

DART Real Time Drilling Data Analytics

**Under supervision of**

Mr. C. Sundararaman

Practice Head - Oil & Gas Downstream Solutions



**Submitted By:**

Anirban Mukherjee

SAP ID- 500040399

B.Tech CSE(Open Source and Standards)

Contents

[Acknowledgment 3](#_Toc489019731)

[Supervisor’s Certificate 4](#_Toc489019732)

[Disclaimer 5](#_Toc489019733)

[About the Company 6](#_Toc489019734)

[About Tech Mahindra Energy and Utilities 7](#_Toc489019735)

[Abstract 10](#_Toc489019736)

[1. Introduction 11](#_Toc489019737)

[2. Problem Statement 13](#_Toc489019738)

[3. Objective 14](#_Toc489019739)

[4. Methodology 15](#_Toc489019740)

[5. Modules 21](#_Toc489019741)

[6. Programming behind the modules 24](#_Toc489019742)

[7. Framework and Tools used 36](#_Toc489019743)

[8. Outputs and Results 38](#_Toc489019744)

[9. Inference and Future Scope of the project 44](#_Toc489019746)

[10. Bibliography and References 45](#_Toc489019747)

# Acknowledgment

On the very outset of this report, we would like to extend our sincere & heartfelt obligation towards all the personages who have helped us in this endeavor. Without their active guidance, help, cooperation & encouragement, we would not have made headway in the project. We extend our gratitude to University of Petroleum & Energy Studies for giving us this opportunity.

We are ineffably indebted to **“Arunaditya Soni” (Business Associate, Tech Mahindra, Hyderabad)** for providing us this great opportunity, **“Mr. C. Sundararaman” (Practice Head - Oil & Gas Downstream Solutions, Tech Mahindra, Chennai)** & **“Mr. Abhilash Naroth” (Business Consultant, CVEU, Tech Mahindra, Hyderabad)** for conscientious guidance and encouragement to accomplish this assignment.

I am Aldo thankful for my fellow interns **“Varun Jamwal”** & **“Vibhas Sharma”** for helping me with this project and Contributing towards it.

# 

# Supervisor’s Certificate

This is to certify that Dissertation Report entitled “**DART: Real Time Drilling Data Analytics**” which is submitted by Anirban Mukherjee for the partial fulfillment of degree **Bachlor of Technology degree in Computer Science and engineering** to University of Petroleum & Energy Studies, Dehradun is a record of the candidate’s own work carried out by him under my supervision. The matter embodied in this report is original and has not been submitted for the award of any other degree.

**Mr. C. Sundararaman,**

**Practice Head -Oil & Gas Downstream Solutions,**

**Tech Mahindra, Chennai**

**Date:**

# Disclaimer

The document below is provided for academic purpose only. Images used in this document are taken from public domain. Hence, a picture credit goes to their originator only. The document does not constitute legal advice or opinions of any kind, or any advertising or solicitation. No lawyer-client, advisory, fiduciary or other relationship is created between Tech Mahindra and any person accessing or otherwise using this document.

Tech Mahindra and its affiliates (and any of their respective directors, officers, agents, contractors, interns, suppliers and employees) will not be liable for any damages, losses or causes of action of any nature arising from any use of this document.

# About the Company



Tech Mahindra Limited is an [Indian](https://en.wikipedia.org/wiki/India) [multinational](https://en.wikipedia.org/wiki/Multinational_corporation) and is a specialist in digital transformation, consulting and business re-engineering solutions. It is a USD 3.7 billion company with 103,000+ professionals across 51 countries. It provides services to 727 global customers including Fortune 500 companies. Tech Mahindra is also amongst the Fab 50 companies in Asia as per the Forbes 2014 List. It is a part of the USD 16.9 billion Mahindra Group that employs more than 200,000 people in over 100 countries. Mahindra operates in the key industries that drive economic growth, enjoying a leadership position in tractors, utility vehicles, information technology, financial services and vacation ownership. Tech Mahindra has operations in more than 51 countries with 40 sales offices and 72 delivery centers. Assessed at SEI CMMi Level 5, its Software headcount stood at 72,952, BPO at 22,693 and Support at 7,636 at the end of the financial year 2015

**Operations**

Tech Mahindra's activities spread across a broad spectrum, including Business Support Systems (BSS), Operations Support Systems (OSS), Network Design & Engineering, Next Generation Networks, Mobility Solutions, Security consulting and testing. The "solutions portfolio" includes Consulting, Application Development & Management, Network Services, Solution Integration, Product Engineering, Infrastructure Managed Services, Remote Infrastructure Management and BSG (comprises BPO, Services and Consulting). Tech Mahindra has implemented more than 15 Greenfield Operations globally and has over 128 active customer engagements mostly in the Telecom sector. The company has been involved in about 8 transformation programs of incumbent telecom operators and an array of service offerings for TSPs, TEMs and ISVs. The major clients of Tech Mahindra are AT&T and British Telecom (BT).

# About Tech Mahindra Energy and Utilities

Tech Mahindra has been helping global industry majors develop next generation products and technologies for several years. They offer **connected energy and utility engineering solutions** across oil and gas, smart grids, micro grids and power equipment. Their key strengths include component-level design of a variety of power equipment, detailed plant layout, design and engineering. We have significant capabilities in product engineering, design, analysis and manufacturing support across the industry in areas such as power generation and oil field equipment, plant engineering and utilities.

For several years now, Tech Mahindra is the key players of the energy and utilities industry, they have been helping global industry majors develop next generation products and technologies. In Energy and Utilities segment we offer the following solutions:

* Engineering consulting, starting from conceptual design, analysis to manufacture and aftermarket support
* Power equipment engineering
* Interlocking Design Algorithms for application development and circuit design
* Utilities equipment engineering
* Oil field equipment engineering
* Geographic Information Solutions
* Plant engineering
* Piping engineering
* Energy Management Solutions
* Model based enterprise solutions
* Design automation solutions and Knowledge Based Engineering
* Connected engineering like Remote Monitoring Solutions
* Controls and Automation Solutions
* Intelligent platforms
* Smart grid solutions
* Data analytics (Digital Oil Field)
* Reliability Engineering
* Product sustenance
* Manufacturing and cost management solutions
* Global sourcing and vendor management solutions
* Product validation and test automation (with sophisticated in-house labs)
* Documentation solutions (data quality management and technical publications)

Tech Mahindra is a preferred outsourcing partner to several industry leaders in the energy and utilities space. They provide a range of energy and utilities services and solutions uniquely tailored to address a broad spectrum of specific industry challenges in:

1. **Power generation**
2. Turbines (steam, gas, wind)
3. Gas engines and turbo compressors
4. Power plants (thermal, nuclear, solar)
5. Turbine control systems

* **Oil and gas**
* Upstream
* Mid-stream
* Downstream
  + **Utilities**
  + Water
  + Gas
  + Electricity

**Team behind this project**



**Sundararaman Chintamani**

**Practice Head - Oil and Gas Solutions**



**Mr. Abhilash Naroth**

**Business Consultant, CVEU, Tech Mahindra, Hyderabad**

# Abstract

Drilling at a rig site required a lot of special attention to details. The data generated form the rig is very crucial that is processed and decisions are generated that it is feasible to go further or not. Previously the analytics used for this kind of project was done at two sites the offshore analyst that predicted the soil structure based on the data sent back to them from rig. And then they communicate to the onsite engineers to carry out the task. The purpose of this project is to provide a real time analytics to the onsite engineers to view a pattern between various mnemonics that are a part of drilling data. The patterns help identify the relation between the change of parameters values as a function of depth. The depth on a particular interval keeps feeding the mongo database with data of various parameters. It is divided into three categories that is time log, mud log and the depth log which contains various mnemonics. The prospect of this project was to compute the enormous kind of drilling data that keeps on coming in the process through precise analytics that give exact precision to onsite engineers as well as analyst to see patterns among various drilling parameters at one shot. The larger the data the better is the pattern library generated.

The idea is to kick start the pattern generation module in a very feasible manner. User should select the preferences and should get the pattern immediately as a feedback from the backend side. It may depend on the kind of network that the system is being operated but if connected adequately it should function in a responsive manner. This project aims to develop a base for the pattern generation library and to some extent connect the frontend to the backend so that it could be one single application and convenient to use.

# Introduction

* 1. **Literature Review**

In order to understand the solution for optimizing drilling parameters in real-time that includes obtaining previously acquired data, correlating the current well formation properties to formation properties determined from the previously acquired data, predicting formation properties for a next segment, optimizing the drilling parameters for the next segment, and returning the optimized drilling parameters to the remote data store, we first of all need to understand what is a current well data ,historical data, and drilling parameters of a drilling system.

The current well is the well for which a drilling parameter optimization method is being performed. Current well data refers to data that is related to the current well.

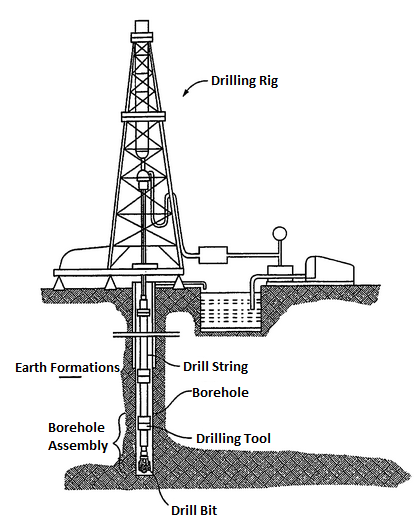
The previously acquired data or the historical data refers to (1) any data related to a well drilled in the same general area as the current well, (2) any data related to a well drilled in a geologically similar area, or (3) other survey data. The historical data may be any data that aids the predictive process. Data from previously or concurrently drilled other well bores in the same area may be used as previously acquired data. Also, data from wells drilled in geologically similar areas may comprise part of the previously acquired data.

A drilling parameter is any parameter that affects the way in which a well is being drilled. For example, the WOB is an important parameter affecting the drilling well. Other drilling parameters includes the torque-on-bit (“TOB”), the rotary speed of the drill bit (“RPM”), differential pressure (∆P), and mud flow rate. There are numerous other drilling parameters, and the term is meant to include any such parameter.

