A Study on Consumer Complaints

Prepared by:

- **Saurabh Dhoble**
- **Priyank Dsilva**

With Guidance from:

- **Jaeki Song, Ph.D**

TEXAS TECH UNIVERSITY
Rawls College *of* Business

October 15, 2015

Executive Summary

As the world progresses together with the increasing purchasing power of the people, all businesses got larger scale of customers to serve their services too. Expansion of businesses along with competition with competitors requires business owners to have good satisfaction index of their clients or customers. Insight from customers about the services being offered would be a key performance indicator about its standing in market. Records of complaints about service being offered by service providers are one of the prime sources to get trajectory of how a business is accepted in the market. For financial companies, it can serve as a regulatory check about its standing in market .It also helps Federal bureau to understand working of financial companies and their impact on people and the market, it will also help them have better governance. Here we would like to analyze a Consumer Complaints database which has complaints received about financial products and services of companies in United States of America. Findings of this analysis would serve Financial companies, Federal Bureau as well as people to understand quality of service being offered by vendors and would help them be in a better decision making position to spend their purchasing power.

Table of Contents

1. Introduction 4
1.1 Why do we need it? 4
1.2 Objective of Report 4
1.3 The target audience for this report: 4
2. What is Consumer Complaint Database? How do we get it? 5
3. Implementation Details: 6
3.1 Data Base Design 6
3.2 Analyze Data 10
3.3 Visualize Data 12
3.3.1 Public: 12
3.3.2 Company: 13
3.3.3 Federal Bureau: 14
3.3.4 Federal Bureau Dashboard: 15
4. Guidelines to use Consumer Complaints Utility 16
4.1 Requirement: 16
4.2 Issue encountered with python Libraries(Anaconda 3.4): 16
4.3 Modules: 17
5. Conclusion 18

1. Introduction

1.1 Why do we need it?

We live in a competitive era, we are being offered a same product or service from various vendors. It becomes at most important to understand the best among the choices which help us have good satisfaction. Financial products and services sector have significant impact on our lives as they are directly related to our finances. It becomes need of hour to have proper understanding about how we can make correct decisions to buy a product or a service. Here in this report we analyze Customer complaints about various financial products to answer these questions.

1.2 Objective of Report

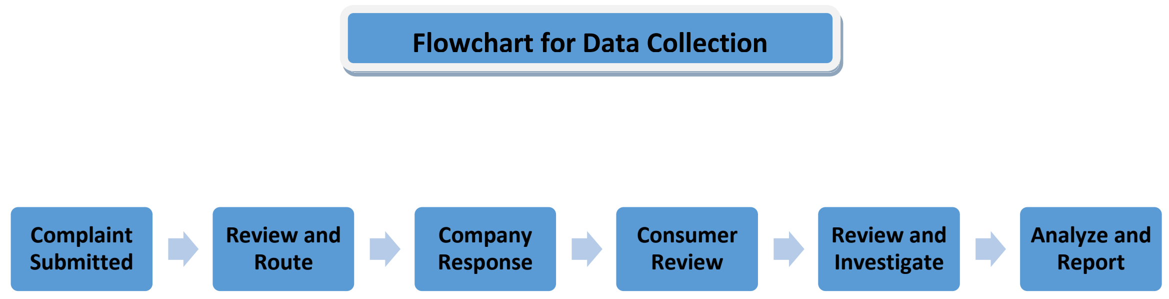
This report aims to visualize findings of complaints received for financial products and services of companies. It helps us understand how each product of a category is comparable to other in terms of number of complaints and user feedbacks on the same. In the process we also learn how this aim was achieved using techniques of python and its strong visualization libraries.

1.3 The target audience for this report:

- Public: It allows common people to have clear understanding about performance and acceptance of a product or service in a market.
- Financial Companies: It allows financial firms to compare product performance with their competitors in market, which help them make better business decisions to improve their market stand.
- Consumer Financial Protection Bureau: It helps government firms to have check and control on financial companies in market and have check on their performance and impact.

2. What is Consumer Complaint Database? How do we get it?

It is important to know how authentic our data is to predict our decision making on, as two of our valuable resources Time and Money are invested on the same. Below is flow diagram of the process is implemented to collect data by consumer Forum.



On technical perspective to use this data, There were various forms of data provided by consumer finance forum example (csv , json using API).We have used API(csv) to get latest dataset from financial consumer forum. This option serves the best if you want to have the most recently updated database synced with your system.

We used an open source database named Firebird to store our data. It is one of the fastest upcoming relational database which is free to use and has a strong community support.

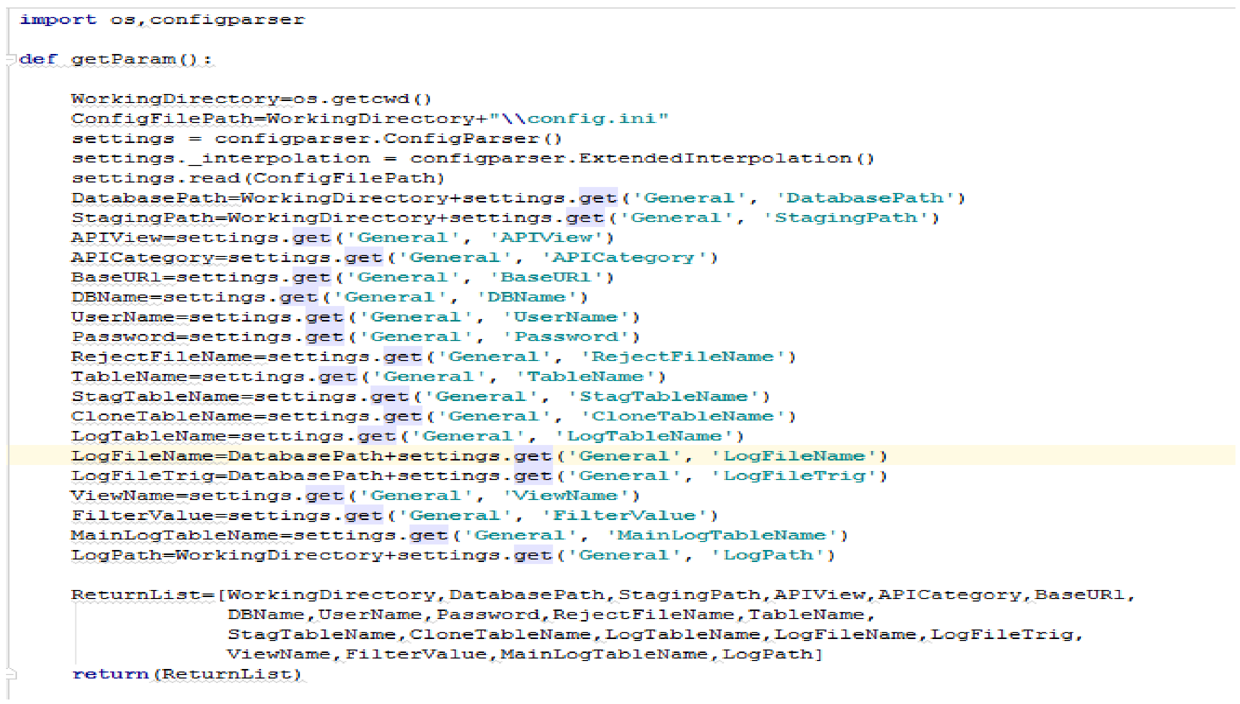
3. Implementation Details:

A three step mechanism was implemented to reach our goal of presenting meaningful interpretations for Consumer complaints data.

