Case Study Data Management using Apache Spark

1. Data Governance creates outline consists of policies and procedures of the structure of the data, meanwhile Data Management executes the outline of policies and procedures made by data governance in order to make decision using the data.
2. A. Data Steward
   ⇨ Is a staff member with the responsibility to oversight for a subset of the data.
   Responsibilities:
   - Implement data standards
   - Monitor data quality
   - Handle inquiries regarding the data
   - Protecting data assets

   B. Data Custodian

   ⇨ Is a system administrator or technical professional in other field who is responsible for a few aspects of management and operation of any of the systems that function as sources of institutional data.
   Responsibilities:
   - Provide support for the data by providing secure infrastructure
   - Monitoring and implementing data access policies.
   - Maintaining system's availability and response time.
   - Help to set data governance priorities

   C. Data Admin

   ⇨ Is an administrator who oversees the implementation of the entire data governance program and also serves to resolve all data-related conflicts
   Responsibiltiies:
   - Ensuring the usefulness of data
   - Enable data analytics for decision-making
   - Ensuring data's credibility, relevance, and update

   D. Data User

   ⇨ Is anyone within the organization who uses the data.
   Responsibilities:
   - Ensuring security measures needed to protect sensitive data
   - Informing the data governance team regarding quality or credibility issues of datasets
   - Attend training and educational sessions on data governance, access, and use

   E. Data Executive

   ⇨ Is the committee that leads the data governance program.
   Responsibilities:
   - Approving enterprise strategic plan and policies
   - Communicating with lines of business the expectations and requirements for data governance.

3. The figure showed the Apache Spark Ecosystem, where the Spark Core Engine is the underlying general execution engine for the Spark platform that all other functionality is

built on top of. The Apache Spark provides high-level APIs in Java, Scala, Python, and R. Apache Spark also provides extensions such as Spark Streaming and Spark SQL for data processing and supplementary packages such as Mlib and GraphX for machine learning modules and graph analytics. Apache Spark is can access data from multiple sources, it creates distributed datasets from the file system that is used for data storage. The popular file systems/sources used by Apache Spark include HBase, Cassandra, HDFS, elasticsearch, etc.

Code

1. Create a directory in verulam-blue directory named YourStudentID-YourName. Display lists files in verulam-blue directory.

```
verulam-blue ~ hdfs dfs -mkdir file:///home/verulam-blue/working/2501975684-ArvioAnandi
verulam-blue ~ hdfs dfs -ls file:///home/verulam-blue/working/
Found 20 items
drwxrwxr-x   - verulam-blue verulam-blue       4096 2022-11-24 08:33 file:///home/verulam-blue/working/2501975684-ArvioAnandi
-rw-rw-r--   1 verulam-blue verulam-blue     550744 2022-10-31 11:38 file:///home/verulam-blue/working/assignment1-code
-rw-rw-r--   1 verulam-blue verulam-blue      44120 2022-10-31 12:41 file:///home/verulam-blue/working/assignment1-code2
drwx------   - verulam-blue verulam-blue       4096 2021-09-19 04:24 file:///home/verulam-blue/working/beginning-apache-spark-3-master
-rw-rw-r--   1 verulam-blue verulam-blue        197 2021-05-29 10:09 file:///home/verulam-blue/working/localFile1.txt
-rw-rw-r--   1 verulam-blue verulam-blue         55 2021-05-29 10:09 file:///home/verulam-blue/working/localFile2.csv
-rw-rw-r--   1 verulam-blue verulam-blue       1755 2021-05-29 10:09 file:///home/verulam-blue/working/localFile3.gz.parquet
drwxrwxr-x   - verulam-blue verulam-blue       4096 2022-10-30 09:27 file:///home/verulam-blue/working/movies
drwxrwxr-x   - verulam-blue verulam-blue       4096 2021-05-24 18:39 file:///home/verulam-blue/working/section_01
drwxrwxr-x   - verulam-blue verulam-blue       4096 2021-05-24 18:39 file:///home/verulam-blue/working/section_02
drwxrwxr-x   - verulam-blue verulam-blue       4096 2021-06-01 18:25 file:///home/verulam-blue/working/section_03
drwxrwxr-x   - verulam-blue verulam-blue       4096 2021-05-24 18:39 file:///home/verulam-blue/working/section_04
drwxrwxr-x   - verulam-blue verulam-blue       4096 2021-05-24 18:39 file:///home/verulam-blue/working/section_05
drwxrwxr-x   - verulam-blue verulam-blue       4096 2021-05-24 18:39 file:///home/verulam-blue/working/section_06
drwxrwxr-x   - verulam-blue verulam-blue       4096 2021-05-24 18:40 file:///home/verulam-blue/working/section_07
drwxrwxr-x   - verulam-blue verulam-blue       4096 2021-05-24 18:40 file:///home/verulam-blue/working/section_08
drwxrwxr-x   - verulam-blue verulam-blue       4096 2021-06-25 11:47 file:///home/verulam-blue/working/section_09
drwxrwxr-x   - verulam-blue verulam-blue       4096 2021-06-25 11:48 file:///home/verulam-blue/working/section_10
drwxrwxr-x   - verulam-blue verulam-blue       4096 2021-06-25 11:48 file:///home/verulam-blue/working/section_11
drwxrwxr-x   - verulam-blue verulam-blue       4096 2021-06-25 11:48 file:///home/verulam-blue/working/section_12
```

