CSP:554 – Project Presentation

Group name: **Panda**
Arinjay Jain: **A20447307**
Ayush Dadhich: **A20449379**
Parth Gupta: **A20449774**
Keerthana Nagula: **A20442972**

- **Project Topic:** Develop a Data handling pipeline with the aid of Big Data SQL tools

- **Application subject area:** Movies & Entertainment

- **Data set Source:** IMDB Movies dataset from Kaggle

  Data set of 1,000 most popular movies on IMDB (Tomiandrep, "IMDB Filmid") in the last 10 years. The data points included are: Title, Genre, Description, Director, Actors, Year, Runtime, Rating, Votes, Revenue, Metascore.

  The main Movies Metadata file (Banik, "The Movies Dataset"). Files contain information on 45,000 movies utilized in the full MovieLens dataset. Data set include plot keywords, languages, production countries and represents a collection of 26 million ratings from all the 45,000 movies.

- **Project Motive / Question:**

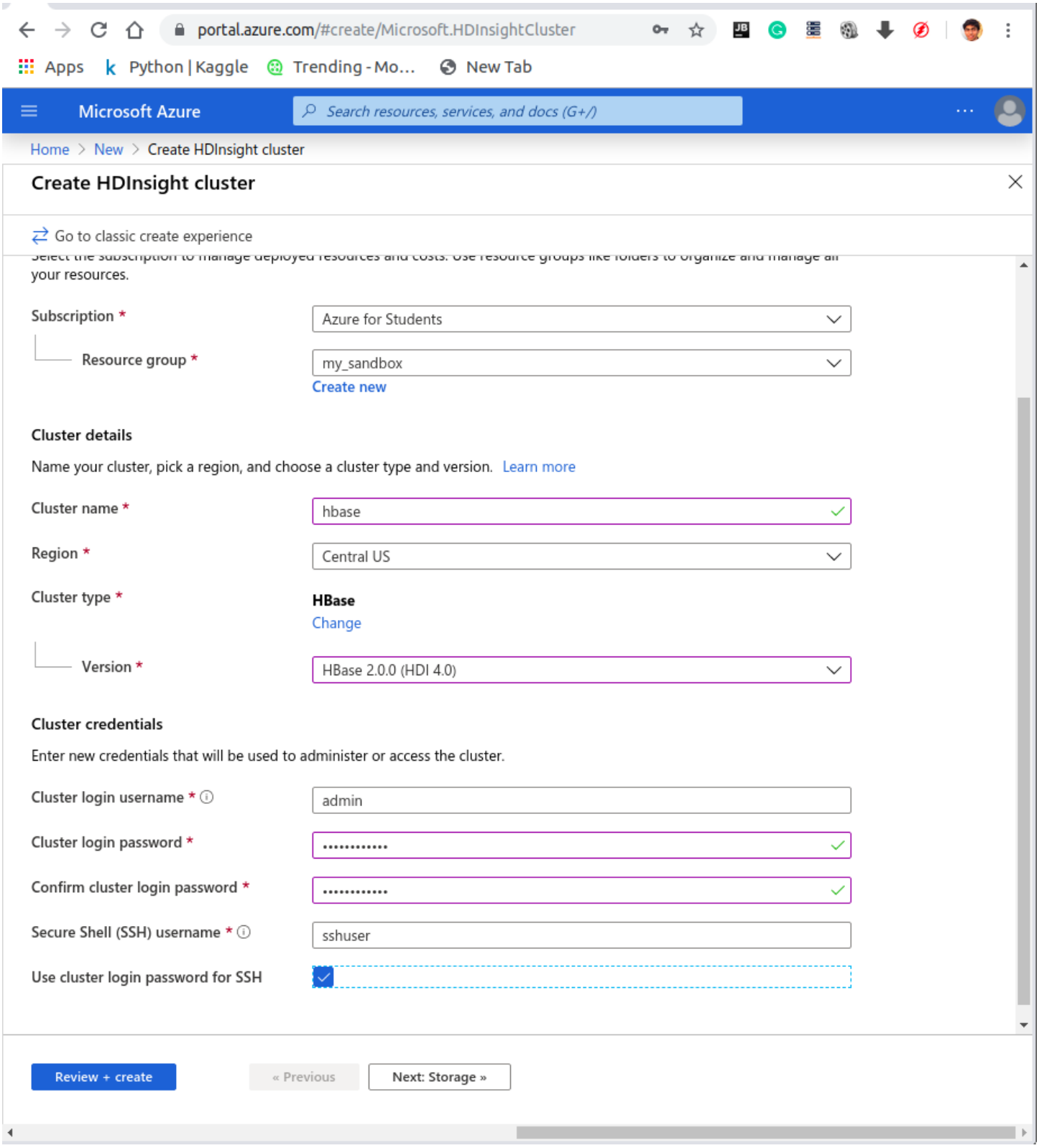
  Develop a data processing pipeline using Big Data technologies to insert, modify, transform and implement Big Data techniques to research relation between proposed data and derive meaningful / relevant understanding of data streaming on movies data from IMDB.

- **Proposed Approach: Tools / Techniques for:**
  - **Data Insertion:** HDFS commands to move data on to Hadoop environment
  - **Data Modification & Transformation:** SparkSQL
  - **Database:** NoSQL HBASE DB
  - **Data Mining & Analysis:** SQL and Python / R on Big Data

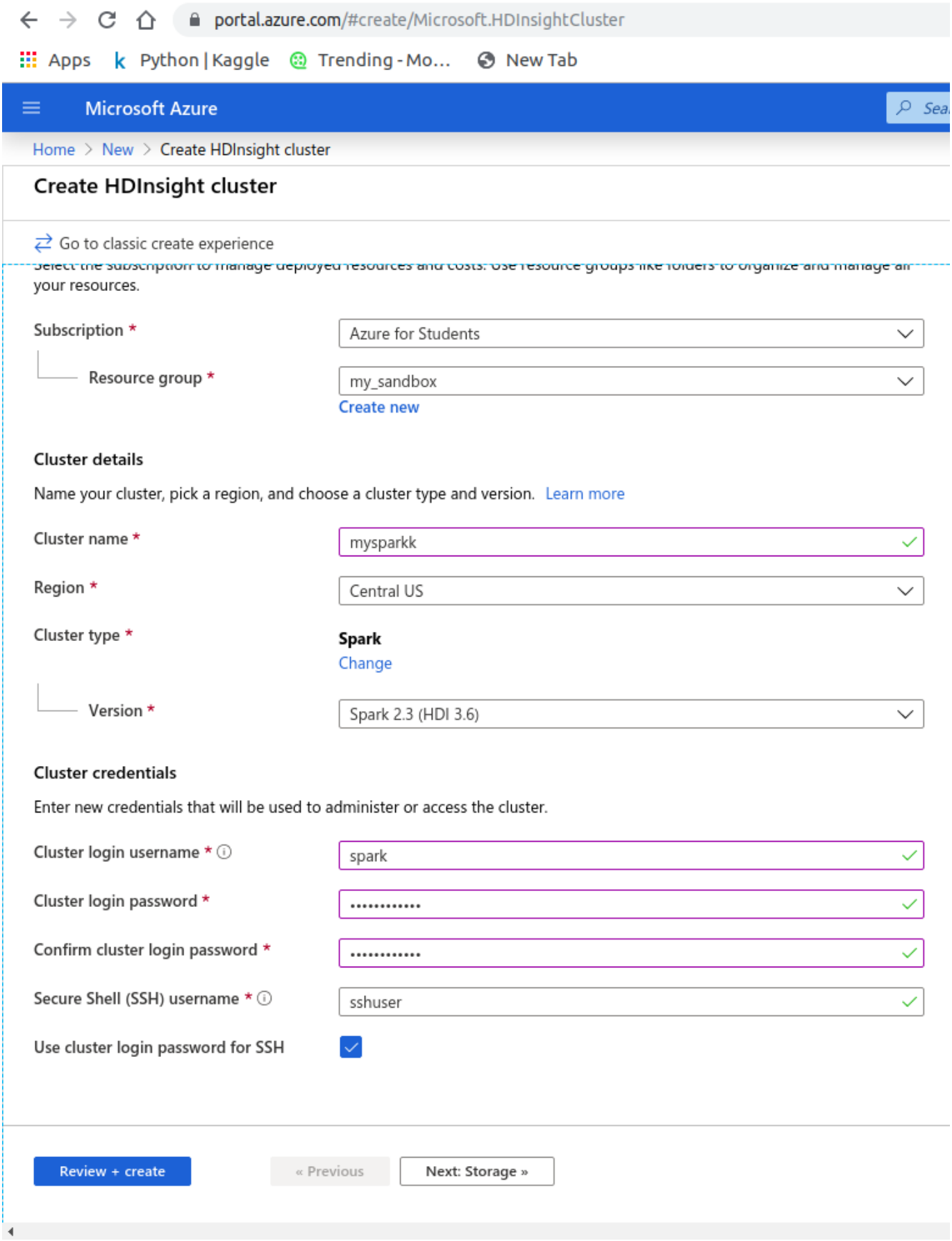
- **Reference resources:**

Tomiandrep. "IMDB Filmid." *Kaggle*, Kaggle, 13 Dec. 2017, https://www.kaggle.com/tomiandrep/imdb-filmid/data.
Banik, Rounak. "The Movies Dataset." *Kaggle*, 10 Nov. 2017, https://www.kaggle.com/rounakbanik/the-movies-dataset/data.

