Big Data Analysis Project Report

Submitted By
Rajashree Sanjay Kumkar
NetID: RSK18000

Table of Contents

Introduction and Problem Description

Looking at the whole dataset, we can say that customers are unhappy with product or service and complaints regarding the same are being raised. All these issues were reported but only some were timely responded to. Each of these complaints are assigned with unique id named as Complaint ID. The distribution of this data is spread over different states and zip codes.

There are various issues when it comes to analyzing customer complaints data. We can identify the patterns in which the issues were reported. There is some correlation between variables which we can identify and remove for better analysis. Also, non-linear effects should be accounted for before any sort of model building is used, if predictions are in the scope of this project for example, relation between zip code and state. Using various analytics, we can figure out the area where most of the complaints were raised from and provide solutions in such a way that these issues will not occur again and reduce the occurrence of at least the repetitive complaints.

The second dataset has the income information of consumers. The reason of taking this dataset is to find out how the income, age or gender affect the product complaints reporting.

The third dataset is about the demographic's information of each consumer. There can be some correlation between the demographic's information and the complaints reporting. The appropriate analysis will help find out the problem area.

The objective of this analysis is to help managers make better decisions. This analysis will find out meaningful information from the given database, deriving insights and finding what are the shortcomings. It will also direct to the path to minimize the number of issues being reported.

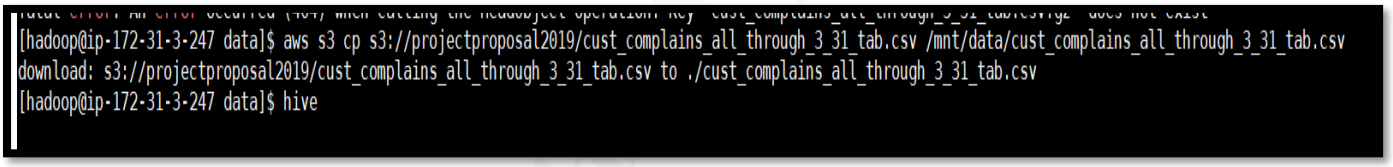
Related Work

To start with the analysis, I firstly created EMR, EC2 Cluster and S3 Bucket to use AWS services. I cleaned the data using python file before uploading it to S3 bucket. Then, I connected to S3 bucket to Jupyter notebook in order to analyze the data using PySpark, SparkAPI.

In PySpark, I imported numerous libraries for various purposes such as to perform operations on columns, to do visualization of my findings.

Additionally, In MobaXterm, I enabled hive and did analysis using SQL queries.

To enable hive support, below steps were performed.
Copy file back and unzip:



Then I created hive tables and performed operation on those tables.

Dataset Description

Below is the description of the three datasets used for the analyses of this project. Below is the screenshot showing the description of Customer Complaints dataset. It has total of 18 columns, the type of each column being string.

```
In [62]: spark.sql("desc customerdata").show()

+--------------------+---------+-------+
|            col_name|data_type|comment|
+--------------------+---------+-------+
|        DateReceived|   string|   null|
|             Product|   string|   null|
|          SubProduct|   string|   null|
|               Issue|   string|   null|
|            SubIssue|   string|   null|
|ConsumerComplaint...|   string|   null|
|CompanyPublicResp...|   string|   null|
|             Company|   string|   null|
|               State|   string|   null|
|             ZIPcode|   string|   null|
|                Tags|   string|   null|
|ConsumerConsentPr...|   string|   null|
|        Submittedvia|   string|   null|
|   DateSentToCompany|   string|   null|
|CompanyResponseto...|   string|   null|
|      TimelyResponse|   string|   null|
|    ConsumerDisputed|   string|   null|
|         ComplaintID|   string|   null|
+--------------------+---------+-------+
```

The second dataset contains the income information of each consumer. Firstly I read the dataset into a data frame. Later I removed the unwanted columns which were not adding much value towards my analysis.

