= ToUnixTime(ToDate('01-JAN-2015', 'dd-MMM-yyyy'));

-- Agrupar avaliações por filme
ratingsByMovie = GROUP ratings BY movieID;
avgRatings = FOREACH ratingsByMovie GENERATE group AS movieID, AVG(ratings.rating) AS avgRating;

-- Filtrar filmes com média de avaliação superior a 4
highRatedMovies = FILTER avgRatings BY avgRating > 4.0;

-- Juntar dados de avaliações com metadados
highRatedRecentMovies = JOIN highRatedMovies BY movieID, recentMoviesFiltered BY movieID;

-- Ordenar por data de lançamento
orderedHighRatedRecentMovies = ORDER highRatedRecentMovies BY releaseTime DESC;

-- Exibir resultado
DUMP orderedHighRatedRecentMovies;
```

PUC Minas