Laporan Eksperimen RDD Big Data di Cluster Spark
dengan data
“imdb_movie_5000.csv”

oleh:
Jonatan Laksamana Purmomo : 2016730081

Pengantar

Eksperimen analisis big data menggunakan SparkSQL dan DataFrame kali ini bertujuan untuk mengenalkan kepada peserta mahasiswa matakuliah Analisis Big Data tentang SQL dalam Spark dan penulisan syntax-nya menggunakan Scala. Pada eksperimen kali ini, digunakan dataset ‘imdb_movie_5000’

Membuat dataframe (tabel sementara) dari data tersebut, lalu melakukan query untuk mencari:
o Top ten director yang paling produktif (membuat film terbanyak)
o Top ten film dengan keuntungan kotor (gross -budget) terbesar.
o Skor IMDB rata-rata darifilm berdasar (group by)aktor pertama(actor_1_name)
Menyimpan hasil query pada sebuah file

Pengerjaan

- Membuat Object Spark yang akan kita gunakan untuk Eksperiment ini

```
//author : Jonathan Laksamana Purnama
//NPM : 2016730081
object bigData{
  def main(args: Array[String]): Unit = {
    val spark = org.apache.spark.sql.SparkSession.builder
      .master( master = "local[*]")
      .appName( name = "Spark CSV Reader")
      .getOrCreate;


  }
}
```

Spark Object ini akan beguna untuk meload data csv , dan menjalankan perintah spark sql

- load data menggunakan spark object
  nama data set : imbd_movie_500
  type : csv
  Berikut ini bentuk format data yang di berikan

color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross	genres	actor_1_name
Color	James Cameron	723	178	0	855	Joel David Moore	1000	760505847	Action|Adventure|Fantasy|Sci-Fi	CCH Pounder
Color	Gore Verbinski	302	169	563	1000	Orlando Bloom	40000	309404152	Action|Adventure|Fantasy	Johnny Depp
Color	Sam Mendes	602	148	0	161	Rory Kinnear	11000	200074175	Action|Adventure|Thriller	Christoph Waltz
Color	Christopher Nolan	813	164	22000	23000	Christian Bale	27000	448130642	Action|Thriller	Tom Hardy
	Doug Walker			131		Rob Walker	131		Documentary	Doug Walker
Color	Andrew Stanton	462	132	475	530	Samantha Morton	640	73058679	Action|Adventure|Sci-Fi	Daryl Sabara
Color	Sam Raimi	392	156	0	4000	James Franco	24000	336530303	Action|Adventure|Romance	J.K. Simmons
Color	Nathan Greno	324	100	15	284	Donna Murphy	799	200807262	Adventure|Animation|Comedy|Family|Fantasy|Musical|Romance	Brad Garrett
Color	Joss Whedon	635	141	0	19000	Robert Downey Jr.	26000	458991599	Action|Adventure|Sci-Fi	Chris Hemsworth
Color	David Yates	375	153	282	10000	Daniel Radcliffe	25000	301956980	Adventure|Family|Fantasy|Mystery	Alan Rickman
Color	Zack Snyder	673	183	0	2000	Lauren Cohan	15000	330249062	Action|Adventure|Sci-Fi	Henry Cavill
Color	Bryan Singer	434	169	0	903	Marlon Brando	18000	200069408	Action|Adventure|Sci-Fi	Kevin Spacey
Color	Marc Forster	403	106	395	393	Mathieu Amalric	451	168368427	Action|Adventure	Giancarlo Giannin
Color	Gore Verbinski	313	151	563	1000	Orlando Bloom	40000	423032628	Action|Adventure|Fantasy	Johnny Depp
Color	Gore Verbinski	450	150	563	1000	Ruth Wilson	40000	89289910	Action|Adventure|Western	Johnny Depp
Color	Zack Snyder	733	143	0	748	Christopher Meloni	15000	291021565	Action|Adventure|Fantasy|Sci-Fi	Henry Cavill
Color	Andrew Adamson	258	150	80	201	Pierfrancesco Favino	22000	141614023	Action|Adventure|Family|Fantasy	Peter Dinklage
Color	Joss Whedon	703	173	0	19000	Robert Downey Jr.	26000	623279547	Action|Adventure|Sci-Fi	Chris Hemsworth
Color	Rob Marshall	448	136	252	1000	Sam Claflin	40000	241063875	Action|Adventure|Fantasy	Johnny Depp
Color	Barry Sonnenfeld	451	106	188	718	Michael Stuhlbarg	10000	179020854	Action|Adventure|Comedy|Family|Fantasy|Sci-Fi	Will Smith
Color	Peter Jackson	422	164	0	773	Adam Brown	5000	255108370	Adventure|Fantasy	Aidan Turner
Color	Marc Webb	599	153	464	963	Andrew Garfield	15000	262030663	Action|Adventure|Fantasy	Emma Stone
Color	Ridley Scott	343	156	0	738	William Hurt	891	105219735	Action|Adventure|Drama|History	Mark Addy
Color	Peter Jackson	509	186	0	773	Adam Brown	5000	258355354	Adventure|Fantasy	Aidan Turner
Color	Chris Weitz	251	113	129	1000	Eva Green	16000	70083519	Adventure|Family|Fantasy	Christopher Lee
Color	Peter Jackson	446	201	0	84	Thomas Kretschmann	6000	218051260	Action|Adventure|Drama|Romance	Naomi Watts
Color	James Cameron	315	194	0	794	Kate Winslet	29000	658672302	Drama|Romance	Leonardo DiCaprio
Color	Anthony Russo	516	147	94	11000	Scarlett Johansson	21000	407197282	Action|Adventure|Sci-Fi	Robert Downey Jr.
Color	Peter Berg	377	131	532	627	Alexander Skarsgård	14000	65173160	Action|Adventure|Sci-Fi|Thriller	Liam Neeson
Color	Colin Trevorrow	644	124	365	1000	Judy Greer	3000	652177271	Action|Adventure|Sci-Fi|Thriller	Bryce Dallas How
Color	Sam Mendes	750	143	0	393	Helen McCrory	883	304360277	Action|Adventure|Thriller	Albert Finney

