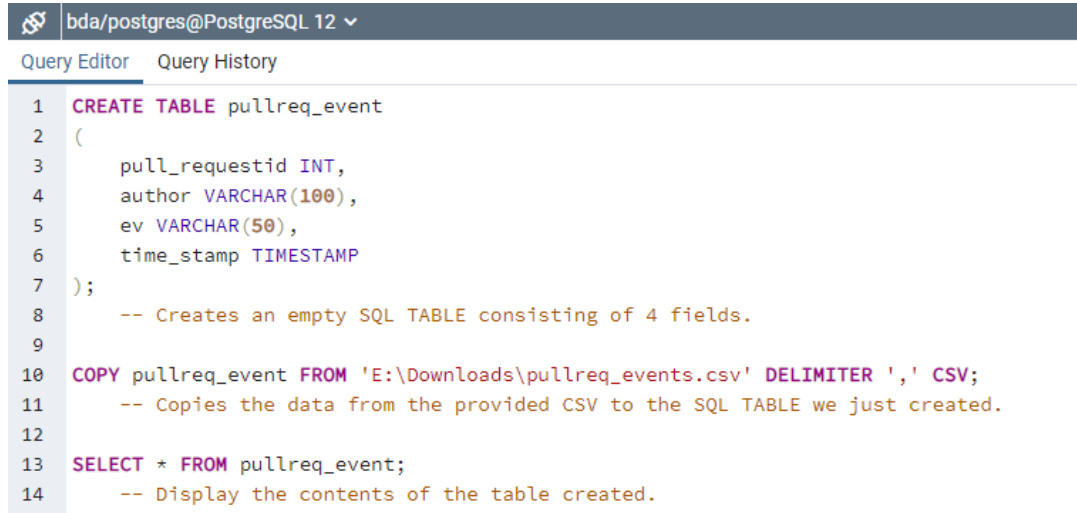


BDA ASSIGNMENT 2

1.Import all the data into postgres used in the first assignment. Now use Apache Spark to answer all the queries with Postgres as the backend storage engine. Write modules in spark to answer all the queries in assignment 1.

METHODOLOGY:

Consider the following screenshot explaining how the given CSV was imported to an SQL TABLE in a POSTGRESQL RELATIONAL DATABASE.



The screenshot shows a PostgreSQL Query Editor window with the following SQL commands:

```
1 CREATE TABLE pullreq_event
2 (
3     pull_requestid INT,
4     author VARCHAR(100),
5     ev VARCHAR(50),
6     time_stamp TIMESTAMP
7 );
8 -- Creates an empty SQL TABLE consisting of 4 fields.
9
10 COPY pullreq_event FROM 'E:\Downloads\pullreq_events.csv' DELIMITER ',' CSV;
11 -- Copies the data from the provided CSV to the SQL TABLE we just created.
12
13 SELECT * FROM pullreq_event;
14 -- Display the contents of the table created.
```

Below the query editor, the 'Data Output' tab is selected, showing the results of the SELECT query. The table has four columns: pull_requestid (integer), author (character varying (100)), ev (character varying (50)), and time_stamp (timestamp without time zone). The data is as follows:

	pull_requestid integer	author character varying (100)	ev character varying (50)	time_stamp timestamp without time zone
1	4	datanoise	opened	2010-09-02 03:34:17
2	5	marsuboss	opened	2010-09-02 07:14:12
3	6	m3talsmith	opened	2010-09-06 16:07:08
4	7	sferik	opened	2010-09-08 19:09:56
5	8	sferik	opened	2010-09-09 04:33:23
6	8	dtrasbo	discussed	2010-09-09 04:44:25
7	8	brianmario	discussed	2010-09-09 04:53:01
8	8	sferik	discussed	2010-09-09 14:22:03
9	9	rsim	opened	2010-09-09 19:18:48
10	10	simonjefford	opened	2010-09-09 19:36:42
11	11	bcardarella	opened	2010-09-09 20:48:38
12	11	wycats	discussed	2010-09-09 23:23:11
13	10	wycats	discussed	2010-09-09 23:26:43
14	9	wycats	discussed	2010-09-09 23:28:24
15	12	marklazz	opened	2010-09-10 00:39:32

- In order to perform operations with the TIMESTAMP related data, we had to perform various operations. Some built-in functions were found to be handy such as EXTRACT() and CAST(). The following resources were referred for the same.
 - <https://www.postgresqltutorial.com/postgresql-extract/>
 - <https://www.postgresql.org/docs/9.2/sql-createcast.html>
 - <https://www.postgresql.org/docs/9.1/functions-datetime.html>

BDA ASSIGNMENT 2

- Spark connection to the postgresql initialized.

```
spark = SparkSession.builder \  
    .appName("PySpark PostgreSQL") \  
    .config("spark.jars",  
"C:/tools/spark-3.0.2-bin-hadoop2.7/jars/postgresql-42.2.19.jar") \  
    .config("spark.local.dir", "C:/tmp") \  
    .config("spark.executor.instances", "2") \  
    .getOrCreate()
```

- The queries were fetched from the backend using as follows:-

```
spark.read \  
    .format("jdbc") \  
    .option("url", "jdbc:postgresql://localhost:5432/bda") \  
    .option("dbtable", "(" + queries[i] + ") AS results") \  
    .option("user", user) \  
    .option("password", password) \  
    .option("driver", "org.postgresql.Driver") \  
    .load() \  
    .show()
```

- 'queries[i]' in the above code contains the SQL queries created in Assignment 1.

