Assignment 18: Intro to Spark Assignment Problems

Problem Statement

--==========

Task 1

Given a list of numbers - List[Int] (1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

Initial Execution :

[acadgild@localhost ~]$ jps
2989 Jps
[acadgild@localhost ~]$ sudo service sshd start
[sudo] password for acadgild:
[acadgild@localhost ~]$ start-all.sh
This script is Deprecated. Instead use start-dfs.sh and start-yarn.sh
18/08/27 13:48:26 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
Starting namenodes on [localhost]
localhost: starting namenode, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-namenode-localhost.localdomain.out
localhost: starting datanode, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-datanode-localhost.localdomain.out
Starting secondary namenodes [0.0.0.0]
0.0.0.0: starting secondarynamenode, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/hadoop-acadgild-secondarynamenode-localhost.localdomain.out
18/08/27 13:49:09 WARN util.NativeCodeLoader: Unable to load native-hadoop library for your platform... using builtin-java classes where applicable
starting yarn daemons
starting resourcemanager, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/yarn-acadgild-resourcemanager-localhost.localdomain.out
localhost: starting nodemanager, logging to /home/acadgild/install/hadoop/hadoop-2.6.5/logs/yarn-acadgild-nodemanager-localhost.localdomain.out
[acadgild@localhost~]$ spark-shell
18/08/27 13:55:43 WARN DataNucleus.Query: Query for candidates of org.apache.hadoop.hive.metastore.model.MPartitionColumnStatistics and subclasses resulted in no

possible candidates
Error(s) were found while auto-creating/validating the datastore for classes. The errors are printed in the log, and are attached to this exception.
org.datanucleus.exceptions.NucleusDataStoreException: Error(s) were found while auto-creating/validating the datastore for classes. The errors are printed in the log, and are attached to this exception.
at org.datanucleus.store.rdbms.RDBMSStoreManager$ClassAdder.verifyErrors(RDBMSStoreManager.java:3602)
at org.datanucleus.store.rdbms.RDBMSStoreManager$ClassAdder.addClassTablesAndValidate(RDBMSStoreManager.java:3205)
at org.datanucleus.store.rdbms.RDBMSStoreManager$ClassAdder.run(RDBMSStoreManager.java:2841)
at org.datanucleus.store.rdbms.AbstractSchemaTransaction.execute(AbstractSchemaTransaction.java:122)
at org.datanucleus.store.rdbms.RDBMSStoreManager.addClasses(RDBMSStoreManager.java:1605)
at org.datanucleus.store.AbstractStoreManager.addClass(AbstractStoreManager.java:954)
at org.datanucleus.store.rdbms.RDBMSStoreManager.getDatastoreClass(RDBMSStoreManager.java:679)
at org.datanucleus.store.rdbms.query.RDBMSQueryUtils.getStatementForCandidates(RDBMSQueryUtils.java:408)
at org.datanucleus.store.rdbms.query.JDOQLQuery.compileQueryFull(JDOQLQuery.java:947)
at org.datanucleus.store.rdbms.query.JDOQLQuery.compileInternal(JDOQLQuery.java:370)
at org.datanucleus.store.query.Query.executeQuery(Query.java:1744)
at org.datanucleus.store.query.Query.executeWithArray(Query.java:1672)
at org.datanucleus.store.query.Query.execute(Query.java:1654)
at org.datanucleus.api.jdo.JDOQuery.execute(JDOQuery.java:221)
at org.apache.hadoop.hive.metastore.MetaStoreDirectSql.ensureDbInit(MetaStoreDirectSql.java:185)
at org.apache.hadoop.hive.metastore.MetaStoreDirectSql.<init>(MetaStoreDirectSql.java:137)
at org.apache.hadoop.hive.metastore.ObjectStore.initialize(ObjectStore.java:295)
at org.apache.hadoop.hive.metastore.ObjectStore.setConf(ObjectStore.java:258)
at org.apache.hadoop.util.ReflectionUtils.setConf(ReflectionUtils.java:73)
at org.apache.hadoop.util.ReflectionUtils.newInstance(ReflectionUtils.java:133)
at org.apache.hadoop.hive.metastore.RawStoreProxy.<init>(RawStoreProxy.java:57)
at org.apache.hadoop.hive.metastore.RawStoreProxy.getProxy(RawStoreProxy.java:66)
at org.apache.hadoop.hive.metastore.HiveMetaStore$HMSHandler.newRawStore(HiveMetaStore.java:593)
at org.apache.hadoop.hive.metastore.HiveMetaStore$HMSHandler.getMS(HiveMetaStore.java:571)