2. Move folder data into your working directory and display lists files in it

```
verulam-blue ~ hdfs dfs -ls file:///home/verulam-blue/working/2501975684-ArvioAnandi
Found 4 items
-rw-rw-r--   1 verulam-blue verulam-blue       8342 2022-11-24 08:37 file:///home/verulam-blue/working/2501975684-ArvioAnandi/README.txt
-rw-rw-r--   1 verulam-blue verulam-blue     494431 2022-11-24 08:37 file:///home/verulam-blue/working/2501975684-ArvioAnandi/movies.csv
-rw-rw-r--   1 verulam-blue verulam-blue    2483723 2022-11-24 08:37 file:///home/verulam-blue/working/2501975684-ArvioAnandi/ratings.csv
-rw-rw-r--   1 verulam-blue verulam-blue     118660 2022-11-24 08:37 file:///home/verulam-blue/working/2501975684-ArvioAnandi/tags.csv
```

3. Show the first 10 contents of movies.csv file

```
verulam-blue ~ 1 hdfs dfs -cat file:///home/verulam-blue/working/2501975684-ArvioAnandi/movies.csv|head -n 11
movieId,title,genres
1,Toy Story (1995),Adventure|Animation|Children|Comedy|Fantasy
2,Jumanji (1995),Adventure|Children|Fantasy
3,Grumpier Old Men (1995),Comedy|Romance
4,Waiting to Exhale (1995),Comedy|Drama|Romance
5,Father of the Bride Part II (1995),Comedy
6,Heat (1995),Action|Crime|Thriller
7,Sabrina (1995),Comedy|Romance
8,Tom and Huck (1995),Adventure|Children
9,Sudden Death (1995),Action
10,GoldenEye (1995),Action|Adventure|Thriller
```

4. Display number of directory, number of files and file sizes of your working directory

```
verulam-blue ~ 2 hdfs dfs -ls file:///home/verulam-blue/working/2501975684-ArvioAnandi
Found 4 items
-rw-rw-r--   1 verulam-blue verulam-blue       8342 2022-11-24 08:37 file:///home/verulam-blue/working/2501975684-ArvioAnandi/README.txt
-rw-rw-r--   1 verulam-blue verulam-blue     494431 2022-11-24 08:37 file:///home/verulam-blue/working/2501975684-ArvioAnandi/movies.csv
-rw-rw-r--   1 verulam-blue verulam-blue    2483723 2022-11-24 08:37 file:///home/verulam-blue/working/2501975684-ArvioAnandi/ratings.csv
-rw-rw-r--   1 verulam-blue verulam-blue     118660 2022-11-24 08:37 file:///home/verulam-blue/working/2501975684-ArvioAnandi/tags.csv
```

