Assignment 21: Spark SQL II Assignment Problems

<u>Initial Terminal Execution:</u>

```
[acadgild@localhost ~]$ jps
3383 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
18/09/04 00:30:08 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.o
ut
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.loc
aldomain.out
18/09/04 00:30:41 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.localdom
ain.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
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
ingMetaStoreClient.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$.main(SparkSubmit.scala:126)
	at org.apache.spark.deploy.SparkSubmit.main(SparkSubmit.scala)
Caused by: com.mysql.jdbc.exceptions.jdbc4.MySQLSyntaxErrorException: Specified key was too
long; max key length is 3072 bytes
	at sun.reflect.NativeConstructorAccessorImpl.newInstance0(Native Method)
	at
sun.reflect.NativeConstructorAccessorImpl.newInstance(NativeConstructorAccessorImpl.java:62)
	at
sun.reflect.DelegatingConstructorAccessorImpl.newInstance(DelegatingConstructorAccessorImpl.ja
va:45)
	at java.lang.reflect.Constructor.newInstance(Constructor.java:423)
	at com.mysql.jdbc.Util.handleNewInstance(Util.java:425)
	at com.mysql.jdbc.Util.getInstance(Util.java:408)
	at com.mysql.jdbc.SQLError.createSQLException(SQLError.java:944)
	at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3976)
	at com.mysql.jdbc.MysqlIO.checkErrorPacket(MysqlIO.java:3912)
	at com.mysql.jdbc.MysqlIO.sendCommand(MysqlIO.java:2530)
	at com.mysql.jdbc.MysqlIO.sqlQueryDirect(MysqlIO.java:2683)
	at com.mysql.jdbc.ConnectionImpl.execSQL(ConnectionImpl.java:2482)
	at com.mysql.jdbc.ConnectionImpl.execSQL(ConnectionImpl.java:2440)
	at com.mysql.jdbc.StatementImpl.executeInternal(StatementImpl.java:845)
	at com.mysql.jdbc.StatementImpl.execute(StatementImpl.java:745)
	at com.jolbox.bonecp.StatementHandle.execute(StatementHandle.java:254)
	at
org.datanucleus.store.rdbms.table.AbstractTable.executeDdlStatement(AbstractTable.java:760)
	at org.datanucleus.store.rdbms.table.TableImpl.createIndices(TableImpl.java:648)
	at org.datanucleus.store.rdbms.table.TableImpl.createConstraints(TableImpl.java:422)
	at
org.datanucleus.store.rdbms.RDBMSStoreManager$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)
18/09/09 12:31:58 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-1536476498992).
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.
```

Data Set Description :

Sports: firstname, lastname, sports, medal_type, age, year, country

Data Sets Present:

1. Sports_data.txt :

Terminal Execution :

```
[acadgild@localhost ~]$ cat /home/acadgild/Desktop/Sports_data.txt
firstname,lastname,sports,medal_type,age,year,country
lisa,cudrow,javellin,gold,34,2015,USA
mathew,louis,javellin,gold,34,2015,RUS
michael,phelps,swimming,silver,32,2016,USA
usha,pt,running,silver,30,2016,IND
serena,williams,running,gold,31,2014,FRA
roger,federer,tennis,silver,32,2016,CHN
jenifer,cox,swimming,silver,32,2014,IND
fernando,johnson,swimming,silver,32,2016,CHN
lisa,cudrow,javellin,gold,34,2017,USA
mathew,louis,javellin,gold,34,2015,RUS
michael,phelps,swimming,silver,32,2017,USA
usha,pt,running,silver,30,2014,IND
serena,williams,running,gold,31,2016,FRA
roger,federer,tennis,silver,32,2017,CHN
jenifer,cox,swimming,silver,32,2014,IND
fernando,johnson,swimming,silver,32,2017,CHN
lisa,cudrow,javellin,gold,34,2014,USA
mathew,louis,javellin,gold,34,2014,RUS
michael,phelps,swimming,silver,32,2017,USA
usha,pt,running,silver,30,2014,IND
serena,williams,running,gold,31,2016,FRA
roger,federer,tennis,silver,32,2014,CHN
jenifer,cox,swimming,silver,32,2017,IND
fernando,johnson,swimming,silver,32,2017,CHN
```

Problem Statement

Task 1

Using spark-sql, Find:

1. What are the total number of gold medal winners every year

2. How many silver medals have been won by USA in each sport

Task 2

Using udfs on dataframe

1. Change firstname, lastname columns into
Mr.first_two_letters_of_firstname<space>lastname
for example - michael, phelps becomes Mr.mi phelps

2. Add a new column called ranking using udfs on dataframe, where :
gold medalist, with age >= 32 are ranked as pro
gold medalists, with age <= 31 are ranked amateur
silver medalist, with age >= 32 are ranked as expert
silver medalists, with age <= 31 are ranked rookie