at
org.apache.hadoop.hive.metastore.HiveMetaStore$HMSHandler.createDefaultDB(HiveMetaStore.java:620)
at
org.apache.hadoop.hive.metastore.HiveMetaStore$HMSHandler.init(HiveMetaStore.java:461)
at
org.apache.hadoop.hive.metastore.RetryingHMSHandler.<init>(RetryingHMSHandler.java:66)
at
org.apache.hadoop.hive.metastore.RetryingHMSHandler.getProxy(RetryingHMSHandler.java:72)
at
org.apache.hadoop.hive.metastore.HiveMetaStore.newRetryingHMSHandler(HiveMetaStore.java:5762)
at
org.apache.hadoop.hive.metastore.HiveMetaStoreClient.<init>(HiveMetaStoreClient.java:199)
at
org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClient.<init>(SessionHiveMetaStoreClient.java:74)
at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45)
at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
at org.apache.hadoop.hive.metastore.MetaStoreUtils.newInstance(MetaStoreUtils.java:1521)
at
org.apache.hadoop.hive.metastore.RetryingMetaStoreClient.<init>(RetryingMetaStoreClient.java:86)
at
org.apache.hadoop.hive.metastore.RetryingMetaStoreClient.getProxy(RetryingMetaStoreClient.java:132)
at
org.apache.hadoop.hive.metastore.RetryingMetaStoreClient.getProxy(RetryingMetaStoreClient.java:104)
at org.apache.hadoop.hive.ql.metadata.Hive.createMetaStoreClient(Hive.java:3005)
at org.apache.hadoop.hive.ql.metadata.Hive.getMSC(Hive.java:3024)
at org.apache.hadoop.hive.ql.metadata.Hive.getAllDatabases(Hive.java:1234)
at org.apache.hadoop.hive.ql.metadata.Hive.reloadFunctions(Hive.java:174)
at org.apache.hadoop.hive.ql.metadata.Hive.<clinit>(Hive.java:166)
at org.apache.hadoop.hive.ql.session.SessionState.start(SessionState.java:503)
at org.apache.spark.sql.hive.client.HiveClientImpl.<init>(HiveClientImpl.scala:192)
at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.java:45)
at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
at
org.apache.spark.sql.hive.client.IsolatedClientLoader.createClient(IsolatedClientLoader.scala:264)

```
	at org.apache.spark.sql.hive.HiveUtils$.newClientForMetadata(HiveUtils.scala:366)
	at org.apache.spark.sql.hive.HiveUtils$.newClientForMetadata(HiveUtils.scala:270)
	at org.apache.spark.sql.hive.HiveExternalCatalog.<init>(HiveExternalCatalog.scala:65)
	at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
	at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
	at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.ja
va:45)
	at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
	at org.apache.spark.sql.internal.SharedState$.org$apache$spark$sql$internal$SharedState$
$reflect(SharedState.scala:166)
	at org.apache.spark.sql.internal.SharedState.<init>(SharedState.scala:86)
	at org.apache.spark.sql.SparkSession$$anonfun$sharedState$1.apply(SparkSession.scala:101)
	at org.apache.spark.sql.SparkSession$$anonfun$sharedState$1.apply(SparkSession.scala:101)
	at scala.Option.getOrElse(Option.scala:121)
	at org.apache.spark.sql.SparkSession.sharedState$lzycompute(SparkSession.scala:101)
	at org.apache.spark.sql.SparkSession.sharedState(SparkSession.scala:100)
	at org.apache.spark.sql.internal.SessionState.<init>(SessionState.scala:157)
	at org.apache.spark.sql.hive.HiveSessionState.<init>(HiveSessionState.scala:32)
	at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
	at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
	at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.ja
va:45)
	at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
	at org.apache.spark.sql.SparkSession$.org$apache$spark$sql$SparkSession$
$reflect(SparkSession.scala:978)
	at org.apache.spark.sql.SparkSession.sessionState$lzycompute(SparkSession.scala:110)
	at org.apache.spark.sql.SparkSession.sessionState(SparkSession.scala:109)
	at org.apache.spark.sql.SparkSession$Builder$
$anonfun$getOrCreate$5.apply(SparkSession.scala:878)
	at org.apache.spark.sql.SparkSession$Builder$
$anonfun$getOrCreate$5.apply(SparkSession.scala:878)
	at scala.collection.mutable.HashMap$$anonfun$foreach$1.apply(HashMap.scala:99)
	at scala.collection.mutable.HashMap$$anonfun$foreach$1.apply(HashMap.scala:99)
	at scala.collection.mutable.HashTable$class.foreachEntry(HashTable.scala:230)
	at scala.collection.mutable.HashMap.foreachEntry(HashMap.scala:40)
	at scala.collection.mutable.HashMap.foreach(HashMap.scala:99)
	at org.apache.spark.sql.SparkSession$Builder.getOrCreate(SparkSession.scala:878)
	at org.apache.spark.repl.Main$.createSparkSession(Main.scala:95)
	at $line3.$read$$iw$$iw.<init>(<console>:15)
	at $line3.$read$$iw.<init>(<console>:42)
	at $line3.$read.<init>(<console>:44)
	at $line3.$read$.<init>(<console>:48)
	at $line3.$read$.<clinit>(<console>)
	at $line3.$eval$.$print$lzycompute(<console>:7)
	at $line3.$eval$.$print(<console>:6)
```

```
        at $line3.$eval.$print(<console>)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at scala.tools.nsc.interpreter.IMain$ReadEvalPrint.call(IMain.scala:786)
        at scala.tools.nsc.interpreter.IMain$Request.loadAndRun(IMain.scala:1047)
        at scala.tools.nsc.interpreter.IMain$WrappedRequest$
$anonfun$loadAndRunReq$1.apply(IMain.scala:638)
        at scala.tools.nsc.interpreter.IMain$WrappedRequest$
$anonfun$loadAndRunReq$1.apply(IMain.scala:637)
        at scala.reflect.internal.util.ScalaClassLoader$class.asContext(ScalaClassLoader.scala:31)
        at
scala.reflect.internal.util.AbstractFileClassLoader.asContext(AbstractFileClassLoader.scala:19)