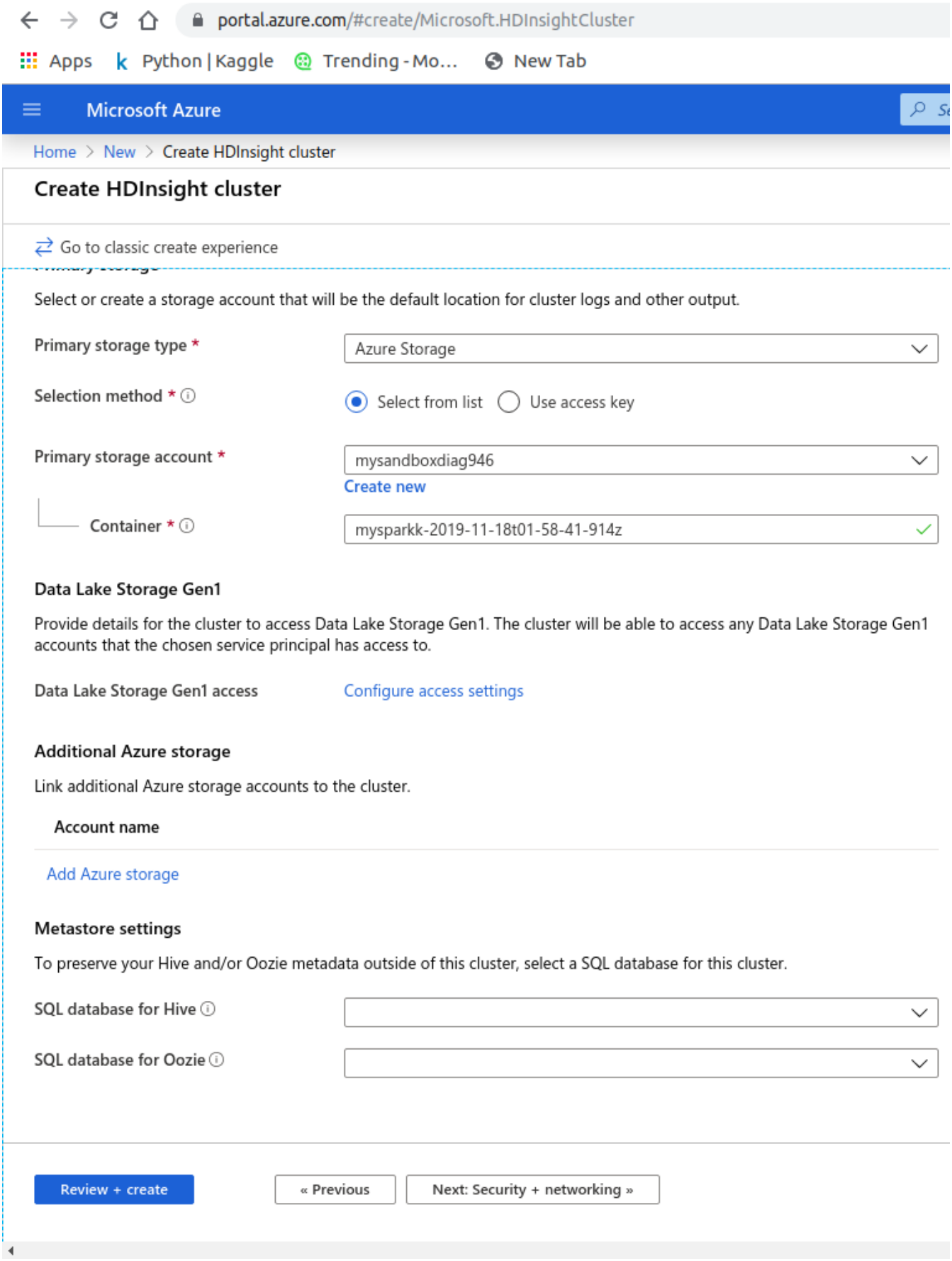
Creating Hbase Cluster:-



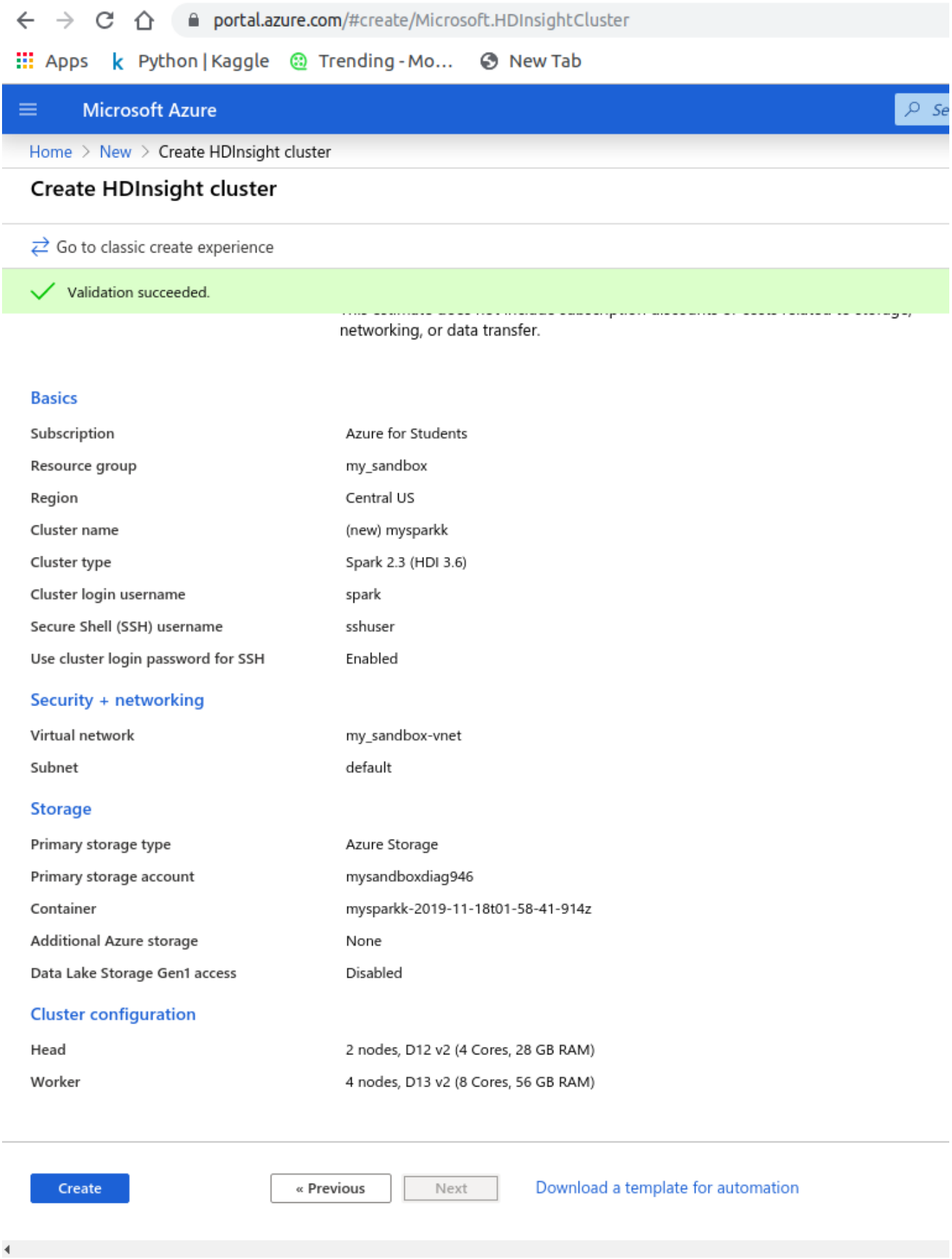
Creating Spark Cluster:- **1**.



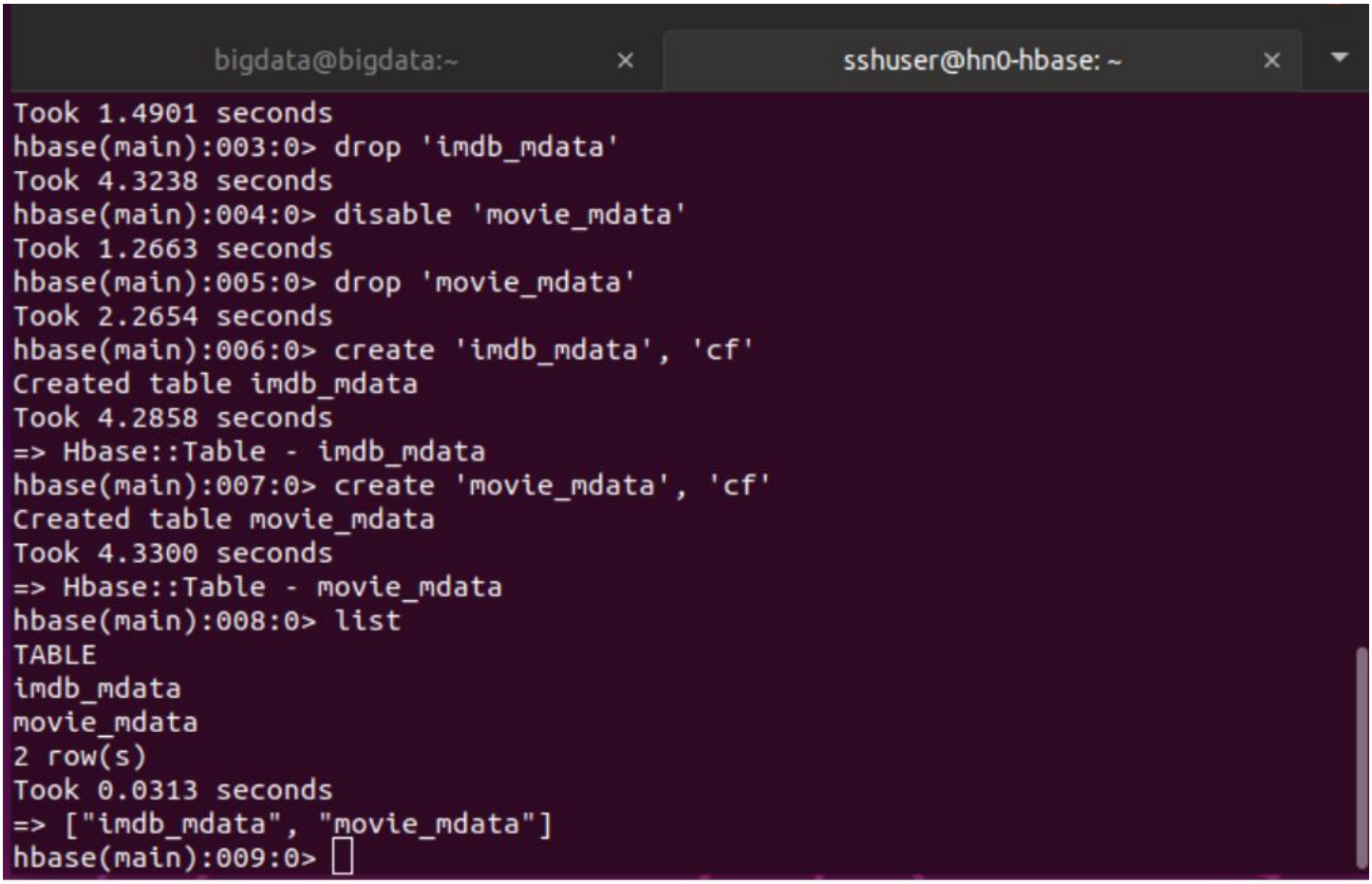
2.


portal.azure.com/#create/Microsoft.HDInsightCluster
Apps
Python | Kaggle
Trending - Mo...
New Tab
Microsoft Azure
Home > New > Create HDInsight cluster
Create HDInsight cluster
Go to classic create experience
Select or create a storage account that will be the default location for cluster logs and other output.
Primary storage type *
Azure Storage
Selection method *
Select from list
Use access key
Primary storage account *
mysandboxdiag946
Create new
Container *
mysparkk-2019-11-18t01-58-41-914z
Data Lake Storage Gen1
Provide details for the cluster to access Data Lake Storage Gen1. The cluster will be able to access any Data Lake Storage Gen1 accounts that the chosen service principal has access to.
Data Lake Storage Gen1 access
Configure access settings
Additional Azure storage
Link additional Azure storage accounts to the cluster.
Account name
Add Azure storage
Metastore settings
To preserve your Hive and/or Oozie metadata outside of this cluster, select a SQL database for this cluster.
SQL database for Hive
SQL database for Oozie
Review + create
« Previous
Next: Security + networking »