- In case of frontend, the table was first imported into a dataframe/RDD:-

```
df = spark.read \  
    .format("jdbc") \  
    .option("url", "jdbc:postgresql://localhost:5432/bda") \  
    .option("dbtable", "pullreq_event") \  
    .option("user", user) \  
    .option("password", password) \  
    .option("driver", "org.postgresql.Driver") \  
    .load()  
df.registerTempTable('pullreq_event')  
df.printSchema()
```

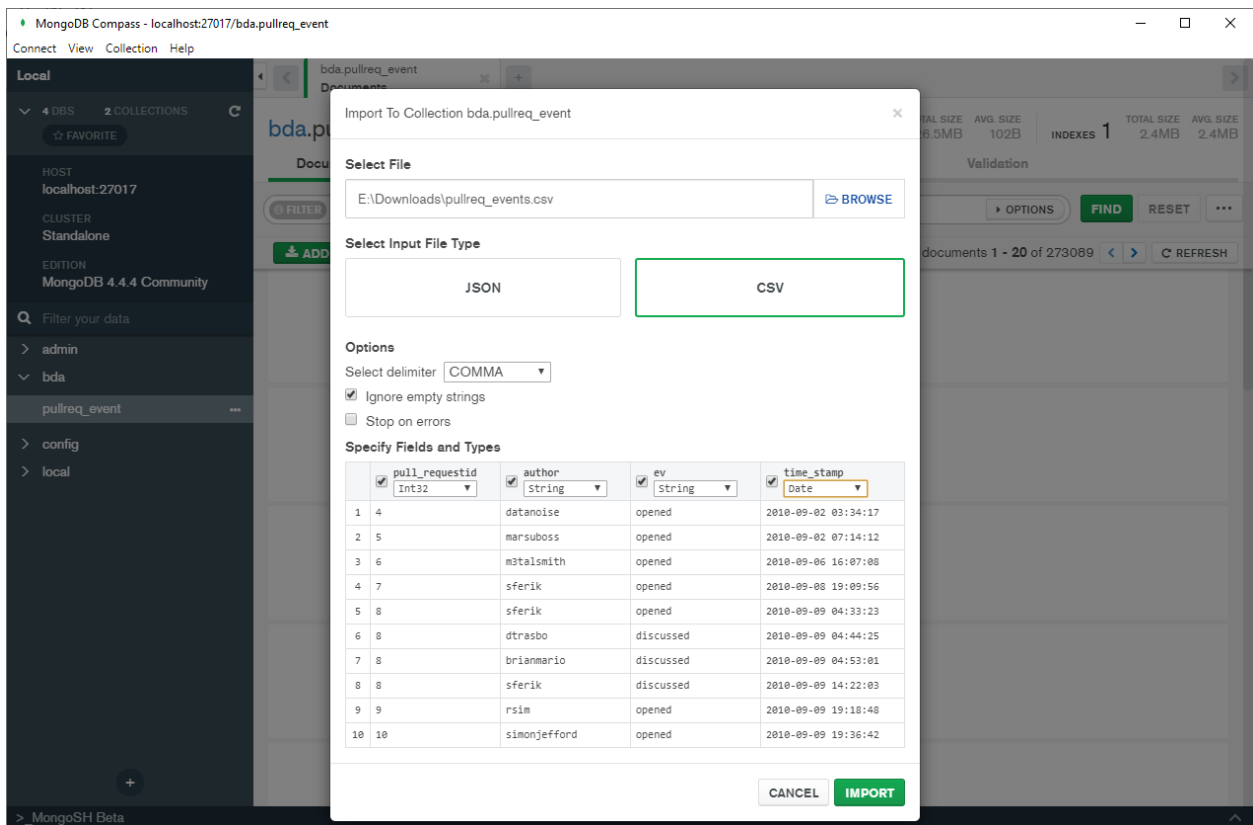
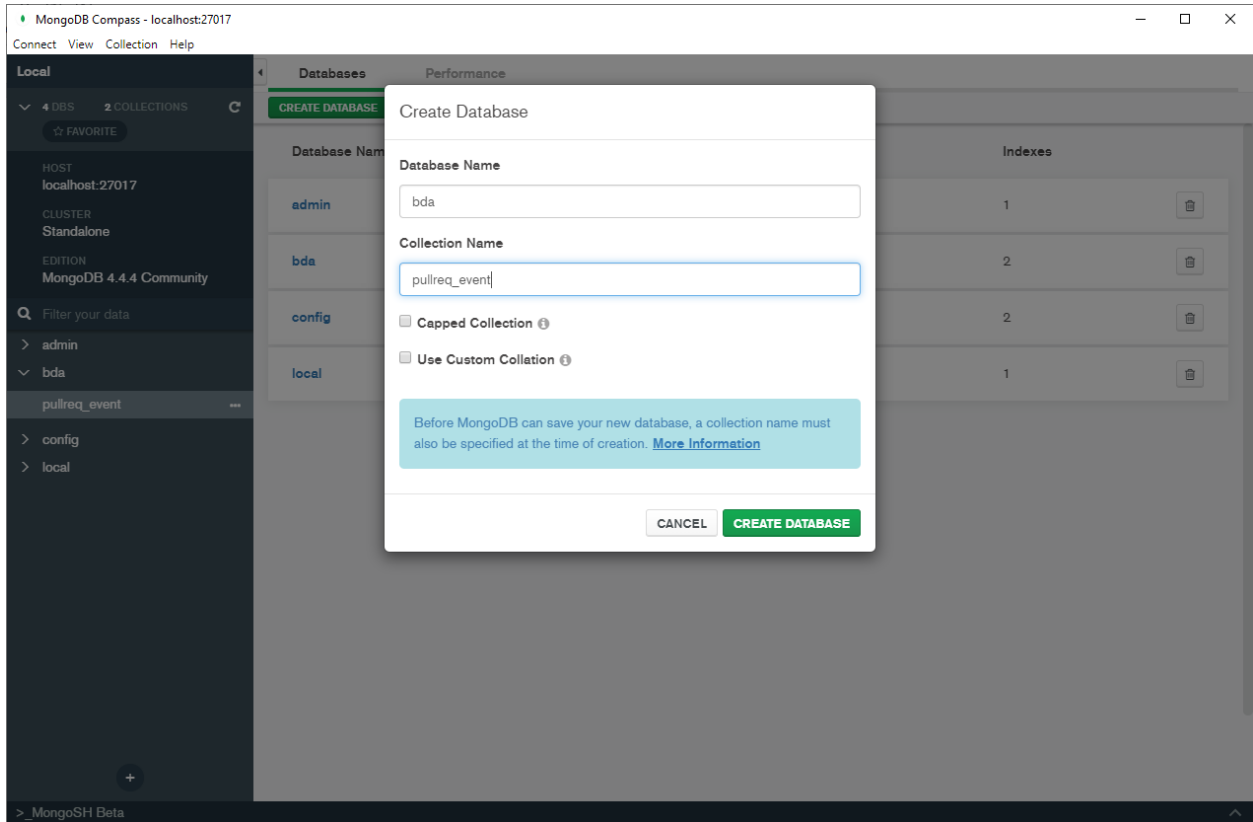
- The queries were run on this RDD/dataframe using:-

```
spark.sql(queries[i]).show()
```