        at scala.tools.nsc.interpreter.IMain$WrappedRequest.loadAndRunReq(IMain.scala:637)
        at scala.tools.nsc.interpreter.IMain.interpret(IMain.scala:569)
        at scala.tools.nsc.interpreter.IMain.interpret(IMain.scala:565)
        at scala.tools.nsc.interpreter.ILoop.interpretStartingWith(ILoop.scala:807)
        at scala.tools.nsc.interpreter.ILoop.command(ILoop.scala:681)
        at scala.tools.nsc.interpreter.ILoop.processLine(ILoop.scala:395)
        at org.apache.spark.repl.SparkILoop$
$anonfun$initializeSpark$1.apply$mcV$sp(SparkILoop.scala:38)
        at org.apache.spark.repl.SparkILoop$$anonfun$initializeSpark$1.apply(SparkILoop.scala:37)
        at org.apache.spark.repl.SparkILoop$$anonfun$initializeSpark$1.apply(SparkILoop.scala:37)
        at scala.tools.nsc.interpreter.IMain.beQuietDuring(IMain.scala:214)
        at org.apache.spark.repl.SparkILoop.initializeSpark(SparkILoop.scala:37)
        at org.apache.spark.repl.SparkILoop.loadFiles(SparkILoop.scala:105)
        at scala.tools.nsc.interpreter.ILoop$$anonfun$process$1.apply$mcZ$sp(ILoop.scala:920)
        at scala.tools.nsc.interpreter.ILoop$$anonfun$process$1.apply(ILoop.scala:909)
        at scala.tools.nsc.interpreter.ILoop$$anonfun$process$1.apply(ILoop.scala:909)
        at
scala.reflect.internal.util.ScalaClassLoader$.savingContextLoader(ScalaClassLoader.scala:97)
        at scala.tools.nsc.interpreter.ILoop.process(ILoop.scala:909)
        at org.apache.spark.repl.Main$.doMain(Main.scala:68)
        at org.apache.spark.repl.Main$.main(Main.scala:51)
        at org.apache.spark.repl.Main.main(Main.scala)
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
        at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:498)
        at org.apache.spark.deploy.SparkSubmit$.org$apache$spark$deploy$SparkSubmit$
$runMain(SparkSubmit.scala:738)
        at org.apache.spark.deploy.SparkSubmit$.doRunMain$1(SparkSubmit.scala:187)
        at org.apache.spark.deploy.SparkSubmit$.submit(SparkSubmit.scala:212)
        at org.apache.spark.deploy.SparkSubmit$.main(SparkSubmit.scala:126)
        at org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)
Caused by: com.mysql.jdbc.exceptions.jdbc4.MySQLSyntaxErrorException: Specified key was too
```

```
long; max key length is 3072 bytes
	at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
	at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
	at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.ja
va:45)
	at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
	at com.mysql.jdbc.Util.handleNewInstance(Util.java:425)
	at com.mysql.jdbc.Util.getInstance(Util.java:408)
	at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:944)
	at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3976)
	at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3912)
	at com.mysql.jdbc.MysqlIO.sendCommand(MysqlIO.java:2530)
	at com.mysql.jdbc.MysqlIO.sqlQueryDirect(MysqlIO.java:2683)
	at com.mysql.jdbc.ConnectionImpl.execSQL(ConnectionImpl.java:2482)
	at com.mysql.jdbc.ConnectionImpl.execSQL(ConnectionImpl.java:2440)
	at com.mysql.jdbc.StatementImpl.executeInternal(StatementImpl.java:845)
	at com.mysql.jdbc.StatementImpl.execute(StatementImpl.java:745)
	at com.jolbox.bonecp.StatementHandle.execute(StatementHandle.java:254)
	at
org.datanucleus.store.rdbms.table.AbstractTable.executeDdlStatement(AbstractTable.java:760)
	at org.datanucleus.store.rdbms.table.TableImpl.createIndices(TableImpl.java:648)
	at org.datanucleus.store.rdbms.table.TableImpl.createConstraints(TableImpl.java:422)
	at
org.datanucleus.store.rdbms.RDBMSStoreManager$ClassAdder.performTablesValidation(RDBMSSt
oreManager.java:3459)
	at
org.datanucleus.store.rdbms.RDBMSStoreManager$ClassAdder.addClassTablesAndValidate(RDBM
SStoreManager.java:3190)
	... 128 more
Nested Throwables StackTrace:
com.mysql.jdbc.exceptions.jdbc4.MySQLSyntaxErrorException: Specified key was too long; max
key length is 3072 bytes
	at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
	at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
	at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.ja
va:45)
	at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
	at com.mysql.jdbc.Util.handleNewInstance(Util.java:425)
	at com.mysql.jdbc.Util.getInstance(Util.java:408)
	at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:944)
	at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3976)
	at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3912)
	at com.mysql.jdbc.MysqlIO.sendCommand(MysqlIO.java:2530)
	at com.mysql.jdbc.MysqlIO.sqlQueryDirect(MysqlIO.java:2683)
	at com.mysql.jdbc.ConnectionImpl.execSQL(ConnectionImpl.java:2482)
```

```
	at com.mysql.jdbc.ConnectionImpl.execSQL(ConnectionImpl.java:2440)
	at com.mysql.jdbc.StatementImpl.executeInternal(StatementImpl.java:845)
	at com.mysql.jdbc.StatementImpl.execute(StatementImpl.java:745)
	at com.jolbox.bonecp.StatementHandle.execute(StatementHandle.java:254)
	at
org.datanucleus.store.rdbms.table.AbstractTable.executeDdlStatement(AbstractTable.java:760)
	at org.datanucleus.store.rdbms.table.TableImpl.createIndices(TableImpl.java:648)
	at org.datanucleus.store.rdbms.table.TableImpl.createConstraints(TableImpl.java:422)
	at
org.datanucleus.store.rdbms.RDBMSStoreManager$ClassAdder.performTablesValidation(RDBMSSt