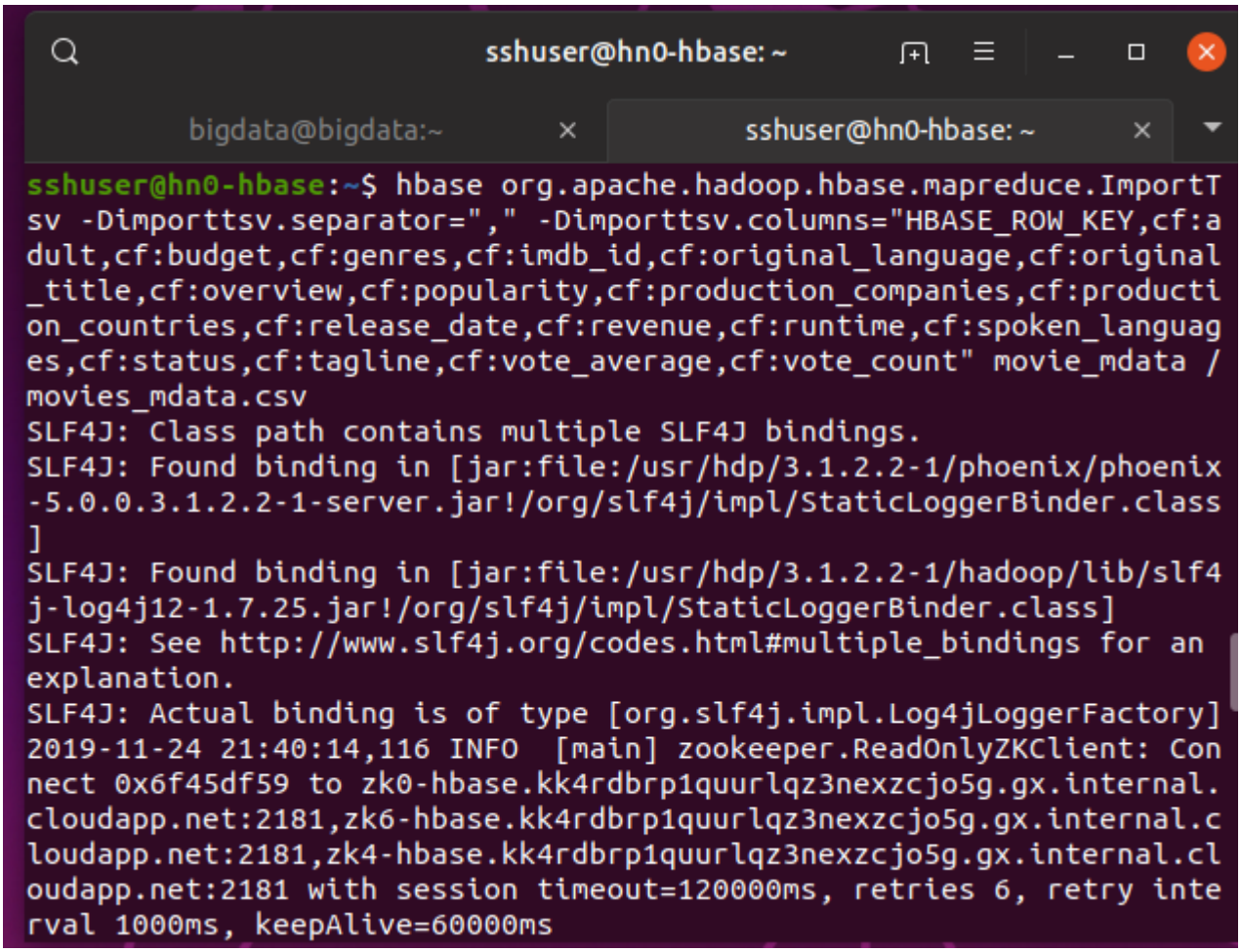
3.


portal.azure.com/#create/Microsoft.HDInsightCluster
Apps
Python | Kaggle
Trending - Mo...
New Tab
Microsoft Azure
Home > New > Create HDInsight cluster
Create HDInsight cluster
Go to classic create experience
Validation succeeded.
networking, or data transfer.
Basics
Subscription
Azure for Students
Resource group
my_sandbox
Region
Central US
Cluster name
(new) mysparkk
Cluster type
Spark 2.3 (HDI 3.6)
Cluster login username
spark
Secure Shell (SSH) username
sshuser
Use cluster login password for SSH
Enabled
Security + networking
Virtual network
my_sandbox-vnet
Subnet
default
Storage
Primary storage type
Azure Storage
Primary storage account
mysandboxdiag946
Container
mysparkk-2019-11-18t01-58-41-914z
Additional Azure storage
None
Data Lake Storage Gen1 access
Disabled
Cluster configuration
Head
2 nodes, D12 v2 (4 Cores, 28 GB RAM)
Worker
4 nodes, D13 v2 (8 Cores, 56 GB RAM)
Create
« Previous
Next
Download a template for automation