--==

Task 1

1. What are the total number of gold medal winners every year

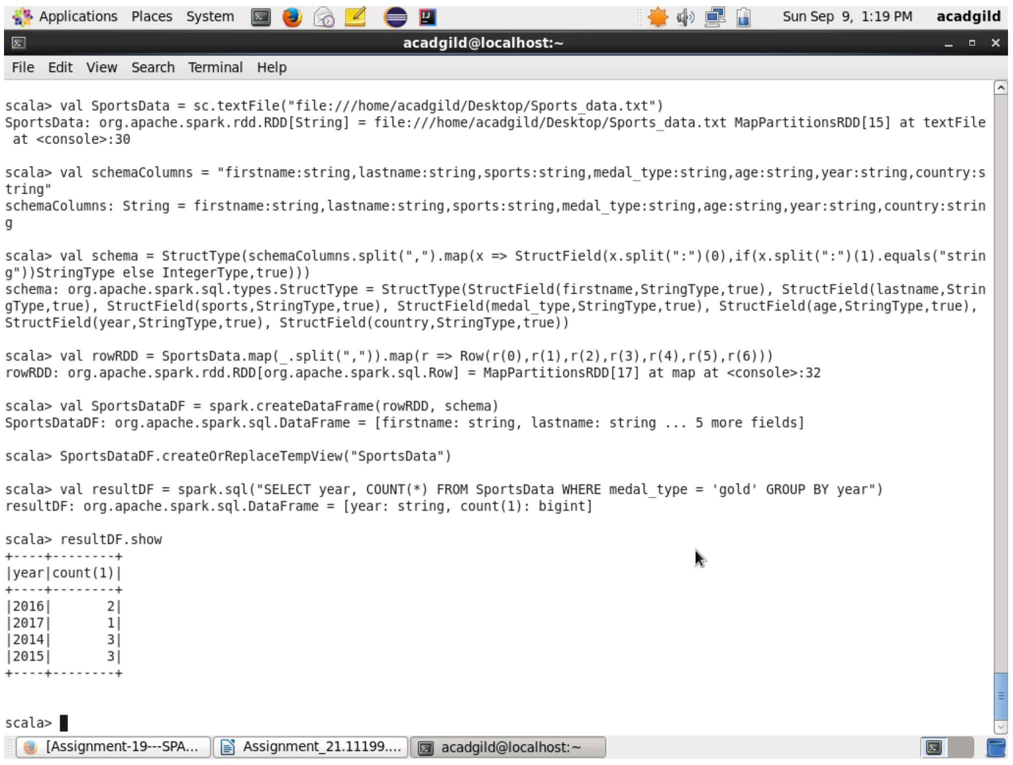
Terminal Execution :

```
scala> import org.apache.spark.sql.Row
import org.apache.spark.sql.Row

scala> import org.apache.spark.sql.types.{StructType,
StructField,StringType,NumericType,IntegerType}
import org.apache.spark.sql.types.{StructType, StructField, StringType, NumericType, IntegerType}
scala> val SportsData = sc.textFile("file:///home/acadgild/Desktop/Sports_data.txt")
SportsData: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Desktop/Sports_data.txt
```

```
MapPartitionsRDD[3] at textFile at <console>:28

scala> SportsData.foreach(println)
firstname,lastname,sports,medal_type,age,year,country
lisa,cudrow,javellin,gold,34,2015,USA
mathew,louis,javellin,gold,34,2015,RUS
michael,phelps,swimming,silver,32,2016,USA
usha,pt,running,silver,30,2016,IND
serena,williams,running,gold,31,2014,FRA
roger,federer,tennis,silver,32,2016,CHN
jenifer,cox,swimming,silver,32,2014,IND
fernando,johnson,swimming,silver,32,2016,CHN
lisa,cudrow,javellin,gold,34,2017,USA
mathew,louis,javellin,gold,34,2015,RUS
michael,phelps,swimming,silver,32,2017,USA
usha,pt,running,silver,30,2014,IND
serena,williams,running,gold,31,2016,FRA
roger,federer,tennis,silver,32,2017,CHN
jenifer,cox,swimming,silver,32,2014,IND
fernando,johnson,swimming,silver,32,2017,CHN
lisa,cudrow,javellin,gold,34,2014,USA
mathew,louis,javellin,gold,34,2014,RUS
michael,phelps,swimming,silver,32,2017,USA
usha,pt,running,silver,30,2014,IND
serena,williams,running,gold,31,2016,FRA
roger,federer,tennis,silver,32,2014,CHN
jenifer,cox,swimming,silver,32,2017,IND
fernando,johnson,swimming,silver,32,2017,CHN

scala> val schemaColumns =
"firstname:string,lastname:string,sports:string,medal_type:string,age:string,year:string,country:string
"
schemaColumns: String =
firstname:string,lastname:string,sports:string,medal_type:string,age:string,year:string,country:string

scala> val schema = StructType(schemaColumns.split(",").map(x => StructField(x.split(":")
(0),if(x.split(":")(1).equals("string"))StringType else IntegerType,true)))
schema: org.apache.spark.sql.types.StructType = StructType(StructField(firstname,StringType,true),
StructField(lastname,StringType,true), StructField(sports,StringType,true),
StructField(medal_type,StringType,true), StructField(age,StringType,true),
StructField(year,StringType,true), StructField(country,StringType,true))
scala> val rowRDD = SportsData.map(_.split(",")).map(r => Row(r(0),r(1),r(2),r(3),r(4),r(5),r(6)))


rowRDD: org.apache.spark.rdd.RDD[org.apache.spark.sql.Row] = MapPartitionsRDD[5] at map at
<console>:30

scala> val SportsDataDF = spark.createDataFrame(rowRDD, schema)
```

```
SportsDataDF: org.apache.spark.sql.DataFrame = [firstname: string, lastname: string ... 5 more
fields]

scala> SportsDataDF.createOrReplaceTempView("SportsData")

scala> val resultDF = spark.sql("SELECT year, COUNT(*) FROM SportsData WHERE medal_type
= 'gold' GROUP BY year")
resultDF: org.apache.spark.sql.DataFrame = [year: string, count(1): bigint]

scala> SportsDataDF.createOrReplaceTempView("SportsData")

scala> val resultDF = spark.sql("SELECT year, COUNT(*) FROM SportsData WHERE medal_type
= 'gold' GROUP BY year")
resultDF: org.apache.spark.sql.DataFrame = [year: string, count(1): bigint]

scala> resultDF.show
+----+--------+
|year|count(1)|
+----+--------+
|2016|       2|
|2017|       1|
|2014|       3|
|2015|       3|
+----+--------+
```