drop unwanted columns from dataframe

```
In [29]: newincomedf= spark.sql("select customerID,age, workclass, \
education,marital_status,occupation,relationship, race, gender,income,ComplaintID from adultincomeData1")

In [30]: newincomedf.show(2)

+----------+---+---------+---------+------------------+-----------------+------------+-----+------+------+-----------+
|customerID|age|workclass|education|    marital_status|       occupation|relationship| race|gender|income|ComplaintID|
+----------+---+---------+---------+------------------+-----------------+------------+-----+------+------+-----------+
|      2200| 25|  Private|     11th|     Never-married|Machine-op-inspct|   Own-child|Black|  Male| <=50K|    3216175|
|      2201| 38|  Private|  HS-grad|Married-civ-spouse|  Farming-fishing|     Husband|White|  Male| <=50K|    3207829|
+----------+---+---------+---------+------------------+-----------------+------------+-----+------+------+-----------+
only showing top 2 rows
```

```
In [33]: spark.sql("desc income").show()
+--------------+---------+-------+
|      col_name|data_type|comment|
+--------------+---------+-------+
|    customerID|   string|   null|
|           age|   string|   null|
|     workclass|   string|   null|
|     education|   string|   null|
|marital_status|   string|   null|
|    occupation|   string|   null|
|  relationship|   string|   null|
|          race|   string|   null|
|        gender|   string|   null|
|        income|   string|   null|
|   ComplaintID|   string|   null|
+--------------+---------+-------+
```

The third dataset contains the demographic information of each consumer.

```
In [259]: demodf.columns
['PanelistID', 'preTaxIncome', 'familysize', 'TypeofResidentialPossession', 'COUNTY', 'MaleWorkingHour', 'AgeGroup', 'FemaleWor
kingHour', 'ChildrenGroup', 'MaritalStatus', 'NumberofTVs', 'ZIPCODE', 'FIPSCODE', 'IRIGeographyNumber', 'EXT_FACT', 'customerI
D']
```

Pre-processing techniques

I started data preprocessing with assigning meaningful column names for all three datasets. It will help to represent data in understandable way.

1. **Preprocessing of Consumer Complaints Data:**
   Below Screenshot shows how I have renamed the column names of each variable.

```
In [135]: cmplDF=cmplDF.withColumnRenamed("_c0","DateReceived")\
          .withColumnRenamed("_c1","Product")\
          .withColumnRenamed("_c2","Sub-product")\
          .withColumnRenamed("_c3",'Issue')\
          .withColumnRenamed("_c4","Sub-issue")\
          .withColumnRenamed("_c5","ConsumerComplaintNarrative")\
          .withColumnRenamed("_c6","CompanyPublicResponse")\
          .withColumnRenamed("_c7",'Company')\
          .withColumnRenamed("_c8",'State')\
          .withColumnRenamed("_c9",'ZIPcode')\
          .withColumnRenamed("_c10",'Tags')\
          .withColumnRenamed("_c11",'ConsumerConsentProvided')\
          .withColumnRenamed("_c12",'Submittedvia')\
          .withColumnRenamed("_c13",'DateSentToCompany')\
          .withColumnRenamed("_c14",'CompanyResponsetoConsumer')\
          .withColumnRenamed("_c15",'TimelyResponse')\
          .withColumnRenamed("_c16",'ConsumerDisputed')\
          .withColumnRenamed("_c17",'ComplaintID')

In [136]: cmplDF.columns

['DateReceived', 'Product', 'Sub-product', 'Issue', 'Sub-issue', 'ConsumerComplaintNarrative', 'CompanyPublicResponse', 'Com
pany', 'State', 'ZIPcode', 'Tags', 'ConsumerConsentProvided', 'Submittedvia', 'DateSentToCompany', 'CompanyResponsetoConsume
r', 'TimelyResponse', 'ConsumerDisputed', 'ComplaintID']
```

   Secondly, I replaced null values with meaningful data as shown below.

```
In [66]: fill_cols_vals = {"Company":"N/A","SubIssue": "N/A", "State" : "N/A", "SubProduct":"N/A", \
                          "ConsumerConsentProvided":"N/A", "Tags":"N/A", \
                         "ConsumerComplaintNarrative":"N/A" , "CompanyPublicResponse": "N/A" , \
                          "ComplaintID" : "0"}
         customerdf = customerdf.na.fill(fill_cols_vals)
```