- 'queries[i]' in the above code contains the SQL queries created in Assignment 1.

2. Use MongoDB as a storage engine. Import the data to MongoDB and connect Spark to mongoDB to answer queries in assignment 1.

BDA ASSIGNMENT 2



2018033, 2018089

BDA ASSIGNMENT 2

- Spark connection to the mongodb initialized.

```
spark = SparkSession\
    .builder\
    .appName("PySpark MongoDB") \
    .config("spark.mongodb.input.uri",
"mongodb://127.0.0.1:27017/bda.pullreq_event") \
    .config("spark.mongodb.output.uri",
"mongodb://127.0.0.1:27017/bda.pullreq_event") \
    .config('spark.jars.packages','org.mongodb.spark:mongo-spark-connector_2.1
2-3.0.1') \
    .config("spark.local.dir", "C:/tmp") \
    .config("spark.executor.instances", "2") \
    .getOrCreate()
```

- For running the queries in the backend engine, the following code was used and aggregation pipeline syntax queries were created as shown further below.

```
spark.read.format("mongo").option("pipeline", queries[i]).load().show()
```

- 'queries[i]' here represent the aggregate pipeline queries passed as shown below.

```
q1_a = [
    { '$match':
      { 'ev': 'opened' }
    },
    { '$group':
      {
        '_id': { '$dateToString': { 'format': '%Y-%m-%d', 'date': '$time_stamp', 'timezone': '+05:30' } },
        'count': { '$sum': 1 }
      }
    },
    { '$sort':
      { '_id': 1 }
    }
]
```

BDA ASSIGNMENT 2

```
q1_b = [
  { '$match':
    { 'ev': 'discussed' }
  },
  { '$group':
    {
      '_id': { '$dateToString': { 'format': '%V-%m-%d', 'date': '$time_stamp', 'timezone': '+05:30' } },
      'pullreqs': { '$addToSet': '$pull_requestid' }
    }
  },
  { '$unwind':
    "$pullreqs"
  },
  { '$group':
    {
      '_id': '$_id',
      'count': { '$sum': 1 }
    }
  },
  { '$sort':
    { '_id': 1 }
  }
]
```

```
q2 = [
  { '$match':
    { 'ev': 'discussed' }
  },
  { '$group':
    {
      '_id': {
        'Month': {
          '$dateToString': { 'format': '%m', 'date': '$time_stamp', 'timezone': '+0530' }
        },
        'author': '$author'
      },
      'numPRs': { '$sum': 1 }
    }
  },
  { '$sort':
    { 'numPRs': -1 }
  },
  { '$group':
    {
      '_id': '$_id.Month',
      'MaxNumPRs': { '$first': '$numPRs' },
      'author': { '$first': '$_id.author' }
    }
  },
  { '$sort':
    { '_id': 1 }
  }
]
```

BDA ASSIGNMENT 2

```
q3 = [
  { '$match':
    { 'ev': 'discussed' }
  },
  { '$project':
    {
      'DayOfWeek': { '$dayOfWeek':
        {
          'date': '$time_stamp',
          'timezone': '+0530'
        }
      },
      'time_stamp': { '$add': [ '$time_stamp', 19800000 ] },
      'ev': '$ev',
      'pull_requestid': '$pull_requestid',
      'author': '$author'
    }
  },
  { '$project':
    {
      'sundayOfWeek': { '$subtract': [ '$time_stamp', { '$multiply': [ { '$subtract': [ '$DayOfWeek', 1 ] }, 86400000 ] } ] },
      'ev': '$ev',
      'pull_requestid': '$pull_requestid',
      'author': '$author'
    }
  },
  { '$group':
    {
      '_id': {
        'sundayOfWeek': { '$dateToString': { 'format': '%Y-%m-%d', 'date': '$sundayOfWeek' } },
        'author': '$author'
      },
      'numPRs': { '$sum': 1 },
    }
  },
  { '$sort':
    { 'numPRs': -1 }
  },
  { '$group':
    {
      '_id': '$_id.sundayOfWeek',
      'MaxNumPRs': { '$first': '$numPRs' },
      'author': { '$first': '$_id.author' }
    }
  },
  { '$sort':
    { '_id': 1 }
  }
]
```

BDA ASSIGNMENT 2

```
q4 = [
  { '$project':
    {
      'DayOfWeek': { '$dayOfWeek':
        {
          'date': '$time_stamp',
          'timezone': '+0530'
        }
      },
      'time_stamp': { '$add': ['$time_stamp', 19800000] },
      'ev': '$ev',
      'pull_requestid': '$pull_requestid',
      'author': '$author'
    }
  },
  { '$project':
    {
      'sundayOfWeek': { '$subtract': ['$time_stamp', { '$multiply': [ { '$subtract': ['$DayOfWeek', 1] }, 86400000 ] } ] },
      'ev': '$ev',
      'pull_requestid': '$pull_requestid',
      'author': '$author'
    }
  },
  { '$match':
    {
      {
        'ev': 'opened'
      }
    }
  },
  { '$group':
    {
      {
        '_id': { '$dateToString': { 'format': '%Y-%m-%d', 'date': '$sundayOfWeek' } },
        'count': { '$sum': 1 }
      }
    }
  },
  { '$sort':
    {
      {
        '_id': 1
      }
    }
  }
]
```