dari data set tersebut kita dapat mengetahui bahwa csv ini memiliki header pada row pertama karena itu ketika kita akan melukan load dengan spark object kita mengunakan option header = true , kurang lebih begini potongan code untuk load data

```
val data = spark.read
  .format( source = "csv")
  .option("header", "true")
  .load( path = "/home/joo/Desktop/imdb_movie_5000.csv")
```

lalu setelah kita meload data pastikan data sudah terload dengan cara melakukan print untuk memastikan data terload

```
val data = spark.read
  .format( source = "csv")
  .option("header", "true")
  .load( path = "/home/joo/Desktop/imdb_movie_5000.csv")

print(data)
```

dan berikut ini hasil output nya

```
20/02/17 17:36:21 INFO SparkContext: Created broadcast 2 from loa
20/02/17 17:36:21 INFO FileSourceScanExec: Planning scan with bin
[color: string, director_name: string ... 26 more fields]20/02/17
20/02/17 17:36:21 INFO SparkUI: Stopped Spark web UI at http://19
20/02/17 17:36:21 INFO MapOutputTrackerMasterEndpoint: MapOutputT
```

- lalu buat table sementara (pada percoaban saya saya akan mengambil seluruh column header untuk saya jadikan atrubte dalam table sementara saya dan saya beri nama df_movies)

```
data.select(data.col( colName = "*")).createTempView( viewName = "df_movies")
```

lalu seperti biasa kita melakukan cross check dengan cara membuat query sederhana untuk membuktikan bahwa table sementara telah terbentuk dan semua atrubte header telah terbuat

```
val sql = "select * from df_movies"
val query = spark.sql(sql).show()
print(query)
```