2. **Preprocessing of Adult Income Dataset.**
   For dataset income dataset, I firstly assign appropriate column names as below.

```
In [119]:
newincome=newincome.withColumnRenamed("_c0","customerID")\
.withColumnRenamed("_c1","age")\
.withColumnRenamed("_c2","workclass")\
.withColumnRenamed("_c3","fnlwgt")\
.withColumnRenamed("_c4","education")\
.withColumnRenamed("_c5","educational_num")\
.withColumnRenamed("_c6","marital_status")\
.withColumnRenamed("_c7","occupation")\
.withColumnRenamed("_c8","relationship")\
.withColumnRenamed("_c9","race")\
.withColumnRenamed("_c10","gender")\
.withColumnRenamed("_c11","capital_gain")\
.withColumnRenamed("_c12","capital_loss")\
.withColumnRenamed("_c13","hours_per_week")\
.withColumnRenamed("_c14","native_country")\
.withColumnRenamed("_c15","income")\
.withColumnRenamed("_c16","ComplaintID")\
.withColumnRenamed("_c17","x")\
.withColumnRenamed("_c18","y")\
.withColumnRenamed("_c19","z")

In [120]:
newincome.columns

['customerID', 'age', 'workclass', 'fnlwgt', 'education', 'educational_num', 'marital_status', 'occupation', 'relationship',
'race', 'gender', 'capital_gain', 'capital_loss', 'hours_per_week', 'native_country', 'income', 'ComplaintID', 'x', 'y',
'z']
```

Later, I dropped unwanted columns which will not provide much meaningful information for my analysis.

```
In [171]:
fill_cols_vals = {"Company":"N/A","Sub-issue": "No Value", "State" : "No Value", "Sub-product":"Not available", \
                  "ConsumerConsentProvided?":"N/A", "Tags":"N/A", \
                 "ConsumerComplaintNarrative":"N/A" , "CompanyPublicResponse": "N/A" , \
                  "ComplaintID" : "0"}
customerdf = customerdf.na.fill(fill_cols_vals)
```

3. Preprocessing of demographic data

For third dataset I repeated the same process.

```
In [257]:
demodf=demodf1.withColumnRenamed("_c0","PanelistID") \
.withColumnRenamed("_c1","preTaxIncome") \
.withColumnRenamed("_c2","familysize") \
.withColumnRenamed("_c3","TypeofResidentialPossession") \
.withColumnRenamed("_c4","COUNTY") \
.withColumnRenamed("_c5","MaleWorkingHour") \
.withColumnRenamed("_c6","AgeGroup") \
.withColumnRenamed("_c7","FemaleWorkingHour") \
.withColumnRenamed("_c8","ChildrenGroup") \
.withColumnRenamed("_c9","MaritalStatus") \
.withColumnRenamed("_c10","NumberofTVs") \
.withColumnRenamed("_c11","ZIPCODE") \
.withColumnRenamed("_c12","FIPSCODE") \
.withColumnRenamed("_c13","IRIGeographyNumber") \
.withColumnRenamed("_c14","EXT_FACT") \
.withColumnRenamed("_c15","customerID")

In [259]:
demodf.columns

['PanelistID', 'preTaxIncome', 'familysize', 'TypeofResidentialPossession', 'COUNTY', 'MaleWorkingHour', 'AgeGroup', 'Female
WorkingHour', 'ChildrenGroup', 'MaritalStatus', 'NumberofTVs', 'ZIPCODE', 'FIPSCODE', 'IRIGeographyNumber', 'EXT_FACT', 'cus
tomerID']
```

Later on the basis of customerID and ComplaintID, I merged these three datasets into one.

Data Analysis and Proposed Solution

Analysis on Consumer Complaints Dataset:
The analysis revealed that the most issues were reported for EQUIFAX, INC. followed by Experian and then by TransUnion. This shows that people are worried about their credit scores. The results of the same are visualized using matplotlib as shown below.

```
In [199]:
spark.sql("select distinct Company, count(Product) as count from customerdata group by  \
Company order by count desc").show(10)
```