oreManager.java:3459)
	at
org.datanucleus.store.rdbms.RDBMSStoreManager$ClassAdder.addClassTablesAndValidate(RDBM
SStoreManager.java:3190)
	at
org.datanucleus.store.rdbms.RDBMSStoreManager$ClassAdder.run(RDBMSStoreManager.java:284
1)
	at
org.datanucleus.store.rdbms.AbstractSchemaTransaction.execute(AbstractSchemaTransaction.java:1
22)
	at
org.datanucleus.store.rdbms.RDBMSStoreManager.addClasses(RDBMSStoreManager.java:1605)
	at org.datanucleus.store.AbstractStoreManager.addClass(AbstractStoreManager.java:954)
	at
org.datanucleus.store.rdbms.RDBMSStoreManager.getDatastoreClass(RDBMSStoreManager.java:6
79)
	at
org.datanucleus.store.rdbms.query.RDBMSQueryUtils.getStatementForCandidates(RDBMSQueryUt
ils.java:408)
	at
org.datanucleus.store.rdbms.query.JDOQLQuery.compileQueryFull(JDOQLQuery.java:947)
	at org.datanucleus.store.rdbms.query.JDOQLQuery.compileInternal(JDOQLQuery.java:370)
	at org.datanucleus.store.query.Query.executeQuery(Query.java:1744)
	at org.datanucleus.store.query.Query.executeWithArray(Query.java:1672)
	at org.datanucleus.store.query.Query.execute(Query.java:1654)
	at org.datanucleus.api.jdo.JDOQuery.execute(JDOQuery.java:221)
	at
org.apache.hadoop.hive.metastore.MetaStoreDirectSql.ensureDbInit(MetaStoreDirectSql.java:185)
	at
org.apache.hadoop.hive.metastore.MetaStoreDirectSql.<init>(MetaStoreDirectSql.java:137)
	at org.apache.hadoop.hive.metastore.ObjectStore.initialize(ObjectStore.java:295)
	at org.apache.hadoop.hive.metastore.ObjectStore.setConf(ObjectStore.java:258)
	at org.apache.hadoop.util.ReflectionUtils.setConf(ReflectionUtils.java:73)
	at org.apache.hadoop.util.ReflectionUtils.newInstance(ReflectionUtils.java:133)
	at org.apache.hadoop.hive.metastore.RawStoreProxy.<init>(RawStoreProxy.java:57)
	at org.apache.hadoop.hive.metastore.RawStoreProxy.getProxy(RawStoreProxy.java:66)
	at
org.apache.hadoop.hive.metastore.HiveMetaStore$HMSHandler.newRawStore(HiveMetaStore.java:
593)
```

```
	at
org.apache.hadoop.hive.metastore.HiveMetaStore$HMSHandler.getMS(HiveMetaStore.java:571)
	at
org.apache.hadoop.hive.metastore.HiveMetaStore$HMSHandler.createDefaultDB(HiveMetaStore.ja
va:620)
	at
org.apache.hadoop.hive.metastore.HiveMetaStore$HMSHandler.init(HiveMetaStore.java:461)
	at
org.apache.hadoop.hive.metastore.RetryingHMSHandler.<init>(RetryingHMSHandler.java:66)
	at
org.apache.hadoop.hive.metastore.RetryingHMSHandler.getProxy(RetryingHMSHandler.java:72)
	at
org.apache.hadoop.hive.metastore.HiveMetaStore.newRetryingHMSHandler(HiveMetaStore.java:57
62)
	at
org.apache.hadoop.hive.metastore.HiveMetaStoreClient.<init>(HiveMetaStoreClient.java:199)
	at
org.apache.hadoop.hive.ql.metadata.SessionHiveMetaStoreClient.<init>(SessionHiveMetaStoreClien
t.java:74)
	at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
	at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
	at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.ja
va:45)
	at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
	at org.apache.hadoop.hive.metastore.MetaStoreUtils.newInstance(MetaStoreUtils.java:1521)
	at
org.apache.hadoop.hive.metastore.RetryingMetaStoreClient.<init>(RetryingMetaStoreClient.java:86)
	at
org.apache.hadoop.hive.metastore.RetryingMetaStoreClient.getProxy(RetryingMetaStoreClient.java:
132)
	at
org.apache.hadoop.hive.metastore.RetryingMetaStoreClient.getProxy(RetryingMetaStoreClient.java:
104)
	at org.apache.hadoop.hive.ql.metadata.Hive.createMetaStoreClient(Hive.java:3005)
	at org.apache.hadoop.hive.ql.metadata.Hive.getMSC(Hive.java:3024)
	at org.apache.hadoop.hive.ql.metadata.Hive.getAllDatabases(Hive.java:1234)
	at org.apache.hadoop.hive.ql.metadata.Hive.reloadFunctions(Hive.java:174)
	at org.apache.hadoop.hive.ql.metadata.Hive.<clinit>(Hive.java:166)
	at org.apache.hadoop.hive.ql.session.SessionState.start(SessionState.java:503)
	at org.apache.spark.sql.hive.client.HiveClientImpl.<init>(HiveClientImpl.scala:192)
	at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
	at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
	at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.ja
va:45)
	at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
```

```
	at
org.apache.spark.sql.hive.client.IsolatedClientLoader.createClient(IsolatedClientLoader.scala:264)
	at org.apache.spark.sql.hive.HiveUtils$.newClientForMetadata(HiveUtils.scala:366)
	at org.apache.spark.sql.hive.HiveUtils$.newClientForMetadata(HiveUtils.scala:270)
	at org.apache.spark.sql.hive.HiveExternalCatalog.<init>(HiveExternalCatalog.scala:65)
	at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
	at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
	at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.ja
va:45)
	at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
	at org.apache.spark.sql.internal.SharedState$.org$apache$spark$sql$internal$SharedState$
$reflect(SharedState.scala:166)
	at org.apache.spark.sql.internal.SharedState.<init>(SharedState.scala:86)
	at org.apache.spark.sql.SparkSession$$anonfun$sharedState$1.apply(SparkSession.scala:101)
	at org.apache.spark.sql.SparkSession$$anonfun$sharedState$1.apply(SparkSession.scala:101)
	at scala.Option.getOrElse(Option.scala:121)
	at org.apache.spark.sql.SparkSession.sharedState$lzycompute(SparkSession.scala:101)
	at org.apache.spark.sql.SparkSession.sharedState(SparkSession.scala:100)