<u>Output :</u>


Applications Places System
Sun Sep 9, 1:19 PM acadgild
acadgild@localhost:~
File Edit View Search Terminal Help
[Assignment-19---SPA...
Assignment_21.11199....
acadgild@localhost:~

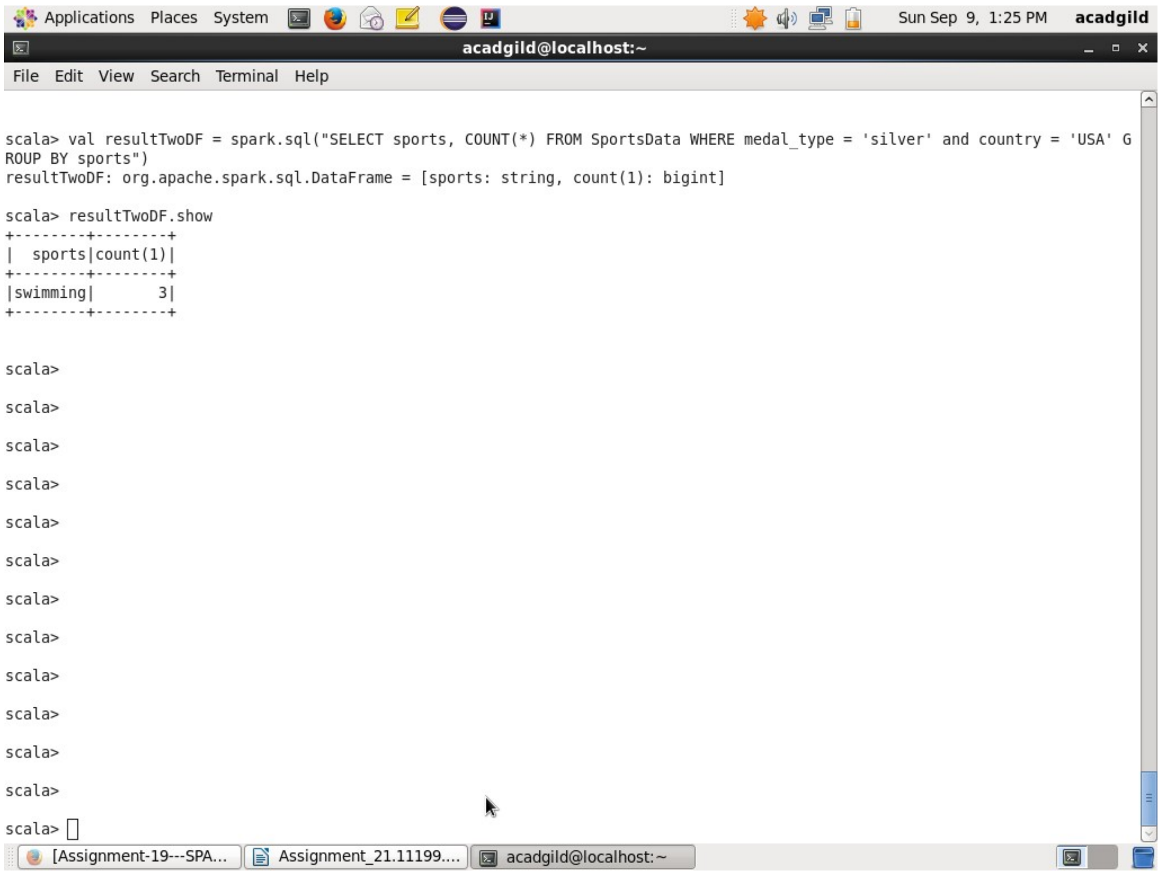
```
scala> val SportsData = sc.textFile("file:///home/acadgild/Desktop/Sports_data.txt")
SportsData: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Desktop/Sports_data.txt MapPartitionsRDD[15] at textFile
 at <console>:30

scala> val schemaColumns = "firstname:string,lastname:string,sports:string,medal_type:string,age:string,year:string,country:s
tring"
schemaColumns: String = firstname:string,lastname:string,sports:string,medal_type:string,age:string,year:string,country:strin
g

scala> val schema = StructType(schemaColumns.split(",").map(x => StructField(x.split(":")(0),if(x.split(":")(1).equals("strin
g"))StringType else IntegerType,true)))
schema: org.apache.spark.sql.types.StructType = StructType(StructField(firstname,StringType,true), StructField(lastname,Strin
gType,true), StructField(sports,StringType,true), StructField(medal_type,StringType,true), StructField(age,StringType,true),
StructField(year,StringType,true), StructField(country,StringType,true))

scala> val rowRDD = SportsData.map(_.split(",")).map(r => Row(r(0),r(1),r(2),r(3),r(4),r(5),r(6)))
rowRDD: org.apache.spark.rdd.RDD[org.apache.spark.sql.Row] = MapPartitionsRDD[17] at map at <console>:32

scala> val SportsDataDF = spark.createDataFrame(rowRDD, schema)
SportsDataDF: org.apache.spark.sql.DataFrame = [firstname: string, lastname: string ... 5 more fields]

scala> SportsDataDF.createOrReplaceTempView("SportsData")

scala> val resultDF = spark.sql("SELECT year, COUNT(*) FROM SportsData WHERE medal_type = 'gold' GROUP BY year")
resultDF: org.apache.spark.sql.DataFrame = [year: string, count(1): bigint]

scala> resultDF.show
+----+--------+
|year|count(1)|
+----+--------+
|2016|       2|
|2017|       1|
|2014|       3|
|2015|       3|
+----+--------+


scala> 
```

