**Wallet Hub assignment for code development**

This utility tool is developed using the maven project on NetBeans IDE. Java jdk 1.8\_121; MySql version: mysql-5.7.21 on MS Windows 10 platform.

The deliverables are available in the dropBox site, <https://www.dropbox.com/sh/ew1pb3dj62eqsgg/AAD4IJS9K7cwO1uY4FqYWflpa?dl=0>

The assignment deliverables are:

1. **config.properties:** configuration file for the mysql database properties
2. parser.jar: this tool works straight out of the box
3. **parser\_project\_sources.zip**: project source code files
4. **SQL Scripts**: This folder contains the DDL file to create the database schema
5. **Wallet Hub assignment.docx**: Read Me document

**Features**

**Features of java:**

1. The program is developed using the Mysql's 'LOAD DATA …' feature which is very efficient on loading the huge data into MySql Tables. The jdbc connection is opened at once and the jdbc data load command is executed by opening a data steam. As per the MySql Documentation, this is 20 times faster than traditional jdbc data inserts. It took about 3 sec to insert all the 116,000+ records when tested.
2. Appropriate validations are implemented to not to provide any irrelevant command line params by the user

**Features of DATABASE**

The table is simple one named, 'server\_log' as attached in the ddl sql. This has a primary constraint implemented on the 2 columns, '**incident\_date' and 'ipaddress'.**

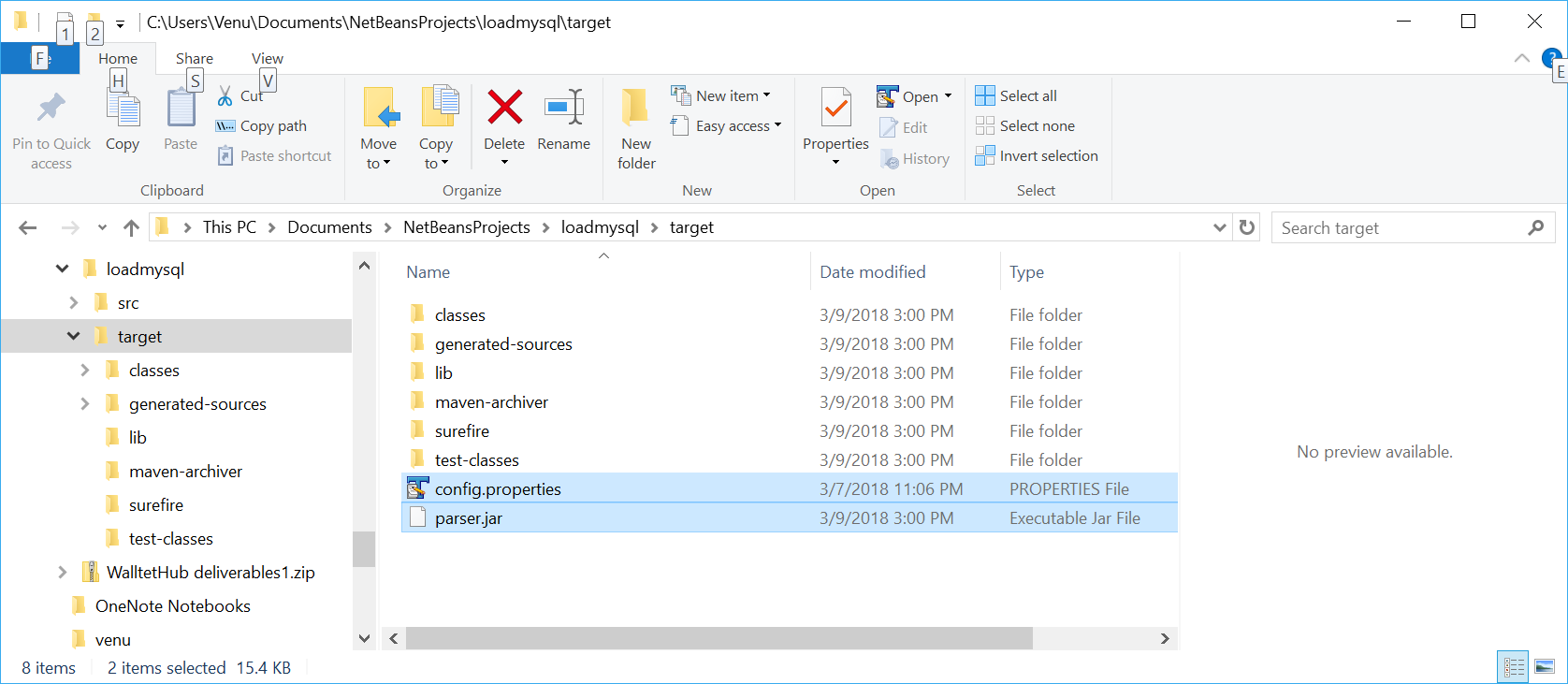
**The purpose is that even the user runs the tool with same command line arg values, the duplicate data does not get inserted. This rule check is applied on the DATABASE server side.**