3.1 Data Base Design

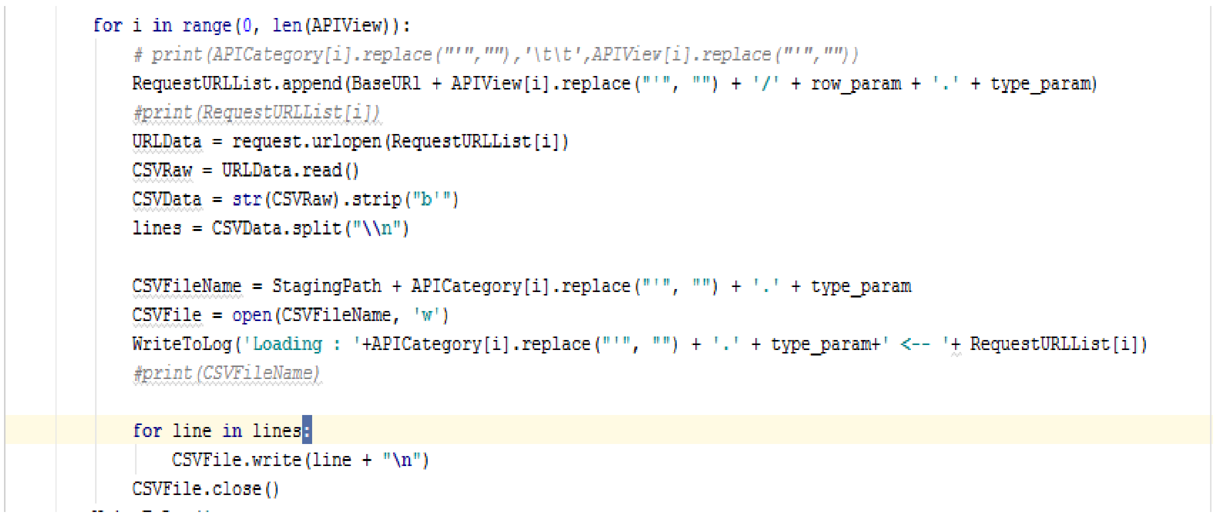
Primary task after selection of data source was to build a database architecture. We have fetched data from the data source using API's provided by consumer finance forum. Segregated data into files depending on categories of complaints registered. As feature of database architecture we have imported data into staging tables. After staging it is stored in our database. Concepts of Database Swap and Database cloning have been implemented so that we have hundred percent application up time and security of our database in case of technical failures. A strong Log maintenance mechanism has been incorporated to have proper record of the process Database used is Firebird. It's an open source database. Factors which led us to choose this relational database were Cost efficiency, Strong user community support, scalability and simple interface for implementation. To know further about Firebird database please follow http://www.firebirdsql.org/en/about-firebird/ ..
Below section of code snippets helps us understand our database architecture in a broader scope.

Configuration data:



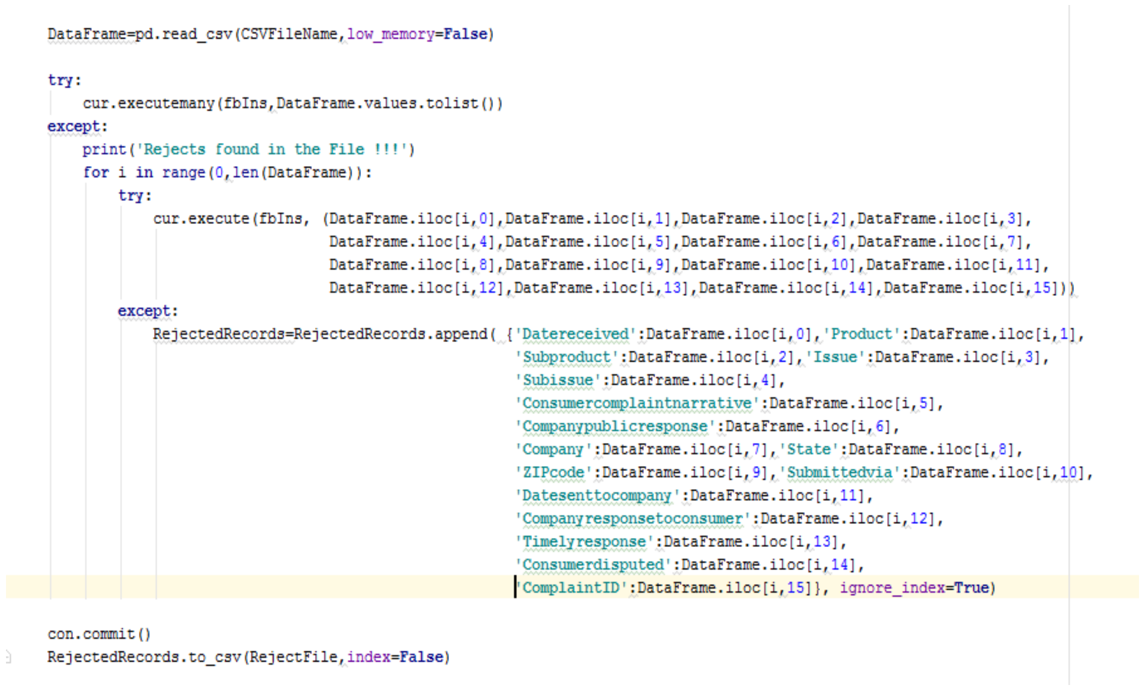
```
import os,configparser

def getParam():

    WorkingDirectory=os.getcwd()
    ConfigFilePath=WorkingDirectory+"\\config.ini"
    settings = configparser.ConfigParser()
    settings._interpolation = configparser.ExtendedInterpolation()
    settings.read(ConfigFilePath)
    DatabasePath=WorkingDirectory+settings.get('General', 'DatabasePath')
    StagingPath=WorkingDirectory+settings.get('General', 'StagingPath')
    APIView=settings.get('General', 'APIView')
    APICategory=settings.get('General', 'APICategory')
    BaseURl=settings.get('General', 'BaseURl')
    DBName=settings.get('General', 'DBName')
    UserName=settings.get('General', 'UserName')
    Password=settings.get('General', 'Password')
    RejectFileName=settings.get('General', 'RejectFileName')
    TableName=settings.get('General', 'TableName')
    StagTableName=settings.get('General', 'StagTableName')
    CloneTableName=settings.get('General', 'CloneTableName')
    LogTableName=settings.get('General', 'LogTableName')
    LogFileName=DatabasePath+settings.get('General', 'LogFileName')
    LogFileTrig=DatabasePath+settings.get('General', 'LogFileTrig')
    ViewName=settings.get('General', 'ViewName')
    FilterValue=settings.get('General', 'FilterValue')
    MainLogTableName=settings.get('General', 'MainLogTableName')
    LogPath=WorkingDirectory+settings.get('General', 'LogPath')

    ReturnList=[WorkingDirectory,DatabasePath,StagingPath,APIView,APICategory,BaseURl,
                DBName,UserName,Password,RejectFileName,TableName,
                StagTableName,CloneTableName,LogTableName,LogFileName,LogFileTrig,
                ViewName,FilterValue,MainLogTableName,LogPath]
    return(ReturnList)
```

Fetching Data from API's:



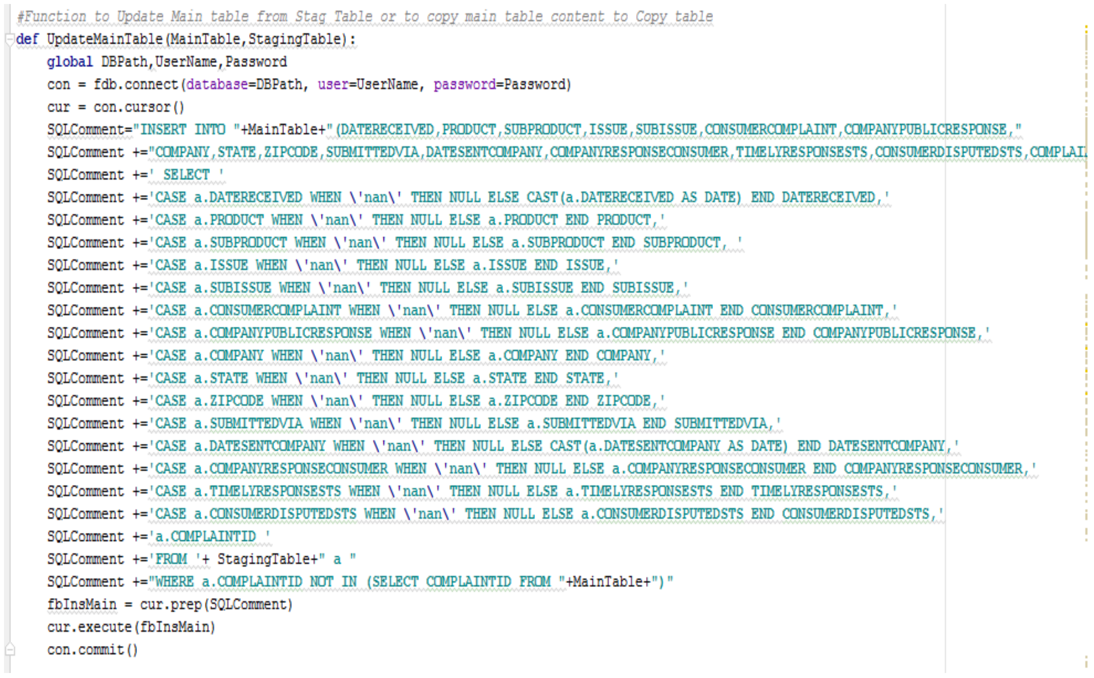
```
for i in range(0, len(APIView)):
    # print(APICategory[i].replace("'",""),'\t\t',APIView[i].replace("'",""))
    RequestURLList.append(BaseURl + APIView[i].replace("'", "") + '/' + row_param + '.' + type_param)
    #print(RequestURLList[i])
    URLData = request.urlopen(RequestURLList[i])
    CSVRaw = URLData.read()
    CSVData = str(CSVRaw).strip("b'")
    lines = CSVData.split("\\n")

    CSVFileName = StagingPath + APICategory[i].replace("'", "") + '.' + type_param
    CSVFile = open(CSVFileName, 'w')
    WriteToLog('Loading : '+APICategory[i].replace("'", "") + '.' + type_param+' <-- '+ RequestURLList[i])
    #print(CSVFileName)

    for line in lines:
        CSVFile.write(line + "\n")
    CSVFile.close()
```