The term optimized drilling parameters refers to values for drilling parameters that have been optimized for a given set of drilling priorities. Optimized does not necessarily mean the best possible drilling parameters because an optimization method may account for one or more drilling priorities. The optimized drilling parameters might be a result of these priorities, and may not represent the drilling parameters that will result in the most economical drilling or the longest bit life.

The data that may be used in a method for optimizing drilling parameters may be collected during the drilling process. Such data relates to current drilling parameters, formation properties, or any other data that will be collected during the drilling process. In the working of a drilling system there is a description of some of the data that may be collected, and how it is related to drilling optimization process.

1.2 **Working of a Drilling System and Data Gathering**



The Figure shows a typical drilling system. The drilling system includes a rig used to suspend a drill string into a borehole. A drill bit at the lower end of the drill string is used to drill through Earth formations. Sensors and other drilling tools may be included in a bottom hole assembly (BHA) near the bottom of the drill string. The drilling system shown in the figure is a land-based drilling system. Other drilling systems, such as deep water drilling systems, are located on floating platforms.

While drilling, it is desirable to capture as much data about the drilling process and about the formations through which the borehole penetrates. The first type of data that are gathered may be classified as near instantaneous measurements, usually called “rig sensed data” as it is sensed on the rig. These include the WOB and the TOB, measured at the surface. Other rig detected data include the RPM, the depth of the drill bit, the casing pressure, the torque, and the drill bit type. In addition, measurements of the drilling fluid (“mud”) are also taken at the surface. For example, the initial mud condition, the mud flow rate, and the pumping pressure, among others. All of these data will be collected on the rig at the surface, and they represent the drilling conditions at the time the information is available.

Different measurements are taken while drilling by instruments and sensors in the Borehole Assembly. These measurements and the resulting data are regularly given by an oilfield services vendor who specializes in making down-hole measurements while drilling.

# Problem Statement

The problem statement is to generate a pattern based on the given data and if it matches it should raise an alarm. The front and the backend should go hand in hand so that the complete project goes in one shot. The generated patterns should be kept in a pattern library and should be used as an inference for building the patters as it comes by from the raw data.

# Objective

**3.1 Project Objective as a technical aspect.**

* Use of modern programming Language

To develop this project we have used modern day programing languages as Java and Python that has object oriented capabilities to keep a level of abstraction from the user. The Project was made into work by dividing every module into the kind of programming language that would solve the purpose to its best.

Java was choose to generate the pattern library, to make the correlation matrix because of it

**3.2 Project Objective from a business point of view.**

* In**crease in reliability of Equipment**

Reliability of equipment determines the life of equipment (example; drill bit). If proper data is present of the amount of drill bit dulling/wearing out that has already occurred in historical datasets and how it has had an effect on the optimized parameters. Then amount of drill bit dulling that has occurred may be estimated based on current well data for those portions of the formation that have already been drilled, as well as data related to such things as WOB, TOB, RPM, mud flow rate, drilling pressure, and data related to measurements of the drill bit properties while drilling. A matched pattern/result obtained allows timely troubleshooting of the problem. And hence reliable operations can be expected.

* **Cost effectiveness**

Major loss a company faces at the time of shutdown of drilling, due to breaking of a drill bit. This would lead to the next procedure of removal of the drill string and changing the drill bit which has a very high cost. Moreover breakdown cost is much higher than the preventive maintenance cost. With the help of Pattern recognition life of the drill bit can be increased. Prediction or forecast can be done with proper data.

* **Improves Process efficiency**

If the parameter settings for the formations and well depth alignment for the new well is displayed on the Electronic Drilling Recorder (EDR) of the back office drilling teams, in Real time as the drilling Parameter Roadmaps, distinctive levels of efficiency can be achieved.

# Methodology

**4.1. The MongoDB**

MongoDB (from *humongous*) is a [free and open-source](https://en.wikipedia.org/wiki/Free_and_open-source_software) [cross-platform](https://en.wikipedia.org/wiki/Cross-platform) [document-oriented database](https://en.wikipedia.org/wiki/Document-oriented_database) program. Classified as a [NoSQL](https://en.wikipedia.org/wiki/NoSQL) database program, MongoDB uses [JSON](https://en.wikipedia.org/wiki/JSON)-like documents with [schemas](https://en.wikipedia.org/wiki/Database_schema). MongoDB is developed by [MongoDB Inc.](https://en.wikipedia.org/wiki/MongoDB_Inc.) and is [free and open-source](https://en.wikipedia.org/wiki/Free_and_open_source_software), published under a combination of the [GNU Affero General Public License](https://en.wikipedia.org/wiki/GNU_Affero_General_Public_License) and the [Apache License](https://en.wikipedia.org/wiki/Apache_License).

**4.1.1 Setting up the mongoDB server.**

The python modules fetch the data from the mongo collection. In order to fetch the data from the mongo collection it must be up and running.

We were given the sample data which was itself a mongo dump “rig\_witsml”. Here the witsml is the server from which the original data was fetched. Wistml is a kind of data exchange protocol that supports the technical data flow across the internet. In this scenario it is used for the retrevial of drilling data from the rig well.

The rig data was saved in the mongo collection in a noSQL or a no schema format where the data was in a JSON file format.

The data was first imported into the mongoDB and cross checked with RoboMongo application by a bridge connection.

Process to import the Mongo collection is pretty simple and straight forward.

1. Start cmd prompt in admin mode
2. Start mongodb server “net start mongod”
3. Then if environment variables are in their place then type the command “mongo”
4. Server should start and we are good to go.

The type the command “mongo –import [/directory/name of the collection]

This should load the collection in the mongo server.

Now we can use pymongo library in python to access the data from the mongo collection

**4.2 Python**

The project begins with a python module named CSV. The job of this python script is generation of csv files. This is the most important step in the project because it creates the csv files that is manipulated in the entire project so it needs to be precise and all the mnemonics searched for should be.

Before beginning the program we must import the libraries in order to make the methods run :

Pprint: pretty print to display the JSON file in a readable format

Pymongo: is the class that allows us to use search and fetch function form the mongo collection.

Csv: The csv class allows us to generate csv files in a fly and dump the data.

Subprocess and os: are used to call some system calls to clean up the redundant files after the job is done.

The python program is one single function and it mainly does two jobs.

**4.2.1. Accessing the headers.**

Before we go and fetch the headers we need to assign variables that holds the connection variables so that we can query the mongodb collection. Once we are don with the connection we simply call this command that gets us all the mnemonic as headers from the collection. Here the collection used is EGFD\_W0\_1. This is the name of the well. The “uidWell” is the key ad “EGFD\_W0\_1” is the value. It’s a kind of hash table where we fetch the values by passing the key as a query. The “UidLog” is the key for the type of log we are trying to get data from. The second set of command refines the first like the “where” clause in a “SQL” query.

The query looks like this and it solves the purpose.

collection\_log.find({**"uidWell"**:**"EGFD\_WO\_1"**,**"uidLog"**:**"TimeLog"**},{**"curvInfoList.mnemonic"**:1,**'\_id'**: **False**}):

The next step is to remove the inverted commas from every key value pair. The JSON file format that the mongodump uses saves everything enclosed in inverted commas. We don’t want it occur in the generated csv files so for that we run a small deletion function that escapes all the inverted commas from the entire file. Not just inverted commas, the extra non used characters are scanned out as well so that the csv becomes more usable and easier to read without throwing any kind of exception in the java program where it is supposed to run.

After the original received query is manipulated and getting rid of unnecessary characters. We save it in a temporary text file that is later removes by a sub process command.

**4.2.2. Accessing the data**

Accessing the data is the same process as of fetching the header only the thing that is changed is the query that is sent to the mongo collection to get the data.

collection\_data.find({**"uidWell"**:**"EGFD\_WO\_1"**,**"uidLog"**:**"TimeLog"**},{**"data"**:1,**'\_id'**: **False**}):

The query format remains the same the only difference is the where clause. And we save this as well in a text file first to carry out the removal of unwanted characters in the text file and one it is done we merge the two files i.e. the header and the data. The number of columns in both the files should be the same which is as expected as they are fetched from the same collection.

And once merging is done we explicitly save it creating a directory naming it after the well and in that directory we save that corresponding csv file. We repeat the process for TimeLog, MudLog and DepthLog for the two wells that is EGFD and Howard. So total 6 files are created in two folders.

And the python module is done for now to generate the csv files for backend processing.

**4.3. Java**

Java in this project is used for about 70% of the task. The Java program or the package that dealt with generation of patterns (as the outcome of the project) comprises 6 modules that has its own individual parts that help the main function to achieve the end result.

**4.3.1 The main method:**

To explain programmatically as the flow of control goes:

1. Read the csv files. The csv file generation was a part of python module where the python script scraped data out of the mongodb collection by find method and fetched all the mnemonics required in their respective header place. Read these files and save it in a variable for operation.
2. Calling a method named “countline” to count the number of header that are available in a csv file. And after fetching the headers we need to display it in the console for user to choose the number of mnemonics and the mnemonics themselves. Another functionality of the this sub program is as we add the columns it parses it into type double. As everything stored in the csv is in string format separated by comma (,).
3. Then the program displays the headers from the selected csv file and prints it on the console for user to see what the different headers available under a particular csv are.
4. The there is an input function that asks the user to enter the number of mnemonic (for testing purpose we have kept the number to be one which when thoroughly tested without any flaws would all to choose multiple mnemonics and generate correlation among them). It displays a small example (4:WOBA) like this to indicate the index number against the displayed header. Once the user inputs the mnemonic the actual calculation can begin.
5. Now we know the number of mnemonic and the name of the mnemonic that is to be worked upon. But we are still dealing with the entire csv file. In order to get rid of the rest of the csv file we save it in an array (1D array)
6. Now the actual calculations are done i.e. running the correlation function where the data sets are passed in a group of 5 in form of an array. The correlation function accepts two arrays of type double and returns a single value in double.