2. How many silver medals have been won by USA in each sport

Terminal Execution :

```
scala> val resultTwoDF = spark.sql("SELECT sports, COUNT(*) FROM SportsData
WHERE medal_type = 'silver' and country = 'USA' GROUP BY sports")
resultTwoDF: org.apache.spark.sql.DataFrame = [sports: string, count(1): bigint]

scala> resultTwoDF.show
+--------+--------+
|  sports|count(1)|
+--------+--------+
|swimming|       3|
+--------+--------+
```

Output :


Applications Places System
Sun Sep 9, 1:25 PM acadgild
acadgild@localhost:~
File Edit View Search Terminal Help
[Assignment-19---SPA...
Assignment_21.11199....
acadgild@localhost:~

```
scala> val resultTwoDF = spark.sql("SELECT sports, COUNT(*) FROM SportsData WHERE medal_type = 'silver' and country = 'USA' G
ROUP BY sports")
resultTwoDF: org.apache.spark.sql.DataFrame = [sports: string, count(1): bigint]

scala> resultTwoDF.show
+--------+--------+
|  sports|count(1)|
+--------+--------+
|swimming|       3|
+--------+--------+


scala>

scala>

scala>

scala>

scala>

scala>

scala>

scala>

scala>

scala>

scala>

scala>

scala> 
```