Loading data to Main table:



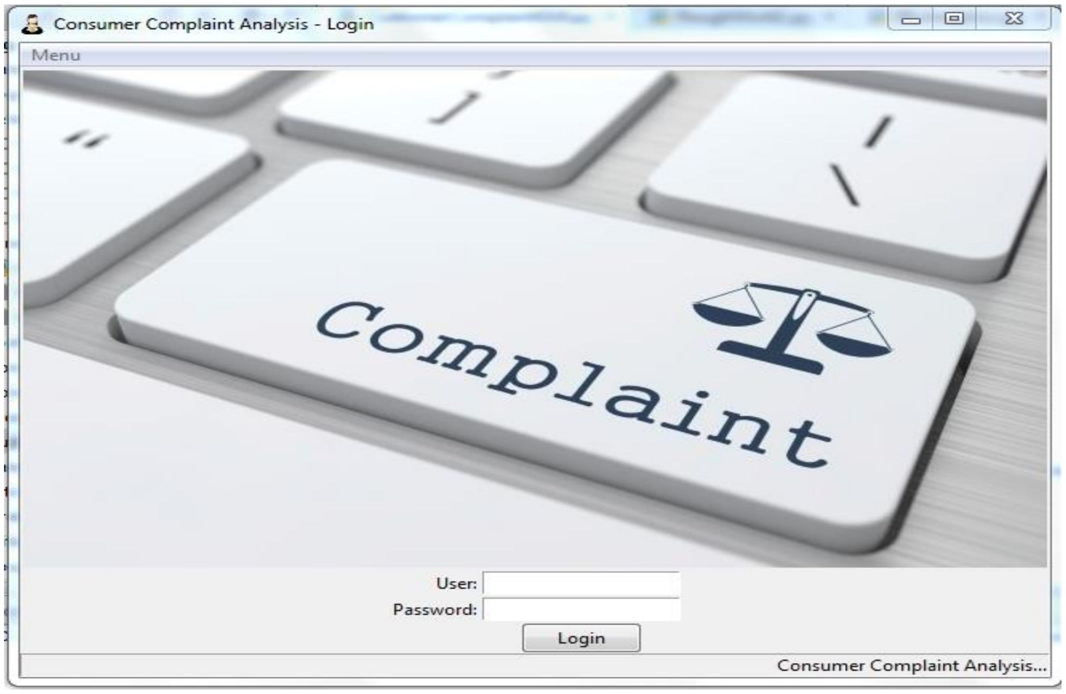
```
DataFrame=pd.read_csv(CSVFileName,low_memory=False)

try:
    cur.executemany(fbIns,DataFrame.values.tolist())
except:
    print('Rejects found in the File !!!')
    for i in range(0,len(DataFrame)):
        try:
            cur.execute(fbIns, (DataFrame.iloc[i,0],DataFrame.iloc[i,1],DataFrame.iloc[i,2],DataFrame.iloc[i,3],
                                DataFrame.iloc[i,4],DataFrame.iloc[i,5],DataFrame.iloc[i,6],DataFrame.iloc[i,7],
                                DataFrame.iloc[i,8],DataFrame.iloc[i,9],DataFrame.iloc[i,10],DataFrame.iloc[i,11],
                                DataFrame.iloc[i,12],DataFrame.iloc[i,13],DataFrame.iloc[i,14],DataFrame.iloc[i,15]))
        except:
            RejectedRecords=RejectedRecords.append( {'Datereceived':DataFrame.iloc[i,0],'Product':DataFrame.iloc[i,1],
                                                    'Subproduct':DataFrame.iloc[i,2],'Issue':DataFrame.iloc[i,3],
                                                    'Subissue':DataFrame.iloc[i,4],
                                                    'Consumercomplaintnarrative':DataFrame.iloc[i,5],
                                                    'Companypublicresponse':DataFrame.iloc[i,6],
                                                    'Company':DataFrame.iloc[i,7],'State':DataFrame.iloc[i,8],
                                                    'ZIPcode':DataFrame.iloc[i,9],'Submittedvia':DataFrame.iloc[i,10],
                                                    'Datesenttocompany':DataFrame.iloc[i,11],
                                                    'Companyresponsetoconsumer':DataFrame.iloc[i,12],
                                                    'Timelyresponse':DataFrame.iloc[i,13],
                                                    'Consumerdisputed':DataFrame.iloc[i,14],
                                                    'ComplaintID':DataFrame.iloc[i,15]}, ignore_index=True)

con.commit()
RejectedRecords.to_csv(RejectFile,index=False)
```

Cleansing Stage Data:

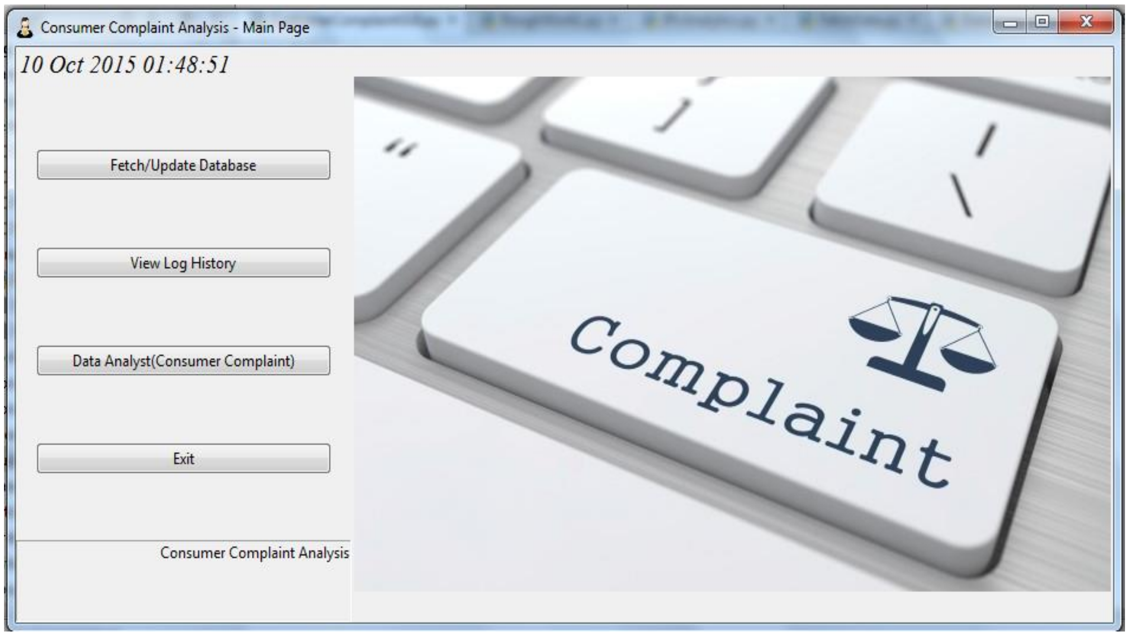


```
#Function to Update Main table from Stag Table or to copy main table content to Copy table
def UpdateMainTable(MainTable,StagingTable):
    global DBPath,UserName,Password
    con = fdb.connect(database=DBPath, user=UserName, password=Password)
    cur = con.cursor()
    SQLComment="INSERT INTO "+MainTable+"(DATERECEIVED,PRODUCT,SUBPRODUCT,ISSUE,SUBISSUE,CONSUMERCOMPLAINT,COMPANYPUBLICRESPONSE,"
    SQLComment +="COMPANY,STATE,ZIPCODE,SUBMITTEDVIA,DATESENTCOMPANY,COMPANYRESPONSECONSUMER,TIMELYRESPONSESTS,CONSUMERDISPUTEDSTS,COMPLAI
    SQLComment +=' SELECT '
    SQLComment +='CASE a.DATERECEIVED WHEN \'nan\' THEN NULL ELSE CAST(a.DATERECEIVED AS DATE) END DATERECEIVED,'
    SQLComment +='CASE a.PRODUCT WHEN \'nan\' THEN NULL ELSE a.PRODUCT END PRODUCT,'
    SQLComment +='CASE a.SUBPRODUCT WHEN \'nan\' THEN NULL ELSE a.SUBPRODUCT END SUBPRODUCT, '
    SQLComment +='CASE a.ISSUE WHEN \'nan\' THEN NULL ELSE a.ISSUE END ISSUE,'
    SQLComment +='CASE a.SUBISSUE WHEN \'nan\' THEN NULL ELSE a.SUBISSUE END SUBISSUE,'
    SQLComment +='CASE a.CONSUMERCOMPLAINT WHEN \'nan\' THEN NULL ELSE a.CONSUMERCOMPLAINT END CONSUMERCOMPLAINT,'
    SQLComment +='CASE a.COMPANYPUBLICRESPONSE WHEN \'nan\' THEN NULL ELSE a.COMPANYPUBLICRESPONSE END COMPANYPUBLICRESPONSE,'
    SQLComment +='CASE a.COMPANY WHEN \'nan\' THEN NULL ELSE a.COMPANY END COMPANY,'
    SQLComment +='CASE a.STATE WHEN \'nan\' THEN NULL ELSE a.STATE END STATE,'
    SQLComment +='CASE a.ZIPCODE WHEN \'nan\' THEN NULL ELSE a.ZIPCODE END ZIPCODE,'
    SQLComment +='CASE a.SUBMITTEDVIA WHEN \'nan\' THEN NULL ELSE a.SUBMITTEDVIA END SUBMITTEDVIA,'
    SQLComment +='CASE a.DATESENTCOMPANY WHEN \'nan\' THEN NULL ELSE CAST(a.DATESENTCOMPANY AS DATE) END DATESENTCOMPANY,'
    SQLComment +='CASE a.COMPANYRESPONSECONSUMER WHEN \'nan\' THEN NULL ELSE a.COMPANYRESPONSECONSUMER END COMPANYRESPONSECONSUMER,'
    SQLComment +='CASE a.TIMELYRESPONSESTS WHEN \'nan\' THEN NULL ELSE a.TIMELYRESPONSESTS END TIMELYRESPONSESTS,'
    SQLComment +='CASE a.CONSUMERDISPUTEDSTS WHEN \'nan\' THEN NULL ELSE a.CONSUMERDISPUTEDSTS END CONSUMERDISPUTEDSTS,'
    SQLComment +='a.COMPLAINTID '
    SQLComment +='FROM '+ StagingTable+" a "
    SQLComment +="WHERE a.COMPLAINTID NOT IN (SELECT COMPLAINTID FROM "+MainTable+")"
    fbInsMain = cur.prep(SQLComment)
    cur.execute(fbInsMain)
    con.commit()
```