5. Create 3 dataframes from CSV filesand get a glimpse of the data.

```
scala> val movies = spark.read.option("inferSchema","true").option("sep",",").option("header","true").csv("file:/home/verulam-blue/working/2501975684-ArvioAnandi/movies.csv").toDF("movieId","title","genres")
movies: org.apache.spark.sql.DataFrame = [movieId: int, title: string ... 1 more field]

scala> movies.show(10)
+-------+--------------------+--------------------+
|movieId|               title|              genres|
+-------+--------------------+--------------------+
|      1|    Toy Story (1995)|Adventure|Animati...|
|      2|      Jumanji (1995)|Adventure|Childre...|
|      3|Grumpier Old Men ...|      Comedy|Romance|
|      4|Waiting to Exhale...|Comedy|Drama|Romance|
|      5|Father of the Bri...|              Comedy|
|      6|         Heat (1995)|Action|Crime|Thri...|
|      7|      Sabrina (1995)|      Comedy|Romance|
|      8| Tom and Huck (1995)|  Adventure|Children|
|      9| Sudden Death (1995)|              Action|
|     10|    GoldenEye (1995)|Action|Adventure|...|
+-------+--------------------+--------------------+
only showing top 10 rows
```

```
scala> val ratings = spark.read.option("inferSchema","true").option("sep",",").option("header","true").csv("file:/home/verulam-blue/working/2501975684-ArvioAnandi/ratings.csv").toDF("userId","movieId","rating","timestamp")
ratings: org.apache.spark.sql.DataFrame = [userId: int, movieId: int ... 2 more fields]

scala> ratings.show(10)
+------+-------+------+---------+
|userId|movieId|rating|timestamp|
+------+-------+------+---------+
|     1|      1|   4.0|964982703|
|     1|      3|   4.0|964981247|
|     1|      6|   4.0|964982224|
|     1|     47|   5.0|964983815|
|     1|     50|   5.0|964982931|
|     1|     70|   3.0|964982400|
|     1|    101|   5.0|964980868|
|     1|    110|   4.0|964982176|
|     1|    151|   5.0|964984041|
|     1|    157|   5.0|964984100|
+------+-------+------+---------+
only showing top 10 rows
```

```
scala> val tags = spark.read.option("inferSchema","true").option("sep",",").option("header","true").csv("file:/home/verulam-blue/working/2501975684-ArvioAnandi/tags.csv").toDF("userId","movieId","tag","timestamp")
tags: org.apache.spark.sql.DataFrame = [userId: int, movieId: int ... 2 more fields]

scala> tags.show(10)
+------+-------+----------------+----------+
|userId|movieId|             tag| timestamp|
+------+-------+----------------+----------+
|     2|  60756|           funny|1445714994|
|     2|  60756|  Highly quotable|1445714996|
|     2|  60756|     will ferrell|1445714992|
|     2|  89774|     Boxing story|1445715207|
|     2|  89774|             MMA|1445715200|
|     2|  89774|        Tom Hardy|1445715205|
|     2| 106782|           drugs|1445715054|
|     2| 106782|Leonardo DiCaprio|1445715051|
|     2| 106782|  Martin Scorsese|1445715056|
|     7|  48516|     way too long|1169687325|
+------+-------+----------------+----------+
only showing top 10 rows
```

6. Rename the following columns using operation for structured Transformation.
   a. [movies dataframe] --> Rename movieId with mId

```
scala> val movies1 = movies.withColumnRenamed("movieId","mId")
movies1: org.apache.spark.sql.DataFrame = [mId: int, title: string ... 1 more field]

scala> movies1.show
+---+--------------------+--------------------+
|mId|               title|              genres|
+---+--------------------+--------------------+
|  1|    Toy Story (1995)|Adventure|Animati...|
|  2|      Jumanji (1995)|Adventure|Childre...|
|  3|Grumpier Old Men ...|      Comedy|Romance|
|  4|Waiting to Exhale...|Comedy|Drama|Romance|
|  5|Father of the Bri...|              Comedy|
|  6|         Heat (1995)|Action|Crime|Thri...|
|  7|      Sabrina (1995)|      Comedy|Romance|
|  8| Tom and Huck (1995)|  Adventure|Children|
|  9| Sudden Death (1995)|              Action|
| 10|    GoldenEye (1995)|Action|Adventure|...|
| 11|American Presiden...|Comedy|Drama|Romance|
| 12|Dracula: Dead and...|       Comedy|Horror|
| 13|        Balto (1995)|Adventure|Animati...|
| 14|        Nixon (1995)|               Drama|
| 15|Cutthroat Island ...|Action|Adventure|...|
| 16|       Casino (1995)|         Crime|Drama|
| 17|Sense and Sensibi...|       Drama|Romance|
| 18|   Four Rooms (1995)|              Comedy|
| 19|Ace Ventura: When...|              Comedy|
| 20|  Money Train (1995)|Action|Comedy|Cri...|
+---+--------------------+--------------------+
only showing top 20 rows
```