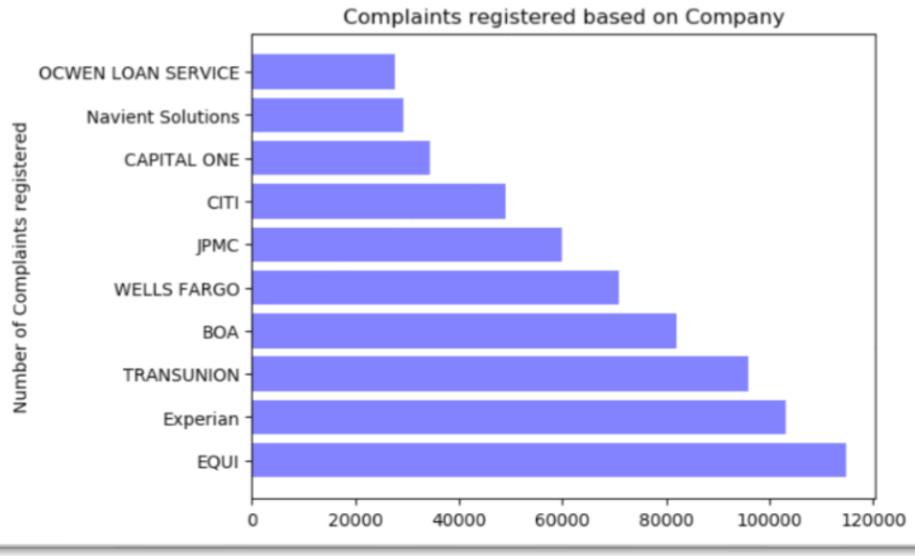
```
+--------------------+------+
|             Company| count|
+--------------------+------+
|       EQUIFAX, INC.|114721|
|Experian Informat...|103061|
|TRANSUNION INTERM...| 95807|
|BANK OF AMERICA, ...| 81910|
|WELLS FARGO & COM...| 70750|
|JPMORGAN CHASE & CO.| 60030|
|      CITIBANK, N.A.| 48925|
|CAPITAL ONE FINAN...| 34463|
|Navient Solutions...| 29227|
|OCWEN LOAN SERVIC...| 27731|
+--------------------+------+
only showing top 10 rows
```

```
In [33]:
import matplotlib.pyplot as plt; plt.rcdefaults()
import numpy as np
import matplotlib.pyplot as plt

objects = ('EQUI', 'Experian','TRANSUNION','BOA','WELLS FARGO','JPMC','CITI', \
           'CAPITAL ONE', 'Navient Solutions','OCWEN LOAN SERVICE')
y_pos = np.arange(len(objects))
count = [114721,103061,95807,81910,70750,60030,48925,34463,29227,27731]

plt.barh(y_pos, count, align='center', alpha=0.5, color="blue")
plt.yticks(y_pos, objects)
plt.ylabel('Number of Complaints registered')
plt.title('Complaints registered based on Company')

plt.show()
```



This analysis was done to find out what are products with maximum number of issues. This will help managers to pay more attention to these products and decide what are the issue areas, whether to release new version or remove this product from market.

This will help to analyze how competitors are performing.

```
In [43]: spark.sql("select distinct Company, product, count(Product) as count from customerdata group by Product, \
         Company order by count desc").show(20)
```

Company	product	count
EQUIFAX, INC.	Credit reporting,...	64629
Experian Informat...	Credit reporting,...	55741
TRANSUNION INTERM...	Credit reporting,...	54208
EQUIFAX, INC.	Credit reporting	48124
Experian Informat...	Credit reporting	45376
BANK OF AMERICA, ...	Mortgage	42905
TRANSUNION INTERM...	Credit reporting	39811
WELLS FARGO & COM...	Mortgage	36629
OCWEN LOAN SERVIC...	Mortgage	26503
Navient Solutions...	Student loan	25107
JPMORGAN CHASE & CO.	Mortgage	20985
NATIONSTAR MORTGAGE	Mortgage	19609
CITIBANK, N.A.	Credit card	16817
BANK OF AMERICA, ...	Bank account or s...	13916
WELLS FARGO & COM...	Bank account or s...	13333
CAPITAL ONE FINAN...	Credit card	12920
Ditech Financial LLC	Mortgage	12894
ENCORE CAPITAL GR...	Debt collection	10487
JPMORGAN CHASE & CO.	Credit card	10373
JPMORGAN CHASE & CO.	Bank account or s...	9816

only showing top 20 rows

Below results indicate that 97.48% times the reported issues were resolved within the time and 2.5% of times these issues were not resolved within expected time.

```
In [195]: spark.sql("select distinct round((count(case when TimelyResponse='Yes' then 1 end)/count(*))*100,3)as Percentageresolved, \
          round((count(case when TimelyResponse='No' then 1 end)/count(*))*100,3) as PercentageNotresolved from customerdata").show(3)
```