Working of the sub module:

* The first 5 elements of the segregated array is saved in a subarray.
* Then another 5 elements are selected the selection is done from 2nd element to the 6th element.
* Now these two sub arrays 1-5 and 2-6 are sent to the correlation method where the correlation coefficient is calculated and it returns a value of type double.
* It goes till the last set of element n-4 to n.
* Once the iteration is completed it shifts the first subarray to 2nd to 6th and the second subarray is iterated from 3rd-7th till n-4 to n.

1. Now we have generated correlation coefficient for the entire column and it forms a kind of upper diagonal 2d matrix that is converted to a lower triangular matrix for convenience of access.
2. The Diagonal of the matrix must always be 1 as the correlation between the same pair of elements must always be 1.

At last part of the sub-module we create a text file so that we can collect the entire output and save it for further processing. The time required for processing a mnemonic at an average case right from picking up the raw csv file t generation of a correlation matrix is about 8 seconds that processes almost 3000 rows at a one shot. This can further be optimized with precise use of algorithm and data structures.

Now to explain how to correlation function works, after all it is the zest of the entire project. The two subarrays are passed as an argument to this class and it returns only one variable output that is the correlation value but the two arrays. It can return +infinity, NaN or –Infinity that can be replaced with 0 in the return part of the module.

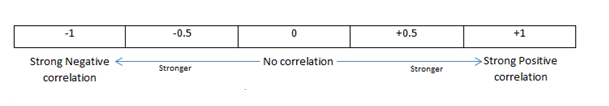
**4.3.2 Correlation Methodology in Details.**

Correlation analysis is a method of statistical evaluation which is used to study the strength of a relationship between two, numerically measured variables (e.g. length and breadth). This particular type of analysis is useful when we have to establish the possible connections between variables. It is often misjudged that correlation analysis determines cause and effect; even though, this is not the case because there may be other variables that are not present in the research which might have impacted the results.

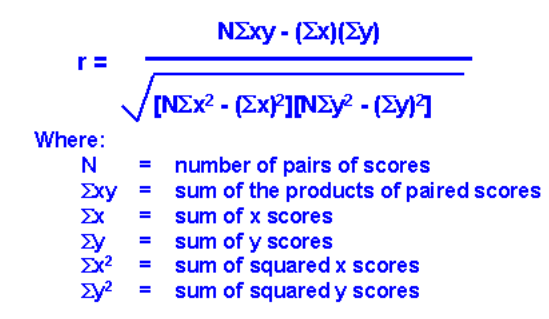
In the event that correlation is found between two variables it implies that when there is a systematic change in one variable, there is also a likewise change in the other; the variables change together over a certain period of time. If there is correlation found between the variables or groups, depending upon the numerical values measured, this may be either positive or negative.

* Positive correlation exists if one variable increases at the same time with the other, i.e. the high numerical values of one variable relate to the high numerical values of the other.
* Negative correlation exists if one variable decreases when the other increases, i.e. the high numerical values of one variable relate to the low numerical values of the other.

Pearson’s product-moment coefficient is the measurement of correlation and ranges between +1 and -1. It is indicated by ‘r’. If r= +1, this indicates the strongest positive correlation possible, and if r= -1, it indicates the strongest negative correlation possible. Therefore the closer the coefficient to either of these numbers the stronger the correlation of the data it represents. On this scale 0 indicates no correlation, thus values which are closer to zero would highlight weaker/poorer correlation than those closer to +1/-1.

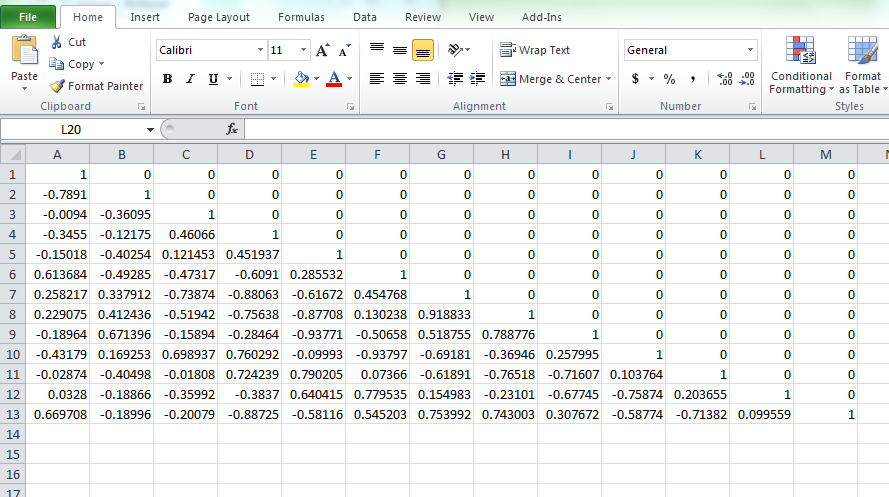


We all know how to compute a correlation between two variables. The formula for the correlation is as shown below



This type of table is called a correlation matrix. It lists the variable names (V0-V10) down the first column and across the first row. The diagonal of a correlation matrix i.e., the numbers that go from the upper left corner to the lower right always consists of ones. That is because these are the correlations between each variable and itself and a variable is always perfectly correlated with itself. This R program only shows the lower triangle of the correlation matrix. In every correlation matrix there are two triangles that are the values below and to the left of the diagonal called the lower triangle and above and to the right of the diagonal called the upper triangle. There is no reason to print both triangles because the two triangles of a correlation matrix are always mirror images of each other as the correlation of variable x with variable y is always equal to the correlation of variable y with variable x. This is referred as a symmetric matrix because it has a mirror-image characteristic above and below the diagonal. A correlation matrix is always a symmetric matrix.

Example of a generated matrix:



Correlation of the mnemonic HKLA form the EGFD\_W0\_1\_DepthLog csv file.

# Modules

5.1 The entire project is divided into two modules i.e. the backend part and the frontend part

**5.1.1 Frontend Design:**

The front end was made by my fellow interns “Vibhas Sharma” and “Varun Jamwal” . The programming languages used for developing the front end was JavaScript and html. A part of Ajax was also used to make the page dynamic instead of rerouting to a different webpage when refreshed.

JavaScript part:

The front end has two views to it one is the Dashboard view and one is the Colum selection view. The Dashboard view gives a very straightforward approach to select the mnemonics based on the names of the wells we promote and demote. One passing after every stage after promoting the names of the well the chevrons is updates as a ratio of what is selected and what is not.

The frontend too needs a mongodb collection to work upon. At first, a mongodb collection is imported on which analytics have to apply through the UI. MongoDb is imported and collection is imported. The imported collection is viewed from the UI and names of mnemonics are selected from a particular log accordingly.

The second part is the column wise selection of the well name, hole size and mnemonics. After selecting one after the other it is updated by clinking on the next button and after the first selection column is down on clicking on the next will show all the data that are selected in the intermediate column and after we click on > arrow it will show the mnemonics under it. After that we click on a particular mnemonic we click on generate that send the data to backend and generate pattern and will redirect to a page that displays the graph based on the feedback given by the beck end engine. And if a particular criterion is matched it will trigger an alarm that it has reached a critical limit and the operation must abort.

This one shot view on pattern based on graph will single handedly allow engineers to decide the well profile and should they continue drilling.

**5.1.2 Backend Design:**

The backend of the application is based on Python and Java application.

To generate the pattern generation it all starts with the creation of csv files. The csv files are nothing but comma separated values that are used to store information in a comma separated readable format. The backend receives the input from the frontend UI. The input received has a set of values upon which the backend engine generates a pattern. The generation of patterns is itself a step by step process.

It can be best described in flowchart:

**Front End Flow:**

**Dashboard**

User selects the name of Well, Hole Size, Log type and Mnemonics to generate pattern

Graph of the pattern is generated

**Back End Flow:**

Mnemonics + Log Type + Field Name

Pattern Library

Graph is generated as a function of parameters

Backend Engine for analytics

**5.1.3 Flow of control in the backend**

Input received from the UI

CSV are made and saved

Python Script to compose the CSV files to process by querying MongoDB

Java module is called to generate the pattern library of correlation values

Output file is generated to save the matrix

Output to UI

Formation of cluster and generating a graph out of the plot pointed by patterns

# Programming behind the modules

**6.1 Programming Languages Used (Backend End):**

**6.1.1. Java:**

Java is a programming language and computing platform first released by Sun Microsystems in 1995. There are lots of applications and websites that will not work unless you have Java installed, and more are created every day. Java is fast, secure, and reliable. From laptops to datacenters, game consoles to scientific supercomputers, cell phones to the Internet, Java is everywhere!