b. [rating dataframe] -->
Rename movieId with mrId & userId with ruId

```
scala> val ratings1 = ratings.withColumnRenamed("movieId","mrId").withColumnRenamed("userId","ruId")
ratings1: org.apache.spark.sql.DataFrame = [ruId: int, mrId: int ... 2 more fields]

scala> ratings1.show
+----+----+------+---------+
|ruId|mrId|rating|timestamp|
+----+----+------+---------+
|   1|   1|   4.0|964982703|
|   1|   3|   4.0|964981247|
|   1|   6|   4.0|964982224|
|   1|  47|   5.0|964983815|
|   1|  50|   5.0|964982931|
|   1|  70|   3.0|964982400|
|   1| 101|   5.0|964980868|
|   1| 110|   4.0|964982176|
|   1| 151|   5.0|964984041|
|   1| 157|   5.0|964984100|
|   1| 163|   5.0|964983650|
|   1| 216|   5.0|964981208|
|   1| 223|   3.0|964980985|
|   1| 231|   5.0|964981179|
|   1| 235|   4.0|964980908|
|   1| 260|   5.0|964981680|
|   1| 296|   3.0|964982967|
|   1| 316|   3.0|964982310|
|   1| 333|   5.0|964981179|
|   1| 349|   4.0|964982563|
+----+----+------+---------+
only showing top 20 rows
```

c. [tags dataframe] --> Rename userId with tuId

```
scala> val tags1 = tags.withColumnRenamed("userId","tuId")
tags1: org.apache.spark.sql.DataFrame = [tuId: int, movieId: int ... 2 more fields]

scala> tags1.show
+----+-------+----------------+----------+
|tuId|movieId|             tag| timestamp|
+----+-------+----------------+----------+
|   2|  60756|           funny|1445714994|
|   2|  60756| Highly quotable|1445714996|
|   2|  60756|    will ferrell|1445714992|
|   2|  89774|    Boxing story|1445715207|
|   2|  89774|             MMA|1445715200|
|   2|  89774|       Tom Hardy|1445715205|
|   2| 106782|           drugs|1445715054|
|   2| 106782|Leonardo DiCaprio|1445715051|
|   2| 106782| Martin Scorsese|1445715056|
|   7|  48516|    way too long|1169687325|
|  18|    431|       Al Pacino|1462138765|
|  18|    431|        gangster|1462138749|
|  18|    431|           mafia|1462138755|
|  18|   1221|       Al Pacino|1461699306|
|  18|   1221|           Mafia|1461699303|
|  18|   5995|       holocaust|1455735472|
|  18|   5995|      true story|1455735479|
|  18|  44665|    twist ending|1456948283|
|  18|  52604| Anthony Hopkins|1457650696|
|  18|  52604| courtroom drama|1457650711|
+----+-------+----------------+----------+
only showing top 20 rows
```

7. Register the data frames as 3 SQL tables.

```
scala> movies1.createOrReplaceTempView("movies1")

scala> ratings1.createOrReplaceTempView("ratings1")

scala> tags1.createOrReplaceTempView("tags1")
```