	at org.apache.spark.sql.internal.SessionState.<init>(SessionState.scala:157)
	at org.apache.spark.sql.hive.HiveSessionState.<init>(HiveSessionState.scala:32)
	at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
	at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
	at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.ja
va:45)
	at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
	at org.apache.spark.sql.SparkSession$.org$apache$spark$sql$SparkSession$
$reflect(SparkSession.scala:978)
	at org.apache.spark.sql.SparkSession.sessionState$lzycompute(SparkSession.scala:110)
	at org.apache.spark.sql.SparkSession.sessionState(SparkSession.scala:109)
	at org.apache.spark.sql.SparkSession$Builder$
$anonfun$getOrCreate$5.apply(SparkSession.scala:878)
	at org.apache.spark.sql.SparkSession$Builder$
$anonfun$getOrCreate$5.apply(SparkSession.scala:878)
	at scala.collection.mutable.HashMap$$anonfun$foreach$1.apply(HashMap.scala:99)
	at scala.collection.mutable.HashMap$$anonfun$foreach$1.apply(HashMap.scala:99)
	at scala.collection.mutable.HashTable$class.foreachEntry(HashTable.scala:230)
	at scala.collection.mutable.HashMap.foreachEntry(HashMap.scala:40)
	at scala.collection.mutable.HashMap.foreach(HashMap.scala:99)
	at org.apache.spark.sql.SparkSession$Builder.getOrCreate(SparkSession.scala:878)
	at org.apache.spark.repl.Main$.createSparkSession(Main.scala:95)
	at $line3.$read$$iw$$iw.<init>(<console>:15)
	at $line3.$read$$iw.<init>(<console>:42)
	at $line3.$read.<init>(<console>:44)
	at $line3.$read$.<init>(<console>:48)
	at $line3.$read$.<clinit>(<console>)
```

```
	at $line3.$eval$.$print$lzycompute(<console>:7)
	at $line3.$eval$.$print(<console>:6)
	at $line3.$eval.$print(<console>)
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
	at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
	at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
	at java.lang.reflect.Method.invoke(Method.java:498)
	at scala.tools.nsc.interpreter.IMain$ReadEvalPrint.call(IMain.scala:786)
	at scala.tools.nsc.interpreter.IMain$Request.loadAndRun(IMain.scala:1047)
	at scala.tools.nsc.interpreter.IMain$WrappedRequest$
$anonfun$loadAndRunReq$1.apply(IMain.scala:638)
	at scala.tools.nsc.interpreter.IMain$WrappedRequest$
$anonfun$loadAndRunReq$1.apply(IMain.scala:637)
	at scala.reflect.internal.util.ScalaClassLoader$class.asContext(ScalaClassLoader.scala:31)
	at
scala.reflect.internal.util.AbstractFileClassLoader.asContext(AbstractFileClassLoader.scala:19)
	at scala.tools.nsc.interpreter.IMain$WrappedRequest.loadAndRunReq(IMain.scala:637)
	at scala.tools.nsc.interpreter.IMain.interpret(IMain.scala:569)
	at scala.tools.nsc.interpreter.IMain.interpret(IMain.scala:565)
	at scala.tools.nsc.interpreter.ILoop.interpretStartingWith(ILoop.scala:807)
	at scala.tools.nsc.interpreter.ILoop.command(ILoop.scala:681)
	at scala.tools.nsc.interpreter.ILoop.processLine(ILoop.scala:395)
	at org.apache.spark.repl.SparkILoop$
$anonfun$initializeSpark$1.apply$mcV$sp(SparkILoop.scala:38)
	at org.apache.spark.repl.SparkILoop$$anonfun$initializeSpark$1.apply(SparkILoop.scala:37)
	at org.apache.spark.repl.SparkILoop$$anonfun$initializeSpark$1.apply(SparkILoop.scala:37)
	at scala.tools.nsc.interpreter.IMain.beQuietDuring(IMain.scala:214)
	at org.apache.spark.repl.SparkILoop.initializeSpark(SparkILoop.scala:37)
	at org.apache.spark.repl.SparkILoop.loadFiles(SparkILoop.scala:105)
	at scala.tools.nsc.interpreter.ILoop$$anonfun$process$1.apply$mcZ$sp(ILoop.scala:920)
	at scala.tools.nsc.interpreter.ILoop$$anonfun$process$1.apply(ILoop.scala:909)
	at scala.tools.nsc.interpreter.ILoop$$anonfun$process$1.apply(ILoop.scala:909)
	at
scala.reflect.internal.util.ScalaClassLoader$.savingContextLoader(ScalaClassLoader.scala:97)
	at scala.tools.nsc.interpreter.ILoop.process(ILoop.scala:909)
	at org.apache.spark.repl.Main$.doMain(Main.scala:68)
	at org.apache.spark.repl.Main$.main(Main.scala:51)
	at org.apache.spark.repl.Main.main(Main.scala)
	at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
	at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)
	at
sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)
	at java.lang.reflect.Method.invoke(Method.java:498)
	at org.apache.spark.deploy.SparkSubmit$.org$apache$spark$deploy$SparkSubmit$
$runMain(SparkSubmit.scala:738)
	at org.apache.spark.deploy.SparkSubmit$.doRunMain$1(SparkSubmit.scala:187)
	at org.apache.spark.deploy.SparkSubmit$.submit(SparkSubmit.scala:212)
	at org.apache.spark.deploy.SparkSubmit$.main(SparkSubmit.scala:126)
```

```
        at org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)
18/08/27 13:55:49 WARN metastore.ObjectStore: Failed to get database global_temp, returning
NoSuchObjectException
Spark context Web UI available at http://10.0.2.15:4040
Spark context available as 'sc' (master = local[*], app id = local-1535358325259).
Spark session available as 'spark'.
Welcome to
      ____              __
     / __/__  ___ _____/ /__
    _\ \/ _ \/ _ `/ __/  '_/
   /___/ .__/\_,_/_/ /_/\_\   version 2.1.0
      /_/

Using Scala version 2.11.8 (Java HotSpot(TM) Client VM, Java 1.8.0_171)
Type in expressions to have them evaluated.
Type :help for more information.

scala> val x = sc.parallelize(List[Int] (1, 2, 3, 4, 5, 6, 7, 8, 9, 10))
x: org.apache.spark.rdd.RDD[Int] = ParallelCollectionRDD[0] at parallelize at <console>:24
```