6.1.2. Java IDE Used: Eclipse EE Neon.3 Release (4.6.3)

6.1.3. Class and Libraries used for compilation of the program:

Dom4j-1.6.1.jar, poi-3.9, poi-ooxml-schemas-3.9, poi-ooxml-3.9, xml-beans-2.3.0.jar

6.1.3. Main Module:

**package** Pattern;

**import** java.io.BufferedReader;

**import** java.io.FileOutputStream;

**import** au.com.bytecode.opencsv.CSVWriter;

**import** java.io.FileReader;

**import** java.io.FileWriter;

**import** java.io.IOException;

**import** java.io.PrintStream;

**import** java.util.ArrayList;

**import** java.util.List;

**import** java.util.Scanner;

**public** **class** Main

{

**public** **static** **void** main(String[] args) **throws** IOException

{

PrintStream out = **new** PrintStream(**new** FileOutputStream("C:/Users/AM0C70368/Desktop/Dart-Java/Project-DART-Java/Pattern/src/output.txt"));

BufferedReader reader = **new** BufferedReader(**new** FileReader("C:/Users/AM0C70368/Desktop/Dart-Java/Project-DART-Java/Pattern/Well/EGFD\_WO\_1/EGFD\_WO\_1\_DepthLog.csv"));

String line = **null**;

List<String> lines = **new** ArrayList<>();

**int** mnemCount =0;

String[] mnem\_tokens = **null**; //store the mnemonics in form of an 1D array

**boolean** header = **true**;

**int** lineCount = countLine.*countLines*("C:/Users/AM0C70368/Desktop/Dart-Java/Project-DART-Java/Pattern/Well/EGFD\_WO\_1/EGFD\_WO\_1\_DepthLog.csv");

Double[][] data = **new** Double[lineCount][200];

**int** i = 0;

**int** j =0;

// Generate Two Dimensional Array of Data Values from the CSV -- Start---------------

**while** ((line = reader.readLine()) != **null**)

{

mnemCount = line.length() - line.replace(",", "").length();

**if**(header == **true**)

{

lines.add(line);

header = **false**;

System.***out***.print("Hedders are : ");

System.***out***.println(lines);

mnem\_tokens = line.split(",");

}

**else**

{

String [] values = line.split(",");

**for** (j=0;j<=mnemCount;j++)

{

**if**(values[j].contains("Z")||values[j].contains("None {

//System.out.println("Z");

}

**else**

{

data[i][j] = Double.*parseDouble*(values[j]);

}

}

}

i++;

}

// Generate Two Dimensional Array of Data Values from the CSV -- End----------------

**int** sample = 4;

Scanner scan = **new** Scanner(System.***in***);

System.***out***.println("");

System.***out***.print("Enter how many mneomics are to be selected (Except TIME) : ");

**int** n = Integer.*parseInt*(scan.nextLine());

System.***out***.println("");

**int** mnemonics[] = **new** **int**[n];

**for** (i = 0; i < mnemonics.length; i++)

{

System.***out***.print("Enter position of mnemonic (e.g "+ sample+ ":"+ mnem\_tokens[sample-1]+") : ");

mnemonics[i] = scan.nextInt();

}

Double[][] arr = **new** Double[lineCount][mnemonics.length];

j=0;

**int** k=0;

**while**(j<mnemCount+1)

{

**while**(k<mnemonics.length)

{

**if**(j+1 == mnemonics[k])

{

**for**(i=1;i<lineCount;i++)

{

arr[i][k] = data[i][j]; //Separating the array for correlation from initial 2d array

}

}

k++;

}

j++;

k=0;

}

**int** rows = arr.length;

System.***out***.println("\nCheck the output file at : C:/Users/AM0C70368/Desktop/Dart-Java/Project-DART-Java/Pattern/src/output.txt");

// Chunking the array into blocks of size 5 each calculating correlation and saving as a 2d array-------------------------------------------------------------

**int** x=0,y=0;

**double**[][] save = **new** **double**[rows][rows];

**double**[] subarr1 = **new** **double**[5];

**double**[] subarr2 = **new** **double**[5];

Correlation obj = **new** Correlation();

**double** rec;

**int** iter = 0;

**int** start = 1;

**int** end = 5;

**int** count = 1 ;

**while**(count < rows -5)

{

k=0;

**for**(i = start ; i<=end ;i++)

{

subarr1[k] = arr[i][0];

k++;

}

start++;

end++;

iter = 0;

**while**(end<rows)

{

k=0;

**for**(j= start; j<=end ;j++)

{

subarr2[k] = arr[j][0];

k++;

}

rec=obj.corr(subarr1, subarr2);

save[x][y]=rec;

x++;

iter = iter + 1;

start++;

end++;

}

x = x-iter;

start = start - iter;

end = end - iter;

count++;

y++;

}

// arranging the 2d array in the lower diagonal--------------------------------------------------------------------------------

//System.out.println(save[5][0]);

**int** temp\_count =0;

**double** temp;

**for**(i=0 ; i <rows-5; i++)

{

**for**(j=0 ; j<rows-5; j++)

{

**if**(save[j][i] == 0.0)

{

temp\_count = j;

**while**(temp\_count > 0)

{

temp = save[temp\_count-1][i];

save[temp\_count-1][i] = save[temp\_count][i];

save[temp\_count][i] =temp;

temp\_count--;

}

}

**else**

{

**continue**;

}

}

}

**for**(i=0 ; i<rows-5; i++)

{

System.***out***.println(" ");

**for**(j=0 ; j<rows-5; j++)

{

**if**(i==j)

{

out.println(1);

System.*setOut*(out);

}

**else**

{

out.println(save[j][i]);

System.*setOut*(out);

}

}

}

reader.close();

scan.close();

out.close();

}

}

6.1.4. Correlation Module:

**package** Pattern;

**import** java.util.\*;

**import** java.lang.\*;

**class** Correlation

{

**public** **double** corr(**double**[] input\_x, **double**[] input\_y) //passing the two arrays

{

**double** nr=0,dr\_1=0,dr\_2=0,dr\_3=0,dr=0;

**double** r;

**double** xx[],xy[],yy[];

xx =**new** **double**[5];

xy =**new** **double**[5];

yy =**new** **double**[5];

//initialise the two array from arguments

**double** x[]= input\_x;

**double** y[]= input\_y;

**double** sum\_y=0,sum\_yy=0,sum\_xy=0,sum\_x=0,sum\_xx=0;

**int** i,n=5;

**for**(i=0;i<n;i++)

{

xx[i]=x[i]\*x[i];

yy[i]=y[i]\*y[i];

}

**for**(i=0;i<n;i++)

{

sum\_x+=x[i];

sum\_y+=y[i];

sum\_xx+= xx[i];

sum\_yy+=yy[i];

sum\_xy+= x[i]\*y[i];

}

nr=(n\*sum\_xy)-(sum\_x\*sum\_y);

**double** sum\_x2=sum\_x\*sum\_x;

**double** sum\_y2=sum\_y\*sum\_y;

dr\_1=(n\*sum\_xx)-sum\_x2;

dr\_2=(n\*sum\_yy)-sum\_y2;

dr\_3=dr\_1\*dr\_2;

dr=Math.*sqrt*(dr\_3);

r=(nr/dr);

**if**(r>1.0 || r<-1.0)

{

**return** 0;

}

**else**

{

**return** r;

}

}

}

6.1.4. countLine Module:

**package** Pattern;

**import** java.io.BufferedInputStream;

**import** java.io.FileInputStream;

**import** java.io.IOException;

**import** java.io.InputStream;

**public** **class** countLine

{

**public** **static** **int** countLines(String filename) **throws** IOException

{

InputStream is = **new** BufferedInputStream(**new** FileInputStream(filename));

**try** {

**byte**[] c = **new** **byte**[1024];

**int** count = 0;

**int** readChars = 0;

**boolean** empty = **true**;

**while** ((readChars = is.read(c)) != -1)

{

empty = **false**;

**for** (**int** i = 0; i < readChars; ++i)

{

**if** (c[i] == '\n')

{

++count;

}

}

}

**return** (count == 0 && !empty) ? 1 : count;

}

**finally**

{

is.close();

}

}

}

6.2.2. Python

Python is a general-purpose interpreted, interactive, object-oriented, and high-level programming language. It was created by Guido van Rossum during 1985- 1990. Like Perl, Python source code is also available under the GNU General Public License (GPL). This tutorial gives enough understanding on Python programming language.

6.2.3. Python IDE and Modules used:

IDE: JetBrains PyCharm Community Edition 2017.1.4

Libraries used: pymongo, subprocess, MongoClient, pprint & os

6.2.4. Main Module:

**import** pprint  
**from** pymongo **import** MongoClient  
**import** csv  
**import** subprocess  
**import** os  
  
client = MongoClient()  
  
*#EGFD\_WO\_1\_TimeLog*db\_log = client.mydb  
collection\_log = db\_log.groupLog  
file=open(**"log.txt"**, **"w"**)  
  
**for** post\_log **in** collection\_log.find({**"uidWell"**:**"EGFD\_WO\_1"**,**"uidLog"**:**"TimeLog"**},{**"curvInfoList.mnemonic"**:1,**'\_id'**: **False**}):  
 pprint.pprint(post\_log)  
 s = str(post\_log)  
 file.write(s + **"\n"**)  
file.close()  
  
infile = **"log.txt"**outfile = **"trim\_log.txt"**delete\_list = [**"{"**,**"}"**,**"mnemonic"**,**":"**,**"'"**,**"curvInfoList"**,**'['**,**']'**, **" "**]  
fin = open(infile)  
fout = open(outfile, **"w+"**)  
**for** line **in** fin:  
 **for** word **in** delete\_list:  
 line = line.replace(word, **""**)  
 fout.write(line)  
fin.close()  
fout.close()  
  
**with** open(**'trim\_log.txt'**, **'r'**) **as** in\_file:  
 stripped = (line.strip() **for** line **in** in\_file)  
 lines = (line.split(**","**) **for** line **in** stripped **if** line)  
 **with** open(**'EGFD\_WO\_1\_TimeLog\_log.csv'**, **'w'**) **as** out\_file:  
 writer = csv.writer(out\_file)  
 writer.writerows(lines)  
  
  
*#for fetching data from mongodb collection*db\_data = client.mydb  
collection\_data = db\_data.data  
file=open(**"data.txt"**, **"w"**)  
  
**for** post\_log **in** collection\_data.find({**"uidWell"**:**"EGFD\_WO\_1"**,**"uidLog"**:**"TimeLog"**},{**"data"**:1,**'\_id'**: **False**}):  
 pprint.pprint(post\_log)  
 s = str(post\_log)  
 file.write(s + **"\n"**)  
file.close()  
  
infile = **"data.txt"**outfile = **"trim\_data.txt"**delete\_list = [**"{"**,**"}"**,**":"**,**"'"**,**"data"**,**'['**,**']'**,**" "**]  
fin = open(infile)  
fout = open(outfile, **"w+"**)  
**for** line **in** fin:  
 **for** word **in** delete\_list:  
 line = line.replace(word, **""**)  
 fout.write(line)  
fin.close()  
fout.close()  
  