BDA ASSIGNMENT 2

```
q5 = [
  { '$project':
    {
      'year': { '$year': '$time_stamp' },
      'time_stamp': '$time_stamp',
      'ev': '$ev',
      'pull_requestid': '$pull_requestid',
      'author': '$author'
    }
  },
  { '$match':
    { 'ev':'merged', 'year':2010 }
  },
  { '$group':
    {
      '_id': {
        '$dateToString': { 'format': '%m', 'date': '$time_stamp', 'timezone': '+0530' }
      },
      'numPRs': { '$sum': 1 },
    }
  },
  { '$sort':
    { '_id': 1 }
  }
]
```

```
q6 = [
  { '$project':
    {
      'time_stamp': '$time_stamp',
      'ev': '$ev',
      'pull_requestid': '$pull_requestid',
      'author': '$author'
    }
  },
  { '$group':
    {
      '_id': { '$dateToString': { 'format': '%Y-%m-%d', 'date': '$time_stamp', 'timezone': '+05:30' } },
      'count': { '$sum':1 }
    }
  },
  { '$sort':
    {
      '_id': 1
    }
  },
]
```


BDA ASSIGNMENT 2

```
q7 = [
  { '$project':
    {
      'year': { '$year': '$time_stamp' },
      'time_stamp': '$time_stamp',
      'ev': '$ev',
      'pull_requestid': '$pull_requestid',
      'author': '$author'
    }
  },
  { '$match':
    { 'ev': 'opened', 'year': 2011 }
  },
  { '$group':
    {
      '_id': '$author',
      'count': { '$sum': 1 }
    }
  },
  { '$sort':
    [
      { 'count': -1 }
    ]
  },
  { '$limit': 1 }
]
```

- For frontend queries, same SQL queries from assignment 1 were passed to 'queries[i]' in the following.

```
df = spark.read.format('com.mongodb.spark.sql.DefaultSource').load()
df.registerTempTable('pullreq_event')

spark.sql(queries[i]).show()
```

3. Now use HDFS to store the data and use apache spark to read from HDFS and perform the queries needed to answer the questions in assignment 1.

BDA ASSIGNMENT 2

```
cmd (running as DUSHYANT-PANCHA\test)

E:\Downloads>start-dfs

E:\Downloads>hdfs dfs -ls /

E:\Downloads>hdfs dfs -put pullreq_events.csv /

E:\Downloads>hdfs dfs -ls /
Found 1 items
-rw-r--r--  1 test supergroup  12349372 2021-03-11 16:19 /pullreq_events.csv

E:\Downloads>

Apache Hadoop Distribution - hadoop namenode

21/03/11 16:18:50 INFO namenode.NameNode: NameNode RPC up at: 0.0.0.0/0.0.0.0:19000
21/03/11 16:18:50 INFO namenode.FSNamesystem: Starting services required for active state
21/03/11 16:18:50 INFO blockmanagement.CacheReplicationMonitor: Starting CacheReplicationMonitor with interval 30000 milliseconds
21/03/11 16:19:00 INFO hdfs.StateChange: BLOCK* registerDatanode: from DatanodeRegistration(192.168.56.1:50010, datanodeUuid=74321ec9-9e43-4495-b319-65d42b4dbaff, infoPort=50075, infoSecurePort=0, ipcPort=50020, storageInfo=lv=-56;cid=CID-44e291b7-d6ce-46b3-b748-3ff1ea846f82;nsid=20937766;c=0) storage 74321ec9-9e43-4495-b319-65d42b4dbaff
21/03/11 16:19:00 INFO blockmanagement.DatanodeDescriptor: Number of failed storage changes from 0 to 0
21/03/11 16:19:00 INFO net.NetworkTopology: Adding a new node: /default-rack/192.168.56.1:50010
21/03/11 16:19:00 INFO blockmanagement.DatanodeDescriptor: Number of failed storage changes from 0 to 0
21/03/11 16:19:00 INFO blockmanagement.DatanodeDescriptor: Adding new storage ID DS-92c06f70-ab0a-45bc-b120-6296e90b1973 for DN 192.168.56.1:50010
21/03/11 16:19:02 INFO BlockStateChange: BLOCK* processReport 0x9a431ec79f74: from storage DS-92c06f70-ab0a-45bc-b120-6296e90b1973 node DatanodeRegistration(192.168.56.1:50010, datanodeUuid=74321ec9-9e43-4495-b319-65d42b4dbaff, infoPort=50075, infoSecurePort=0, ipcPort=50020, storageInfo=lv=-56;cid=CID-44e291b7-d6ce-46b3-b748-3ff1ea846f82;nsid=20937766;c=0), blocks: 0, hasStaleStorage: false, processing time: 8 msec
21/03/11 16:19:46 INFO hdfs.StateChange: BLOCK* allocate blk_1073741825_1001{UCState=UNDER_CONSTRUCTION, truncateBlock=null, primaryNodeIndex=-1, replicas=[ReplicaUC[[DISK]DS-92c06f70-ab0a-45bc-b120-6296e90b1973:NORMAL:192.168.56.1:50010|RBW]]} for /pullreq_events.csv._COPYING_
21/03/11 16:19:46 INFO namenode.FSEditLog: Number of transactions: 5 Total time for transactions(ms): 174 Number of transactions batched in Syncs: 0 Number of syncs: 3 SyncTimes(ms): 1896
21/03/11 16:19:47 INFO namenode.FSNamesystem: BLOCK* blk_1073741825_1001{UCState=COMMITTED, truncateBlock=null, primaryNodeIndex=-1, replicas=[ReplicaUC[[DISK]DS-92c06f70-ab0a-45bc-b120-6296e90b1973:NORMAL:192.168.56.1:50010|RBW]]} is not COMPLETE (ucState = COMMITTED, replication# = 0 < minimum = 1) in file /pullreq_events.csv._COPYING_
21/03/11 16:19:47 INFO BlockStateChange: BLOCK* addStoredBlock: blockMap updated: 192.168.56.1:50010 is added to blk_1073741825_1001{UCState=COMMITTED, truncateBlock=null, primaryNodeIndex=-1, replicas=[ReplicaUC[[DISK]DS-92c06f70-ab0a-45bc-b120-6296e90b1973:NORMAL:192.168.56.1:50010|RBW]]} size 12349372
21/03/11 16:19:48 INFO hdfs.StateChange: DIR* completeFile: /pullreq_events.csv._COPYING_ is closed by DFSClient_NONMAPREDUCE_-623955460_1
```