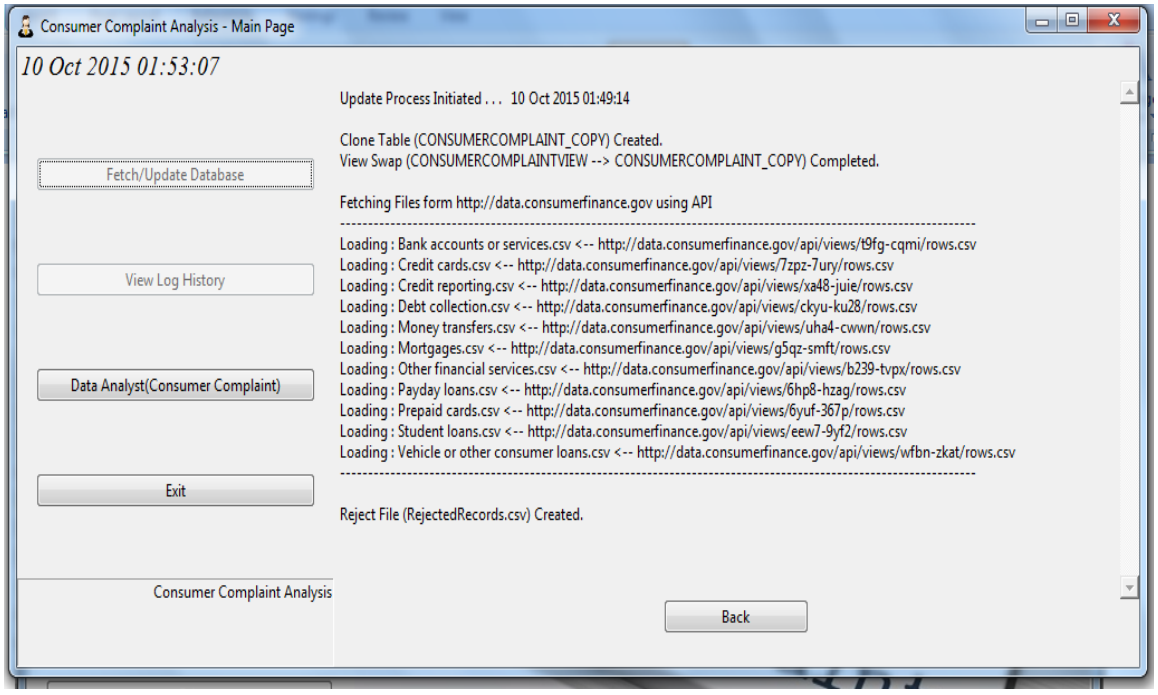
Log In Screen:


Consumer Complaint Analysis - Login
Menu
Complaint
User:
Password:
Login
Consumer Complaint Analysis...

First Screen Displaying Menus:

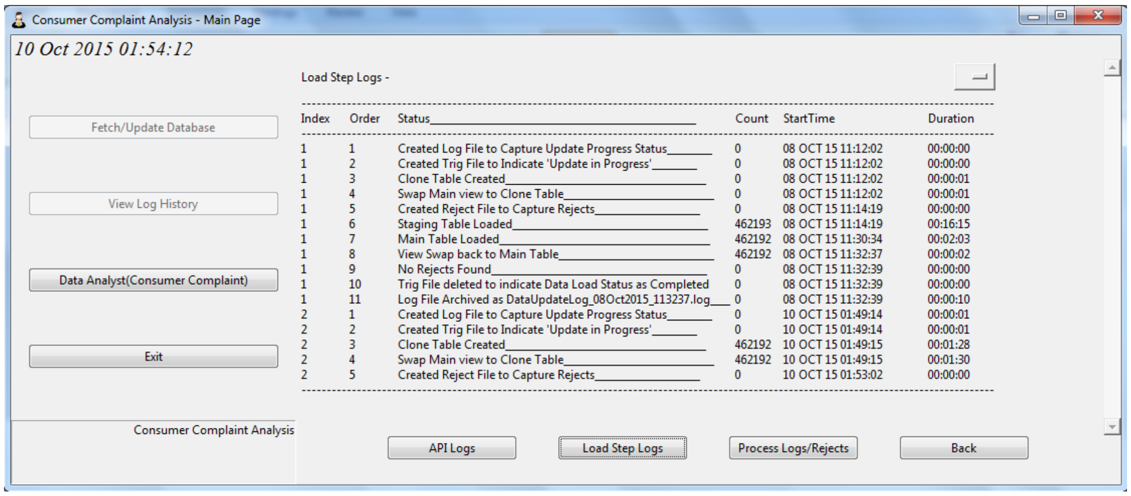

Consumer Complaint Analysis - Main Page
10 Oct 2015 01:48:51
Fetch/Update Database
View Log History
Data Analyst(Consumer Complaint)
Exit
Consumer Complaint Analysis
Complaint

Fetch/Update Database Stage:


Consumer Complaint Analysis - Main Page
10 Oct 2015 01:53:07
Fetch/Update Database
View Log History
Data Analyst(Consumer Complaint)
Exit
Consumer Complaint Analysis
Back

```
Update Process Initiated . . .  10 Oct 2015 01:49:14

Clone Table (CONSUMERCOMPLAINT_COPY) Created.
View Swap (CONSUMERCOMPLAINTVIEW --> CONSUMERCOMPLAINT_COPY) Completed.

Fetching Files form http://data.consumerfinance.gov using API
-----------------------------------------------------------------------------------------------------------
Loading : Bank accounts or services.csv <-- http://data.consumerfinance.gov/api/views/t9fg-cqmi/rows.csv
Loading : Credit cards.csv <-- http://data.consumerfinance.gov/api/views/7zpz-7ury/rows.csv
Loading : Credit reporting.csv <-- http://data.consumerfinance.gov/api/views/xa48-juie/rows.csv
Loading : Debt collection.csv <-- http://data.consumerfinance.gov/api/views/ckyu-ku28/rows.csv
Loading : Money transfers.csv <-- http://data.consumerfinance.gov/api/views/uha4-cwwn/rows.csv
Loading : Mortgages.csv <-- http://data.consumerfinance.gov/api/views/g5qz-smft/rows.csv
Loading : Other financial services.csv <-- http://data.consumerfinance.gov/api/views/b239-tvpx/rows.csv
Loading : Payday loans.csv <-- http://data.consumerfinance.gov/api/views/6hp8-hzag/rows.csv
Loading : Prepaid cards.csv <-- http://data.consumerfinance.gov/api/views/6yuf-367p/rows.csv
Loading : Student loans.csv <-- http://data.consumerfinance.gov/api/views/eew7-9yf2/rows.csv
Loading : Vehicle or other consumer loans.csv <-- http://data.consumerfinance.gov/api/views/wfbn-zkat/rows.csv
-----------------------------------------------------------------------------------------------------------

Reject File (RejectedRecords.csv) Created.
```

Log Details:


Consumer Complaint Analysis - Main Page
Fetch/Update Database
View Log History
Data Analyst(Consumer Complaint)
Exit
Consumer Complaint Analysis
API Logs
Load Step Logs
Process Logs/Rejects
Back

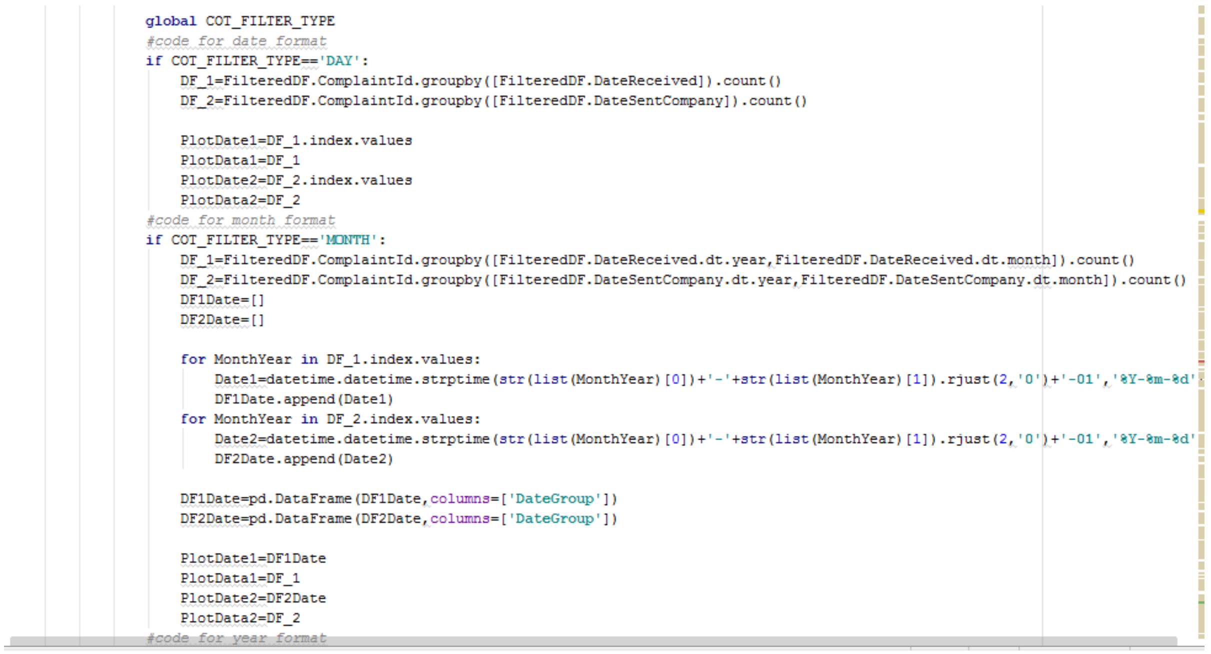
10 Oct 2015 01:54:12

```
Load Step Logs -

-----------------------------------------------------------------------------------------------------------------
Index  Order  Status____________________________________              Count   StartTime              Duration
-----------------------------------------------------------------------------------------------------------------
1      1      Created Log File to Capture Update Progress Status________  0       08 OCT 15 11:12:02     00:00:00
1      2      Created Trig File to Indicate 'Update in Progress'________  0       08 OCT 15 11:12:02     00:00:00
1      3      Clone Table Created_______________________________________  0       08 OCT 15 11:12:02     00:00:01
1      4      Swap Main view to Clone Table_____________________________  0       08 OCT 15 11:12:02     00:00:01
1      5      Created Reject File to Capture Rejects____________________  0       08 OCT 15 11:14:19     00:00:00
1      6      Staging Table Loaded______________________________________  462193  08 OCT 15 11:14:19     00:16:15
1      7      Main Table Loaded_________________________________________  462192  08 OCT 15 11:30:34     00:02:03
1      8      View Swap back to Main Table______________________________  462192  08 OCT 15 11:32:37     00:00:02
1      9      No Rejects Found__________________________________________  0       08 OCT 15 11:32:39     00:00:00
1      10     Trig File deleted to indicate Data Load Status as Completed 0       08 OCT 15 11:32:39     00:00:00
1      11     Log File Archived as DataUpdateLog_08Oct2015_113237.log___  0       08 OCT 15 11:32:39     00:00:10
2      1      Created Log File to Capture Update Progress Status________  0       10 OCT 15 01:49:14     00:00:01
2      2      Created Trig File to Indicate 'Update in Progress'________  0       10 OCT 15 01:49:14     00:00:01
2      3      Clone Table Created_______________________________________  462192  10 OCT 15 01:49:15     00:01:28
2      4      Swap Main view to Clone Table_____________________________  462192  10 OCT 15 01:49:15     00:01:30
2      5      Created Reject File to Capture Rejects____________________  0       10 OCT 15 01:53:02     00:00:00
-----------------------------------------------------------------------------------------------------------------
```

3.2 Analyze Data

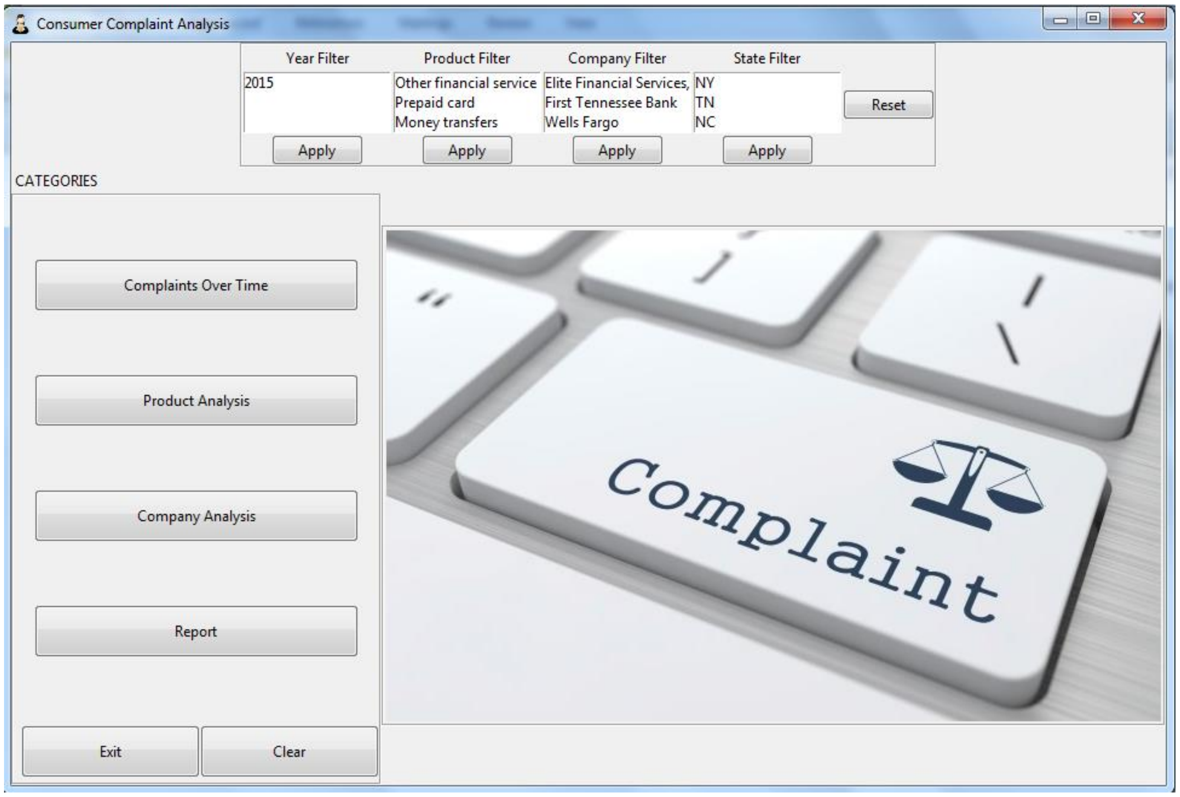
After setting up a strong and reliable database next step included cleaning data and storing the non redundant data in proper format which would help us draw logical insights.
Data staging table were implemented ,which serves as a layer before final database entries are made, views were created which hides from user the core implementation details about database swap, in short logic behind how we provide hundred percent application availability. Lets consider important code snippet we implemented to have dynamic data frames, followed by User Interface of the same in the utility.