Procedure to generate the tool from provided java sources.

1. Import the java sources, (extract the sources of parser\_project.zip) into the IDE as a maven project.
2. Initiate the maven build command to generate the jar file. This file has all the required jar library dependencies packaged and the manifest info specifies the runnable main class. This single jar file is all we need to run the tool.

In our case, the generated executable jar is '**parser.jar**'. I used the maven's spring-boot plugin to achieve this.

Below screen shot shows the generated file which is highlighted. Also, the config.properties is manually copied from the same path where the 'src' folder exists.



1. Copy the 'config.properties' from the path where the src is co located. Make necessary changes to this file as needed by the MySql database properties. The tool accesses the file, 'access.log' as mentioned by the property, 'server.data.log.file.name' described in the config.properties. The location of the access.log can be either absolute path or relative. When you wanted to make it relative, then, specify just the file name and leave this file, 'access.log' in the same path where you are placing the tool, 'parser.jar'. (Please read the comments describing it).

Note: make sure that both the config.properties and parser.jar are located in the same path before launching the tool

Here are the screen shots for the java and SQL questions.

**Java assignment**

**Here is how I executed the tool on command line. After building the project, (maven clean install) I switched to the "target" folder. Made sure the file, config.properties is manually updated with appropriate changes to it.**

**Question 1**:

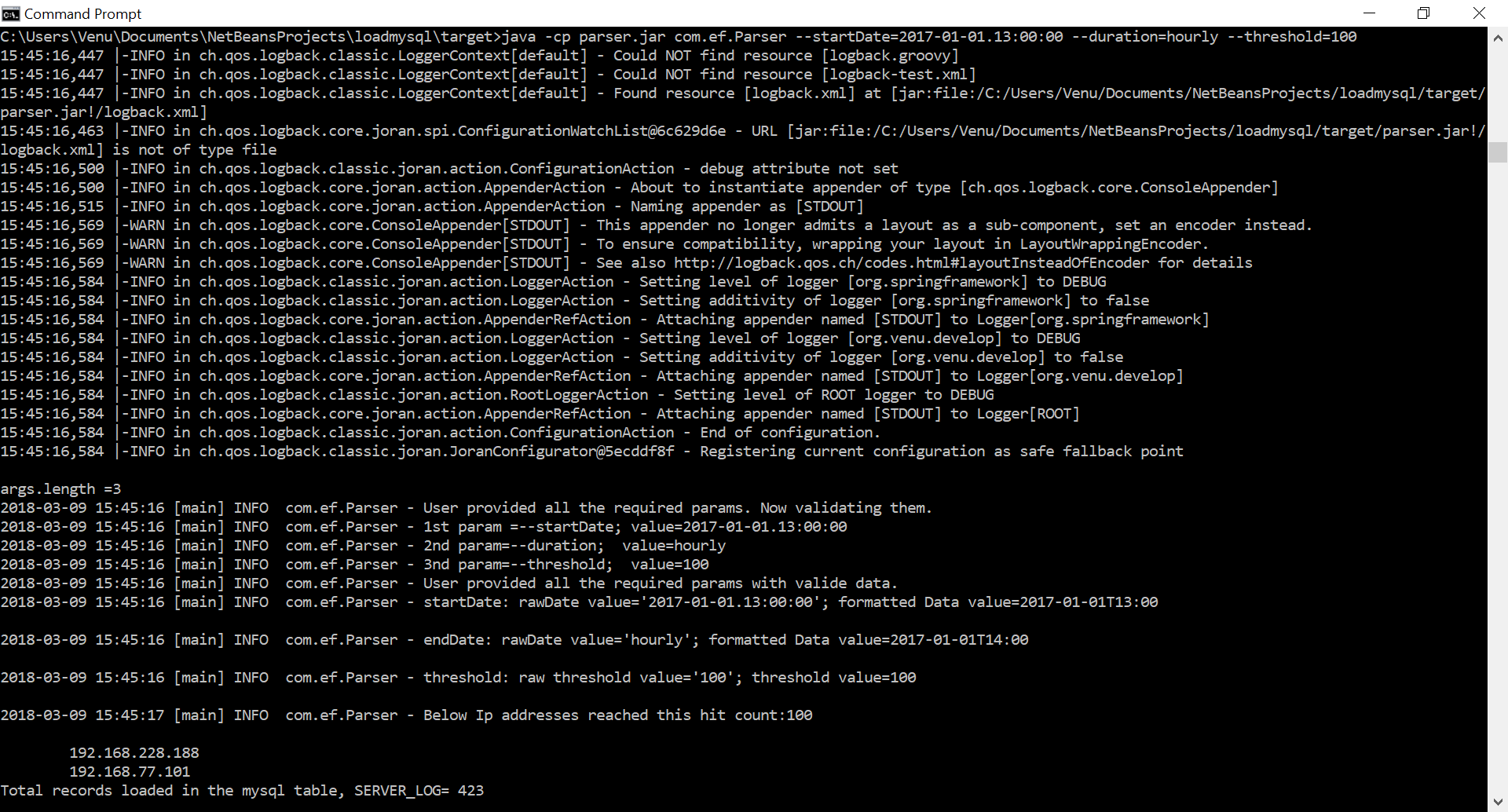
Run the tool against the access.log file to save all the records matching the criteria specified by the user. Criteria is: --startDate=2017-01-01.13:00:00 --duration=hourly --threshold=100