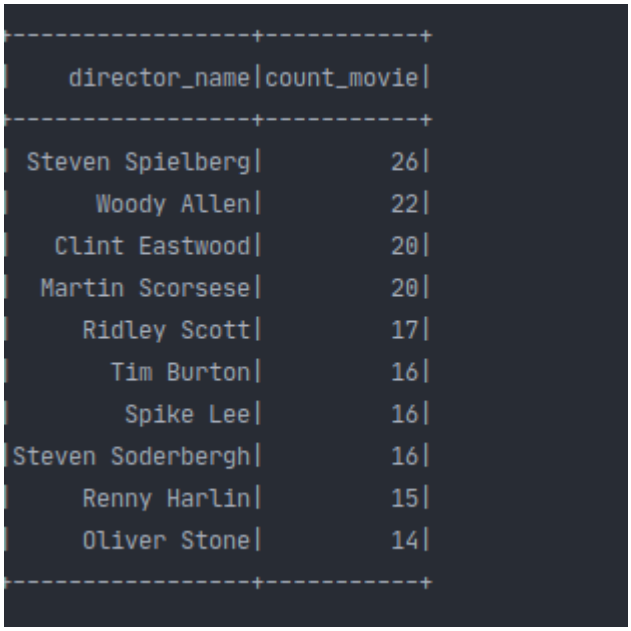
dan output

color	director_name	num_critic_for_reviews	duration	director_facebook_likes	actor_3_facebook_likes	actor_2_name	actor_1_facebook_likes	gross	genres
Color	James Cameron	723	178	0	855	Joel David Moore	1000	760505847	Action|Adventure|...
Color	Gore Verbinski	302	169	563	1000	Orlando Bloom	40000	309404152	Action|Adventure|...
Color	Sam Mendes	602	148	0	161	Rory Kinnear	11000	200074175	Action|Adventure|...
Color	Christopher Nolan	813	164	22000	23000	Christian Bale	27000	448130642	Action|Thriller
null	Doug Walker	null	null	131	null	Rob Walker	131	null	Documentary
Color	Andrew Stanton	462	132	475	530	Samantha Morton	640	73058679	Action|Adventure|...
Color	Sam Raimi	392	156	0	4000	James Franco	24000	336530303	Action|Adventure|...
Color	Nathan Greno	324	100	15	284	Donna Murphy	799	200807262	Adventure|Animati...
Color	Joss Whedon	635	141	0	19000	Robert Downey Jr.	26000	458991599	Action|Adventure|...
Color	David Yates	375	153	282	10000	Daniel Radcliffe	25000	301956980	Adventure|Family|...
Color	Zack Snyder	673	183	0	2000	Lauren Cohan	15000	330249062	Action|Adventure|...
Color	Bryan Singer	434	169	0	903	Marlon Brando	18000	200069408	Action|Adventure|...

setelah kita membuat table sementara maka kita akan membuat query untuk mendapatkan data
o Top ten director yang paling produktif (membuat film terbanyak)
o Top ten film dengan keuntungan kotor (gross -budget) terbesar.
o Skor IMDB rata-rata darifilm berdasar (group by)aktor
pertama(actor_1_name)
Menyimpan hasil query pada sebuah file

```
val OUTPUT_BASE_PATH = "../output/"
//    no 1
val sql_top_10_productive_director = "SELECT" +
  " director_name , count(movie_title) as count_movie" +
  " from df_movies" +
  " where director_name is not null " +
  " group by director_name " +
  " order by count_movie desc LIMIT 10"
var query = spark.sql(sql_top_10_productive_director)
query.show()
query.write.json(OUTPUT_BASE_PATH + "top_10_director_name")
//    no 2
val sql_top_ten_gross_by_movie = "SELECT" +
  " movie_title , gross " +
  " FROM df_movies ORDER BY" +
  " gross desc limit 10 "
query = spark.sql(sql_top_ten_gross_by_movie)
query.show()
query.write.json(OUTPUT_BASE_PATH + "top_10_gross_by_movie")
//    no 3
val sql_average_imbd_by_first_actor_name = "SELECT" +
  " actor_1_name ,  avg(imdb_score) as average_imbd_score " +
  " FROM df_movies group by actor_1_name "
query = spark.sql(sql_average_imbd_by_first_actor_name)
query.show()
query.write.json(OUTPUT_BASE_PATH + "average_imdb_score_by_first_actor_name")
```

Print Output untuk soal no 1



director_name	count_movie
Steven Spielberg	26
Woody Allen	22
Clint Eastwood	20
Martin Scorsese	20
Ridley Scott	17
Tim Burton	16
Spike Lee	16
Steven Soderbergh	16
Renny Harlin	15
Oliver Stone	14