Task 2

Using udfs on dataframe

1. Change firstname, lastname columns into Mr.first_two_letters_of_firstname<space>lastname

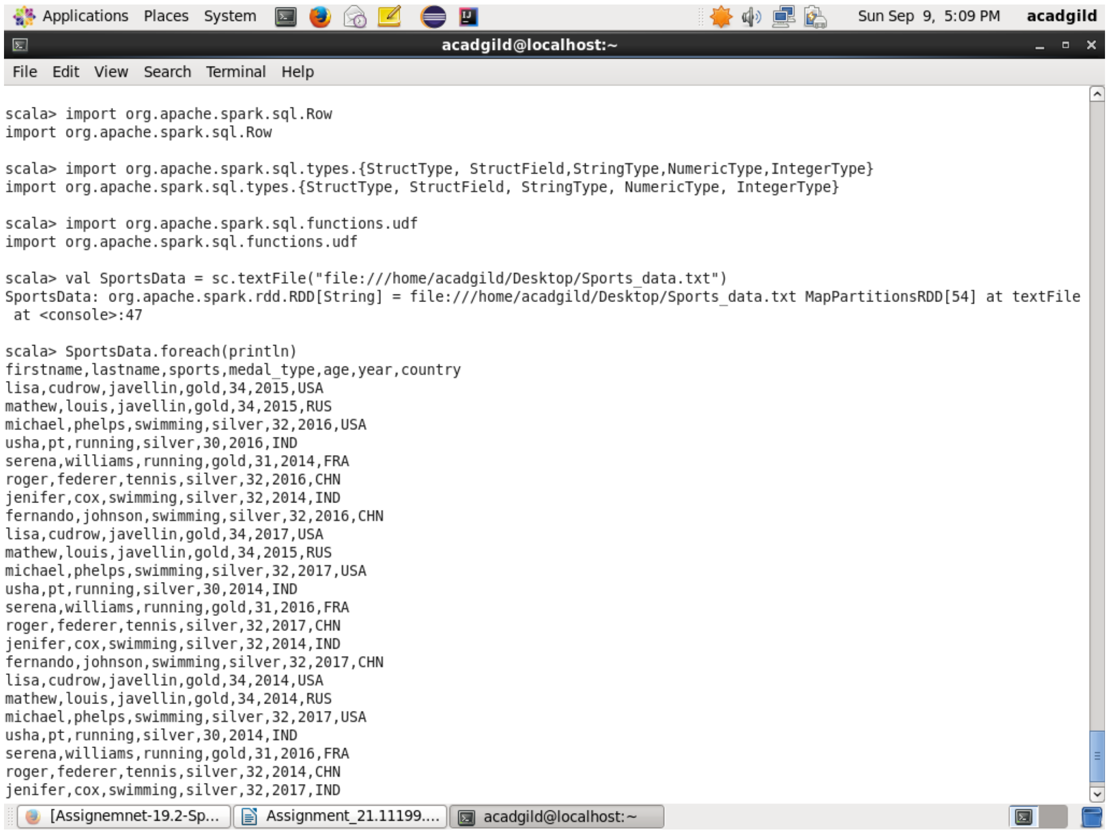
for example - michael, phelps becomes Mr.mi phelps

Terminal Execution :