8. Combine all records from three tables into one, named df. You just need records whenever movieId values from movies table are matching with movieId from rating and tag tables as well as userId values from rating table are matching with those in tag table, using an SQL Query

```
scala> val df = spark.sql("select * from movies1 m join ratings1 r on m.mId = r.mrId join tags1 t on m.mId = t.movieId and r.ruId = t.tuId")
df: org.apache.spark.sql.DataFrame = [mId: int, title: string ... 9 more fields]

scala>

scala> df.show
+-----+--------------------+--------------------+----+------+------+----------+----+-------+----------------+----------+
|  mId|               title|              genres|ruId|  mrId|rating| timestamp|tuId|movieId|             tag| timestamp|
+-----+--------------------+--------------------+----+------+------+----------+----+-------+----------------+----------+
|60756|Step Brothers (2008)|              Comedy|   2| 60756|   5.0|1445714980|   2|  60756|    will ferrell|1445714992|
|60756|Step Brothers (2008)|              Comedy|   2| 60756|   5.0|1445714980|   2|  60756| Highly quotable|1445714996|
|60756|Step Brothers (2008)|              Comedy|   2| 60756|   5.0|1445714980|   2|  60756|           funny|1445714994|
|89774|      Warrior (2011)|               Drama|   2| 89774|   5.0|1445715189|   2|  89774|      Tom Hardy|1445715205|
|89774|      Warrior (2011)|               Drama|   2| 89774|   5.0|1445715189|   2|  89774|             MMA|1445715200|
|89774|      Warrior (2011)|               Drama|   2| 89774|   5.0|1445715189|   2|  89774|   Boxing story|1445715207|
|106782|Wolf of Wall Stre...|  Comedy|Crime|Drama|   2|106782|   5.0|1445714966|   2| 106782| Martin Scorsese|1445715056|
|106782|Wolf of Wall Stre...|  Comedy|Crime|Drama|   2|106782|   5.0|1445714966|   2| 106782|Leonardo DiCaprio|1445715051|
|106782|Wolf of Wall Stre...|  Comedy|Crime|Drama|   2|106782|   5.0|1445714966|   2| 106782|           drugs|1445715054|
| 48516|Departed, The (2006)|Crime|Drama|Thriller|   7| 48516|   1.0|1169687318|   7|  48516|    way too long|1169687325|
|  431|Carlito's Way (1993)|         Crime|Drama|  18|   431|   4.0|1462138790|  18|    431|           mafia|1462138755|
|  431|Carlito's Way (1993)|         Crime|Drama|  18|   431|   4.0|1462138790|  18|    431|        gangster|1462138749|
|  431|Carlito's Way (1993)|         Crime|Drama|  18|   431|   4.0|1462138790|  18|    431|       Al Pacino|1462138765|
| 1221|Godfather: Part I...|         Crime|Drama|  18|  1221|   5.0|1460242083|  18|   1221|           Mafia|1461699303|
| 1221|Godfather: Part I...|         Crime|Drama|  18|  1221|   5.0|1460242083|  18|   1221|       Al Pacino|1461699306|
| 5995| Pianist, The (2002)|           Drama|War|  18|  5995|   4.5|1455735416|  18|   5995|      true story|1455735479|
| 5995| Pianist, The (2002)|           Drama|War|  18|  5995|   4.5|1455735416|  18|   5995|       holocaust|1455735472|
|44665|Lucky Number Slev...| Crime|Drama|Mystery|  18| 44665|   4.5|1455049870|  18|  44665|    twist ending|1456948283|
|52604|     Fracture (2007)|Crime|Drama|Myste...|  18| 52604|   4.5|1457650649|  18|  52604|    twist ending|1457650682|
|52604|     Fracture (2007)|Crime|Drama|Myste...|  18| 52604|   4.5|1457650649|  18|  52604| courtroom drama|1457650711|
+-----+--------------------+--------------------+----+------+------+----------+----+-------+----------------+----------+
only showing top 20 rows
```