Create Table

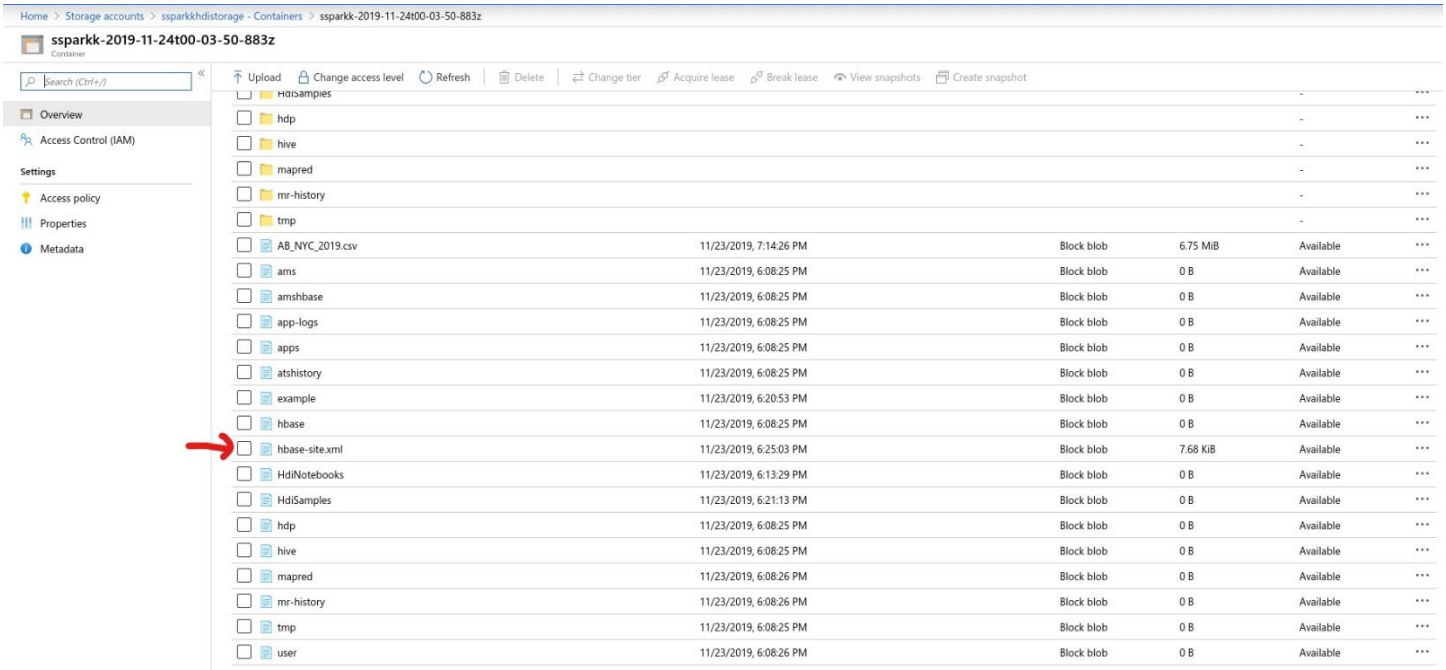


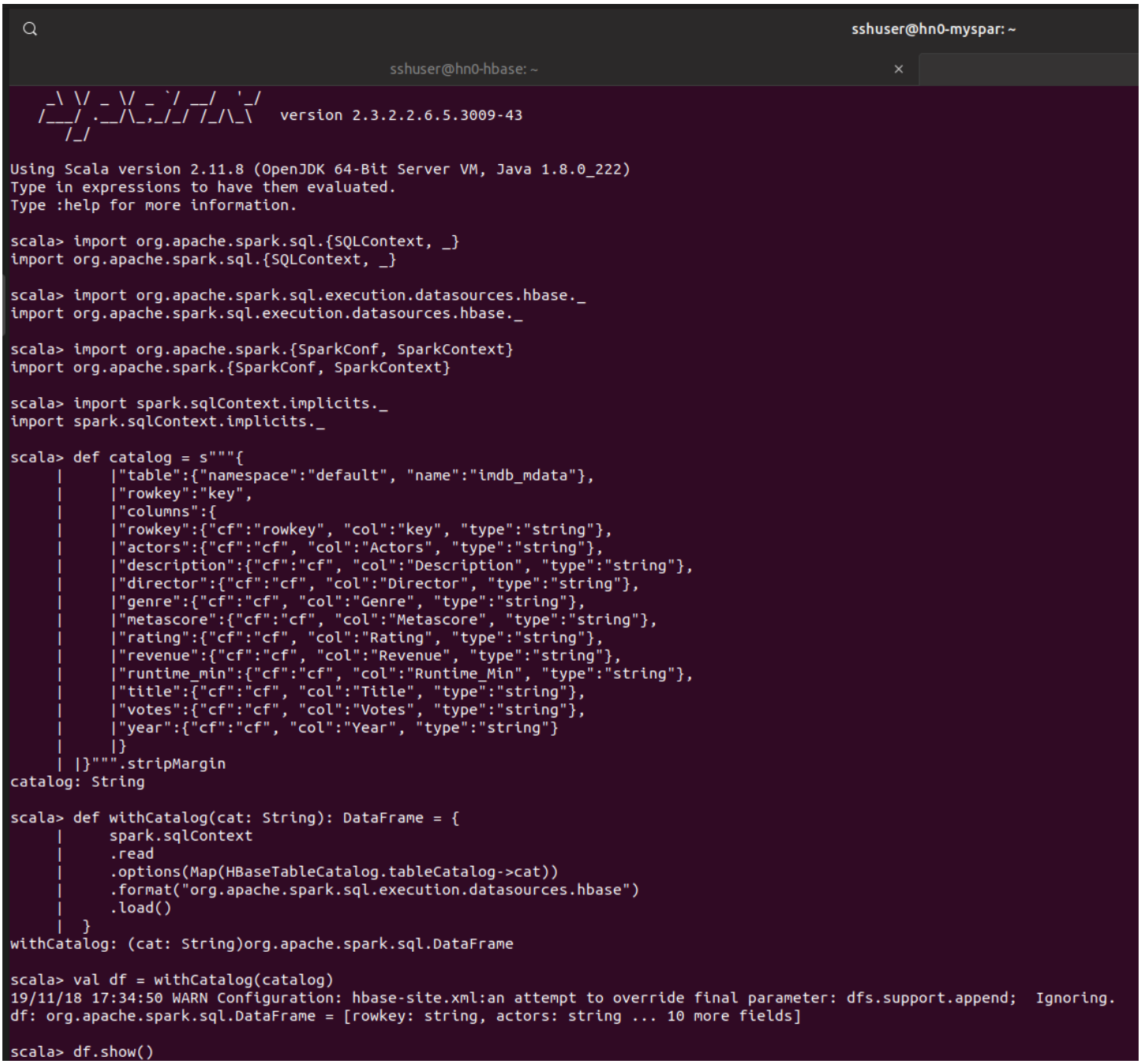
Load Data


sshuser@hn0-hbase: ~
bigdata@bigdata:~
sshuser@hn0-hbase: ~
sshuser@hn0-hbase:~$ hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimporttsv.separator="," -Dimporttsv.columns="HBASE_ROW_KEY,cf:adult,cf:budget,cf:genres,cf:imdb_id,cf:original_language,cf:original_title,cf:overview,cf:popularity,cf:production_companies,cf:production_countries,cf:release_date,cf:revenue,cf:runtime,cf:spoken_languages,cf:status,cf:tagline,cf:vote_average,cf:vote_count" movie_mdata /movies_mdata.csv
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/hdp/3.1.2.2-1/phoenix/phoenix-5.0.0.3.1.2.2-1-server.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/hdp/3.1.2.2-1/hadoop/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
2019-11-24 21:40:14,116 INFO [main] zookeeper.ReadOnlyZKClient: Connect 0x6f45df59 to zk0-hbase.kk4rdbrp1quurlqz3nexzcjo5g.gx.internal.cloudapp.net:2181,zk6-hbase.kk4rdbrp1quurlqz3nexzcjo5g.gx.internal.cloudapp.net:2181,zk4-hbase.kk4rdbrp1quurlqz3nexzcjo5g.gx.internal.cloudapp.net:2181 with session timeout=120000ms, retries 6, retry interval 1000ms, keepAlive=60000ms

```
sshuser@hn0-hbase:~$ hbase org.apache.hadoop.hbase.mapreduce.ImportTsv -Dimpor
ttsv.separator="," -Dimporttsv.columns="HBASE_ROW_KEY,cf:Rank,cf:Title,cf:Genr
e,cf:Description,cf:Director,cf:Actors,cf:Year,cf:Runtime (Minutes),cf:Rating,
cf:Votes,cf:Revenue (Millions),cf:Metascore" imdb_mdata /imdb_movie.csv
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/hdp/3.1.2.2-1/phoenix/phoenix-5.0.0.3.1
.2.2-1-server.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/usr/hdp/3.1.2.2-1/hadoop/lib/slf4j-log4j12-
1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanatio
n.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
2019-11-24 21:37:42,706 INFO  [main] zookeeper.ReadOnlyZKClient: Connect 0x6f4
5df59 to zk0-hbase.kk4rdbrp1quurlqz3nexzcjo5g.gx.internal.cloudapp.net:2181,zk
6-hbase.kk4rdbrp1quurlqz3nexzcjo5g.gx.internal.cloudapp.net:2181,zk4-hbase.kk4
rdbrp1quurlqz3nexzcjo5g.gx.internal.cloudapp.net:2181 with session timeout=120
000ms, retries 6, retry interval 1000ms, keepAlive=60000ms
2019-11-24 21:37:42,734 INFO  [ReadOnlyZKClient-zk0-hbase.kk4rdbrp1quurlqz3nex
zcjo5g.gx.internal.cloudapp.net:2181,zk6-hbase.kk4rdbrp1quurlqz3nexzcjo5g.gx.i
nternal.cloudapp.net:2181,zk4-hbase.kk4rdbrp1quurlqz3nexzcjo5g.gx.internal.clo
udapp.net:2181@0x6f45df59] zookeeper.ZooKeeper: Client environment:zookeeper.v
ersion=3.4.6-1--1, built on 08/13/2019 20:34 GMT
```

Connection has been established between Hbase and Spark after copying **hbase-site.xml** from Hbase cluster storage to spark cluster storage.




sshuser@hn0-myspar: ~
sshuser@hn0-hbase: ~
version 2.3.2.2.6.5.3009-43
Using Scala version 2.11.8 (OpenJDK 64-Bit Server VM, Java 1.8.0_222)
Type in expressions to have them evaluated.
Type :help for more information.
scala> import org.apache.spark.sql.{SQLContext, _}
import org.apache.spark.sql.{SQLContext, _}
scala> import org.apache.spark.sql.execution.datasources.hbase._
import org.apache.spark.sql.execution.datasources.hbase._
scala> import org.apache.spark.{SparkConf, SparkContext}
import org.apache.spark.{SparkConf, SparkContext}
scala> import spark.sqlContext.implicits._
import spark.sqlContext.implicits._
scala> def catalog = s"""{
|"table":{"namespace":"default", "name":"imdb_mdata"},
|"rowkey":"key",
|"columns":{
|"rowkey":{"cf":"rowkey", "col":"key", "type":"string"},
|"actors":{"cf":"cf", "col":"Actors", "type":"string"},
|"description":{"cf":"cf", "col":"Description", "type":"string"},
|"director":{"cf":"cf", "col":"Director", "type":"string"},
|"genre":{"cf":"cf", "col":"Genre", "type":"string"},
|"metascore":{"cf":"cf", "col":"Metascore", "type":"string"},
|"rating":{"cf":"cf", "col":"Rating", "type":"string"},
|"revenue":{"cf":"cf", "col":"Revenue", "type":"string"},
|"runtime_min":{"cf":"cf", "col":"Runtime_Min", "type":"string"},
|"title":{"cf":"cf", "col":"Title", "type":"string"},
|"votes":{"cf":"cf", "col":"Votes", "type":"string"},
|"year":{"cf":"cf", "col":"Year", "type":"string"}
|}
|}""".stripMargin
catalog: String
scala> def withCatalog(cat: String): DataFrame = {
spark.sqlContext
.read
.options(Map(HBaseTableCatalog.tableCatalog->cat))
.format("org.apache.spark.sql.execution.datasources.hbase")
.load()
}
withCatalog: (cat: String)org.apache.spark.sql.DataFrame
scala> val df = withCatalog(catalog)
19/11/18 17:34:50 WARN Configuration: hbase-site.xml:an attempt to override final parameter: dfs.support.append; Ignoring.
df: org.apache.spark.sql.DataFrame = [rowkey: string, actors: string ... 10 more fields]
scala> df.show()

```
sshuser@hn0-myspar: ~

sshuser@hn0-hbase: ~                                                    sshuser@hn0-myspar: ~

19/11/18 17:36:26 WARN Configuration: hbase-site.xml:an attempt to override final parameter: dfs.support.append;  Ignoring.
19/11/18 17:36:27 WARN Configuration: hbase-site.xml:an attempt to override final parameter: dfs.support.append;  Ignoring.
+------+--------------------+--------------------+----------------+--------------------+---------+------+-------+----------+--------------------+------+----+
|rowkey|              actors|         description|        director|               genre|metascore|rating|revenue|runtime_min|               title| votes|year|
+------+--------------------+--------------------+----------------+--------------------+---------+------+-------+----------+--------------------+------+----+
|     1|Chris Pratt; Vin ...|A group of interg...|       James Gunn|Action;Adventure;...|       76|   8.1| 333.13|       121|Guardians of the ...|757074|2014|
|    10|Jennifer Lawrence...|A spacecraft trav...|    Morten Tyldum|Adventure;Drama;R...|       41|     7| 100.01|       116|          Passengers|192177|2016|
|   100|Leonardo DiCaprio...|An undercover cop...|  Martin Scorsese|Crime;Drama;Thriller|       85|   8.5| 132.37|       151|        The Departed|937414|2006|
|  1000|Kevin Spacey; Jen...|A stuffy business...| Barry Sonnenfeld|Comedy;Family;Fan...|       11|   5.3|  19.64|        87|          Nine Lives| 12435|2016|
|   101|Tom Hardy; Emily ...|Identical twin ga...|  Brian Helgeland|Biography;Crime;D...|       55|     7|   1.87|       132|              Legend|108836|2015|
|   102|Chris Hemsworth; ...|The powerful but ...|  Kenneth Branagh|Action;Adventure;...|       57|     7| 181.02|       115|                Thor|570814|2011|
|   103|Matt Damon; Jessi...|An astronaut beco...|     Ridley Scott|Adventure;Drama;S...|       80|     8| 228.43|       144|         The Martian|556097|2015|
|   104|Mario Casas; Ana ...|A young businessm...|      Oriol Paulo|Crime;Mystery;Thr...|     null|   7.9|   null|       106|        Contratiempo|  7204|2016|
|   105|Henry Cavill; Arm...|In the early 1960...|      Guy Ritchie|Action;Adventure;...|       56|   7.3|  45.43|       116|The Man from U.N....|202973|2015|
|   106|Chris Pine; Ben F...|A divorced father...|  David Mackenzie|Crime;Drama;Thriller|       88|   7.7|  26.86|       102|  Hell or High Water|115546|2016|
|   107|Robert De Niro; L...|A look at the lif...|   Taylor Hackford|              Comedy|       40|   5.4|   1.66|       120|         The Comedian|  1954|2016|
|   108|Alexander Skarsgå...|Tarzan having acc...|      David Yates|Action;Adventure;...|       44|   6.3| 126.59|       110|The Legend of Tarzan|117590|2016|
|   109|Eve Lindley; Rich...|A mother struggle...|     Katie Holmes|               Drama|       48|   5.8|   null|       105|          All We Had|  1004|2016|
|    11|Eddie Redmayne; K...|The adventures of...|      David Yates|Adventure;Family;...|       66|   7.5| 234.02|       133|Fantastic Beasts ...|232072|2016|
|   110|Alicia Vikander; ...|A young programme...|     Alex Garland|Drama;Mystery;Sci-Fi|       78|   7.7|  25.44|       108|          Ex Machina|339797|2014|
|   111|John Gallagher Jr...|In a twisted soci...|      Greg McLean|Action;Horror;Thr...|       44|   6.3|  10.16|        89|The Belko Experiment|  3712|2016|
|   112|Chiwetel Ejiofor;...|In the antebellum...|   Steve McQueen|Biography;Drama;H...|       96|   8.1|  56.67|       134|    12 Years a Slave|486338|2013|
|   113|Keanu Reeves; Jas...|A dystopian love ...|Ana Lily Amirpour|      Romance;Sci-Fi|       65|   6.1|   null|       118|       The Bad Batch|   512|2016|
|   114|Gerard Butler; Le...|King Leonidas of ...|      Zack Snyder|  Action;Fantasy;War|       52|   7.7| 210.59|       117|                 300|637104|2006|
|   115|Daniel Radcliffe;...|Harry Ron and Her...|      David Yates|Adventure;Drama;F...|       87|   8.1| 380.96|       130|Harry Potter and ...|590595|2011|
+------+--------------------+--------------------+----------------+--------------------+---------+------+-------+----------+--------------------+------+----+
only showing top 20 rows


scala> df.createTempView("imdb_Mdata")

scala> spark.sqlContext.sql("select title, rating from imdb_Mdata").show
+--------------------+------+
|               title|rating|
+--------------------+------+
|Guardians of the ...|   8.1|
|          Passengers|     7|
|        The Departed|   8.5|
|          Nine Lives|   5.3|
|              Legend|     7|
|                Thor|     7|
|         The Martian|     8|
|        Contratiempo|   7.9|
|The Man from U.N....|   7.3|
|  Hell or High Water|   7.7|
|         The Comedian|   5.4|
|The Legend of Tarzan|   6.3|
|          All We Had|   5.8|
|Fantastic Beasts ...|   7.5|
|          Ex Machina|   7.7|
|The Belko Experiment|   6.3|
|    12 Years a Slave|   8.1|
|       The Bad Batch|   6.1|
|                 300|   7.7|
```

```
scala> def withCatalog(cat: String): DataFrame = {
     |     spark.sqlContext
     |     .read
     |     .options(Map(HBaseTableCatalog.tableCatalog->cat))
     |     .format("org.apache.spark.sql.execution.datasources.hbase")
     |     .load()
     | }
withCatalog: (cat: String)org.apache.spark.sql.DataFrame

scala> val df = withCatalog(catalog)
19/11/18 21:51:01 WARN Configuration: hbase-site.xml:an attempt to override final parameter: dfs.support.append;  Ignoring.
df: org.apache.spark.sql.DataFrame = [rowkey: string, adult: string ... 17 more fields]

scala> df.show()
```

```
+-----+-----+--------+--------------------+---------+----------------+--------------------+--------------------+---------+--------------------+--------------------+-----------+--------+------+-------
-------------+--------+--------------------+-----------+----------+
|   id|adult|  budget|              genres|  imdb_id|original_language|      original_title|            overview|popularity|production_companies|production_countries|release_date|  revenue|runtime|    spo
ken_languages|  status|             tagline|vote_average|vote_count|
|  862|FALSE|30000000|[{'id': 16 'name'...|tt0114709|               en|           Toy Story|Led by Woody Andy...| 21.946943|[{'name': 'Pixar ...|[{'iso_3166_1': '...|  10/30/1995|373554033|     81|[{'iso_
639_1': 'e...|Released|            null|         7.7|      5415|
| 8844|FALSE|65000000|[{'id': 12 'name'...|tt0113497|               en|             Jumanji|When siblings Jud...| 17.015539|[{'name': 'TriSta...|[{'iso_3166_1': '...|  12/15/1995|262797249|    104|[{'iso_
639_1': 'e...|Released|Roll the dice and...|         6.9|      2413|
|15602|FALSE|       0|[{'id': 10749 'na...|tt0113228|               en|    Grumpier Old Men|A family wedding ...|   11.7129|[{'name': 'Warner...|[{'iso_3166_1': '...|  12/22/1995|        0|    101|[{'iso_
639_1': 'e...|Released|Still Yelling. St...|         6.5|        92|
|31357|FALSE|16000000|[{'id': 35 'name'...|tt0114885|               en|   Waiting to Exhale|"Cheated on mistr...|  3.859495|[{'name': 'Twenti...|[{'iso_3166_1': '...|  12/22/1995| 81452156|    127|[{'iso_
639_1': 'e...|Released|Friends are the p...|         6.1|        34|
|11862|FALSE|       0|[{'id': 35 'name'...|tt0113041|               en|Father of the Bri...|Just when George ...|  8.387519|[{'name': 'Sandol...|[{'iso_3166_1': '...|   2/10/1995| 76578911|    106|[{'iso_
639_1': 'e...|Released|Just When His Wor...|         5.7|       173|
|  949|FALSE|60000000|[{'id': 28 'name'...|tt0113277|               en|                Heat|Obsessive master ...| 17.924927|[{'name': 'Regenc...|[{'iso_3166_1': '...|  12/15/1995|187436818|    170|[{'iso_
639_1': 'e...|Released|A Los Angeles Cri...|         7.7|      1886|
|11860|FALSE|58000000|[{'id': 35 'name'...|tt0114319|               en|             Sabrina|An ugly duckling ...|  6.677277|[{'name': 'Paramo...|[{'iso_3166_1': '...|  12/15/1995|        0|    127|[{'iso_
639_1': 'f...|Released|You are cordially...|         6.2|       141|
|45325|FALSE|       0|[{'id': 28 'name'...|tt0112302|               en|        Tom and Huck|A mischievous you...|  2.561161|[{'name': 'Walt D...|[{'iso_3166_1': '...|  12/22/1995|        0|     97|[{'iso_
639_1': 'e...|Released|The Original Bad ...|         5.4|        45|
| 9091|FALSE|35000000|[{'id': 28 'name'...|tt0114576|               en|        Sudden Death|International act...|   5.23158|[{'name': 'Univer...|[{'iso_3166_1': '...|  12/22/1995| 64350171|    106|[{'iso_
639_1': 'e...|Released|Terror goes into ...|         5.5|       174|
|  710|FALSE|58000000|[{'id': 12 'name'...|tt0113189|               en|           GoldenEye|James Bond must u...| 14.686036|[{'name': 'United...|[{'iso_3166_1': '...|  11/16/1995|352194034|    130|[{'iso_
639_1': 'e...|Released|No limits. No fea...|         6.6|      1194|
| 9087|FALSE|62000000|[{'id': 35 'name'...|tt0112346|               en|The American Pres...|Widowed U.S. pres...|  6.318445|[{'name': 'Columb...|[{'iso_3166_1': '...|  11/17/1995|107879496|    106|[{'iso_
639_1': 'e...|Released|Why can't the mos...|         6.5|       199|
|12110|FALSE|       0|[{'id': 35 'name'...|tt0112896|               en|            Dracula|Dracula: Dead and...|When a lawyer sho...|  5.430331|[{'name': 'Columb...|[{'iso_3166_1': '...|  12/22/1995|        0|     88|[{'iso_
639_1': 'e...|Released|                null|         5.7|       210|
|21032|FALSE|       0|[{'id': 10751 'na...|tt0112453|               en|               Balto|An outcast half-w...| 12.140733|[{'name': 'Univer...|[{'iso_3166_1': '...|  12/22/1995| 11348324|     78|[{'iso_
639_1': 'e...|Released|Part Dog. Part Wo...|         7.1|       423|
|10858|FALSE|44000000|[{'id': 36 'name'...|tt0113987|               en|              Nixon|An all-star cast ...|     5.092|[{'name': 'Hollyw...|[{'iso_3166_1': '...|  12/22/1995| 13681765|    192|[{'iso_
639_1': 'e...|Released|Triumphant in Vic...|         7.1|        72|
| 1408|FALSE|98000000|[{'id': 28 'name'...|tt0112760|               en|    Cutthroat Island|Morgan Adams and ...|  7.284477|[{'name': 'Le Stu...|[{'iso_3166_1': '...|  12/22/1995| 10017322|    119|[{'iso_
639_1': 'e...|Released|The Course Has Be...|         5.7|       137|
|  524|FALSE|52000000|[{'id': 18 'name'...|tt0112641|               en|              Casino|The life of the g...| 10.137389|[{'name': 'Univer...|[{'iso_3166_1': '...|  11/22/1995|116112375|    178|[{'iso_
639_1': 'e...|Released|No one stays at t...|         7.8|      1343|
| 4584|FALSE|16500000|[{'id': 18 'name'...|tt0114388|               en|Sense and Sensibi...|Rich Mr. Dashwood...| 10.673167|[{'name': 'Columb...|[{'iso_3166_1': '...|  12/13/1995|135000000|    136|[{'iso_
639_1': 'e...|Released|Lose your heart a...|         7.2|       364|
|    5|FALSE| 4000000|[{'id': 80 'name'...|tt0113101|               en|          Four Rooms|It's Ted the Bell...|  9.026586|[{'name': 'Mirama...|[{'iso_3166_1': '...|   12/9/1995|  4300000|     98|[{'iso_
639_1': 'e...|Released|Twelve outrageous...|         6.5|       539|
| 9273|FALSE|30000000|[{'id': 80 'name'...|tt0112281|               en|Ace Ventura: When...|Summoned from an ...|  8.205448|[{'name': 'O Ente...|[{'iso_3166_1': '...|  11/10/1995|212385533|     90|[{'iso_
639_1': 'e...|Released|New animals. New ...|         6.1|      1128|
+-----+-----+--------+--------------------+---------+----------------+--------------------+--------------------+---------+--------------------+--------------------+------------+---------+-------+-------
-------------+--------+--------------------+-----------+----------+
only showing top 20 rows

>>>
```

