This project is based on a dataset containing 17 years worth of sales transactions for a welding materials warehouse. A sample of the data is shown in Figure 1, which shows three columns titled 'Line Description', 'Item ID', and 'Item Description', respectively. Line Description represents the welding material characteristics, Item ID, is a unique value that is supposed to be extracted from the line description, and item description is a truncated string that contains the line description.

Line Description	Item ID	Item Description
ER308L .030 X 15"	308L03036	308L 030 X 36"
ER321/AMS 5689E .030 X 2#SPOOL	3210302	AMS 5689 321 .030 X 2#Spool
502 1/16 X 36 AMS 6466 FT1E CERTS	50206236	ER80S-B6 (502) 1/16 X 36
502 3/32 X 36 AMS 6466 FTIE CERTS	50209336	ER80S-B6 (502) 3/32 X 36
718 062 X 36 AMS 5832 FT1E CERTS	71806236	718 062 X 36 AMS 5832

Figure 1: A sample dataset containing 5 entries.

The actual dataset has about 50,000 entries, and contains a lot of inconsistencies in character spacing, and the line description is supposed to be logically consistent with the unique Item ID.

The goal of this project was to clean up the data by removing unnecessary spaces, removing typos and excess information about the welding material from the line description, generate a unique item ID, extract the data and store it into a new table, export the data to MYSQL, and then stream the data, so that if a new entry is made, the streamed database is automatically adjusted. To achieve all of this, Pig, Hive, Sqoop, and Flume were used.

To clean up the data, and logically relate the line description, Pig was used. The Pig UDF created md5 hashing of the line descriptions. The shell script removed excess spaces, and created a hash mapping between the old line description and the hashed line description, and generated a unique item ID for each line description. To ensure that a duplicate item ID did not exist, if there were duplicate hashes, the line descriptions were ranked. In terms of Hive, a data warehouse and HCatalog were created with columns for the hash key, item ID, and line description. To query the data in the data warehouse, HCatalog connector was used. In the shell script, using HCatStorer allowed for writing data to the HCatalog-managed table.

The next step consisted of creating a MYSQL database similar to the HCatalog in columns, and exporting the data. In the existing shell script, Sqoop was introduced, and the MYSQL database and Hive database were accessed, and the data was exported into MYSQL. A sample of the MYSQL data is shown in Figure 2

hash_key	item_id	line_desc
4c37cea24d2ccf4bddbbe10159de6c49	10009586	"ER630/17-4PH .035"" X 36"" AWS A5.9M:2006 Heat# Mfg# Actual Cert"
4c38fef4ce576fb0e6b465a05140e08e	10009587	"AMS 5675G Alloy 92 3/32"" X 36"" Heat# Mfg#0014 Lot# Actual Cert Flag Tagged 1 end pe
4c3a81fe8de003ac3d6d48972edea9a1	10009588	"ERCuSi-A 3/32"" X 36"" AWS A5.7-84"
4c3c892d542f14b99334f14ad20f0daf	10009589	"Alloy 112 1/8"" X 14""CTD AWS A5.11/ENiCrMo-3 Heat#047360 MFg# Inco Certs"
4c3db93a24176b3754555a3ab26956d9	10009590	"Alloy 625 1/8"" X 36"" AWS A5.14M-05/ERNiCrMo-3 Heat # 210793 Mfg # 0078 Actual cert

Figure 2: A sample of the MYSQL table containing 5 entries.

The last step was to stream the data to HDFS, thus Flume was introduced. To connect to the MYSQL server and adjust channels, sink sources, server sources, and time parameters, a Flume configuration file was written.

After executing the Flume command the output looked as such:

-Djava.library.path=:/usr/lib/hadoop/lib/native:/usr/lib/hadoop/lib/native

org.apache.flume.node.Application -f /usr/lib/flume-ng/conf/flume2.conf -n agent

2016-01-26 18:11:50,151 (lifecycleSupervisor-1-0) [INFO - org.apache.flume.node.PollingPropertiesFileConfigurationProvider.start(PollingPropertiesFileConfigurationProvider.java:61)] Configuration provider starting
2016-01-26 18:11:50,255 (conf-file-poller-0) [INFO - org.apache.flume.node.PollingPropertiesFileConfigurationProvider$FileWatcherRunnable.run(PollingPropertiesFileConfigurationProvider.java:133)] Reloading configuration file:/usr/lib/flume-ng/conf/flume2.conf
2016-01-26 18:11:50,327 (conf-file-poller-0) [INFO - org.apache.flume.conf.FlumeConfiguration$AgentConfiguration.addProperty(FlumeConfiguration.java:1017)] Processing:HDFS
...

2016-01-26 18:11:54,652 (lifecycleSupervisor-1-3) [INFO - org.apache.flume.instrumentation.MonitoredCounterGroup.register(MonitoredCounterGroup.java:120)] Monitored counter group for type: SOURCE, name: SOURCESQL.sql-source: Successfully registered new MBean.
2016-01-26 18:11:54,652 (lifecycleSupervisor-1-3) [INFO - org.apache.flume.instrumentation.MonitoredCounterGroup.start(MonitoredCounterGroup.java:96)] Component type: SOURCE, name: SOURCESQL.sql-source started
2016-01-26 18:11:54,661 (lifecycleSupervisor-1-2) [INFO - org.apache.flume.instrumentation.MonitoredCounterGroup.register(MonitoredCounterGroup.java:120)] Monitored counter group for type: SINK, name: HDFS: Successfully registered new MBean.
2016-01-26 18:11:54,662 (lifecycleSupervisor-1-2) [INFO - org.apache.flume.instrumentation.MonitoredCounterGroup.start(MonitoredCounterGroup.java:96)] Component type: SINK, name: HDFS started
2016-01-26 18:11:57,789 (SinkRunner-PollingRunner-DefaultSinkProcessor) [INFO - org.apache.flume.sink.hdfs.HDFSDataStream.configure(HDFSDataStream.java:58)] Serializer = TEXT, UseRawLocalFileSystem = false
2016-01-26 18:11:59,254 (SinkRunner-PollingRunner-DefaultSinkProcessor) [INFO - org.apache.flume.sink.hdfs.BucketWriter.open(BucketWriter.java:234)] Creating /flume/mysql/FlumeData.1453860717790.tmp

To test that the stream was working properly, the MYSQL command “select count(*) from item_id_hash_mapping;” was used, and outputted:

```
+----------+
| count(*) |
+----------+
|  38412 |
+----------+
1 row in set (0.00 sec)
```

Next, a sample entry was created and added to the data using the command “Insert into item_id_hash_mapping values”, and the command “select count(*) from item_id_hash_mapping;” was used again, which outputted:

```
+----------+
| count(*) |
+----------+
|  38413 |
+----------+
1 row in set (0.00 sec)
```

Thus showing that the Flume stream was successful upon adding a new entry.

This project allowed for welding material sales transaction of big data to be understood and manipulated with reduced human error using a Hadoop. Further steps with the project include looking back at the original data set and using Apache Mahout and machine learning algorithms such as K means clustering to find patterns in the line description to and try to rewrite the line description to even further reduce human error, and then to repeat the steps in this project.