**with** open(**'trim\_data.txt'**, **'r'**) **as** in\_file:  
 stripped = (line.strip() **for** line **in** in\_file)  
 lines = (line.split(**","**) **for** line **in** stripped **if** line)  
 **with** open(**'EGFD\_WO\_1\_TimeLog\_data.csv'**, **'w'**) **as** out\_file:  
 writer = csv.writer(out\_file)  
 writer.writerows(lines)  
  
file\_object = open(**'EGFD\_WO\_1\_TimeLog\_data.csv'**, **'r'**)  
lines = csv.reader(file\_object, delimiter=**','**, quotechar=**'"'**)  
flag = 0  
data=[]  
**for** line **in** lines:  
 **if** line == []:  
 flag =1  
 **continue  
 else**:  
 data.append(line)  
file\_object.close()  
**if** flag ==1:  
 file\_object = open(**'EGFD\_WO\_1\_TimeLog\_data.csv'**, **'w'**)  
 **for** line **in** data:  
 str1 = **','**.join(line)  
 file\_object.write(str1+**"\n"**)  
file\_object.close()  
  
  
subprocess.call(**"type EGFD\_WO\_1\_TimeLog\_log.csv EGFD\_WO\_1\_TimeLog\_data.csv > EGFD\_WO\_1\_TimeLog\_1.csv"**, shell=**True**)  
os.remove(**"trim\_data.txt"**)  
os.remove(**"trim\_log.txt"**)  
os.remove(**"data.txt"**)  
os.remove(**"log.txt"**)  
os.remove(**"EGFD\_WO\_1\_TimeLog\_log.csv"**)  
os.remove(**"EGFD\_WO\_1\_TimeLog\_data.csv"**)  
  
**with** open(**'EGFD\_WO\_1\_TimeLog\_1.csv'**) **as** infile, open(**'EGFD\_WO\_1\_TimeLog.csv'**, **'w'**) **as** outfile:  
 **for** line **in** infile:  
 **if not** line.strip(): **continue** outfile.write(line)  
  
os.remove(**"EGFD\_WO\_1\_TimeLog\_1.csv"**)  
  
subprocess.call(**"move EGFD\_WO\_1\_TimeLog.csv C:\\Users\AM0C70368\python\_projects\CSV\Well\EGFD\_WO\_1"**, shell =**True**)  
  
*#EGFD\_WO\_1\_MudLog*db\_log = client.mydb  
collection\_log = db\_log.groupLog  
file=open(**"log.txt"**, **"w"**)  
  
**for** post\_log **in** collection\_log.find({**"uidWell"**:**"EGFD\_WO\_1"**,**"uidLog"**:**"MudLog"**},{**"curvInfoList.mnemonic"**:1,**'\_id'**: **False**}):  
 pprint.pprint(post\_log)  
 s = str(post\_log)  
 file.write(s + **"\n"**)  
file.close()  
  
infile = **"log.txt"**outfile = **"trim\_log.txt"**delete\_list = [**"{"**,**"}"**,**"mnemonic"**,**":"**,**"'"**,**"curvInfoList"**,**'['**,**']'**, **" "**]  
fin = open(infile)  
fout = open(outfile, **"w+"**)  
**for** line **in** fin:  
 **for** word **in** delete\_list:  
 line = line.replace(word, **""**)  
 fout.write(line)  
fin.close()  
fout.close()  
  
**with** open(**'trim\_log.txt'**, **'r'**) **as** in\_file:  
 stripped = (line.strip() **for** line **in** in\_file)  
 lines = (line.split(**","**) **for** line **in** stripped **if** line)  
 **with** open(**'EGFD\_WO\_1\_MudLog\_log.csv'**, **'w'**) **as** out\_file:  
 writer = csv.writer(out\_file)  
 writer.writerows(lines)  
  
  
db\_data = client.mydb  
collection\_data = db\_data.data  
file=open(**"data.txt"**, **"w"**)  
  
**for** post\_log **in** collection\_data.find({**"uidWell"**:**"EGFD\_WO\_1"**,**"uidLog"**:**"MudLog"**},{**"data"**:1,**'\_id'**: **False**}):  
 pprint.pprint(post\_log)  
 s = str(post\_log)  
 file.write(s + **"\n"**)  
file.close()  
  
infile = **"data.txt"**outfile = **"trim\_data.txt"**delete\_list = [**"{"**,**"}"**,**":"**,**"'"**,**"data"**,**'['**,**']'**,**" "**]  
fin = open(infile)  
fout = open(outfile, **"w+"**)  
**for** line **in** fin:  
 **for** word **in** delete\_list:  
 line = line.replace(word, **""**)  
 fout.write(line)  
fin.close()  
fout.close()  
  
**with** open(**'trim\_data.txt'**, **'r'**) **as** in\_file:  
 stripped = (line.strip() **for** line **in** in\_file)  
 lines = (line.split(**","**) **for** line **in** stripped **if** line)  
 **with** open(**'EGFD\_WO\_1\_MudLog\_data.csv'**, **'w'**) **as** out\_file:  
 writer = csv.writer(out\_file)  
 writer.writerows(lines)  
  
file\_object = open(**'EGFD\_WO\_1\_MudLog\_data.csv'**, **'r'**)  
lines = csv.reader(file\_object, delimiter=**','**, quotechar=**'"'**)  
flag = 0  
data=[]  
**for** line **in** lines:  
 **if** line == []:  
 flag =1  
 **continue  
 else**:  
 data.append(line)  
file\_object.close()  
**if** flag ==1:  
 file\_object = open(**'EGFD\_WO\_1\_MudLog\_data.csv'**, **'w'**)  
 **for** line **in** data:  
 str1 = **','**.join(line)  
 file\_object.write(str1+**"\n"**)  
file\_object.close()  
  
subprocess.call(**"type EGFD\_WO\_1\_MudLog\_log.csv EGFD\_WO\_1\_MudLog\_data.csv > EGFD\_WO\_1\_MudLog\_1.csv"**, shell=**True**)  
os.remove(**"trim\_data.txt"**)  
os.remove(**"trim\_log.txt"**)  
os.remove(**"data.txt"**)  
os.remove(**"log.txt"**)  
os.remove(**"EGFD\_WO\_1\_MudLog\_log.csv"**)  
os.remove(**"EGFD\_WO\_1\_MudLog\_data.csv"**)  
  
**with** open(**'EGFD\_WO\_1\_MudLog\_1.csv'**) **as** infile, open(**'EGFD\_WO\_1\_MudLog.csv'**, **'w'**) **as** outfile:  
 **for** line **in** infile:  
 **if not** line.strip(): **continue** outfile.write(line)  
  
os.remove(**"EGFD\_WO\_1\_MudLog\_1.csv"**)  
  
subprocess.call(**"move EGFD\_WO\_1\_MudLog.csv C:\\Users\AM0C70368\python\_projects\CSV\Well\EGFD\_WO\_1"**, shell =**True**)  
  
*#DepthLog\_EGFD\_WO\_1*db\_log = client.mydb  
collection\_log = db\_log.groupLog  
file=open(**"log.txt"**, **"w"**)  
  
**for** post\_log **in** collection\_log.find({**"uidWell"**:**"EGFD\_WO\_1"**,**"uidLog"**:**"DepthLog"**},{**"curvInfoList.mnemonic"**:1,**'\_id'**: **False**}):  
 pprint.pprint(post\_log)  
 s = str(post\_log)  
 file.write(s + **"\n"**)  
file.close()  
  
infile = **"log.txt"**outfile = **"trim\_log.txt"**delete\_list = [**"{"**,**"}"**,**"mnemonic"**,**":"**,**"'"**,**"curvInfoList"**,**'['**,**']'**, **" "**]  
fin = open(infile)  
fout = open(outfile, **"w+"**)  
**for** line **in** fin:  
 **for** word **in** delete\_list:  
 line = line.replace(word, **""**)  
 fout.write(line)  
fin.close()  
fout.close()  
  
**with** open(**'trim\_log.txt'**, **'r'**) **as** in\_file:  
 stripped = (line.strip() **for** line **in** in\_file)  
 lines = (line.split(**","**) **for** line **in** stripped **if** line)  
 **with** open(**'EGFD\_WO\_1\_DepthLog\_log.csv'**, **'w'**) **as** out\_file:  
 writer = csv.writer(out\_file)  
 writer.writerows(lines)  
  
  
db\_data = client.mydb  
collection\_data = db\_data.data  
file=open(**"data.txt"**, **"w"**)  
  
**for** post\_log **in** collection\_data.find({**"uidWell"**:**"EGFD\_WO\_1"**,**"uidLog"**:**"DepthLog"**},{**"data"**:1,**'\_id'**: **False**}):  
 pprint.pprint(post\_log)  
 s = str(post\_log)  
 file.write(s + **"\n"**)  
file.close()  
  
infile = **"data.txt"**outfile = **"trim\_data.txt"**delete\_list = [**"{"**,**"}"**,**":"**,**"'"**,**"data"**,**'['**,**']'**,**" "**]  
fin = open(infile)  
fout = open(outfile, **"w+"**)  
**for** line **in** fin:  
 **for** word **in** delete\_list:  
 line = line.replace(word, **""**)  
 fout.write(line)  
fin.close()  
fout.close()  
  
**with** open(**'trim\_data.txt'**, **'r'**) **as** in\_file:  
 stripped = (line.strip() **for** line **in** in\_file)  
 lines = (line.split(**","**) **for** line **in** stripped **if** line)  
 **with** open(**'EGFD\_WO\_1\_DepthLog\_data.csv'**, **'w'**) **as** out\_file:  
 writer = csv.writer(out\_file)  
 writer.writerows(lines)  
  
file\_object = open(**'EGFD\_WO\_1\_DepthLog\_data.csv'**, **'r'**)  
lines = csv.reader(file\_object, delimiter=**','**, quotechar=**'"'**)  
flag = 0  
data=[]  
**for** line **in** lines:  
 **if** line == []:  
 flag =1  
 **continue  
 else**:  
 data.append(line)  
file\_object.close()  
**if** flag ==1:  
 file\_object = open(**'EGFD\_WO\_1\_DepthLog\_data.csv'**, **'w'**)  
 **for** line **in** data:  
 str1 = **','**.join(line)  
 file\_object.write(str1+**"\n"**)  
file\_object.close()  
  
subprocess.call(**"type EGFD\_WO\_1\_DepthLog\_log.csv EGFD\_WO\_1\_DepthLog\_data.csv > EGFD\_WO\_1\_DepthLog\_1.csv"**, shell=**True**)  
os.remove(**"trim\_data.txt"**)  
os.remove(**"trim\_log.txt"**)  
os.remove(**"data.txt"**)  
os.remove(**"log.txt"**)  
os.remove(**"EGFD\_WO\_1\_DepthLog\_log.csv"**)  
os.remove(**"EGFD\_WO\_1\_DepthLog\_data.csv"**)  
**with** open(**'EGFD\_WO\_1\_DepthLog\_1.csv'**) **as** infile, open(**'EGFD\_WO\_1\_DepthLog.csv'**, **'w'**) **as** outfile:  
 **for** line **in** infile:  
 **if not** line.strip(): **continue** outfile.write(line)  
  
os.remove(**"EGFD\_WO\_1\_DepthLog\_1.csv"**)  
  
subprocess.call(**"move EGFD\_WO\_1\_DepthLog.csv C:\\Users\AM0C70368\python\_projects\CSV\Well\EGFD\_WO\_1"**, shell =**True**)