SQL Results:

-------movie count each release_year----------

```
>>> result2 = spark.sql("SELECT _c6, count(*) as _c1 from df group by _c6 order by _c1 DESC")
>>> result2.collect()
[Row(_c6=u'2016', _c1=297), Row(_c6=u'2015', _c1=127), Row(_c6=u'2014', _c1=98), Row(_c6=u'2013', _c1=91), Row(_c6=u'2012', _c1=64), Row(_c6=u'2011', _c1=63), Row(_c6=u'2010', _c1=60), Row(_c6=u'2007', _c
1=53), Row(_c6=u'2008', _c1=52), Row(_c6=u'2009', _c1=51), Row(_c6=u'2006', _c1=44)]
>>>
>>>
>>>
>>>
>>>
>>>
```

-------Movie count each director directed----------

```
>>>
>>>
>>> result4 = spark.sql("SELECT _c4, count(*) as _c1 from df group by _c4 having _c1>4 order by _c1 DESC")
>>> result4.collect()
[Stage 18:==============================>                          (115 + 6) / 2[Stage 18:=============================================>         (165 + 7) / 2
                        [Stage 20:=============================================>                           (145 + 4) / 2[Stage 20:=================================================>     (181 + 4) / 2
                                                                [Row(_c4=u'Ridley Scott', _c1=8), Row(_c4=u'Paul W.S. Anderson', _c1=6), Row(_c4=u'Michael Bay', _c1=6), Row(_c4=u'David Yates', _c1=6), Row(_c4=u'M.
Night Shyamalan', _c1=6), Row(_c4=u'Antoine Fuqua', _c1=5), Row(_c4=u'Zack Snyder', _c1=5), Row(_c4=u'Danny Boyle', _c1=5), Row(_c4=u'Justin Lin', _c1=5), Row(_c4=u'J.J. Abrams', _c1=5), Row(_c4=u'Woody A
llen', _c1=5), Row(_c4=u'Peter Berg', _c1=5), Row(_c4=u'David Fincher', _c1=5), Row(_c4=u'Christopher Nolan', _c1=5), Row(_c4=u'Martin Scorsese', _c1=5), Row(_c4=u'Denis Villeneuve', _c1=5)]
>>>
```

--------movie count each year with rating---------------------------------

```
', 'CLUSTER', 'DISTRIBUTE')(line 1, pos 21)\n\n== SQL ==\nSELECT _c6, _c8 count(*) as _c1 from df group by _c6,_c8 having _c1>4 order by _c1 DESC\n---------------------^^^\n"
>>> result6 = spark.sql("SELECT _c6, _c8, count(*) as _c1 from df group by _c6,_c8 having _c1>4 order by _c1 DESC")
>>> result6.collect()
[Stage 23:=============================================>           (156 + 4) /                                                                                  [Stage 25:=====================================>
          (141 + 4) / [Stage 25:=============================================>   (187 + 5) /                                                                                  [Row(_c6=u'2016', _c8=u'6.3'
, _c1=19), Row(_c6=u'2016', _c8=u'7.4', _c1=14), Row(_c6=u'2016', _c8=u'6.1', _c1=14), Row(_c6=u'2016', _c8=u'7.5', _c1=13), Row(_c6=u'2016', _c8=u'5.8', _c1=12), Row(_c6=u'2016', _c8=u'6.8', _c1=12), Row
(_c6=u'2016', _c8=u'7.1', _c1=11), Row(_c6=u'2016', _c8=u'6.7', _c1=11), Row(_c6=u'2016', _c8=u'7.2', _c1=11), Row(_c6=u'2016', _c8=u'6', _c1=10), Row(_c6=u'2016', _c8=u'6.5', _c1=10), Row(_c6=u'2016', _c
8=u'6.9', _c1=10), Row(_c6=u'2016', _c8=u'5.7', _c1=10), Row(_c6=u'2015', _c8=u'7.1', _c1=10), Row(_c6=u'2016', _c8=u'6.4', _c1=10), Row(_c6=u'2016', _c8=u'7.3', _c1=9), Row(_c6=u'2016', _c8=u'7', _c1=9),
 Row(_c6=u'2015', _c8=u'6.3', _c1=9), Row(_c6=u'2013', _c8=u'7', _c1=9), Row(_c6=u'2016', _c8=u'6.6', _c1=8), Row(_c6=u'2015', _c8=u'6.7', _c1=8), Row(_c6=u'2014', _c8=u'6.2', _c1=8), Row(_c6=u'2016', _c8
=u'6.2', _c1=8), Row(_c6=u'2016', _c8=u'7.9', _c1=8), Row(_c6=u'2007', _c8=u'7.1', _c1=7), Row(_c6=u'2016', _c8=u'5.6', _c1=7), Row(_c6=u'2014', _c8=u'8.1', _c1=7), Row(_c6=u'2015', _c8=u'7.3', _c1=7), Ro
w(_c6=u'2015', _c8=u'6.5', _c1=7), Row(_c6=u'2016', _c8=u'5.9', _c1=7), Row(_c6=u'2013', _c8=u'7.3', _c1=6), Row(_c6=u'2014', _c8=u'6', _c1=6), Row(_c6=u'2016', _c8=u'5.3', _c1=6), Row(_c6=u'2015', _c8=u'
6.6', _c1=6), Row(_c6=u'2014', _c8=u'6.7', _c1=6), Row(_c6=u'2013', _c8=u'6.5', _c1=6), Row(_c6=u'2013', _c8=u'7.8', _c1=6), Row(_c6=u'2007', _c8=u'7.2', _c1=6), Row(_c6=u'2015', _c8=u'6', _c1=6), Row(_c6
=u'2013', _c8=u'6.7', _c1=6), Row(_c6=u'2015', _c8=u'7', _c1=6), Row(_c6=u'2014', _c8=u'6.5', _c1=6), Row(_c6=u'2015', _c8=u'7.2', _c1=5), Row(_c6=u'2012', _c8=u'7', _c1=5), Row(_c6=u'2014', _c8=u'7.8', _c
1=5), Row(_c6=u'2008', _c8=u'6.6', _c1=5), Row(_c6=u'2007', _c8=u'7.5', _c1=5), Row(_c6=u'2013', _c8=u'7.5', _c1=5), Row(_c6=u'2011', _c8=u'7.1', _c1=5), Row(_c6=u'2013', _c8=u'6.2', _c1=5), Row(_c6=u'20
16', _c8=u'5.2', _c1=5), Row(_c6=u'2016', _c8=u'7.7', _c1=5), Row(_c6=u'2012', _c8=u'7.2', _c1=5), Row(_c6=u'2013', _c8=u'6.6', _c1=5), Row(_c6=u'2013', _c8=u'7.1', _c1=5), Row(_c6=u'2008', _c8=u'7.1', _c
1=5), Row(_c6=u'2016', _c8=u'5.4', _c1=5), Row(_c6=u'2010', _c8=u'6.8', _c1=5), Row(_c6=u'2014', _c8=u'6.3', _c1=5), Row(_c6=u'2015', _c8=u'5.7', _c1=5)]
>>>
```