Data frame code:

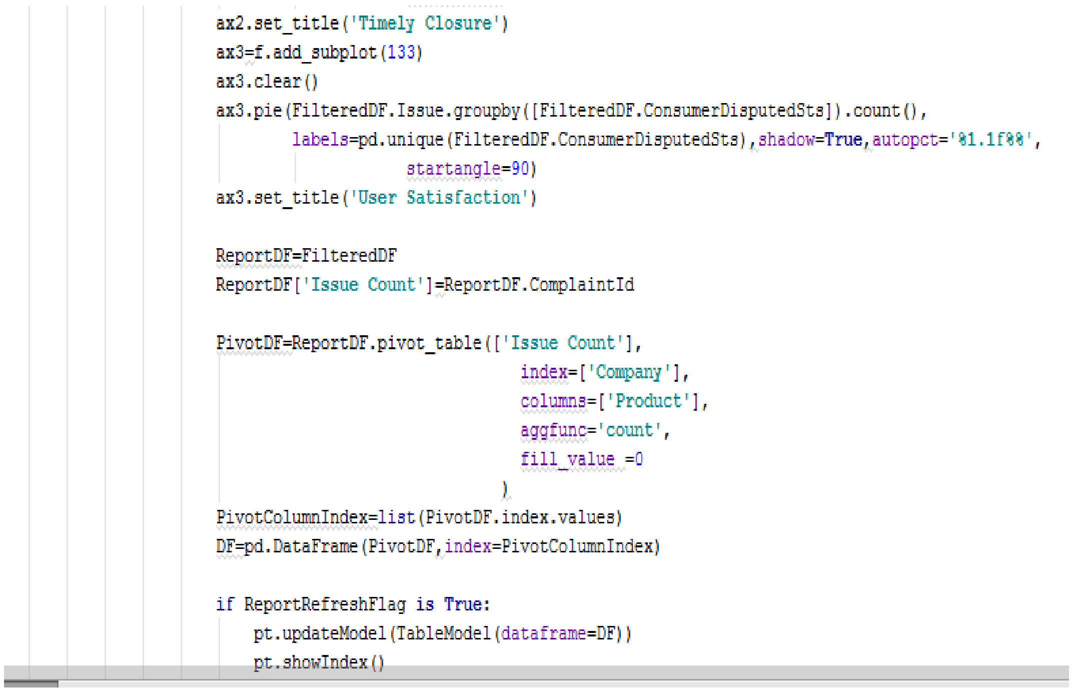


```
global COT_FILTER_TYPE
#code for date format
if COT_FILTER_TYPE=='DAY':
    DF_1=FilteredDF.ComplaintId.groupby([FilteredDF.DateReceived]).count()
    DF_2=FilteredDF.ComplaintId.groupby([FilteredDF.DateSentCompany]).count()

    PlotDate1=DF_1.index.values
    PlotData1=DF_1
    PlotDate2=DF_2.index.values
    PlotData2=DF_2
#code for month format
if COT_FILTER_TYPE=='MONTH':
    DF_1=FilteredDF.ComplaintId.groupby([FilteredDF.DateReceived.dt.year,FilteredDF.DateReceived.dt.month]).count()
    DF_2=FilteredDF.ComplaintId.groupby([FilteredDF.DateSentCompany.dt.year,FilteredDF.DateSentCompany.dt.month]).count()
    DF1Date=[]
    DF2Date=[]

    for MonthYear in DF_1.index.values:
        Date1=datetime.datetime.strptime(str(list(MonthYear)[0])+'-'+str(list(MonthYear)[1]).rjust(2,'0')+'-01','%Y-%m-%d'
        DF1Date.append(Date1)
    for MonthYear in DF_2.index.values:
        Date2=datetime.datetime.strptime(str(list(MonthYear)[0])+'-'+str(list(MonthYear)[1]).rjust(2,'0')+'-01','%Y-%m-%d'
        DF2Date.append(Date2)

    DF1Date=pd.DataFrame(DF1Date,columns=['DateGroup'])
    DF2Date=pd.DataFrame(DF2Date,columns=['DateGroup'])

    PlotDate1=DF1Date
    PlotData1=DF_1
    PlotDate2=DF2Date
    PlotData2=DF_2
#code for year format
```

Data Analysis Stage:



Report Generation:



```
ax2.set_title('Timely Closure')
ax3=f.add_subplot(133)
ax3.clear()
ax3.pie(FilteredDF.Issue.groupby([FilteredDF.ConsumerDisputedSts]).count(),
        labels=pd.unique(FilteredDF.ConsumerDisputedSts),shadow=True,autopct='%1.1f%%',
                    startangle=90)
ax3.set_title('User Satisfaction')

ReportDF=FilteredDF
ReportDF['Issue Count']=ReportDF.ComplaintId

PivotDF=ReportDF.pivot_table(['Issue Count'],
                                index=['Company'],
                                columns=['Product'],
                                aggfunc='count',
                                fill_value =0
                             )
PivotColumnIndex=list(PivotDF.index.values)
DF=pd.DataFrame(PivotDF,index=PivotColumnIndex)

if ReportRefreshFlag is True:
    pt.updateModel(TableModel(dataframe=DF))
    pt.showIndex()
```

3.3 Visualize Data

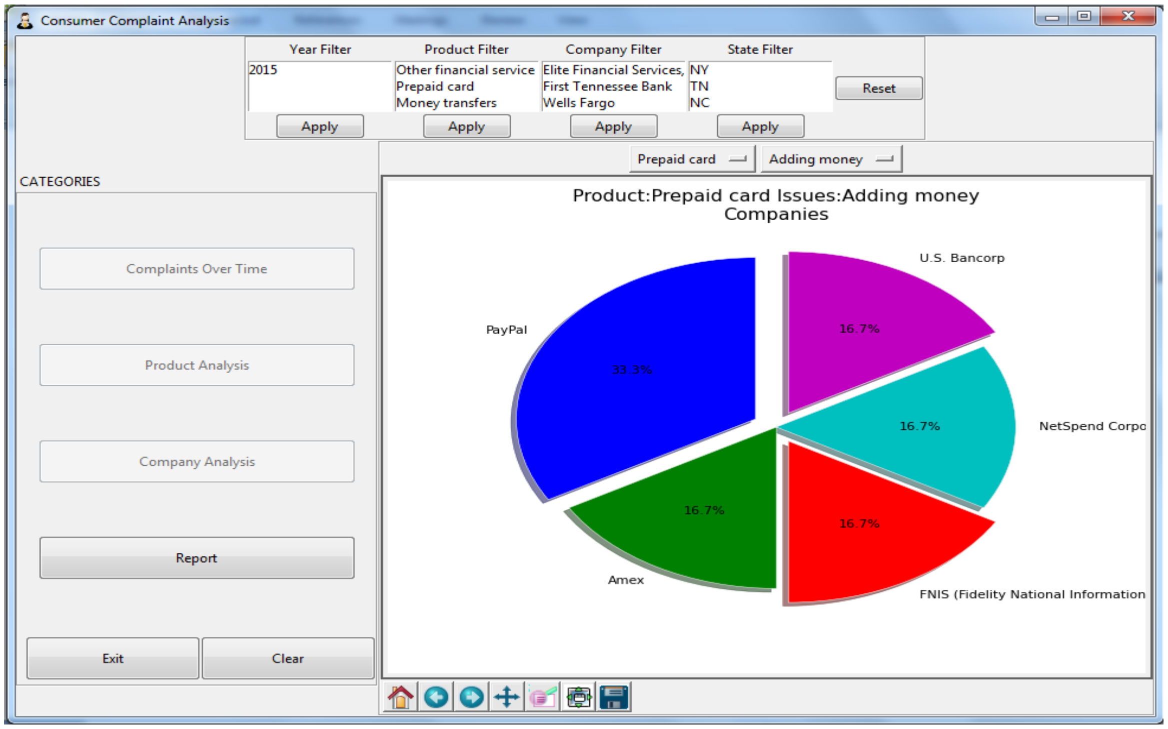
It is of at most importance to answer questions that provides insights to business problems .We have implemented visualizations for our three target audience as stated earlier. Let's visualize and analyse few sample findings for each category below.