- Spark initialized.

```
spark = SparkSession.builder \
    .appName("PySpark HDFS") \
    .config("spark.local.dir", "C:/tmp") \
    .config("spark.executor.instances", "2") \
    .getOrCreate()
```

- The data was loaded from csv file stored in the hadoop file system and stored in a dataframe/RDD.

```
df = spark.read.csv("hdfs://127.0.0.1:19000/pullreq_events.csv",
header=True)
df.registerTempTable('pullreq_event')
```

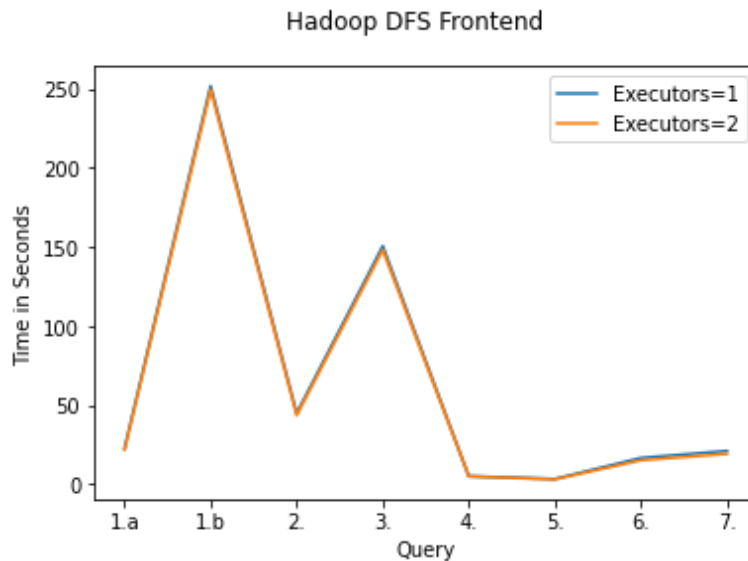
BDA ASSIGNMENT 2

- Again the same queries were taken from assignment 1 in the SQL syntax and passed to 'queries[i]' in the following.