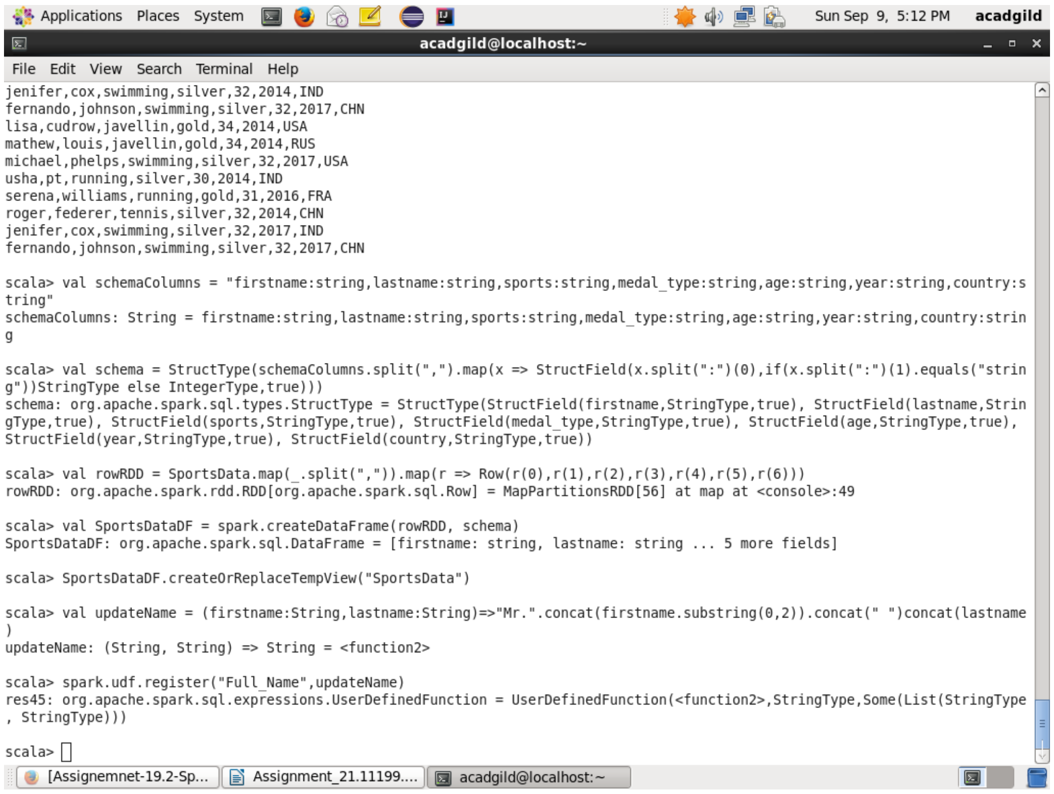
```
scala> import org.apache.spark.sql.Row
import org.apache.spark.sql.Row

scala> import org.apache.spark.sql.types.{StructType,
StructField,StringType,NumericType,IntegerType}
import org.apache.spark.sql.types.{StructType, StructField, StringType, NumericType, IntegerType}

scala> import org.apache.spark.sql.functions.udf
import org.apache.spark.sql.functions.udf

scala> val SportsData = sc.textFile("file:///home/acadgild/Desktop/Sports_data.txt")
SportsData: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Desktop/Sports_data.txt
MapPartitionsRDD[48] at textFile at <console>:44

scala> SportsData.foreach(println)
firstname,lastname,sports,medal_type,age,year,country
lisa,cudrow,javellin,gold,34,2015,USA
mathew,louis,javellin,gold,34,2015,RUS
michael,phelps,swimming,silver,32,2016,USA
usha,pt,running,silver,30,2016,IND
serena,williams,running,gold,31,2014,FRA
roger,federer,tennis,silver,32,2016,CHN
jenifer,cox,swimming,silver,32,2014,IND
fernando,johnson,swimming,silver,32,2016,CHN
lisa,cudrow,javellin,gold,34,2017,USA
mathew,louis,javellin,gold,34,2015,RUS
michael,phelps,swimming,silver,32,2017,USA
usha,pt,running,silver,30,2014,IND
serena,williams,running,gold,31,2016,FRA
roger,federer,tennis,silver,32,2017,CHN
jenifer,cox,swimming,silver,32,2014,IND
fernando,johnson,swimming,silver,32,2017,CHN
lisa,cudrow,javellin,gold,34,2014,USA
mathew,louis,javellin,gold,34,2014,RUS
michael,phelps,swimming,silver,32,2017,USA
```

```
usha,pt,running,silver,30,2014,IND
serena,williams,running,gold,31,2016,FRA
roger,federer,tennis,silver,32,2014,CHN
jenifer,cox,swimming,silver,32,2017,IND
fernando,johnson,swimming,silver,32,2017,CHN

scala> val schemaColumns =
"firstname:string,lastname:string,sports:string,medal_type:string,age:string,year:string,country:string
"
schemaColumns: String =
firstname:string,lastname:string,sports:string,medal_type:string,age:string,year:string,country:string

scala> val schema = StructType(schemaColumns.split(",").map(x => StructField(x.split(":")
(0),if(x.split(":")(1).equals("string"))StringType else IntegerType,true)))
schema: org.apache.spark.sql.types.StructType = StructType(StructField(firstname,StringType,true),
StructField(lastname,StringType,true), StructField(sports,StringType,true),
StructField(medal_type,StringType,true), StructField(age,StringType,true),
StructField(year,StringType,true), StructField(country,StringType,true))

scala> val rowRDD = SportsData.map(_.split(",")).map(r => Row(r(0),r(1),r(2),r(3),r(4),r(5),r(6)))
rowRDD: org.apache.spark.rdd.RDD[org.apache.spark.sql.Row] = MapPartitionsRDD[50] at map at
<console>:46

scala> val SportsDataDF = spark.createDataFrame(rowRDD, schema)
SportsDataDF: org.apache.spark.sql.DataFrame = [firstname: string, lastname: string ... 5 more
fields]

scala> SportsDataDF.createOrReplaceTempView("SportsData")

scala> val updateName =
(firstname:String,lastname:String)=>"Mr.".concat(firstname.substring(0,2)).concat("
")concat(lastname)
updateName: (String, String) => String = <function2>

scala> spark.udf.register("Full_Name",updateName)
res45: org.apache.spark.sql.expressions.UserDefinedFunction =
UserDefinedFunction(<function2>,StringType,Some(List(StringType, StringType)))

scala> val fname = spark.sql("SELECT Full_Name(firstname,lastname) FROM SportsData").show()
+-----------------------+
|UDF(firstname, lastname)|
+-----------------------+
|        Mr.fi lastname|
|          Mr.li cudrow|
|           Mr.ma louis|
|          Mr.mi phelps|
|              Mr.us pt|
|        Mr.se williams|
|         Mr.ro federer|
|             Mr.je cox|
|          Mr.fe johnson|
|           Mr.li cudrow|
|            Mr.ma louis|
|           Mr.mi phelps|
|              Mr.us pt|
|         Mr.se williams|
|          Mr.ro federer|
|             Mr.je cox|
|          Mr.fe johnson|
|           Mr.li cudrow|
|            Mr.ma louis|
|           Mr.mi phelps|
+------------------------+
only showing top 20 rows
```

Output :