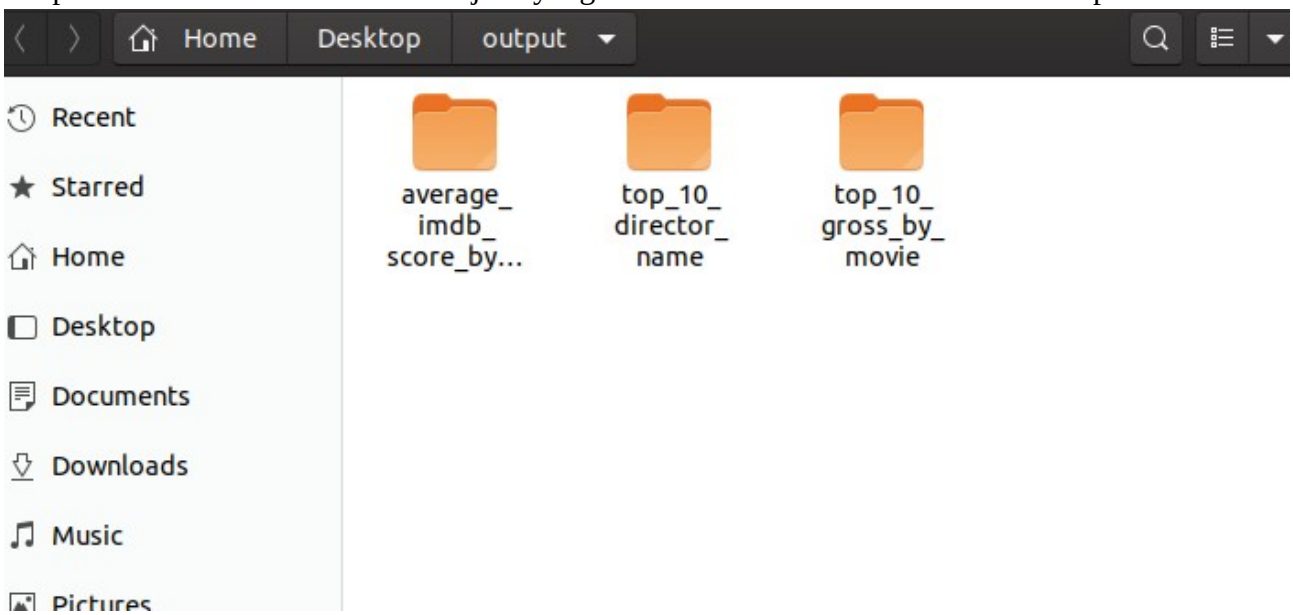
Print output untuk soal no 2

movie_title	gross
8: The Mormon Pro...	99851
Dirty Work	9975684
Pandora's Box	9950
The Young and Pro...	99462
Brazil	9929000
My Summer of Love	992238
Desert Blue	99147
As It Is in Heaven	9910
Olympus Has Fallen	98895417
The Green Hornet	98780042

print Output untuk soal no 3

```
+-------------+-------------------+
|  actor_1_name|average_imbd_score|
+-------------+-------------------+
|    Doug Walker|                7.1|
|   Oliver Platt|  6.2099999999999999|
|     Snoop Dogg|  6.6999999999999999|
|   Stephen Root|               5.88|
| Laurence Olivier|  5.9666666666666666|
|    Jordi Mollà|               6.15|
|      Ian Whyte|                2.7|
|   Tyler Labine|                6.1|
|  Caroline Munro|                7.1|
|   Necati Sasmaz|                6.0|
|   Julia Jentsch|                7.4|
|Geraldine Chaplin|                7.5|
|     Emma Stone|  6.5583333333333334|
|  Frances Conroy|  5.3999999999999995|
|   Matthew Perry|  5.9333333333333333|
|Takeshi Kaneshiro|                7.5|
|   Darcy Donavan|                7.2|
| Violante Placido|                6.3|
|        Dan Byrd|               6.15|
|  Tadanobu Asano|                7.3|
+-------------+-------------------+
only showing top 20 rows
```

Output di tulis dalam bentuk format json yang secara default ada terbentuk di Desktop



dimana di dalamanya kita akan membuat 3 buah file berdasarkan perintah code di atas

dan di dalamnya terdapat file json dengan format nama seperti :

berikut ini contoh file output json yang di buat mengunakan spark sql


```
{"movie_title":"8: The Mormon Proposition ","gross":"99851"}
{"movie_title":"Dirty Work ","gross":"9975684"}
{"movie_title":"Pandora's Box ","gross":"9950"}
{"movie_title":"The Young and Prodigious T.S. Spivet ","gross":"99462"}
{"movie_title":"Brazil ","gross":"9929000"}
{"movie_title":"My Summer of Love ","gross":"992238"}
{"movie_title":"Desert Blue ","gross":"99147"}
{"movie_title":"As It Is in Heaven ","gross":"9910"}
{"movie_title":"Olympus Has Fallen ","gross":"98895417"}
{"movie_title":"The Green Hornet ","gross":"98780042"}
```