--------Revenue Stylized Facts per year----------------

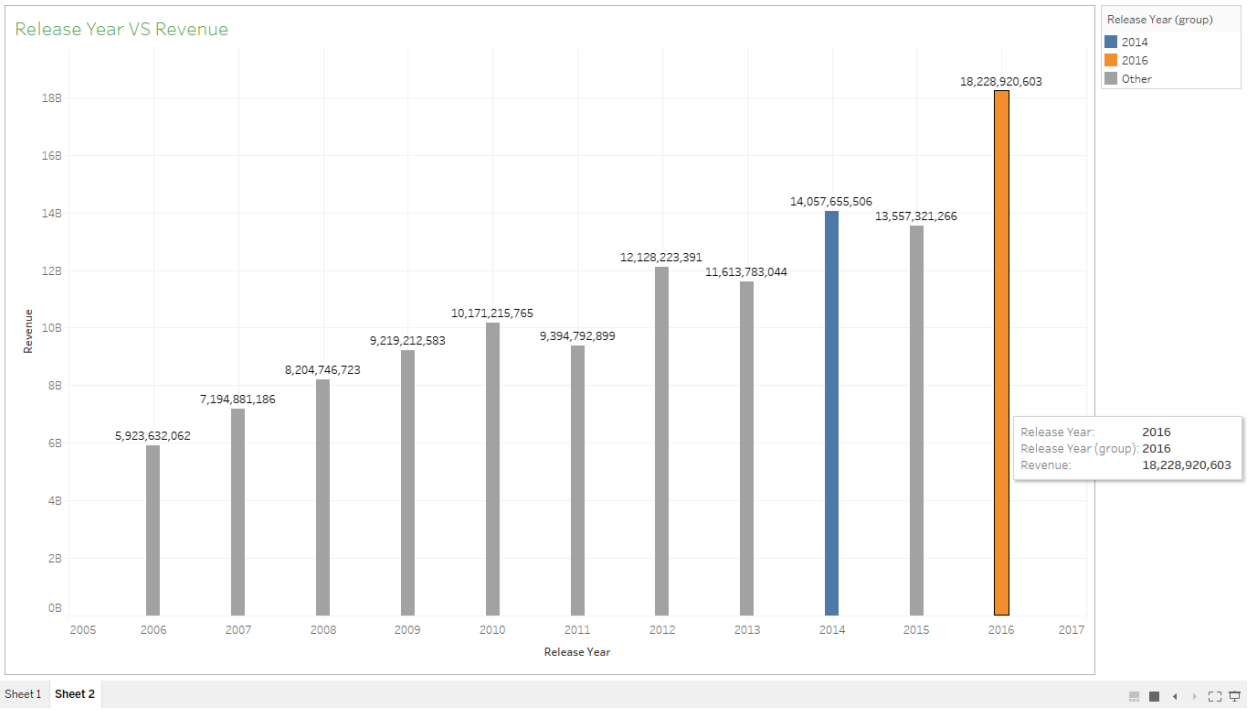
```
SyntaxError: invalid syntax
>>> result5 = spark.sql("SELECT _c6, _c8 count(*) as _c1 from df group>>> r>>> result7 = spark.sql("SELECT _c6, sum(_c10) as sum_revenue, avg(_c10) as Avg_revenue, min(_c10) as min_revenue, max(_c10) as m
ax_revenue from df group by _c6 order by _c6 DESC")
>>> result7.collect()
[Stage 28:=====================>                                   (79 + 4) /[Stage 28:================================>                (116 + 4) /[Stage 28:=====================================>
         (142 + 4) /[Stage 28:================================================>     (179 + 4) /[Stage 28:=========================================================> (193 + 4) /
                                    [Stage 30:============================>                            (114 + 4) /[Stage 30:=======================================>             (149 + 4) /[Stage 30:==
==============================================>  (190 + 4) /                                                                  [Row(_c6=u'2016', sum_revenue=11211.650000000003, Avg_revenue=54.6
90975609756116, min_revenue=u'0', max_revenue=u'97.66'), Row(_c6=u'2015', sum_revenue=8854.1200000000026, Avg_revenue=78.355044247787632, min_revenue=u'0.01', max_revenue=u'936.63'), Row(_c6=u'2014', sum_
revenue=7997.3999999999978, Avg_revenue=85.0787234042553, min_revenue=u'0.01', max_revenue=u'91.12'), Row(_c6=u'2013', sum_revenue=7666.7199999999966, Avg_revenue=87.121818181818142, min_revenue=u'0.03',
 max_revenue=u'98.9'), Row(_c6=u'2012', sum_revenue=6910.2900000000027, Avg_revenue=107.97328125000004, min_revenue=u'0.02', max_revenue=u'95.72'), Row(_c6=u'2011', sum_revenue=5431.9600000000009, Avg_reve
nue=87.612258064516141, min_revenue=u'0.03', max_revenue=u'85.46'), Row(_c6=u'2010', sum_revenue=5989.6500000000005, Avg_revenue=105.08157894736843, min_revenue=u'0.02', max_revenue=u'96.92'), Row(_c6=u'2
009', sum_revenue=5292.2600000000011, Avg_revenue=112.60127659574471, min_revenue=u'0.06', max_revenue=u'97.03'), Row(_c6=u'2008', sum_revenue=5053.2200000000021, Avg_revenue=99.082745098039254, min_reven
ue=u'0.07', max_revenue=u'9.03'), Row(_c6=u'2007', sum_revenue=4306.2300000000005, Avg_revenue=87.882244897959197, min_revenue=u'0.04', max_revenue=u'82.23'), Row(_c6=u'2006', sum_revenue=3624.46000000000
09, Avg_revenue=86.296666666666695, min_revenue=u'0.44', max_revenue=u'88.5')]
>>>
```

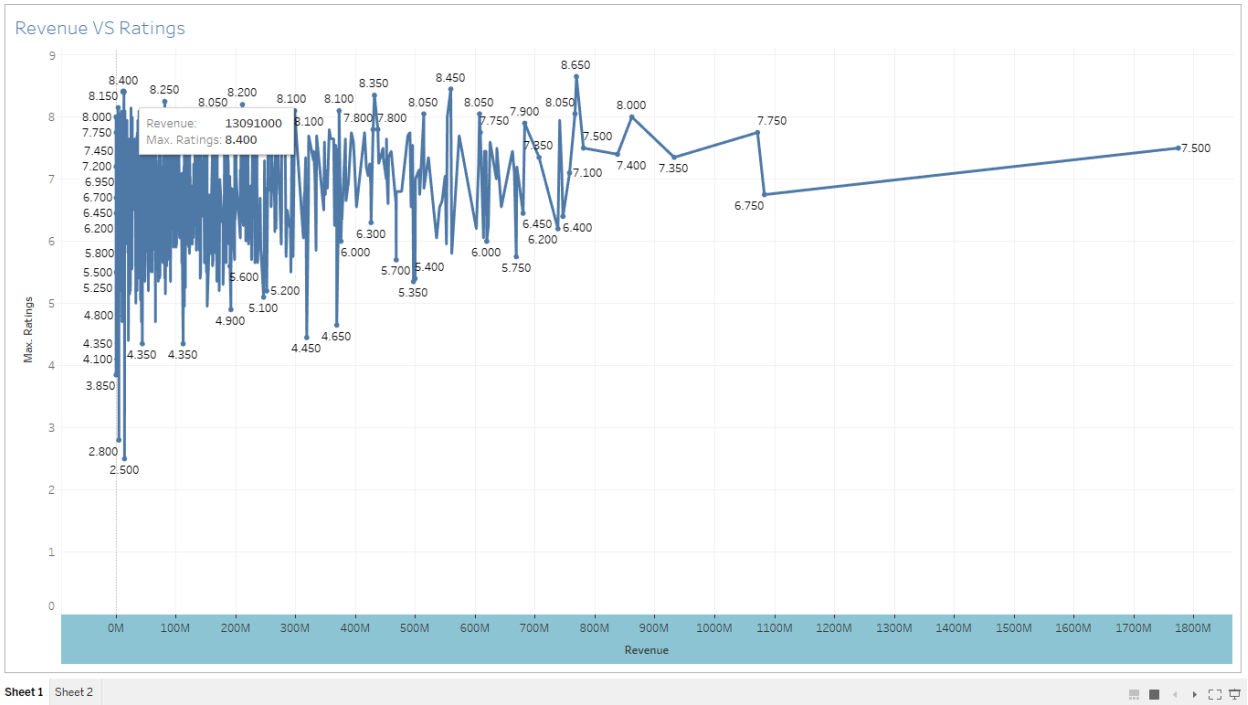
--------Avg. Revenue of movies per director---------------