**SOLUTION:**

java -cp "parser.jar" com.ef.Parser --startDate=2017-01-01.13:00:00 --duration=hourly --threshold=100

**Alternatively**,

java -jar parser.jar --startDate=2017-01-01.13:00:00 --duration=hourly --threshold=100



The tool will find any IPs that made more than 100 requests starting from 2017-01-01.13:00:00 to 2017-01-01.14:00:00 (one hour) and print them to console AND also load them to another MySQL table with comments on why it's blocked.

**Question 2**

The tool will find any IPs that made more than 250 requests starting from 2017-01-01.13:00:00 to 2017-01-02.13:00:00 (24 hours) and print them to console AND also load them to another MySQL table with comments on why it's blocked.

**SOLUTION:**

java -cp "parser.jar" com.ef.Parser --startDate=2017-01-01.13:00:00 --duration=daily --threshold=250

**Alternatively**,

java -jar "parser.jar" --startDate=2017-01-01.13:00:00 --duration=daily --threshold=250

**Standalone executable jar:**

This jar contains all the dependencies and the main runnable java class,com.ef.Parser is specified as the Main launcher in the Manifest file.

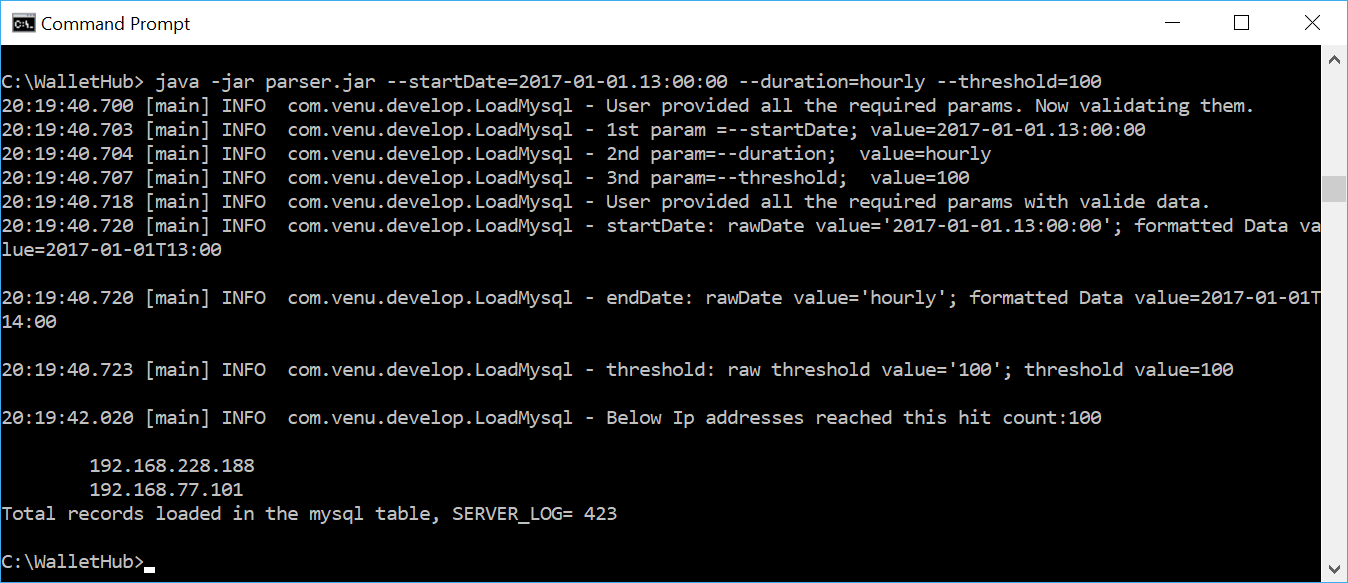
To generate this jar file, just uncomment the 1st build tag (Line 44 to 75) and comment the 2nd build tag (Line 82 to 153) in the "pom.xml".

After this change run the maven to build the project.

1. Copy the generated parser.jar file from target folder to any where to a directory and can be launched as a standalone tool.
2. Copy the 'config.properties' from the path where the src is located to one next immediate parent dir where the "parser.jar" is moved to. Make changes to the config.properties.

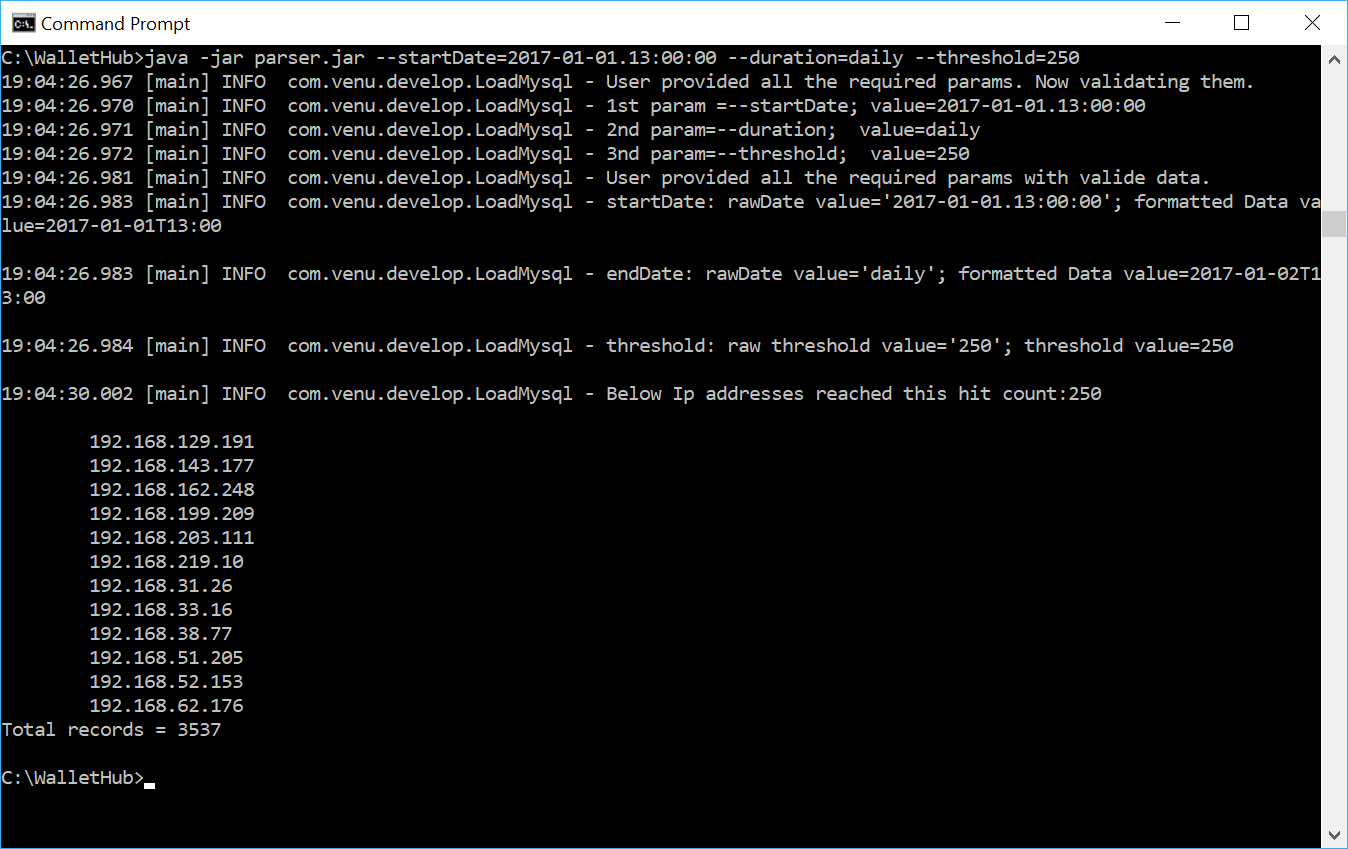
Screenshot showing the standalone execution of the tool:

java -jar parser.jar --startDate=2017-01-01.13:00:00 --duration=hourly --threshold=100



2.Below screen shot for the second assignment task,

java -jar "parser.jar" --startDate=2017-01-01.13:00:00 --duration=daily --threshold=250



**SQL Assignment**

(1) Write MySQL query to find IPs that mode more than a certain number of requests for a given time period.

Ex: Write SQL to find IPs that made more than 100 requests starting from 2017-01-01.13:00:00 to 2017-01-01.14:00:00.

**SOLUTION**:

1. Sql query:

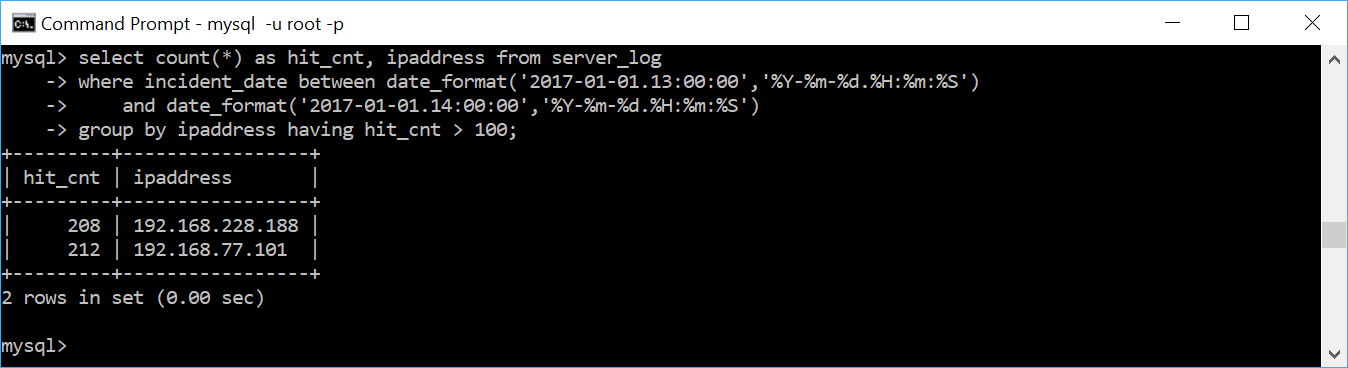
**select count(\*) as hit\_cnt, ipaddress from server\_log**

**where incident\_date between date\_format('2017-01-01.13:00:00','%Y-%m-%d.%H:%m:%S')**

**and date\_format('2017-01-01.14:00:00','%Y-%m-%d.%H:%m:%S')**

**group by ipaddress having hit\_cnt > 100;**

**2. Screen shot on mysql client showing the results**



(2) Write MySQL query to find requests made by a given IP.

**SOLUTION**:

1. **Sql query:**

select count(\*) as hit\_cnt, ipaddress from server\_log where ipaddress='192.168.129.191';

1. **Screen shot on mysql client showing the results**