Problem 1:

- find the sum of all numbers

Terminal Execution :

```
scala> x.sum
18/08/27 14:22:31 WARN util.SizeEstimator: Failed to check whether UseCompressedOops
is set; assuming yes
res0: Double = 55.0
```

Problem 2:

- find the total elements in the list

Terminal Execution :

```
scala> x.count
res1: Long = 10
```

Problem 3:

- calculate the average of the numbers in the list

Terminal Execution :

```
scala> val avg = x.sum / x.count
avg: Double = 5.5
```

Problem 4:

- Find the sum of all the even numbers in the list

Terminal Execution :

```
scala> val even = x.filter(num => num%2 == 0)
even: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[3] at filter at <console>:26

scala> even.collect
res2: Array[Int] = Array(2, 4, 6, 8, 10)

scala> even.sum
res3: Double = 30.0
```

Problem 5:

- Find the total number of elements in the list divisible by both 5 and 3

Terminal Execution :

```
scala> val div = x.filter(num => (num%3 ==0)&&(num%5==0))
div: org.apache.spark.rdd.RDD[Int] = MapPartitionsRDD[5] at filter at <console>:26

scala> div.collect
res4: Array[Int] = Array()
scala> div.count
res6: Long = 0
```

Task 2

Question 1:

Pen down the limitations of MapReduce.

Solution:

1. Issue with Small Files

Hadoop is not suited for small data. (HDFS) Hadoop distributed file system lacks the ability to efficiently support the random reading of small files because of its high capacity design.

2. Slow Processing Speed

In Hadoop, with a parallel and distributed algorithm, MapReduce process large data sets. There are tasks that need to be performed: Map and Reduce and, MapReduce requires a lot of time to perform these tasks thereby increasing latency. Data is distributed and processed over the cluster in MapReduce which increases the time and reduces processing speed.

3. Support for Batch Processing only

Hadoop supports batch processing only, it does not process streamed data, and hence overall performance is slower. MapReduce framework of Hadoop does not leverage the memory of the Hadoop cluster to the maximum.