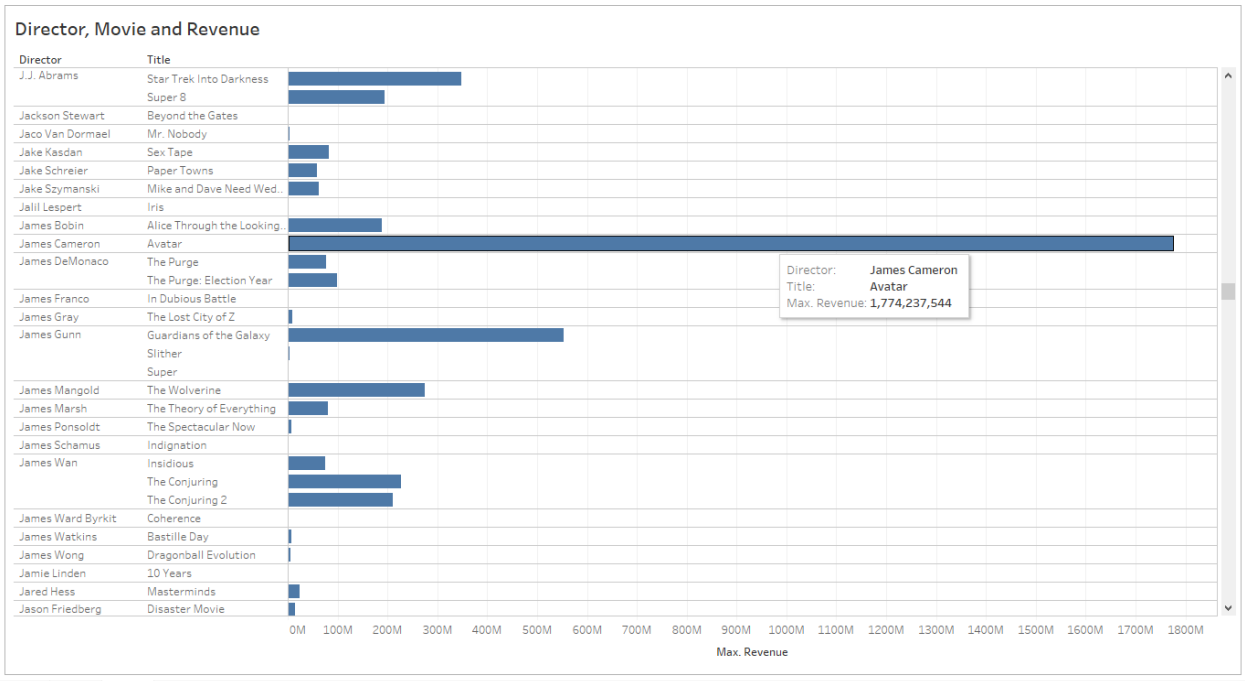
```
>>> result8 = spark.sql("SELECT _c4, avg(_c10) as Avg_revenue from df group by _c4 HAVING Avg_revenue > 100 order by Avg_revenue DESC")
>>> result8.collect()
[Stage 33:=================================>                       (126 + 4) [Stage 33:=============================================>      (177 + 5)
          [Stage 35:=====================>                                 (80 + 5) [Stage 35:================================>                (118 + 4) [Stage 35:=============================
==============>      (157 + 4)                                                  [Row(_c4=u'James Cameron', Avg_revenue=760.50999999999999), Row(_c4=u'Colin Trevorrow', Avg_re
venue=652.17999999999995), Row(_c4=u'Joss Whedon', Avg_revenue=541.13499999999999), Row(_c4=u'Lee Unkrich', Avg_revenue=414.98000000000002), Row(_c4=u'Gary Ross', Avg_revenue=408.0), Row(_c4=u'Chris Buck'
, Avg_revenue=400.74000000000001), Row(_c4=u'Chris Renaud', Avg_revenue=368.31), Row(_c4=u'Gareth Edwards', Avg_revenue=366.41499999999996), Row(_c4=u'Tim Miller', Avg_revenue=363.01999999999998), Row(_c4
=u'Byron Howard', Avg_revenue=341.25999999999999), Row(_c4=u'J.J. Abrams', Avg_revenue=336.68999999999994), Row(_c4=u'Kyle Balda', Avg_revenue=336.02999999999997), Row(_c4=u'Anthony Russo', Avg_revenue=33
3.91499999999996), Row(_c4=u'Francis Lawrence', Avg_revenue=324.95249999999999), Row(_c4=u'Pete Docter', Avg_revenue=324.71500000000003), Row(_c4=u'Pierre Coffin', Avg_revenue=309.77499999999998), Row(_c4
=u'Christopher Nolan', Avg_revenue=303.01800000000003), Row(_c4=u'David Slade', Avg_revenue=300.51999999999998), Row(_c4=u'Bill Condon', Avg_revenue=286.78999999999996), Row(_c4=u'Sam Raimi', Avg_revenue=
285.71499999999997), Row(_c4=u'David Yates', Avg_revenue=271.75166666666667), Row(_c4=u'Christophe Lourdelet', Avg_revenue=270.31999999999999), Row(_c4=u'Dan Scanlon', Avg_revenue=268.49000000000001), Row
(_c4=u'Andrew Stanton', Avg_revenue=261.05333333333334), Row(_c4=u'Jon Favreau', Avg_revenue=256.39999999999998), Row(_c4=u'Robert Stromberg', Avg_revenue=241.41), Row(_c4=u'Mark Andrews', Avg_revenue=237
.28), Row(_c4=u'Michael Bay', Avg_revenue=236.88666666666666), Row(_c4=u'Shane Black', Avg_revenue=222.62), Row(_c4=u'Don Hall', Avg_revenue=222.49000000000001), Row(_c4=u'John Lasseter', Avg_revenue=217.
75), Row(_c4=u'Mark Osborne', Avg_revenue=215.40000000000001), Row(_c4=u'Peter Jackson', Avg_revenue=215.11250000000001), Row(_c4=u'Gore Verbinski', Avg_revenue=207.45499999999998), Row(_c4=u'Nathan Greno
', Avg_revenue=200.81), Row(_c4=u'Dean DeBlois', Avg_revenue=197.19499999999999), Row(_c4=u'Bryan Singer', Avg_revenue=196.43666666666664), Row(_c4=u'Phil Lord', Avg_revenue=195.94333333333333), Row(_c4=u'
Zack Snyder', Avg_revenue=195.148), Row(_c4=u'Catherine Hardwicke', Avg_revenue=191.44999999999999), Row(_c4=u'Rich Moore', Avg_revenue=189.41), Row(_c4=u'Marc Forster', Avg_revenue=185.36000000000001), R
ow(_c4=u'Rob Marshall', Avg_revenue=184.53), Row(_c4=u'Tim Johnson', Avg_revenue=177.34), Row(_c4=u'Joe Johnston', Avg_revenue=176.63999999999999), Row(_c4=u'Ron Clements', Avg_revenue=176.56), Row(_c4=u'
George Miller', Avg_revenue=175.81), Row(_c4=u'Sam Mendes', Avg_revenue=175.77000000000001), Row(_c4=u'Brad Bird', Avg_revenue=169.74000000000001), Row(_c4=u'Genndy Tartakovsky', Avg_revenue=169.69), Row(
_c4=u'Theodore Melfi', Avg_revenue=169.27000000000001), Row(_c4=u'Sam Taylor-Johnson', Avg_revenue=166.15000000000001), Row(_c4=u'Marc Webb', Avg_revenue=165.75666666666666), Row(_c4=u'Paul Greengrass', A
vg_revenue=165.46666666666667), Row(_c4=u'Justin Lin', Avg_revenue=164.958), Row(_c4=u'Clint Eastwood', Avg_revenue=164.7475), Row(_c4=u'James Wan', Avg_revenue=160.96749999999997), Row(_c4=u'Todd Phillip
s', Avg_revenue=160.16499999999999), Row(_c4=u'Steven Spielberg', Avg_revenue=156.7475), Row(_c4=u'Brad Peyton', Avg_revenue=155.18000000000001), Row(_c4=u'Rupert Sanders', Avg_revenue=155.11000000000001)
, Row(_c4=u'Alfonso Cuar\xf3n', Avg_revenue=154.685), Row(_c4=u'Walt Dohrn', Avg_revenue=153.69), Row(_c4=u'Brett Ratner', Avg_revenue=153.50999999999999), Row(_c4=u'David Ayer', Avg_revenue=150.569999999
99999), Row(_c4=u'Stephen Sommers', Avg_revenue=150.16999999999999), Row(_c4=u'Peyton Reed', Avg_revenue=149.435), Row(_c4=u'Tom McGrath', Avg_revenue=148.34), Row(_c4=u'Alan Taylor', Avg_revenue=148.0450
00000000002), Row(_c4=u'Kenneth Branagh', Avg_revenue=144.24000000000001), Row(_c4=u'Phyllida Lloyd', Avg_revenue=143.69999999999999), Row(_c4=u'Carlos Saldanha', Avg_revenue=143.62), Row(_c4=u'Alessandro
Carloni', Avg_revenue=143.52000000000001), Row(_c4=u'Paul Feig', Avg_revenue=141.95500000000001), Row(_c4=u'Martin Campbell', Avg_revenue=141.80000000000001), Row(_c4=u'Scott Derrickson', Avg_revenue=140.
32999999999998), Row(_c4=u'Rawson Marshall Thurber', Avg_revenue=138.875), Row(_c4=u'Tim Burton', Avg_revenue=138.505), Row(_c4=u'Jonathan Liebesman', Avg_revenue=137.255), Row(_c4=u'James Mangold', Avg_r
evenue=132.55000000000001), Row(_c4=u'John Stockwell', Avg_revenue=131.56), Row(_c4=u'Joseph Kosinski', Avg_revenue=130.535), Row(_c4=u'Kevin Lima', Avg_revenue=127.70999999999999), Row(_c4=u'Christopher
McQuarrie', Avg_revenue=126.84), Row(_c4=u'Josh Boone', Avg_revenue=124.87), Row(_c4=u'Dennis Dugan', Avg_revenue=124.67999999999999), Row(_c4=u'Michael Patrick King', Avg_revenue=123.98499999999999), Row
(_c4=u'Tate Taylor', Avg_revenue=122.51000000000001), Row(_c4=u'Seth Gordon', Avg_revenue=117.53), Row(_c4=u'John Lee Hancock', Avg_revenue=117.34666666666668), Row(_c4=u'F. Gary Gray', Avg_revenue=117.18
5), Row(_c4=u'Gabriele Muccino', Avg_revenue=116.27000000000001), Row(_c4=u'Mark Steven Johnson', Avg_revenue=115.8), Row(_c4=u'Angelina Jolie', Avg_revenue=115.59999999999999), Row(_c4=u'Anne Fletcher',
Avg_revenue=114.60999999999999), Row(_c4=u'Seth MacFarlane', Avg_revenue=114.17), Row(_c4=u'Roland Emmerich', Avg_revenue=114.11666666666667), Row(_c4=u'James Gunn', Avg_revenue=113.73999999999999), Row(_
c4=u'Jason Reitman', Avg_revenue=113.65000000000001), Row(_c4=u'Tony Gilroy', Avg_revenue=113.17), Row(_c4=u'Jon Lucas', Avg_revenue=113.08), Row(_c4=u'Quentin Tarantino', Avg_revenue=112.48), Row(_c4=u'G
uy Ritchie', Avg_revenue=111.74250000000001), Row(_c4=u'Ryan Coogler', Avg_revenue=109.70999999999999), Row(_c4=u'Adam McKay', Avg_revenue=109.535), Row(_c4=u'Peter Billingsley', Avg_revenue=109.180000000
00001), Row(_c4=u'Judd Apatow', Avg_revenue=108.75333333333333), Row(_c4=u'David O. Russell', Avg_revenue=108.05500000000001), Row(_c4=u'Clay Kaytis', Avg_revenue=107.51000000000001), Row(_c4=u'Noam Murro
', Avg_revenue=106.37), Row(_c4=u'Louis Leterrier', Avg_revenue=105.56750000000001), Row(_c4=u'David Fincher', Avg_revenue=105.54400000000001), Row(_c4=u'Rupert Wyatt', Avg_revenue=105.185), Row(_c4=u'Har
ald Zwart', Avg_revenue=103.88), Row(_c4=u'Sean Anders', Avg_revenue=102.36499999999999), Row(_c4=u'Peter Berg', Avg_revenue=102.26599999999999), Row(_c4=u'Steven Soderbergh', Avg_revenue=102.163333333333
34), Row(_c4=u'Matt Reeves', Avg_revenue=100.23333333333333), Row(_c4=u'Tom Hooper', Avg_revenue=100.09666666666668)]
>>>
```