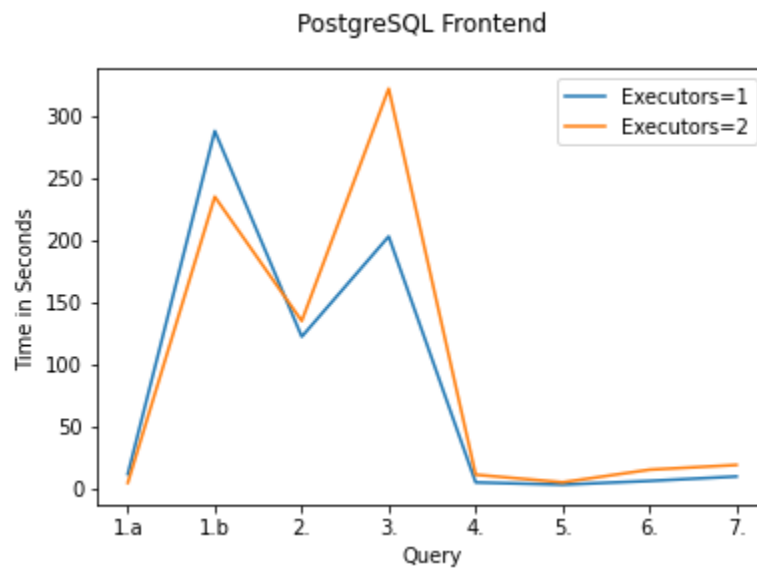
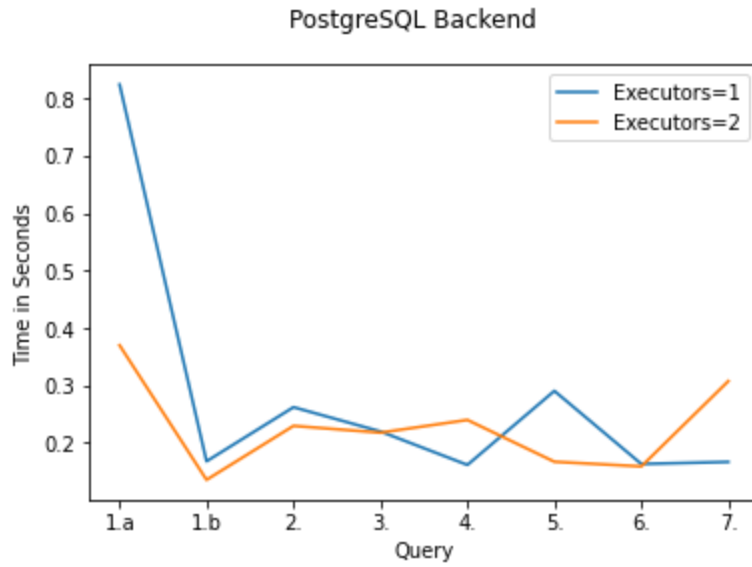
```
spark.sql(queries[i]).show()
```

```
Apache Hadoop Distribution - hadoop datanode
21/03/11 16:18:55 INFO datanode.VolumeScanner: VolumeScanner(\tmp\hadoop-test\dfs\data, DS-92c06f70-ab0a-45bc-b120-6296e90b1973): finished scanning block pool BP-537602078-192.168.56.1-1615459699945
21/03/11 16:18:55 ERROR datanode.DirectoryScanner: dfs.datanode.directoryscan.throttle.limit.ms.per.sec set to value below 1 ms/sec. Assuming default value of 1000
21/03/11 16:18:55 INFO datanode.DirectoryScanner: Periodic Directory Tree Verification scan starting at 1615469280767ms with interval of 21600000ms
21/03/11 16:18:55 INFO datanode.DataNode: Block pool BP-537602078-192.168.56.1-1615459699945 (Datanode Uuid null) service to /0.0.0.0:19000 beginning handshake with NN
21/03/11 16:18:55 INFO datanode.VolumeScanner: VolumeScanner(\tmp\hadoop-test\dfs\data, DS-92c06f70-ab0a-45bc-b120-6296e90b1973): no suitable block pools found to scan. Waiting 1814399926 ms.
21/03/11 16:19:00 INFO datanode.DataNode: Block pool BP-537602078-192.168.56.1-1615459699945 (Datanode Uuid null) service to /0.0.0.0:19000 successfully registered with NN
21/03/11 16:19:00 INFO datanode.DataNode: For namenode /0.0.0.0:19000 using BLOCKREPORT_INTERVAL of 21600000msec CACHEREPORT_INTERVAL of 10000msec Initial delay: 0msec; heartBeatInterval=3000
21/03/11 16:19:01 INFO datanode.DataNode: Namenode block pool BP-537602078-192.168.56.1-1615459699945 (Datanode Uuid 74321ec9-9e43-4495-b319-65d42b4dbaff) service to /0.0.0.0:19000 trying to claim ACTIVE state with txid=1
21/03/11 16:19:01 INFO datanode.DataNode: Acknowledging ACTIVE Namenode Block pool BP-537602078-192.168.56.1-1615459699945 (Datanode Uuid 74321ec9-9e43-4495-b319-65d42b4dbaff) service to /0.0.0.0:19000
21/03/11 16:19:02 INFO datanode.DataNode: Successfully sent block report 0x9a431ec79f74, containing 1 storage report(s), of which we sent 1. The reports had 0 total blocks and used 1 RPC(s). This took 13 msec to generate and 218 msec for RPC and NN processing. Got back one command: FinalizeCommand/5.
21/03/11 16:19:02 INFO datanode.DataNode: Got finalize command for block pool BP-537602078-192.168.56.1-1615459699945
21/03/11 16:19:47 INFO datanode.DataNode: Receiving BP-537602078-192.168.56.1-1615459699945:blk_1073741825_1001 src: /192.168.56.1:18537 dest: /192.168.56.1:50010
21/03/11 16:19:47 INFO DataNode.clienttrace: src: /192.168.56.1:18537, dest: /192.168.56.1:50010, bytes: 12349372, op: HDFS_WRITE, cliID: DFSClient_NONMAPREDUCE_-623955460_1, offset: 0, srvID: 74321ec9-9e43-4495-b319-65d42b4dbaff, blockid: BP-537602078-192.168.56.1-1615459699945:blk_1073741825_1001, duration: 332554900
21/03/11 16:19:47 INFO datanode.DataNode: PacketResponder: BP-537602078-192.168.56.1-1615459699945:blk_1073741825_1001, type=LAST_IN_PIPELINE, downstreams=0:[] terminating
```

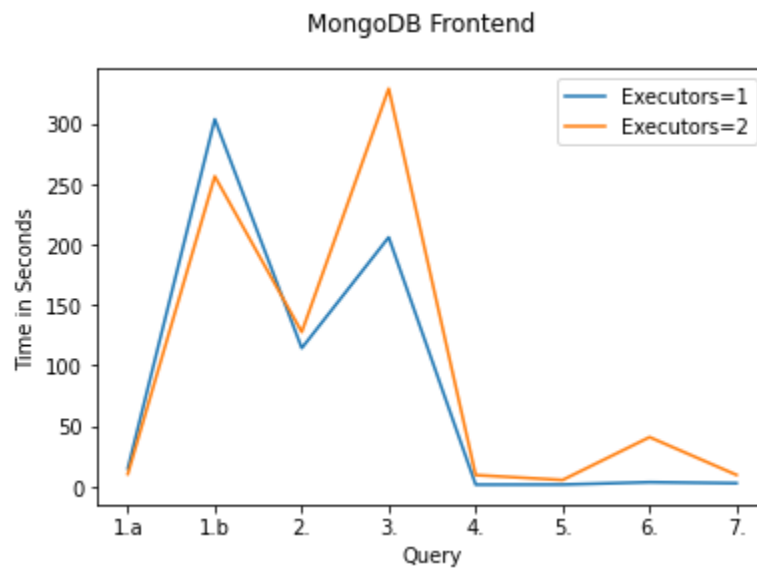
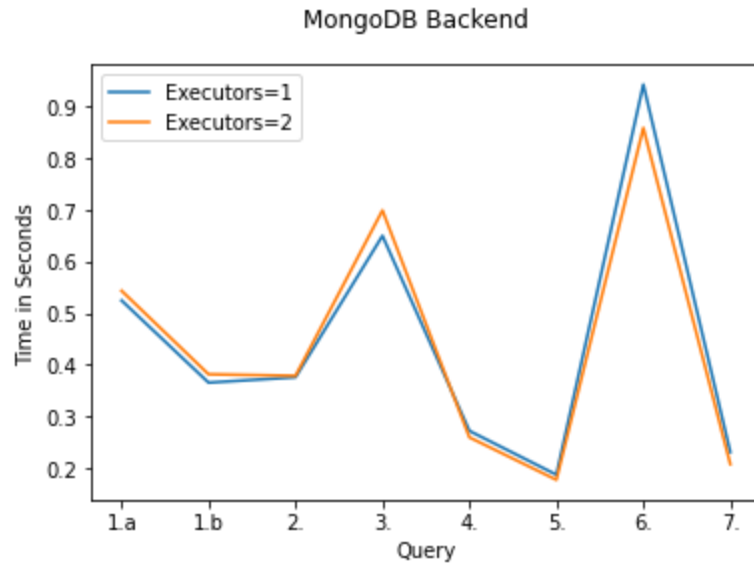
4. Execute commands for all parts in two settings: with only one executor and with two executors. Report your average execution time (over some runs of the commands) for each part in both settings.