Percentageresolved	PercentageNotresolved
97.48	2.513

This is done to find out the quality of service being provided for zip code. This quality is determined based on number of issues resolved and not resolved.
It is found out that for Equifax, the maximum complaints have been registered for zip code 300XX, 770XX, 303XX. This indicates that the quality of service being provided in these areas is not good. Hence the company should focus more on improving the service in these areas.
Similar solution must be implemented for the other companies listed in the table below.

```
In [60]: spark.sql("select distinct Company, ZIPcode, count(case when TimelyResponse='Yes' then 1 end) as resolved, \
count(case when TimelyResponse='No' then 1 end) as Notresolved \
from customerdata where ZIPcode!='' group by Company, ZIPcode order by resolved desc").show()
```

Company	ZIPcode	resolved	Notresolved
EQUIFAX, INC.	300XX	947	24
EQUIFAX, INC.	330XX	859	9
Experian Informat...	330XX	805	0
EQUIFAX, INC.	770XX	779	15
TRANSUNION INTERM...	330XX	759	0
TRANSUNION INTERM...	300XX	703	1
Experian Informat...	300XX	703	0
EQUIFAX, INC.	331XX	689	8
Experian Informat...	770XX	669	0
EQUIFAX, INC.	303XX	662	10
TRANSUNION INTERM...	770XX	648	2
Experian Informat...	331XX	608	0
TRANSUNION INTERM...	331XX	602	3
EQUIFAX, INC.	606XX	595	8
Experian Informat...	334XX	542	0
EQUIFAX, INC.	334XX	531	11
EQUIFAX, INC.	302XX	523	7
Experian Informat...	303XX	510	0
TRANSUNION INTERM...	303XX	497	0
TRANSUNION INTERM...	606XX	493	0

only showing top 20 rows

Below screenshot indicates the company, its product and number of issues reported under that product.

```
In [43]: spark.sql("select distinct Company, product, count(Product) as count from customerdata group by Product, \
Company order by count desc").show(20)
```

Company	product	count
EQUIFAX, INC.	Credit reporting,...	64629
Experian Informat...	Credit reporting,...	55741
TRANSUNION INTERM...	Credit reporting,...	54208
EQUIFAX, INC.	Credit reporting	48124
Experian Informat...	Credit reporting	45376
BANK OF AMERICA, ...	Mortgage	42905
TRANSUNION INTERM...	Credit reporting	39811
WELLS FARGO & COM...	Mortgage	36629
OCWEN LOAN SERVIC...	Mortgage	26503
Navient Solutions...	Student loan	25107
JPMORGAN CHASE & CO.	Mortgage	20985
NATIONSTAR MORTGAGE	Mortgage	19609
CITIBANK, N.A.	Credit card	16817
BANK OF AMERICA, ...	Bank account or s...	13916
WELLS FARGO & COM...	Bank account or s...	13333
CAPITAL ONE FINAN...	Credit card	12920
Ditech Financial LLC	Mortgage	12894
ENCORE CAPITAL GR...	Debt collection	10487
JPMORGAN CHASE & CO.	Credit card	10373
JPMORGAN CHASE & CO.	Bank account or s...	9816

only showing top 20 rows

The above analysis illustrates that the maximum complaints have been reported for the credit reporting, hence the company providing the credit reporting should make measures to further analyze the cause of these complaints and try not to repeat the mistakes done in the past. Equifax, Experian and TransUnion have the most complaints in the credit reporting domain.

Analysis on Income Data:

Upon analysis of income data, it shows us that income generally increases with age.

```
In [271]: from pyspark.sql.functions import countDistinct, approx_count_distinct, count, sum, mean, round
incomedf.groupBy(col("occupation"), col("income")).agg(round(mean("age"),3).alias("Average-age"))\
.sort("Average-age", ascending=False).show(10)
```

```
+-----------------+------+-----------+
|       occupation|income|Average-age|
+-----------------+------+-----------+
|                ?|  >50K|       56.0|
|  Farming-fishing|  >50K|     46.471|
|  Priv-house-serv| <=50K|     45.107|
| Transport-moving|  >50K|     45.097|
|  Exec-managerial|  >50K|     44.713|
|     Adm-clerical|  >50K|     44.224|
|            Sales|  >50K|     44.134|
|   Prof-specialty|  >50K|     43.761|
|Handlers-cleaners|  >50K|     43.528|
|Machine-op-inspct|  >50K|     43.414|
+-----------------+------+-----------+
only showing top 10 rows
```

```
In [272]: incomedf.groupBy(col("occupation"), col("income")) \
.agg(round(mean("age"),3).alias("Average-age"))\
.sort("Average-age", ascending=True).show(10)
```

```
+-----------------+------+-----------+
|       occupation|income|Average-age|
+-----------------+------+-----------+
|     Armed-Forces| <=50K|       25.0|
|Handlers-cleaners| <=50K|     32.054|
|     Tech-support| <=50K|     34.832|
|    Other-service| <=50K|     34.839|
|            Sales| <=50K|     35.002|
|     Adm-clerical| <=50K|     36.442|
|Machine-op-inspct| <=50K|     37.054|
|   Prof-specialty| <=50K|     37.706|
|                ?| <=50K|     37.713|
|     Craft-repair| <=50K|     37.923|
+-----------------+------+-----------+
only showing top 10 rows
```