9. Remove the specified columns from dataframes mrId, movieId(from tags dataframe), timestamp, and tuId. Then, register the dataframe as an SQL table named df1

```
scala> val df1 = df.drop("mrId","movieId","timestamp","tuId")
df1: org.apache.spark.sql.DataFrame = [mId: int, title: string ... 4 more fields]

scala>

scala> df1.createOrReplaceTempView("df1")

scala>

scala> df1.show
+------+--------------------+--------------------+----+------+-----------------+
|   mId|               title|              genres|ruId|rating|              tag|
+------+--------------------+--------------------+----+------+-----------------+
| 60756|Step Brothers (2008)|              Comedy|   2|   5.0|      will ferrell|
| 60756|Step Brothers (2008)|              Comedy|   2|   5.0|   Highly quotable|
| 60756|Step Brothers (2008)|              Comedy|   2|   5.0|            funny|
| 89774|      Warrior (2011)|               Drama|   2|   5.0|         Tom Hardy|
| 89774|      Warrior (2011)|               Drama|   2|   5.0|              MMA|
| 89774|      Warrior (2011)|               Drama|   2|   5.0|      Boxing story|
|106782|Wolf of Wall Stre...|  Comedy|Crime|Drama|   2|   5.0|   Martin Scorsese|
|106782|Wolf of Wall Stre...|  Comedy|Crime|Drama|   2|   5.0|Leonardo DiCaprio|
|106782|Wolf of Wall Stre...|  Comedy|Crime|Drama|   2|   5.0|            drugs|
| 48516|Departed, The (2006)|Crime|Drama|Thriller|   7|   1.0|      way too long|
|   431|Carlito's Way (1993)|         Crime|Drama|  18|   4.0|            mafia|
|   431|Carlito's Way (1993)|         Crime|Drama|  18|   4.0|         gangster|
|   431|Carlito's Way (1993)|         Crime|Drama|  18|   4.0|         Al Pacino|
|  1221|Godfather: Part I...|         Crime|Drama|  18|   5.0|            Mafia|
|  1221|Godfather: Part I...|         Crime|Drama|  18|   5.0|         Al Pacino|
|  5995| Pianist, The (2002)|           Drama|War|  18|   4.5|        true story|
|  5995| Pianist, The (2002)|           Drama|War|  18|   4.5|         holocaust|
| 44665|Lucky Number Slev...| Crime|Drama|Mystery|  18|   4.5|      twist ending|
| 52604|     Fracture (2007)|Crime|Drama|Myste...|  18|   4.5|      twist ending|
| 52604|     Fracture (2007)|Crime|Drama|Myste...|  18|   4.5|  courtroom drama|
+------+--------------------+--------------------+----+------+-----------------+
only showing top 20 rows
```

10. Select the 10th most rated 5 star movies in a decresing count of times the movie got 5 star rating, using an SQL query

```
scala> spark.sql("select title,count(title) as count from df1 where rating = 5 group by title").orderBy('count.desc).show(10)
+--------------------+-----+
|               title|count|
+--------------------+-----+
| Pulp Fiction (1994)|  176|
|   Fight Club (1999)|   49|
|2001: A Space Ody...|   39|
|L?on: The Profess...|   32|
|Big Lebowski, The...|   31|
|Eternal Sunshine ...|   24|
|   Eraserhead (1977)|   16|
| Mary and Max (2009)|   13|
|    Inception (2010)|   13|
|Talented Mr. Ripl...|   12|
+--------------------+-----+
only showing top 10 rows
```

11. Which user gave most '5' ratings? Return the userID and number of ratings, using an SQL query

```
scala> spark.sql("select ruId,count(rating) as count from df1 where rating = 5 group by ruId").orderBy('count.desc).show()
+----+-----+
|ruId|count|
+----+-----+
| 599|  323|
| 567|   96|
| 477|   92|
| 474|   81|
| 537|   55|
|  62|   51|
| 424|   45|
| 125|   41|
| 357|   12|
|   2|    9|
| 318|    6|
| 419|    5|
| 462|    5|
| 184|    5|
| 305|    4|
| 166|    4|
| 573|    4|
| 319|    3|
| 112|    3|
| 177|    3|
+----+-----+
only showing top 20 rows
```