# Framework and Tools used

**7.1 Pycharm**

PyCharm is an [Integrated Development Environment](https://en.wikipedia.org/wiki/Integrated_Development_Environment) (IDE) used in [computer programming](https://en.wikipedia.org/wiki/Computer_programming), specifically for the [Python](https://en.wikipedia.org/wiki/Python_%28programming_language%29) language. It is developed by the Czech company [JetBrains](https://en.wikipedia.org/wiki/JetBrains). It provides code analysis, a graphical debugger, an integrated unit tester, integration with [version control systems](https://en.wikipedia.org/wiki/Revision_control) (VCSes), and supports web development with [Django](https://en.wikipedia.org/wiki/Django_%28web_framework%29). It was used because :

 Coding Assistance and [Analysis](https://en.wikipedia.org/wiki/Code_analysis), with [code completion](https://en.wikipedia.org/wiki/Autocomplete), syntax and error highlighting, linter integration, and quick fixes

 Project and Code Navigation: specialized project views, file structure views and quick jumping between files, classes, methods and usages

 Python [Refactoring](https://en.wikipedia.org/wiki/Refactoring): including rename, extract method, introduce variable, introduce constant, pull up, push down and others

 Integrated Python [Debugger](https://en.wikipedia.org/wiki/Debugger)

 Integrated [Unit Testing](https://en.wikipedia.org/wiki/Unit_Testing), with line-by-line [code coverage](https://en.wikipedia.org/wiki/Code_coverage)

**7.2 MongoDB**

* MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time
* The document model maps to the objects in your application code, making data easy to work with
* Ad hoc queries, indexing, and real time aggregation provide powerful ways to access and analyze your data
* MongoDB is a distributed database at its core, so high availability, horizontal scaling, and geographic distribution are built in and easy to use
* MongoDB is free and open-source, published under the GNU Affero General Public License

**7.3 Eclipse IDE**

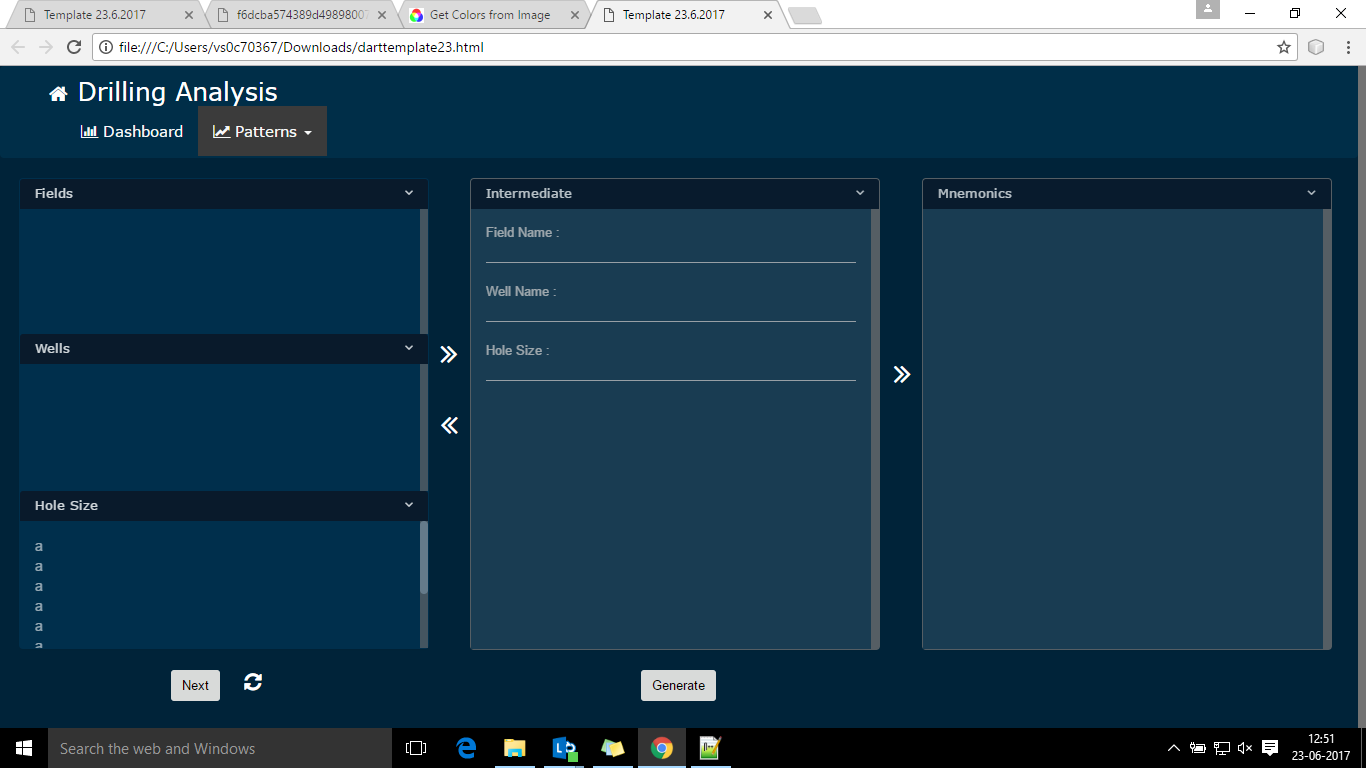
Eclipse is an [integrated development environment](https://en.wikipedia.org/wiki/Integrated_development_environment) (IDE) used in [computer programming](https://en.wikipedia.org/wiki/Computer_programming), and is the most widely used Java IDE. It contains a base [workspace](https://en.wikipedia.org/wiki/Workspace) and an extensible [plug-in](https://en.wikipedia.org/wiki/Plug-in_%28computing%29) system for customizing the environment. Eclipse is written mostly in [Java](https://en.wikipedia.org/wiki/Java_%28programming_language%29) and its primary use is for developing Java applications, but it may also be used to develop applications in other [programming languages](https://en.wikipedia.org/wiki/Programming_language) via plug-ins, including [Ada](https://en.wikipedia.org/wiki/Ada_%28programming_language%29), [ABAP](https://en.wikipedia.org/wiki/ABAP), [C](https://en.wikipedia.org/wiki/C_%28programming_language%29), [C++](https://en.wikipedia.org/wiki/C%2B%2B), [COBOL](https://en.wikipedia.org/wiki/COBOL), [D](https://en.wikipedia.org/wiki/D_%28programming_language%29), [Fortran](https://en.wikipedia.org/wiki/Fortran), [Haskell](https://en.wikipedia.org/wiki/Haskell_%28programming_language%29), [JavaScript](https://en.wikipedia.org/wiki/JavaScript), [Julia](https://en.wikipedia.org/wiki/Julia_%28programming_language%29), [Lasso](https://en.wikipedia.org/wiki/Lasso_%28programming_language%29), [Lua](https://en.wikipedia.org/wiki/Lua_%28programming_language%29), NATURAL, [Perl](https://en.wikipedia.org/wiki/Perl), [PHP](https://en.wikipedia.org/wiki/PHP), [Prolog](https://en.wikipedia.org/wiki/Prolog), [Python](https://en.wikipedia.org/wiki/Python_%28programming_language%29), [R](https://en.wikipedia.org/wiki/R_%28programming_language%29), [Ruby](https://en.wikipedia.org/wiki/Ruby_%28programming_language%29) (including [Ruby on Rails](https://en.wikipedia.org/wiki/Ruby_on_Rails) framework), [Rust](https://en.wikipedia.org/wiki/Rust_%28programming_language%29), [Scala](https://en.wikipedia.org/wiki/Scala_%28programming_language%29), [Clojure](https://en.wikipedia.org/wiki/Clojure), [Groovy](https://en.wikipedia.org/wiki/Groovy_%28programming_language%29), [Scheme](https://en.wikipedia.org/wiki/Scheme_%28programming_language%29), and [Erlang](https://en.wikipedia.org/wiki/Erlang_%28programming_language%29). It can also be used to develop documents with [LaTeX](https://en.wikipedia.org/wiki/LaTeX) (via a TeXlipse plug-in) and packages for the software [Mathematica](https://en.wikipedia.org/wiki/Mathematica). Development environments include the Eclipse Java development tools (JDT) for Java and Scala, Eclipse CDT for C/C++, and Eclipse PDT for PHP, among others.