3.3.1 Public:

This example graph gives a view about how public can use this utility to understand about products standing in market

.

Graph:

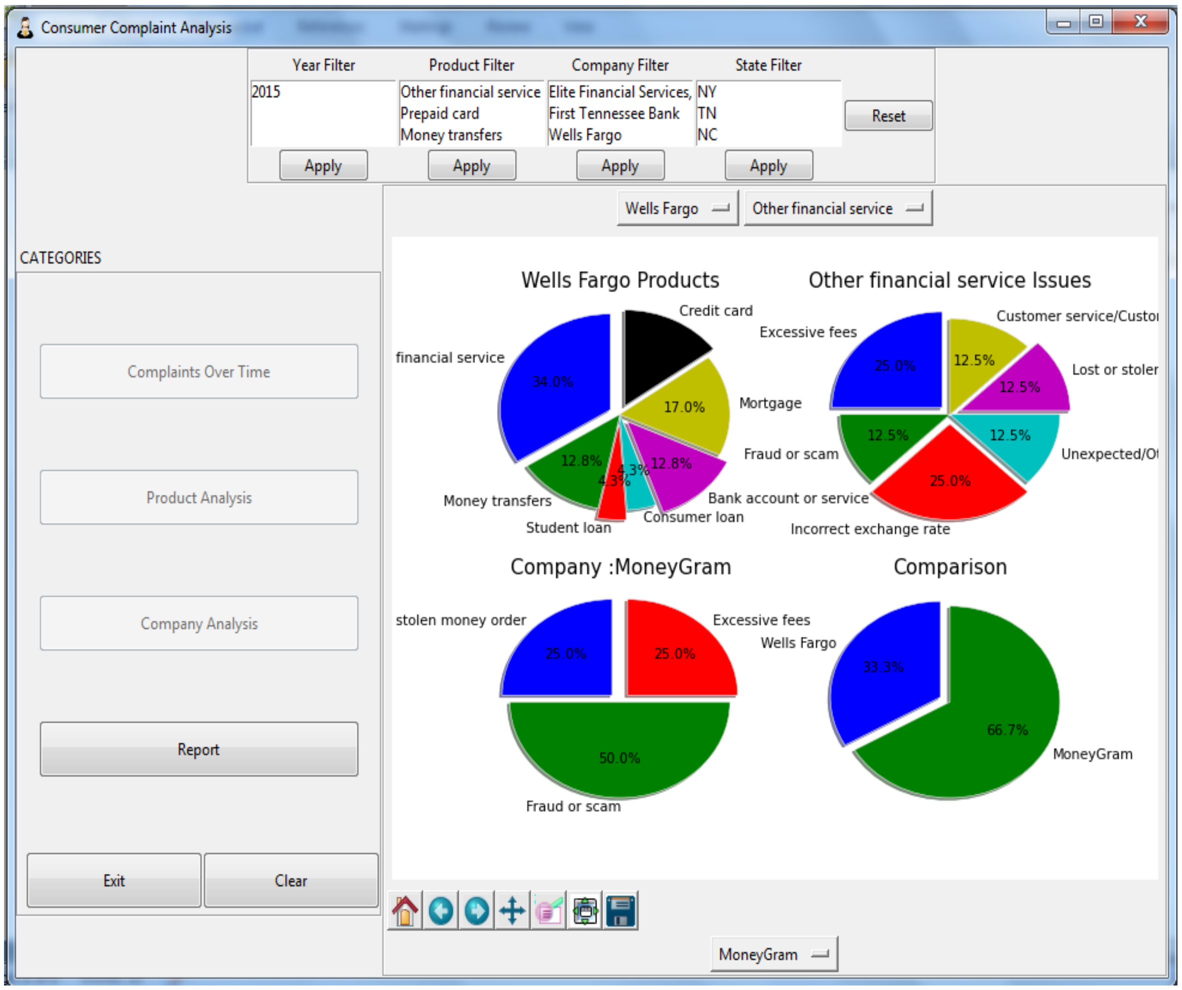


Findings: After applying appropriate filters above we can see the graph presented provides information about product category Prepaid Card and issue category for Adding money to companies. It shows details for all companies having complaints registered for these product and issue details. It gives a one stop dashboard for customers for analyse which company is receiving maximum number or minimum number of complaints. Then they can make their decision accordingly.

3.3.2 Company:

This example graph helps financial firms compare their market standing with their co-competitors.

Graph:



Consumer Complaint Analysis

Year Filter	Product Filter	Company Filter	State Filter
2015	Other financial service	Elite Financial Services,	NY
	Prepaid card	First Tennessee Bank	TN
	Money transfers	Wells Fargo	NC

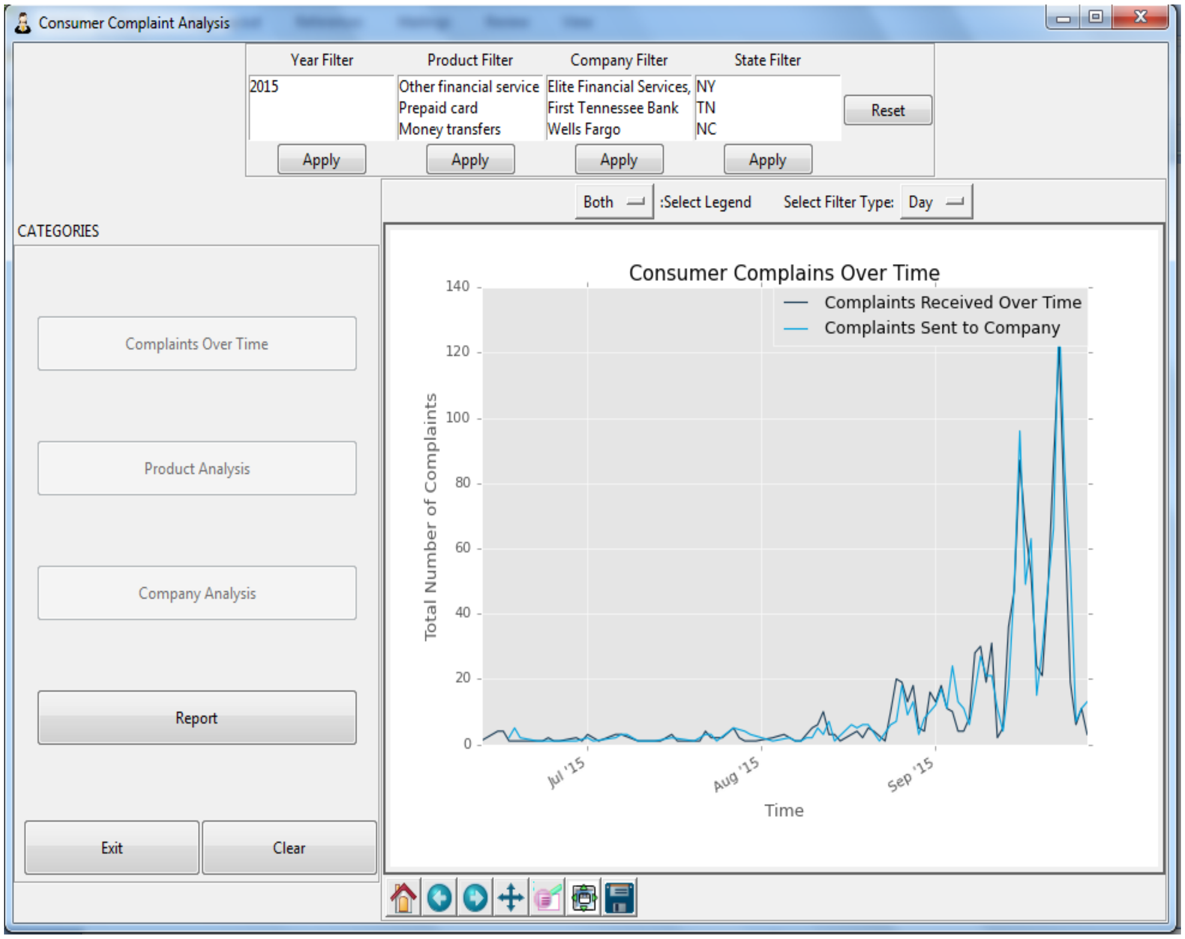
Wells Fargo — Other financial service — MoneyGram

Findings: After applying appropriate filters above we can see the graph presented provides information about similar companies having complaints registered for same set of products and issues. This gives companies a one stop dashboard to compare its performance with its competitors about how its product is being accepted in the market as compared to its peers.