BDA ASSIGNMENT 2



BDA ASSIGNMENT 2



BDA ASSIGNMENT 2

HADOOP

Executors=1

[22.509477397328745, 251.862155380873476, 44.75394942582140,
150.540043388599092, 4.95314047859909, 3.1987861174369, 16.4950235980529,
20.912178345980529]

HADOOP

Executors=2

[21.773667732874554, 249.58760873476663, 43.515235821406044,
148.13695859909058, 4.853235165278117, 2.98285174369812,
15.053032159805298, 19.114360014597576]

POSTGRES-BACKEND

Executors=1

[0.8241069793701172, 0.1673967123031616, 0.26167662143707277,
0.21938045024871827, 0.1610717296600342, 0.290036678314209,
0.16273510456085205, 0.16629159450531006]

POSTGRES-BACKEND

Executors=2

[0.36961138248443604, 0.13508906364440917, 0.2291285753250122,
0.2173015832901001, 0.2394742488861084, 0.16658904552459716,
0.15846605300903321, 0.30710530281066895]

POSTGRES-FRONTEND

Executors=1

[11.372584899266561, 287.56904260317486, 121.9718411763509,
202.72570045789084, 4.612216234207153, 2.820171038309733,
5.820555051167806, 9.393634875615438]

POSTGRES-FRONTEND

Executors=2

[4.305006821950276, 234.6917590300242, 134.63548787434897,
321.71466700236004, 10.84055725733439, 4.718230565388997,
14.864173173904419, 18.67643944422404]

MONGODB-BACKEND

BDA ASSIGNMENT 2

Executors=1

```
[0.5248032331466674,      0.36573240756988523,      0.37603073120117186,
0.650059700012207,      0.2722919464111328,      0.18742387294769286,
0.943103289604187, 0.23130388259887696]
```

MONGODB-BACKEND

Executors=2

```
[0.5436521053314209,      0.38195674419403075,      0.378934907913208,
0.6992531299591065,      0.2596954107284546,      0.1775022029876709,
0.859039044380188, 0.20791006088256836]
```

MONGODB-FRONTEND

Executors=1

```
[14.948185364405314,      303.7211236159007,      114.44110067685445,
206.15040723482767,      1.5650041898091633,      1.6539669036865234,
3.59218692779541, 2.78354541460673]
```

MONGODB-FRONTEND

Executors=2

```
[10.347276290257772,      256.56013584136963,      127.79095546404521,
329.0319859981537,      9.485195398330688,      5.4332239627838135,
40.85181665420532, 9.593047300974527]
```

1. There are two ways you can compute your queries: 1) Import the data in the main memory data structure (RDD) and evaluate your queries, 2) Directly execute your queries on the backend if it is some database engine at the backend. Include the merits and demerits of each approach in your report. Include some empirical results using the queries to substantiate the claims.

Merits and Demerits:

- RDD is higher in speed, as it is stored in the main memory and fetches data from there only. RDDs can handle petabytes of data over a large number of nodes, which is not possible in databases like SQL. For large databases RDDs run much faster than other database engines.

BDA ASSIGNMENT 2

- No inbuilt optimisation engine is available for RDDs. When working with structured data, RDDs cannot take advantage of spark's advanced optimizers. For query engines like SQL, each query is optimised.
- RDD provides OOPs type programming and compile time safety.
- Spark evaluates RDDs lazily. Spark computes Transformations only when an action needs a result to send to the driver program.
- RDDs are fairly recent instead, query engines like SQL are very old and hence used more and are user-friendly.
- Empirical results with the help of graphs are shown above.

2. In your report include details on how you stored the data in the respective storage variants and also the challenges you faced in integrating Spark with various storage mediums.

- First of all, setting up Hadoop DFS was a challenging task, due to setting up of various path variables and configurations.
 - Hadoop installation on windows requires the username to not have any space characters, thus we had to create a new user account and setup hadoop on that user.
 - Unfortunately, we could not find any fix for including space in the username.
- Installation of MongoDB was pretty smooth, but sending a query to the backend through pyspark was another big challenge.
 - We came across a new syntax called the aggregation pipeline syntax, which we had to learn to write queries in.
- PostgreSQL was not much challenging to say, once we found how to contact the server through JDBC via pyspark.

OUTPUT for all sections:

For output of each section, please see the postgres.txt, mongo.txt, hadoop.txt for output for corresponding techniques. These files have backend output first, followed by output obtained by the frontend.