The initial [codebase](https://en.wikipedia.org/wiki/Codebase) originated from [IBM VisualAge](https://en.wikipedia.org/wiki/IBM_VisualAge). The Eclipse [software development kit](https://en.wikipedia.org/wiki/Software_development_kit) (SDK), which includes the Java development tools, is meant for Java developers. Users can extend its abilities by installing plug-ins written for the Eclipse Platform, such as development toolkits for other programming languages, and can write and contribute their own plug-in modules. Since the introduction of the [OSGi](https://en.wikipedia.org/wiki/OSGi) implementation ( [Equinox](https://en.wikipedia.org/wiki/Equinox_%28OSGi%29" \o "Equinox (OSGi))) in version 3 of Eclipse, plug-ins can be plugged-stopped dynamically and are termed (OSGI) bundles[[9]](https://en.wikipedia.org/wiki/Eclipse_%28software%29#cite_note-9)

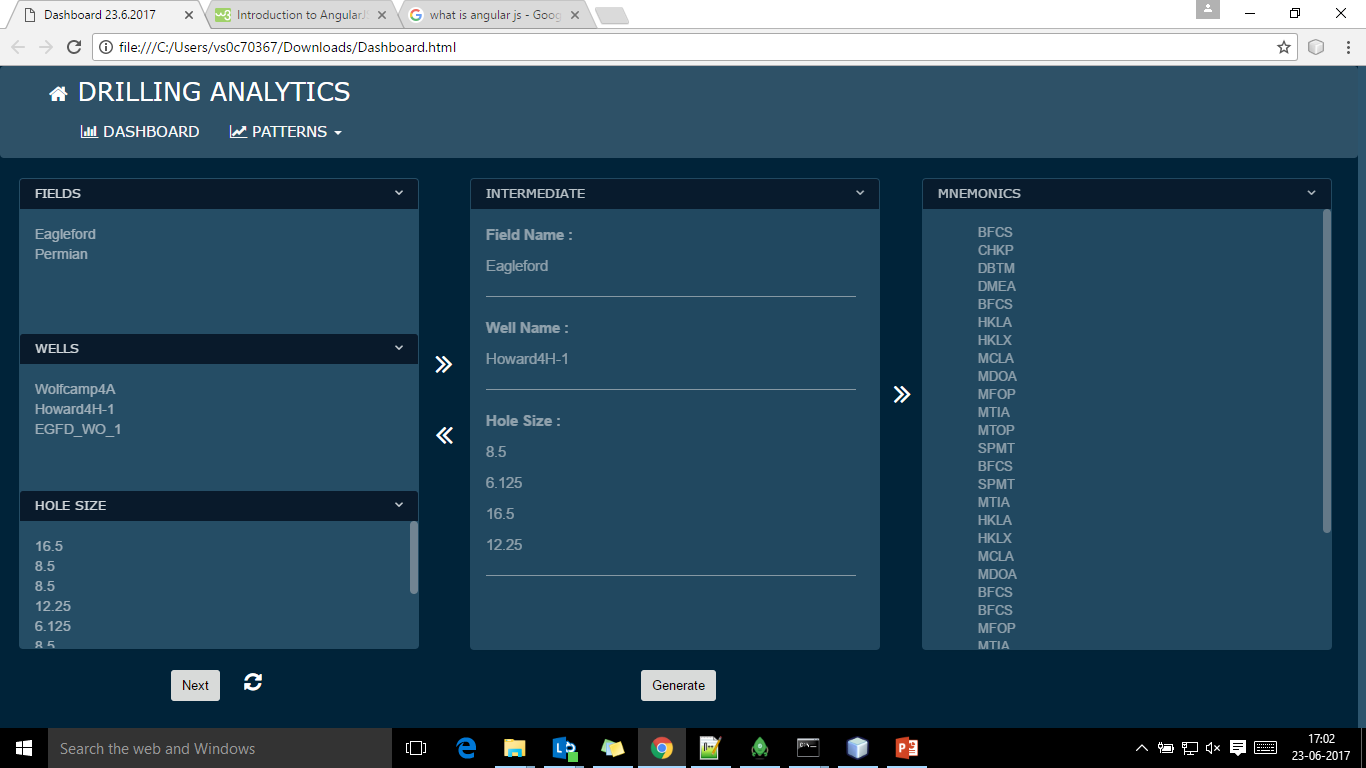
Eclipse [software development kit](https://en.wikipedia.org/wiki/Software_development_kit) (SDK) is [free and open-source software](https://en.wikipedia.org/wiki/Free_and_open-source_software), released under the terms of the [Eclipse Public License](https://en.wikipedia.org/wiki/Eclipse_Public_License), although it is incompatible with the [GNU General Public License](https://en.wikipedia.org/wiki/GNU_General_Public_License).[[10]](https://en.wikipedia.org/wiki/Eclipse_%28software%29#cite_note-10) It was one of the first IDEs to run under [GNU Classpath](https://en.wikipedia.org/wiki/GNU_Classpath) and it runs without problems under [IcedTea](https://en.wikipedia.org/wiki/IcedTea).

# Outputs and Results

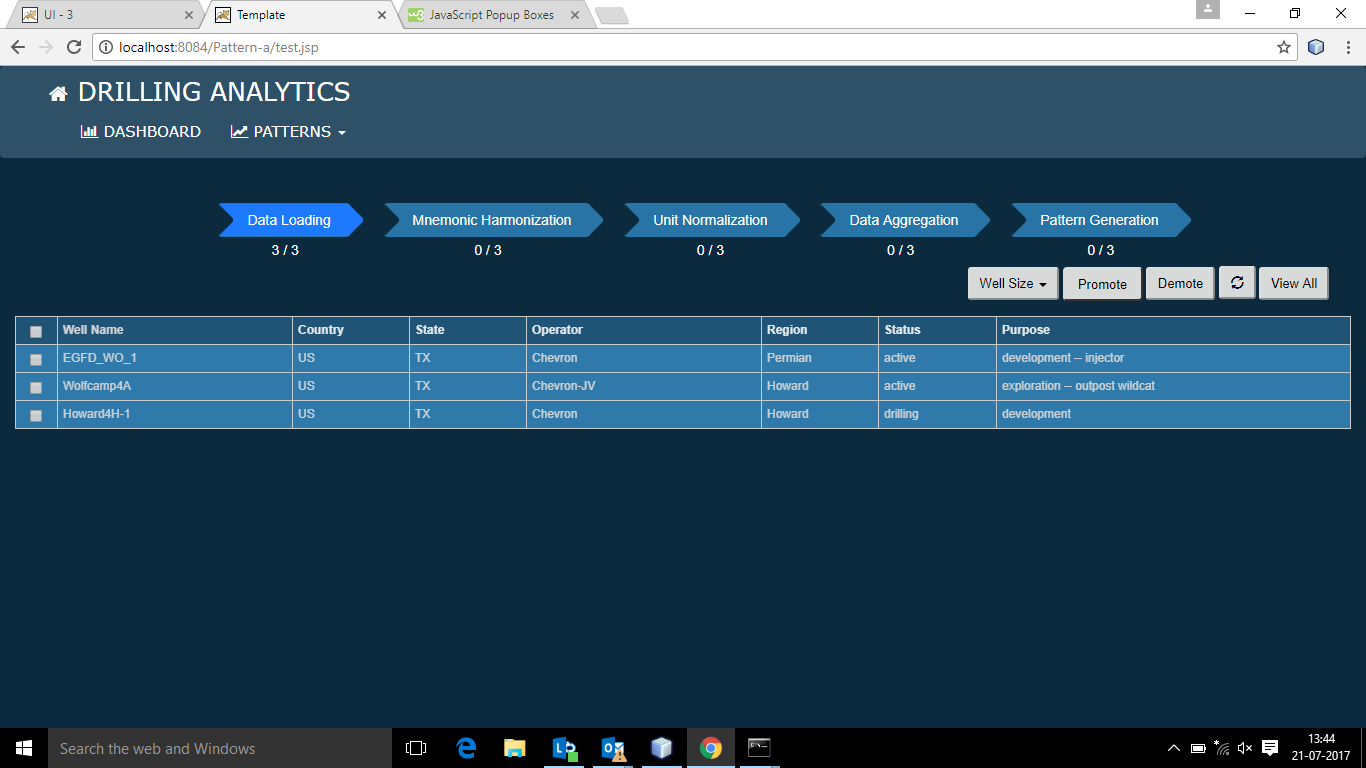
**8.1 Result of the frontend**



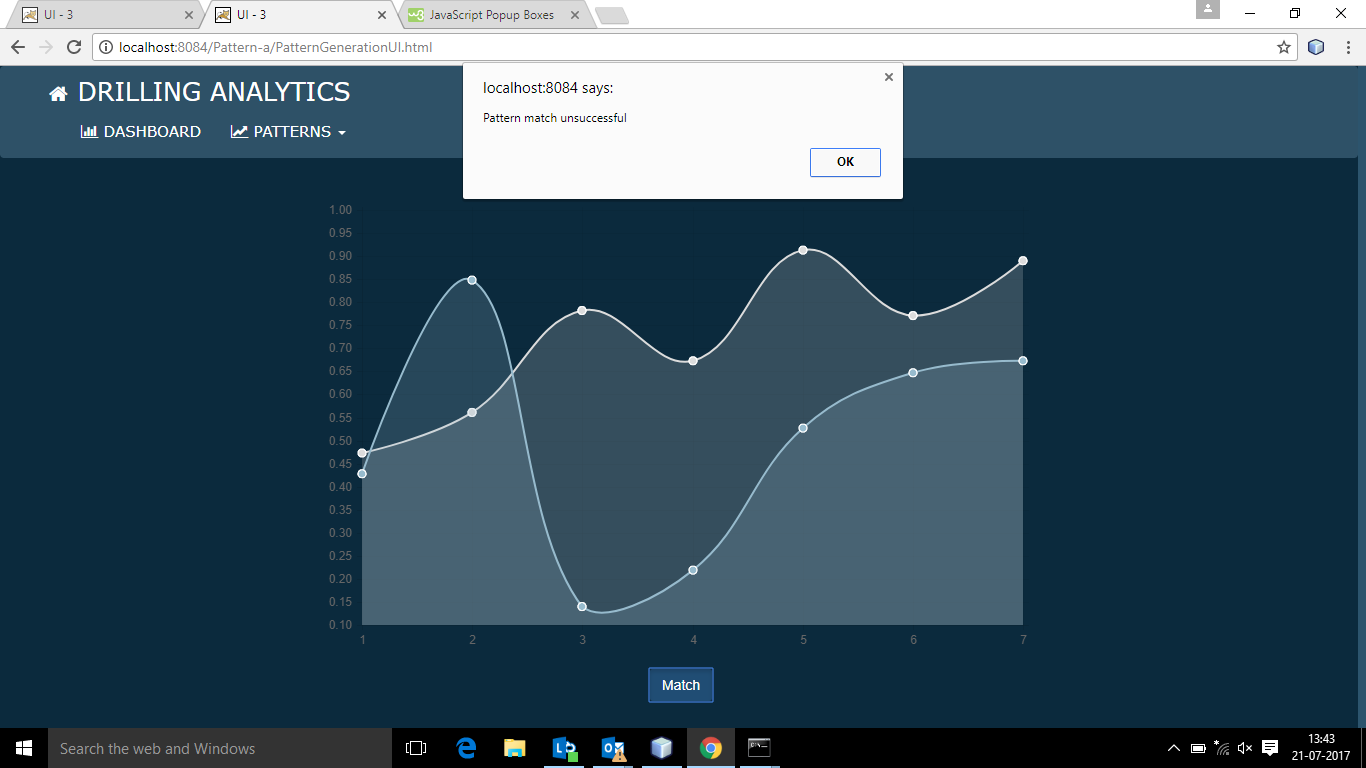
This is the sample UI that is made using jsp and html. Currently the lists are empty. This is how the first screen would look like.