3.3.3 Federal **Bureau:**

This example graph helps Federal Bureau have insight about how financial companies are performing in the market. These insights can help them have knowledge about satisfaction of public about the service being provided by these financial companies to them. Can help them have implement necessary regulatory measures in the market for the same.

Graph:



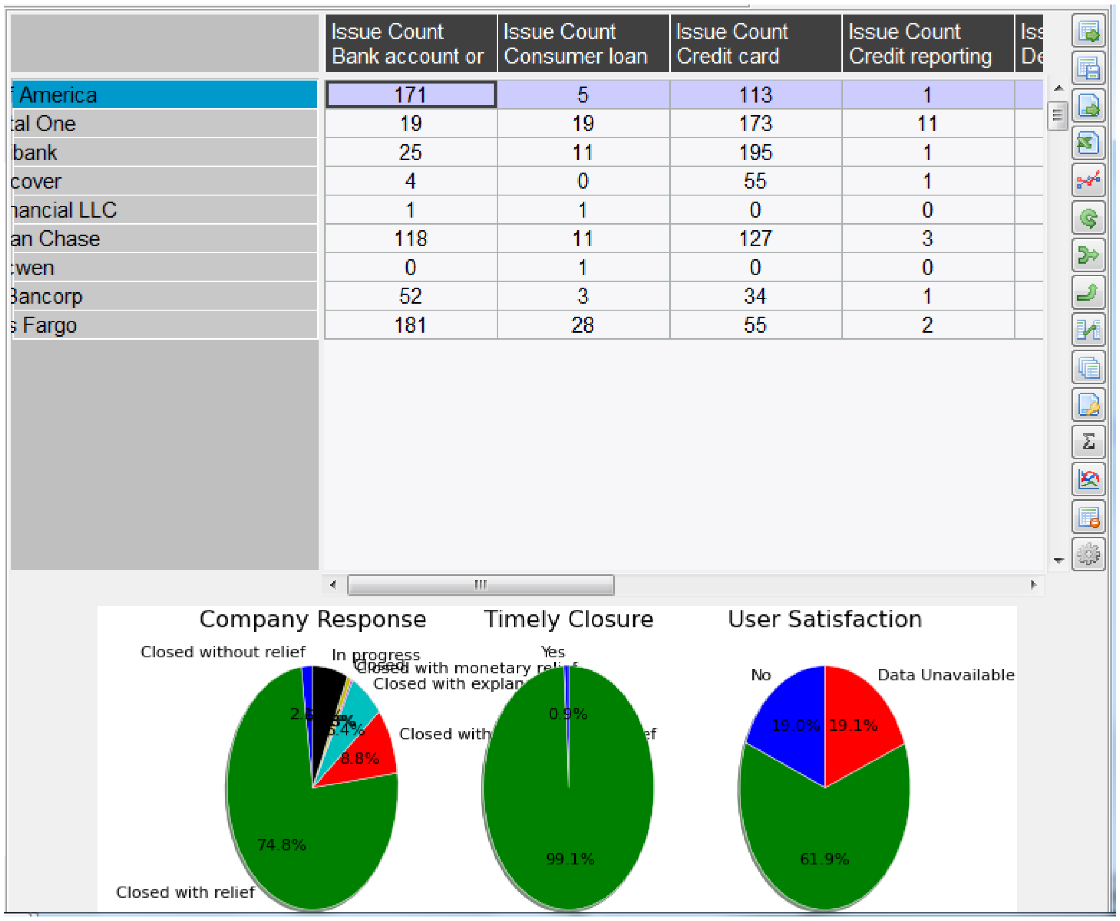
Year Filter	Product Filter	Company Filter	State Filter
2015	Other financial service	Elite Financial Services,	NY
	Prepaid card	First Tennessee Bank	TN
	Money transfers	Wells Fargo	NC

Findings: After applying appropriate filters above we can see the graph presented provides information about how is the trend between the time a complaint is received and the how long it takes for the complaint to be taken to company. This view helps financial bureau to keep check on its working by informing about the time it takes for complaints to be sent to companies for resolution.

3.3.4 Federal **Bureau Dashboard:**

This example visualizes one step dashboard to understand and study number of complaints registered for each company and how was the responses time from companies about the issue, followed by the customer satisfaction.

Graph:



	Issue Count Bank account or	Issue Count Consumer loan	Issue Count Credit card	Issue Count Credit reporting	Iss De
America	171	5	113	1	
tal One	19	19	173	11	
bank	25	11	195	1	
cover	4	0	55	1	
nancial LLC	1	1	0	0	
an Chase	118	11	127	3	
wen	0	1	0	0	
Bancorp	52	3	34	1	
s Fargo	181	28	55	2	

Findings: As we can infer from the dashboard above we have obtained list of companies with their respective counts of number of complaints received for each category of products they deal with. Pie charts depict three critical findings about Company response or the way the issue got resolved; timely closure status and the ultimate aim of User satisfaction index stating how user was satisfied for the issue raised. This serves as an indicator to understand entire process in one dashboard since it includes company names, products, company response, and time for issue resolution and user satisfaction together.

4. Guidelines to use Consumer Complaints Utility

Shared on Github repository online: https://github.com/PriyankDsilva/Data-Analysis---Consumer-Complaint /

Username and Password for the GUI is admin/admin

**Note-For testing purpose we have also created a proportional sample of source and saved it in ManualOverrideSource folder. If there exists a ConsumerComplaints.csv file in this folder the utility will override the data fetching process from database and load the source from the csv file*

4.1 Requirement:

- Python 3.4/Anaconda with python 3.4
- FireBird:

  Install Firebird Database and its python Library(fdb)

  Create an account to be used for database operations using below command:

  goto the installed dir using dos(cmd prmt):

  gsec -user sysdba -pass masterkey -add <User Name> -pw <Password>

- Additional Python Libraries used are as follows:

  fdb,pandas,os,configparser,urllib,datetime,shutil,time,tkinter,pandastable,

  matplotlib,numpy,threading

4.2 Issue encountered with python Libraries(Anaconda 3.4):

An Issue encountered while using tkinter, we were unable to plot graphs using GUI because the matplotlib lib was having an issue with tkinter while plotting graphs on canvas(TkAgg Error).

Solution - uninstalled matplotlib and pillow libraries from Anaconda and reinstall the core libraries from sourceforge

**Note-the .whl files for these libraries are uploaded in Github folder to install them just write pip install .whl file*

4.3 Modules:

CreateDBStructure.py - The python code will create initial DB Structure required for the Project (one time run)

(Make sure to create the username and password for Firebird DB and update the config.ini file)

ConsumerComplaintIcon.ico - Icon image for GUI

ConsumerComplaint.png - Image to display in GUI

config.ini - store the general variable values in it, so that the main code doesn't impact if anything needs to be changed.

Initialize.py - Loads the variables with initial value and stores it in a List to be accessed from anywhere.

FirebirdDB.py - This python file has the functions associated with the DB operation for Firebird.

FetchData.py - The Python code will fetch data from the online API and load it into Main tables while maintaining the log of the entire process.

CustomerComplaintGUI.py - python code to built GUI for Customer Complaint project. Starts with the Login page (admin/admin) then has Fetch/Update Data page which loads data from online API to main tables, View Logs to view logs maintained at each level, Data Analyst to perform analysis on the fetched data and Exit to quit.

DataAnalyst.py-python code to provide a GUI and Data Analysis for the Data Fetched from the Consumer Complaint site.

Database - This Folder have working Log ,Trig file(to indicate that update is in progress),Firebird Database and Staging folder for Staging Activities

Database\Staging -This folder will be used to store downloaded csv files and reject file.

Logs - This folder will be used to archive log and Reject Files if any

5. Conclusion

This report introduced about customer complaints database, subsequently we were introduced about three stages of implementation namely design database, analyze data and visualization of data. Efficient usage of visualization was made to notify real world problems in present process of analyzing customer data. Questions for three target audience namely public, companies and federal bureau were answered and effective trends and patterns were recognized, in the process we also learned how strong python and panda libraries are for implementing this utility.