Analysis on Demographics Dataset:

Analyzing the data shows that Male total working hours are more than female total working hours. And as the family size increases the total working hours for both males and females decrease.

```
In [288]: demodf2.agg(sum("MaleWorkingHour").alias("total-MaleWorkingHour") ,sum("FemaleWorkingHour").alias("total-FemaleWorkingHour"))\
          .sort("total-MaleWorkingHour", ascending=False).show()

+---------------------+-----------------------+
|total-MaleWorkingHour|total-FemaleWorkingHour|
+---------------------+-----------------------+
|              28374.0|                22289.0|
+---------------------+-----------------------+

In [289]: demodf2.groupBy("familysize").agg(sum("MaleWorkingHour").alias("total-MaleWorkingHour") \
                                ,sum("FemaleWorkingHour").alias("total-FemaleWorkingHour"))\
          .sort("total-MaleWorkingHour", ascending=False).show()

+----------+---------------------+-----------------------+
|familysize|total-MaleWorkingHour|total-FemaleWorkingHour|
+----------+---------------------+-----------------------+
|         1|               9883.0|                 6236.0|
|         2|               9793.0|                 8534.0|
|         3|               3851.0|                 3009.0|
|         4|               3043.0|                 2837.0|
|         5|               1288.0|                 1170.0|
|         6|                516.0|                  503.0|
+----------+---------------------+-----------------------+
```

Analysis on Merged Dataset:

Interestingly, the below analysis shows that most complaints have been submitted via web and it can also be seen that people with income less than 50K file more complaints.

```
In [145]: df= mrgeddf.groupBy(col("Submittedvia"), col("income")).agg(count("Submittedvia").alias("total-Submittedvia"))\
          .sort("total-Submittedvia", ascending=False).show()

+------------+------+-------------------+
|Submittedvia|income|total-Submittedvia|
+------------+------+-------------------+
|         Web| <=50K|              11171|
|         Web|  >50K|               3497|
|    Referral| <=50K|               2132|
|       Phone| <=50K|                879|
| Postal mail| <=50K|                810|
|    Referral|  >50K|                624|
|       Phone|  >50K|                311|
| Postal mail|  >50K|                248|
|         Fax| <=50K|                240|
|         Fax|  >50K|                 80|
|       Email|  >50K|                  5|
|       Email| <=50K|                  3|
|        null| <=50K|                  0|
|        null|  >50K|                  0|
+------------+------+-------------------+
```

The same code is written using SPARK API.

```
In [144]: spark.sql("select Submittedvia,count(Submittedvia) as countSubmittedVia, \
income from mergedDF group by income, Submittedvia  order by countSubmittedVia desc").show()
```

```
+------------+-----------------+------+
|Submittedvia|countSubmittedVia|income|
+------------+-----------------+------+
|         Web|            11171| <=50K|
|         Web|             3497|  >50K|
|    Referral|             2132| <=50K|
|       Phone|              879| <=50K|
| Postal mail|              810| <=50K|
|    Referral|              624|  >50K|
|       Phone|              311|  >50K|
| Postal mail|              248|  >50K|
|         Fax|              240| <=50K|
|         Fax|               80|  >50K|
|       Email|                5|  >50K|
|       Email|                3| <=50K|
|        null|                0| <=50K|
|        null|                0|  >50K|
+------------+-----------------+------+
```