4. No Real-time Data Processing

Apache Hadoop is designed for batch processing, that means it take a huge amount of data in input, process it and produce the result. Although batch processing is very efficient for processing a high volume of data, but depending on the size of the data being processed and computational power of the system, an output can be delayed significantly. Hadoop is not suitable for Real-time data processing.

5. No Delta Iteration

Hadoop is not so efficient for iterative processing, as Hadoop does not support cyclic data flow(i.e. a chain of stages in which each output of the previous stage is the input to the next stage).

6. Latency

In Hadoop, MapReduce framework is comparatively slower, since it is designed to support different format, structure and huge volume of data. In MapReduce, Map takes a set of data and converts it into another set of data, where individual element are broken down into key value pair and Reduce takes the output from the map as input and process further and MapReduce requires a lot of time to perform these tasks thereby increasing latency.

7. Not Easy to Use

In Hadoop, MapReduce developers need to hand code for each and every operation which makes it very difficult to work. MapReduce has no interactive mode, but adding one such as hive and pig makes working with MapReduce a little easier for adopters.

8. No Caching

Hadoop is not efficient for caching. In Hadoop, MapReduce cannot cache the intermediate data in memory for a further requirement which diminishes the performance of Hadoop.

Question 2:

What is RDD? Explain few features of RDD?

Solution:

RDD :

1. RDD (Resilient Distributed Dataset) is the fundamental data structure of Apache Spark which are an immutable collection of objects which computes on the different node of the cluster. Each and every dataset in Spark RDD is logically partitioned across many servers so that they can be computed on different nodes of the cluster.

2. Apache Spark should be installed first in pseudo distributed mode or in a multi-node cluster to play with RDD.

3. Decomposing the name RDD:

3.1. **Resilient :** It is referred to as fault-tolerant with the help of RDD lineage graph(DAG) and so able to recompute missing or damaged partitions due to node failures.

3.2. **Distributed :** It means that Data resides on multiple nodes.

3.3. Dataset represents records of the data you work with. The user can load the data set externally which can be either JSON file, CSV file, text file or database via JDBC with no specific data structure.

Hence, each and every dataset in RDD is logically partitioned across many servers so that they can be computed on different nodes of the cluster. RDDs are fault tolerant i.e. It posses self-recovery in the case of failure.

4. There are three ways to create RDDs in Spark such as –

4.1. Data in stable storage,
4.2. Data in other RDDs
4.3. Parallelizing already existing collection in driver program. One can also operate Spark RDDs in parallel with a low-level API that offers transformations and actions.

5. Spark RDD can also be cached and manually partitioned. Caching is beneficial when we use RDD several times. And manual partitioning is important to correctly balance partitions. Generally, smaller partitions allow distributing RDD data more equally, among more executors. Hence, fewer partitions make the work easy.

6. Programmers can also call a persist method to indicate which RDDs they want to reuse in future operations. Spark keeps persistent RDDs in memory by default, but it can spill them to disk if there is not enough RAM. Users can also request other persistence strategies, such as storing the RDD only on disk or replicating it across machines, through flags to persist.

7. **The Purpose of using RDD are as follows :**

The key motivations behind the concept of RDD are-

Iterative algorithms.
Interactive data mining tools.

DSM (Distributed Shared Memory) is a very general abstraction, but this generality makes it harder to implement in an efficient and fault tolerant manner on commodity clusters. Here the need of RDD comes into the picture.

In distributed computing system data is stored in intermediate stable distributed store such as HDFS or Amazon S3. This makes the computation of job slower since it involves many IO operations, replications, and serializations in the process.

In first two cases we keep data in-memory, it can improve performance by an order of magnitude.

The main challenge in designing RDD is defining a program interface that provides fault tolerance efficiently. To achieve fault tolerance efficiently, RDDs provide a restricted form of shared memory, based on coarse-grained transformation rather than fine-grained updates to shared state.

Spark exposes RDD through language integrated API. In integrated API each data set is represented as an object and transformation is involved using the method of these objects.

Apache Spark evaluates RDDs lazily. It is called when needed, which saves lots of time and improves efficiency. The first time they are used in an action so that it can pipeline the transformation. Also, the programmer can call a persist method to state which RDD they want to use in future operations.

8. RDD vs DSM (Distributed Shared Memory)

In this Spark RDD tutorial, we are going to get to know the difference between RDD and DSM which will take RDD in Apache Spark into the limelight.

8.1. Read

RDD – The read operation in RDD is either coarse grained or fine grained. Coarse-grained meaning we can transform the whole dataset but not an individual element on the dataset. While fine-grained means we can transform individual element on the dataset.
DSM – The read operation in Distributed shared memory is fine-grained.

8.2. Write

RDD – The write operation in RDD is coarse grained.
DSM – The Write operation is fine grained in distributed shared system.

8.3. Consistency

RDD – The consistency of RDD is trivial meaning it is immutable in nature. Any changes on RDD is permanent i.e we can not realtor the content of RDD. So the level of consistency is high.
DSM – In Distributed Shared Memory the system guarantees that if the programmer follows the rules, the memory will be consistent and the results of memory operations will be predictable.

8.4. Fault-Recovery Mechanism

RDD – The lost data can be easily recovered in Spark RDD using lineage graph at any moment. Since for each transformation, new RDD is formed and RDDs are immutable in nature so it is easy to recover.
DSM – Fault tolerance is achieved by a checkpointing technique which allows applications to roll back to a recent checkpoint rather than restarting.