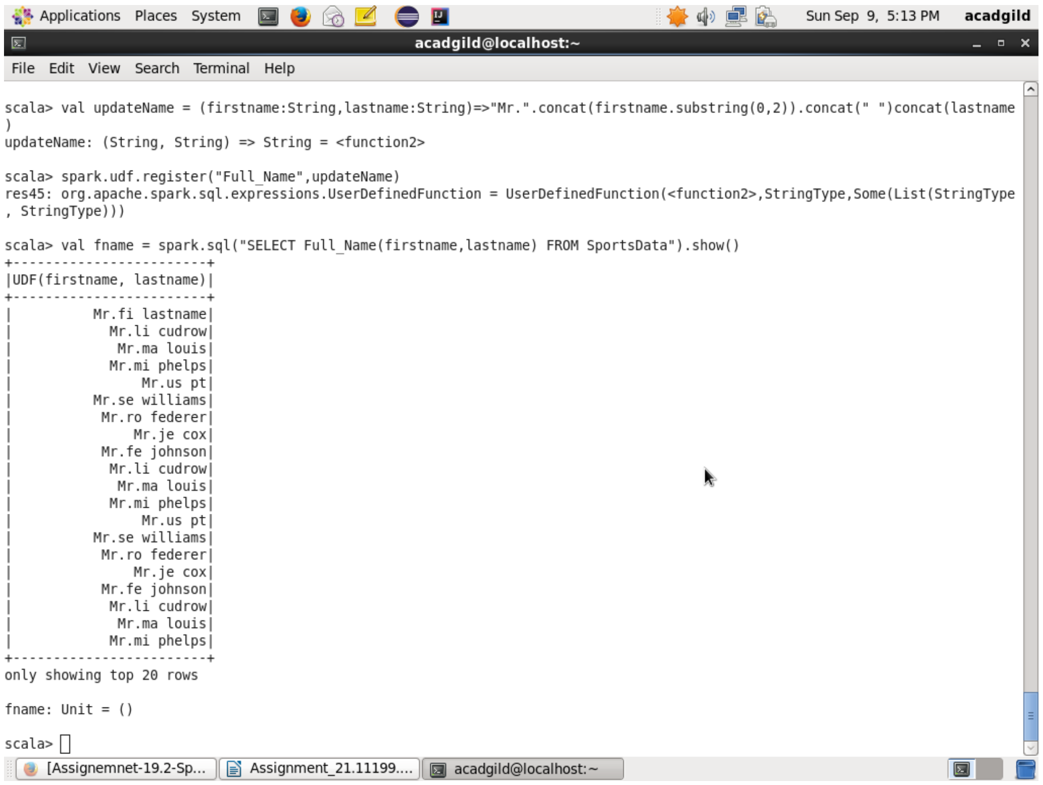


```
scala> import org.apache.spark.sql.Row
import org.apache.spark.sql.Row

scala> import org.apache.spark.sql.types.{StructType, StructField,StringType,NumericType,IntegerType}
import org.apache.spark.sql.types.{StructType, StructField, StringType, NumericType, IntegerType}

scala> import org.apache.spark.sql.functions.udf
import org.apache.spark.sql.functions.udf

scala> val SportsData = sc.textFile("file:///home/acadgild/Desktop/Sports_data.txt")
SportsData: org.apache.spark.rdd.RDD[String] = file:///home/acadgild/Desktop/Sports_data.txt MapPartitionsRDD[54] at textFile
 at <console>:47

scala> SportsData.foreach(println)
firstname,lastname,sports,medal_type,age,year,country
lisa,cudrow,javellin,gold,34,2015,USA
mathew,louis,javellin,gold,34,2015,RUS
michael,phelps,swimming,silver,32,2016,USA
usha,pt,running,silver,30,2016,IND
serena,williams,running,gold,31,2014,FRA
roger,federer,tennis,silver,32,2016,CHN
jenifer,cox,swimming,silver,32,2014,IND
fernando,johnson,swimming,silver,32,2016,CHN
lisa,cudrow,javellin,gold,34,2017,USA
mathew,louis,javellin,gold,34,2015,RUS
michael,phelps,swimming,silver,32,2017,USA
usha,pt,running,silver,30,2014,IND
serena,williams,running,gold,31,2016,FRA
roger,federer,tennis,silver,32,2017,CHN
jenifer,cox,swimming,silver,32,2014,IND
fernando,johnson,swimming,silver,32,2017,CHN
lisa,cudrow,javellin,gold,34,2014,USA
mathew,louis,javellin,gold,34,2014,RUS
michael,phelps,swimming,silver,32,2017,USA
usha,pt,running,silver,30,2014,IND
serena,williams,running,gold,31,2016,FRA
roger,federer,tennis,silver,32,2014,CHN
jenifer,cox,swimming,silver,32,2017,IND
```