After the selection of Fields, Wells and Hole size the selected parameters are displayed in the intermediate stage. And then it gives the names of all the mnemonics that were under it.

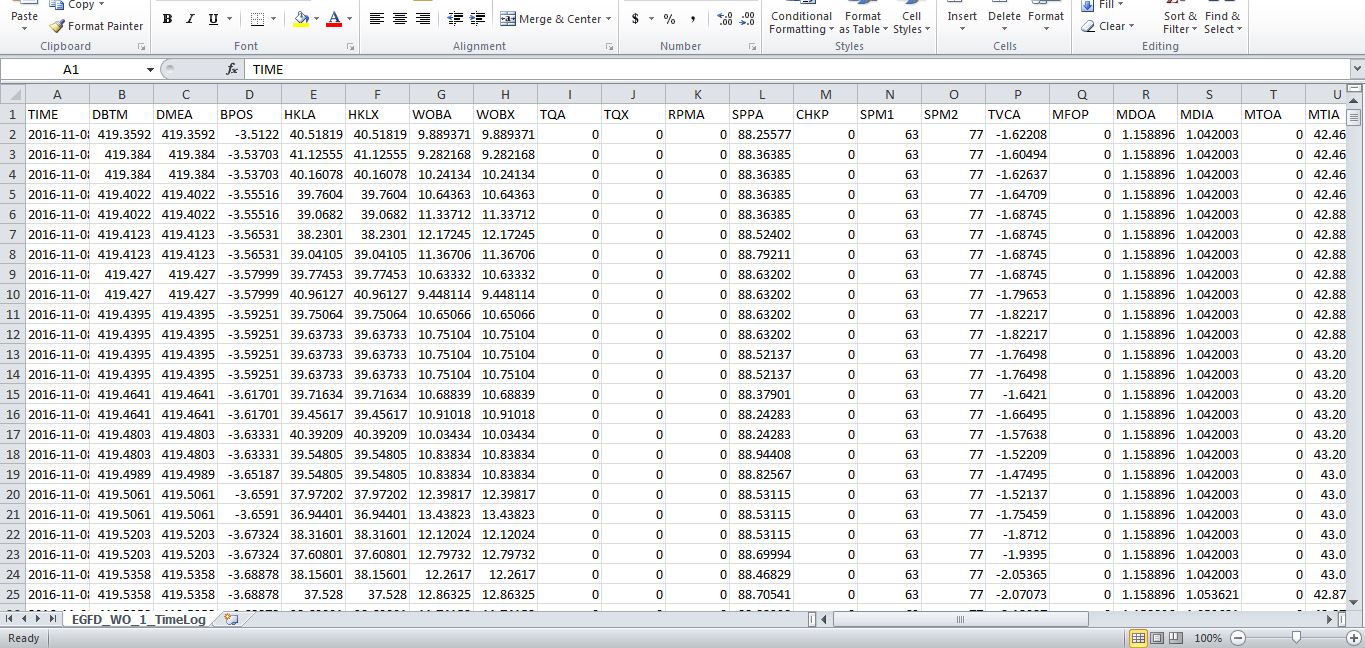


This is a dashboard or everything at a glance which shows the field selection and what happened when we promote and demote the rigs.

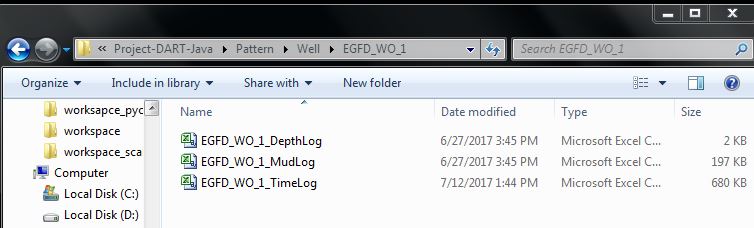


This is a sample pattern generation graph which will fetch the data from the backend and plot the points.

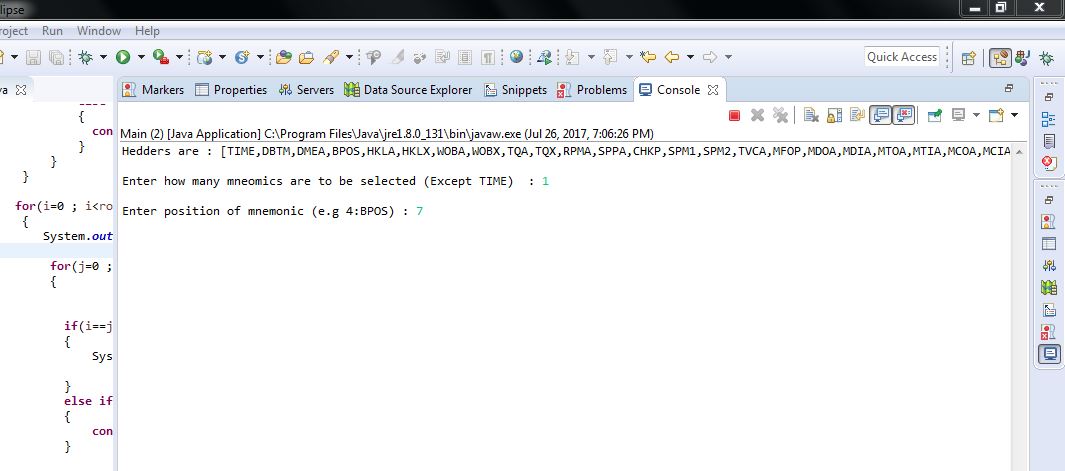
**8.2 Result and outcome from Backend.**



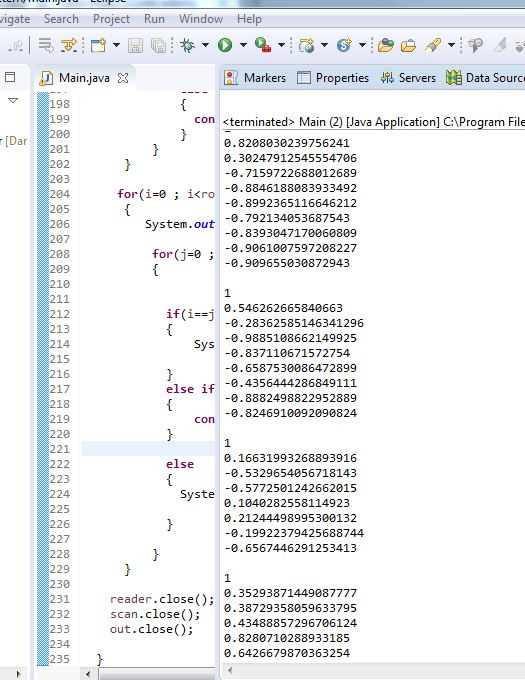
The output of the python script is this csv file.



The sub-module of python creates a directory structure that puts the Log files in their respective parent folder i.e. their well names.



Java Program (Main module) asking for number of mnemonics and the index of it.



Correlation output generated which is further saved into a text file.

# 

# 

The Pattern Library is initiated by generating the correlation matrix.

# Inference and Future Scope of the project

**9.1 Inference**

After the correlations are generated among the selected mnemonics in a log table which had more than 2000 rows, all the data was found to be correct and they were tallied manually in another correlation module of the test of integrity and found to be true. All over it preformed according to the planned modules and delivered the outputs in the expected format. There was the issue with time and space complexity as generating a correlation among more than 2000 mnemonics values in a matrix format i.e. generating correlation value for arrays in a number as large as factorial of >2000. So time required was bit more for computation. In future with use of better algorithms we can comply with original standards and deliver the output as fast as possible by optimizing the existing algorithms that are used in the program.

In terms of space complexity the output file of the correlation matrix which was computer against ~= 2000 rows was found to be a size of 45 MB. So this was the problem in keeping the data in place as the heap space would run out of space of any console. In order to avoid that a txt file is created so that we can compute from that file later on.

Another aspect is the data that we are getting out of the csv have to be entered manually (location of the file). So an another approach can be as soon as we get the input request from the UI we can segregate the values like name mnemonics and the well name and then pass these two as arguments to the main module of java. This is fully automating the task and no further intrusion is required in the backend to manually map the files.

**9.2 Future Scope of the project**

The modules completed so far has only the correlation matrix generated. Now the task would be to generate a cluster and make patterns out of it. It should be an easy task to make cluster out of a correlation matrix as we need to pick only the cells those have a value more than 0.97 and store then in a separate matrix and then run a graph plot function. And there it is, a pattern is generated. This will be sent to the UI (front end).

This was only done with emprises on the drilling data analytics it can be further stretched to stock market trading and in other domain of this kind of nature that needs to generate pattern based matching for decision making.

# Bibliography and References

1. [www.google.com](http://www.google.com)
2. [www.wikipedia.com](http://www.wikipedia.com)
3. [www.stackoverflow.com](http://www.stackoverflow.com)
4. <https://docs.python.org/2/library/csv.html>
5. <https://poi.apache.org/>