--------Ratings Stylized Facts per year----------

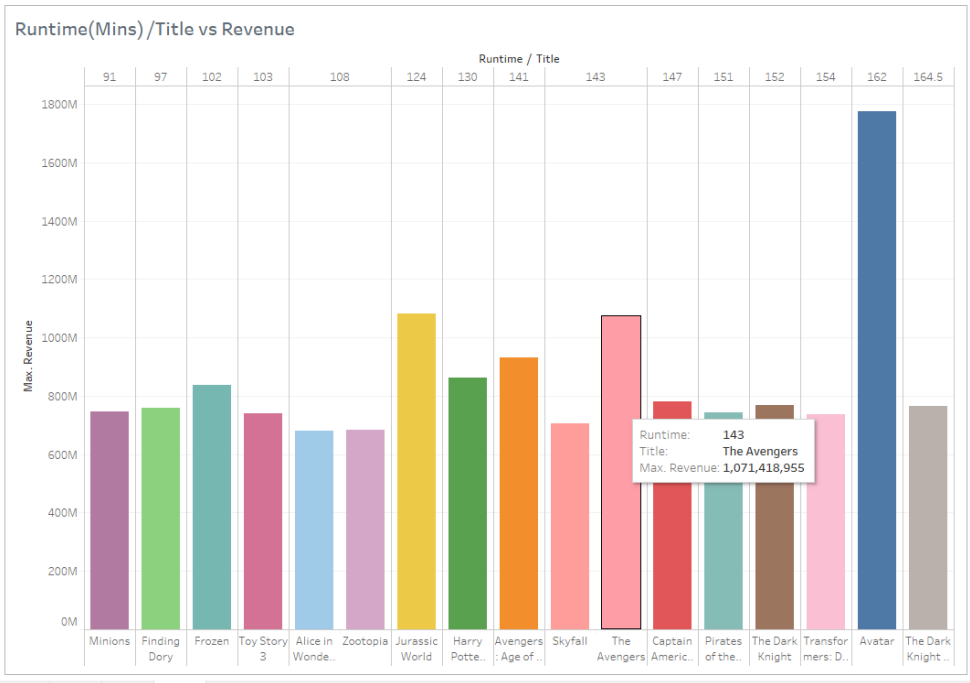
```
>>> result9 = spark.sql("SELECT _c6, sum(_c8) as sum_rating, avg(_c8) as avg_rating, min(_c8) as min_ratings, max(_c8) as max_rating from df group by _c6 order by _c6 DESC")
>>> result9.collect()
[Stage 38:=============================================>          (160 + 4[Stage 38:=================================================>(197 + 3
          [Stage 40:============================>                          (100 + 4[Stage 40:=======================================>          (146 + 4[Stage 40:=================================
=============>  (191 + 4                                                  [Row(_c6=u'2016', sum_rating=1911.7, avg_rating=6.436700336700337, min_ratings=u'2.7', max_rating=u'8.8'), R
ow(_c6=u'2015', sum_rating=838.5000000000023, avg_rating=6.602362204724411, min_ratings=u'3.5', max_rating=u'8.3'), Row(_c6=u'2014', sum_rating=670.1000000000002, avg_rating=6.8377551020408163, min_rati
ngs=u'5.1', max_rating=u'8.6'), Row(_c6=u'2013', sum_rating=619.8999999999986, avg_rating=6.812087912087911, min_ratings=u'4.3', max_rating=u'8.2'), Row(_c6=u'2012', sum_rating=443.19999999999993, avg_ra
ting=6.924999999999989, min_ratings=u'5.3', max_rating=u'8.5'), Row(_c6=u'2011', sum_rating=430.80000000000007, avg_rating=6.838095238095239 6, min_ratings=u'4.9', max_rating=u'8.6'), Row(_c6=u'2010', sum
_rating=409.60000000000008, avg_rating=6.826666666666668, min_ratings=u'4.2', max_rating=u'8.8'), Row(_c6=u'2009', sum_rating=355.0, avg_rating=6.9607843137254903, min_ratings=u'2.7', max_rating=u'8.4'),
Row(_c6=u'2008', sum_rating=352.79999999999995, avg_rating=6.784615384615383 6, min_ratings=u'1.9', max_rating=u'9'), Row(_c6=u'2007', sum_rating=378.09999999999991, avg_rating=7.133962264150941 5, min_rati
ngs=u'4.7', max_rating=u'8.5'), Row(_c6=u'2006', sum_rating=313.50000000000006, avg_rating=7.125000000000009, min_ratings=u'5.6', max_rating=u'8.5')]
>>>
```



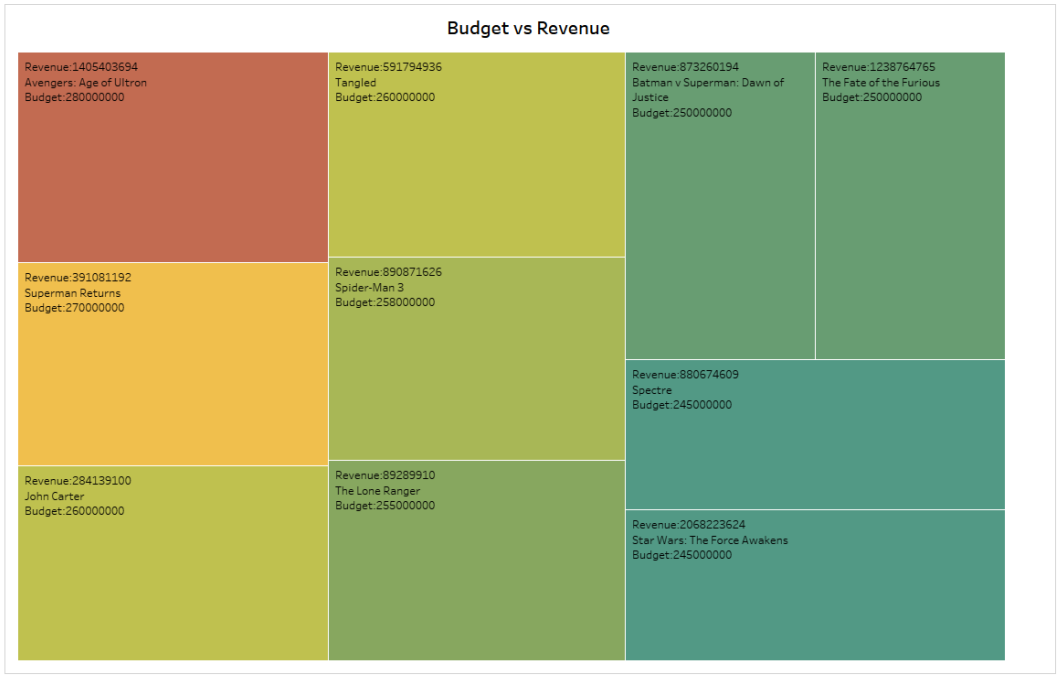



Director, Movie and Revenue
Director
Title
J.J. Abrams
Star Trek Into Darkness
Super 8
Jackson Stewart
Beyond the Gates
Jaco Van Dormael
Mr. Nobody
Jake Kasdan
Sex Tape
Jake Schreier
Paper Towns
Jake Szymanski
Mike and Dave Need Wed..
Jalil Lespert
Iris
James Bobin
Alice Through the Looking..
James Cameron
Avatar
James DeMonaco
The Purge
The Purge: Election Year
James Franco
In Dubious Battle
James Gray
The Lost City of Z
James Gunn
Guardians of the Galaxy
Slither
Super
James Mangold
The Wolverine
James Marsh
The Theory of Everything
James Ponsoldt
The Spectacular Now
James Schamus
Indignation
James Wan
Insidious
The Conjuring
The Conjuring 2
James Ward Byrkit
Coherence
James Watkins
Bastille Day
James Wong
Dragonball Evolution
Jamie Linden
10 Years
Jared Hess
Masterminds
Jason Friedberg
Disaster Movie
Director: James Cameron
Title: Avatar
Max. Revenue: 1,774,237,544
0M 100M 200M 300M 400M 500M 600M 700M 800M 900M 1000M 1100M 1200M 1300M 1400M 1500M 1600M 1700M 1800M
Max. Revenue
Sheet 1
Sheet 2
Sheet 3


Top-10 Movie and Revenue
Frozen
Harry Potter and the Deathly Hallows: Part 2
The Dark Knight Rises
Avatar
Avengers: Age of Ultron
Jurassic World
The Dark Knight
Captain America: Civil War
Minions
The Avengers
Title
Avatar
Avengers: Age of Ultron
Captain America: Civil War
Frozen
Harry Potter and the Deathly Hallows: Part 2
Jurassic World
Minions
The Avengers
The Dark Knight
The Dark Knight Rises
Sheet 1
Sheet 2
Sheet 3

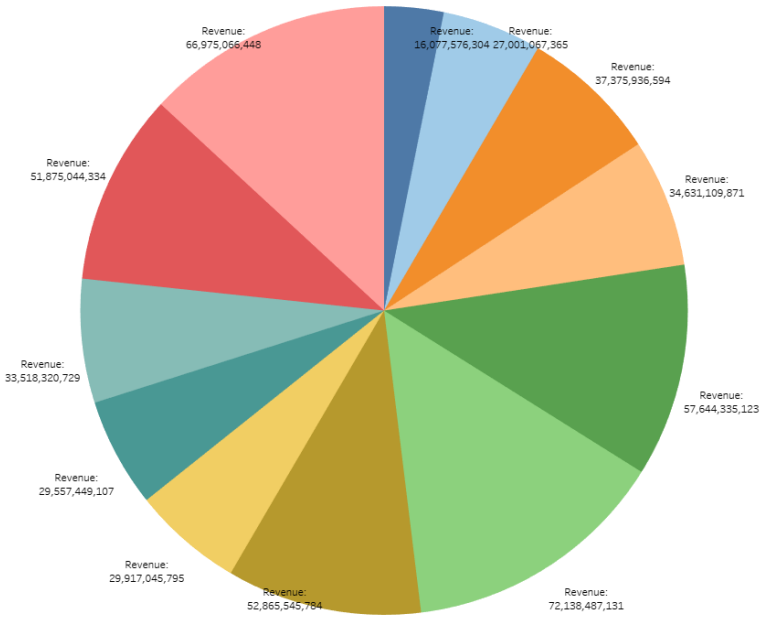


Sheet 1 | Sheet 2 | Sheet 3 | **Sheet 4**



Sheet 1 | Sheet 2 | Sheet 3 | Sheet 4 | Sheet 5 | **Sheet 6**

Months with maximum Revenue

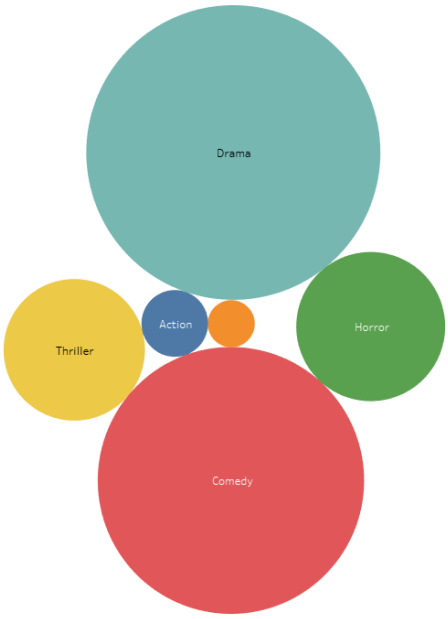


SUM(Revenue)
509,576,984,585

Month
- January
- February
- March
- April
- May
- June
- July
- August
- September
- October
- November
- December

Sheet 1 | Sheet 2 | Sheet 3 | Sheet 4 | Sheet 5 | **Sheet 7**

Genre vs Movie Title



Genre
- Action
- Adventure
- Comedy
- Drama
- Horror
- Thriller

Sheet 1 | Sheet 2 | Sheet 3 | Sheet 4 | Sheet 5 | Sheet 7 | **Sheet 9**