Applications Places System
Sun Sep 9, 5:12 PM acadgild
acadgild@localhost:~
File Edit View Search Terminal Help

```
jenifer,cox,swimming,silver,32,2014,IND
fernando,johnson,swimming,silver,32,2017,CHN
lisa,cudrow,javellin,gold,34,2014,USA
mathew,louis,javellin,gold,34,2014,RUS
michael,phelps,swimming,silver,32,2017,USA
usha,pt,running,silver,30,2014,IND
serena,williams,running,gold,31,2016,FRA
roger,federer,tennis,silver,32,2014,CHN
jenifer,cox,swimming,silver,32,2017,IND
fernando,johnson,swimming,silver,32,2017,CHN

scala> val schemaColumns = "firstname:string,lastname:string,sports:string,medal_type:string,age:string,year:string,country:s
tring"
schemaColumns: String = firstname:string,lastname:string,sports:string,medal_type:string,age:string,year:string,country:strin
g

scala> val schema = StructType(schemaColumns.split(",").map(x => StructField(x.split(":")(0),if(x.split(":")(1).equals("strin
g"))StringType else IntegerType,true)))
schema: org.apache.spark.sql.types.StructType = StructType(StructField(firstname,StringType,true), StructField(lastname,Strin
gType,true), StructField(sports,StringType,true), StructField(medal_type,StringType,true), StructField(age,StringType,true),
StructField(year,StringType,true), StructField(country,StringType,true))

scala> val rowRDD = SportsData.map(_.split(",")).map(r => Row(r(0),r(1),r(2),r(3),r(4),r(5),r(6)))
rowRDD: org.apache.spark.rdd.RDD[org.apache.spark.sql.Row] = MapPartitionsRDD[56] at map at <console>:49

scala> val SportsDataDF = spark.createDataFrame(rowRDD, schema)
SportsDataDF: org.apache.spark.sql.DataFrame = [firstname: string, lastname: string ... 5 more fields]

scala> SportsDataDF.createOrReplaceTempView("SportsData")

scala> val updateName = (firstname:String,lastname:String)=>"Mr.".concat(firstname.substring(0,2)).concat(" ")concat(lastname
)
updateName: (String, String) => String = <function2>

scala> spark.udf.register("Full_Name",updateName)
res45: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function2>,StringType,Some(List(StringType
, StringType)))

scala> 
```

[Assignemnet-19.2-Sp...
Assignment_21.11199....
acadgild@localhost:~


Applications Places System
Sun Sep 9, 5:13 PM acadgild
acadgild@localhost:~
File Edit View Search Terminal Help

```
scala> val updateName = (firstname:String,lastname:String)=>"Mr.".concat(firstname.substring(0,2)).concat(" ")concat(lastname
)
updateName: (String, String) => String = <function2>

scala> spark.udf.register("Full_Name",updateName)
res45: org.apache.spark.sql.expressions.UserDefinedFunction = UserDefinedFunction(<function2>,StringType,Some(List(StringType
, StringType)))

scala> val fname = spark.sql("SELECT Full_Name(firstname,lastname) FROM SportsData").show()
+-----------------------+
|UDF(firstname, lastname)|
+-----------------------+
|          Mr.fi lastname|
|            Mr.li cudrow|
|             Mr.ma louis|
|            Mr.mi phelps|
|                Mr.us pt|
|          Mr.se williams|
|           Mr.ro federer|
|               Mr.je cox|
|           Mr.fe johnson|
|            Mr.li cudrow|
|             Mr.ma louis|
|            Mr.mi phelps|
|                Mr.us pt|
|          Mr.se williams|
|           Mr.ro federer|
|               Mr.je cox|
|           Mr.fe johnson|
|            Mr.li cudrow|
|             Mr.ma louis|
|            Mr.mi phelps|
+-----------------------+
only showing top 20 rows

fname: Unit = ()

scala> 
```

2. Add a new column called ranking using udfs on dataframe, where :

gold medalist, with age >= 32 are ranked as pro
gold medalists, with age <= 31 are ranked amateur
silver medalist, with age >= 32 are ranked as expert
silver medalists, with age <= 31 are ranked rookie

<u>Terminal Execution</u> :

```
scala> val Ranking = (medal:String,age:Int)=> (medal,age) match
     | {
     | case (medal,age) if medal == "gold" && age >= 32 => "Pro"
     | case (medal,age) if medal == "gold" && age <= 32 => "Amateur"
     | case (medal,age) if medal == "silver" && age >= 32 => "Expert"
     | case (medal,age) if medal == "silver" && age <= 32 => "Rookie"
     | }
Ranking: (String, Int) => String = <function2>

scala> spark.udf.register("Ranks", Ranking)
res46: org.apache.spark.sql.expressions.UserDefinedFunction =
UserDefinedFunction(<function2>,StringType,Some(List(StringType, IntegerType)))

scala> val rankingRDD = SportsDataDF.withColumn("Ranks",
Ranking(SportsDataDF.col("medal"),SportsDataDF.col("age")))
```