The answer in every case is :

BDA ASSIGNMENT 2

1. Report the number of pull requests

a. "opened" per day.

```
+-----+-----+
|count|timestamp_date|
+-----+-----+
|    2|    2010-09-02|
|    1|    2010-09-06|
|    1|    2010-09-08|
|    4|    2010-09-09|
|    3|    2010-09-10|
|    3|    2010-09-11|
|    3|    2010-09-12|
|    3|    2010-09-13|
|    2|    2010-09-15|
|    2|    2010-09-16|
|    6|    2010-09-18|
|    4|    2010-09-19|
|    2|    2010-09-20|
|    1|    2010-09-22|
|    4|    2010-09-23|
|    5|    2010-09-24|
|    5|    2010-09-25|
|    4|    2010-09-27|
|    2|    2010-09-28|
|    2|    2010-09-29|
+-----+-----+
only showing top 20 rows
```

b. "Discussed" per day.

```
+-----+-----+
|count|timestamp_date|
+-----+-----+
|    4|    2010-09-09|
|    4|    2010-09-10|
|    7|    2010-09-11|
|    3|    2010-09-12|
|    2|    2010-09-13|
|    1|    2010-09-14|
|    1|    2010-09-15|
|    1|    2010-09-16|
|    1|    2010-09-17|
|    1|    2010-09-21|
|    1|    2010-09-22|
|    4|    2010-09-23|
|    3|    2010-09-24|
```

BDA ASSIGNMENT 2

```
|      5|      2010-09-25|
|      3|      2010-09-27|
|      1|      2010-09-29|
|      1|      2010-09-30|
|      1|      2010-10-01|
|      1|      2010-10-04|
|      5|      2010-10-06|
+-----+-----+
only showing top 20 rows
```

2. Output the person who has the highest number of comments per month. Assume a month has 30 days. The comments refer to events of the type “discussed”. (Points:10)

```
+-----+-----+-----+
|date_month|      author|numprs|
+-----+-----+-----+
|      1.0|rafaelfranca|  828|
|      2.0|rafaelfranca|  555|
|      3.0|rafaelfranca|  580|
|      4.0|rafaelfranca|  758|
|      5.0|rafaelfranca|  915|
|      6.0|rafaelfranca|  582|
|      7.0|rafaelfranca|  579|
|      8.0|rafaelfranca|  651|
|      9.0|rafaelfranca|  585|
|     10.0|rafaelfranca|  667|
|     11.0|rafaelfranca|  590|
|     12.0|   rails-bot|  546|
+-----+-----+-----+
```

3. Output the person who has the highest number of comments per week. Assume a week has 7 days. The comments refer to events of the type “discussed”. (Points: 10)

```
+-----+-----+-----+
|sunday_of_week|numprs|      author|
+-----+-----+-----+
|     2010-09-05|    13|      mikel|
|     2010-09-12|     9|      mikel|
|     2010-09-19|     9|   josevalim|
|     2010-09-26|     6|   josevalim|
|     2010-10-03|    11|   josevalim|
|     2010-10-10|     4|   josevalim|
|     2010-10-17|     6|   krekoten|
|     2010-10-24|     1|         fxn|
|     2010-10-31|     2|spastorino|
|     2010-11-07|     3|   josevalim|
```

BDA ASSIGNMENT 2

```
| 2010-11-14| 6| josevalim|
| 2010-11-21| 6| tenderlove|
| 2010-11-28| 4| josevalim|
| 2010-12-05| 2| drogus|
| 2010-12-12| 2| josevalim|
| 2010-12-12| 2| tenderlove|
| 2010-12-19| 6| josevalim|
| 2011-01-02| 2| tenderlove|
| 2011-01-09| 21| jeremy|
| 2011-01-16| 5| tenderlove|
+-----+-----+
only showing top 20 rows
```

4. Output the number of Pull Requests (PRs) opened per week. (Points:10)

```
+-----+-----+
|sunday_of_week|numprs|
+-----+-----+
| 2014-12-21| 70|
| 2013-01-06| 91|
| 2016-08-28| 51|
| 2016-09-04| 48|
| 2017-04-30| 42|
| 2011-03-06| 23|
| 2016-08-14| 63|
| 2018-02-11| 46|
| 2012-08-12| 32|
| 2016-07-03| 20|
| 2018-05-27| 26|
| 2017-08-20| 76|
| 2011-10-09| 41|
| 2011-12-18| 111|
| 2014-10-19| 54|
| 2018-03-25| 36|
| 2016-01-03| 59|
| 2011-09-25| 34|
| 2015-08-09| 53|
| 2010-09-19| 21|
+-----+-----+
only showing top 20 rows
```

5. Count the number of Pull Requests merged per month in the year 2010. (Points:10)

```
+-----+-----+
|date_month|coalesce|
+-----+-----+
```

BDA ASSIGNMENT 2

	1.0	0
	2.0	0
	3.0	0
	4.0	0
	5.0	0
	6.0	0
	7.0	0
	8.0	0
	9.0	0
	10.0	0
	11.0	0
	12.0	0
+-----+-----+		

NOTE: for mongodb backend the output received was:

```
++  
| |  
++  
++
```

That is, an empty table as there are no answers to the query.

6. Per day report the total number of events. (Points: 5)

+-----+-----+	
count	timestamp_date
+-----+-----+	
2	2010-09-02
1	2010-09-06
1	2010-09-08
10	2010-09-09
13	2010-09-10
16	2010-09-11
8	2010-09-12
10	2010-09-13
1	2010-09-14
5	2010-09-15
4	2010-09-16
1	2010-09-17
6	2010-09-18
4	2010-09-19
2	2010-09-20
6	2010-09-21
4	2010-09-22
9	2010-09-23
8	2010-09-24
10	2010-09-25

BDA ASSIGNMENT 2

```
+-----+-----+
only showing top 20 rows
```

7. Report the user who has opened the highest number of PRs in the year 2011. (Points: 15)

```
+-----+-----+
|num_prs| author|
+-----+-----+
|      228|arunagw|
+-----+-----+
```

Make a section “Learning”, which describes your learning in doing this assignment.

- Setup PostgreSQL, MongoDB, Hadoop DFS
- Pyspark library and its functions
- Integrate these databases with apache spark to fetch data into RDD.
- RDD and SQL query engines and difference between the two and how to use them via the spark framework.
- Aggregate pipeline query mongodb engine queries.
- Functions such as \$year, \$month etc. in pipeline for Mongodb for extracting date and month
- Syntax for Mongodb pipeline for querying and different stages such as project, match, group etc.