8.5. Straggler Mitigation

Stragglers, in general, are those that take more time to complete than their peers. This could happen due to many reasons such as load imbalance, I/O blocks, garbage collections, etc.

The problem with stragglers is that when the parallel computation is followed by synchronizations such as reductions. This would cause all the parallel tasks to wait for others.

RDD – In RDD it is possible to mitigate stragglers using backup task.
DSM – It is quite difficult to achieve straggler mitigation.

8.6. Behavior if not enough RAM

RDD – If there is not enough space to store RDD in RAM then the RDDs are shifted to disk.
DSM – In this type of system, the performance decreases if the RAM runs out of storage.

9. The Features of RDD in Spark are as follows :

9.1. In-memory Computation

Spark RDDs have a provision of in-memory computation. It stores intermediate results in distributed memory(RAM) instead of stable storage(disk).

9.2. Lazy Evaluations

All transformations in Apache Spark are lazy, in that they do not compute their results right away. Instead, they just remember the transformations applied to some base data set.

Spark computes transformations when an action requires a result for the driver

program.

9.3. Fault Tolerance

Spark RDDs are fault tolerant as they track data lineage information to rebuild lost data automatically on failure. They rebuild lost data on failure using lineage, each RDD remembers how it was created from other datasets (by transformations like a map, join or groupBy) to recreate itself.

9.4. Immutability

Data is safe to share across processes. It can also be created or retrieved anytime which makes caching, sharing & replication easy. Thus, it is a way to reach consistency in computations.

9.5. Partitioning

Partitioning is the fundamental unit of parallelism in Spark RDD. Each partition is one logical division of data which is mutable. One can create a partition through some transformations on existing partitions.

9.6. Persistence

Users can state which RDDs they will reuse and choose a storage strategy for them (e.g., in-memory storage or on Disk).

9.7. Coarse-grained Operations

It applies to all elements in datasets through maps or filter or group by operation.

9.8. Location-Stickiness

RDDs are capable of defining placement preference to compute partitions. Placement preference refers to information about the location of RDD. The DAGScheduler places the partitions in such a way that task is close to data as much as possible. Thus, speed up computation.

Question 3:

List down few Spark RDD operations and explain each of them.

Solution:

RDD in Apache Spark supports two types of operations:

- **Transformation**
- **Actions**

1. Transformations

1.1. Spark RDD Transformations are functions that take an RDD as the input and produce one or many RDDs as the output. They do not change the input RDD (since RDDs are immutable and hence one cannot change it), but always produce one or more new RDDs by applying the computations they represent e.g. Map(), filter(), reduceByKey() etc.

1.2. Transformations are lazy operations on an RDD in Apache Spark. It creates one or many new RDDs, which executes when an Action occurs. Hence, Transformation creates a new dataset from an existing one.

1.3. Certain transformations can be pipelined which is an optimization method, that Spark uses to improve the performance of computations. There are two kinds of transformations: narrow transformation, wide transformation.

1.3.1. Narrow Transformations

It is the result of map, filter and such that the data is from a single partition only, i.e. it is self-sufficient. An output RDD has partitions with records that originate from a single partition in the parent RDD. Only a limited subset of partitions used to calculate the result.

Spark groups narrow transformations as a stage known as pipelining.
Spark RDD - Narrow Transformation

1.3.2. Wide Transformations

It is the result of groupByKey() and reduceByKey() like functions. The data required to compute the records in a single partition may live in many partitions of the parent RDD. Wide transformations are also known as shuffle transformations because they may or may not

depend on a shuffle.

1.4. Various functions in RDD transformation :

1. map(func)

The map function iterates over every line in RDD and split into new RDD. Using **map()** transformation we take in any function, and that function is applied to every element of RDD.

In the map, we have the flexibility that the input and the return type of RDD may differ from each other.

For example, in RDD {1, 2, 3, 4, 5} if we apply “rdd.map(x=>x+2)” we will get the result as (3, 4, 5, 6, 7).

2. flatMap()

With the help of **flatMap()** function, to each input element, we have many elements in an output RDD. The most simple use of flatMap() is to split each input string into words.

Key difference between map() and flatMap() is map() returns only one element, while flatMap() can return a list of elements.

3.3. filter(func)

Spark RDD **filter()** function returns a new RDD, containing only the elements that meet a predicate. It is a *narrow operation* because it does not shuffle data from one partition to many partitions.

2. Actions

2.1. An Action in Spark returns final result of RDD computations. It triggers execution using lineage graph to load the data into original RDD, carry out all intermediate transformations and return final results to Driver program or write it out to file system. Lineage graph is dependency graph of all parallel RDDs of RDD.

2.2. Actions are RDD operations that produce non-RDD values. They materialize a value in a Spark program. An Action is one of the ways to send result from executors to the driver. First(), take(), reduce(), collect(), the count() is some of the Actions in spark.

2.3. Using transformations, one can create RDD from the existing one. But when we want to work with the actual dataset, at that point we use Action. When the Action occurs it does not create the new RDD, unlike transformation. Thus, actions are RDD operations that give no RDD values. Action stores its value either to drivers or to the external storage system. It brings laziness of RDD into motion.