12. check the users that have rated the most and the least number of movies, using an SQL Query

```
scala> // Showing the userId with the most number of ratings

scala> spark.sql("select ruId,count(ruId) as count from df1 group by ruId").orderBy('count.desc).show()
+----+-----+
|ruId|count|
+----+-----+
| 474| 1414|
| 567|  432|
|  62|  370|
| 599|  323|
| 477|  267|
| 424|  223|
| 537|   90|
| 125|   48|
| 357|   45|
| 318|   36|
| 184|   35|
| 193|   17|
|  18|   16|
| 119|   12|
| 573|   12|
|   2|    9|
| 336|    9|
| 305|    8|
| 327|    7|
| 166|    6|
scala> // Showing the userId with the least number of ratings

scala> spark.sql("select ruId,count(ruId) as count from df1 group by ruId").orderBy('count.asc).show()
+----+-----+
|ruId|count|
+----+-----+
| 300|    1|
| 161|    1|
| 167|    1|
| 274|    1|
|  21|    1|
|   7|    1|
|  63|    2|
|  76|    2|
| 106|    2|
| 520|    2|
| 256|    2|
| 138|    2|
| 205|    3|
| 319|    3|
|  49|    3|
| 112|    3|
| 439|    3|
| 435|    3|
| 610|    3|
| 513|    3|
+----+-----+
only showing top 20 rows
```

13. Get a distinct list of all movie titles that are five-star rated in the Crime/Drama genre, using an SQL Query

```
scala> spark.sql("select distinct title from df1 where (genres like '%Crime%' or genres like '%Drama%') and rating = 5 ").orderBy('title.desc).show()
+--------------------+
|               title|
+--------------------+
|Zero Dark Thirty ...|
|Wolf of Wall Stre...|
|  Whale Rider (2002)|
|West Side Story (...|
|       Warrior (2011)|
|Vicky Cristina Ba...|
|Usual Suspects, T...|
|  Unbreakable (2000)|
|Truly, Madly, Dee...|
|There Will Be Blo...|
|The Butterfly Eff...|
|Talented Mr. Ripl...|
|Sixth Sense, The ...|
|Silence of the La...|
|Shutter Island (2...|
|Shawshank Redempt...|
|  Shadowlands (1993)|
|Sense and Sensibi...|
|Schindler's List ...|
|Say Anything... (...|
+--------------------+
only showing top 20 rows
```

14. Get the movie titles tagged with 'Disney' with minimum three-star rated in the Adventure genre, the number of ratings for each movie, ordered by the highest rating, using an SQL Query

```
scala> spark.sql("select title, rating from df1 where tag like '%Disney%' and rating >= 3 and genres like '%Adventure%'").orderBy('rating.desc).show()
+--------------------+------+
|               title|rating|
+--------------------+------+
|Lion King, The (1...|   5.0|
|Lion King, The (1...|   5.0|
|Lion King, The (1...|   4.5|
|      Aladdin (1992)|   4.0|
|101 Dalmatians (O...|   4.0|
|Honey, I Shrunk t...|   4.0|
| Finding Nemo (2003)|   4.0|
|Incredibles, The ...|   4.0|
|  Toy Story 2 (1999)|   3.0|
+--------------------+------+
```

15. Get a sorted count of the number of people who gave each Crime/Thriller movie a four-star or more rating, using an SQL Query

```
scala> spark.sql("select title,count(ruId) as count from df1 where (genres like '%Crime%' or genres like '%Thriller%') and rating >= 4 group by title").orderBy('count
.desc).show()
+--------------------+-----+
|               title|count|
+--------------------+-----+
| Pulp Fiction (1994)|  181|
|   Fight Club (1999)|   54|
|L?on: The Profess...|   34|
|Big Lebowski, The...|   31|
|Suicide Squad (2016)|   19|
|    Inception (2010)|   16|
| Donnie Darko (2001)|   15|
|Talented Mr. Ripl...|   12|
|Terminator Salvat...|   12|
|Burn After Readin...|   11|
|      Memento (2000)|   11|
|Battle Royale (Ba...|   10|
|      Psycho (1960)|    9|
|  I Am Legend (2007)|    9|
|  Margin Call (2011)|    9|
|Departed, The (2006)|    9|
|    John Wick (2014)|    9|
|Twelve Monkeys (a...|    9|
|Catch Me If You C...|    8|
|    Prisoners (2013)|    8|
+--------------------+-----+
only showing top 20 rows
```