Further analysis on the age and complaints reveal that the complaints based on product does not depend upon age. People from all age groups file complaints regarding all product category though younger people seem to file complaints marginally less than older people.

```
In [351]: allmergeddf.groupBy(col("Product"), col("AgeGroup")) \
.agg(count(col("Product")).alias("total-Product"))  \
.where(col("total-Product")>250).where(col("AgeGroup")>=5)  \
.orderBy(col("AgeGroup"), ascending=False) \
.show()
```

```
+--------------------+--------+-------------+
|             Product|AgeGroup|total-Product|
+--------------------+--------+-------------+
|            Mortgage|       6|          425|
|Credit reporting,...|       6|          337|
|     Debt collection|       6|          346|
|            Mortgage|       5|          257|
+--------------------+--------+-------------+
```

```
In [352]: allmergeddf.groupBy(col("Product"), col("AgeGroup")) \
.agg(count(col("Product")).alias("total-Product"))  \
.where(col("total-Product")>200) .where(col("AgeGroup")<5)\
.orderBy(col("AgeGroup"), ascending=True) \
.show()
```

```
+--------------------+--------+-------------+
|             Product|AgeGroup|total-Product|
+--------------------+--------+-------------+
|            Mortgage|       3|          249|
|            Mortgage|       4|          296|
|     Debt collection|       4|          277|
|Credit reporting,...|       4|          278|
+--------------------+--------+-------------+
```

Further analysis reveals Lesser the income more the working hours for male and female and larger the family size, lesser the working hours and greater the income.

```
In [369]: allmergeddf.groupBy(col("income"), col("familysize")) \
          .agg(sum(col("MaleWorkingHour")).alias("HoursMalework"), \
              sum(col("FemaleWorkingHour")).alias("HoursFemalework"), \
              count("income").alias("total"))\
          .sort(col("HoursFemalework"), col("HoursMalework"), ascending= False).show()
```

```
+------+----------+-------------+---------------+-----+
|income|familysize|HoursMalework|HoursFemalework|total|
+------+----------+-------------+---------------+-----+
| <=50K|         2|       7581.0|         6597.0| 1990|
| <=50K|         1|       7410.0|         4674.0| 1126|
| <=50K|         3|       2995.0|         2285.0|  745|
| <=50K|         4|       2335.0|         2130.0|  737|
|  >50K|         2|       2212.0|         1937.0|  577|
|  >50K|         1|       2473.0|         1562.0|  377|
| <=50K|         5|        953.0|          858.0|  294|
|  >50K|         3|        856.0|          724.0|  234|
|  >50K|         4|        708.0|          707.0|  223|
| <=50K|         6|        386.0|          379.0|  122|
|  >50K|         5|        335.0|          312.0|  101|
|  >50K|         6|        130.0|          124.0|   39|
+------+----------+-------------+---------------+-----+
```

Visualizations

- Visualization for the number of customers having income either >50k or <=50k.

```
In [185]: df=spark.sql("select distinct income, count(*) from mergedDF group by income")
          df.show()
```

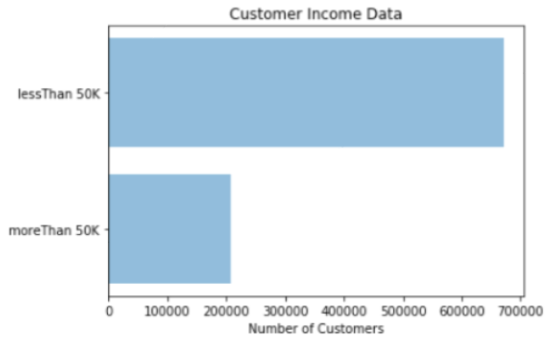
```
+------+--------+
|income|count(1)|
+------+--------+
| <=50K|  671759|
|  >50K|  208155|
+------+--------+
```

```
In [8]: import matplotlib.pyplot as plt;
        import numpy as np
        import matplotlib.pyplot as plt

        objects = ('moreThan 50K', 'lessThan 50K')
        y_pos = np.arange(len(objects))
        numberOfCustomer = [208155,671759]

        plt.barh(y_pos, numberOfCustomer, align='center', alpha=0.5)
        plt.yticks(y_pos, objects)
        plt.xlabel('Number of Customers')
        plt.title('Customer Income Data')

        plt.show()
```



- Visualization for the number of complaints registered based on gender of a consumer.

```
In [56]: mrgeddf.groupBy("gender").count().show()
```

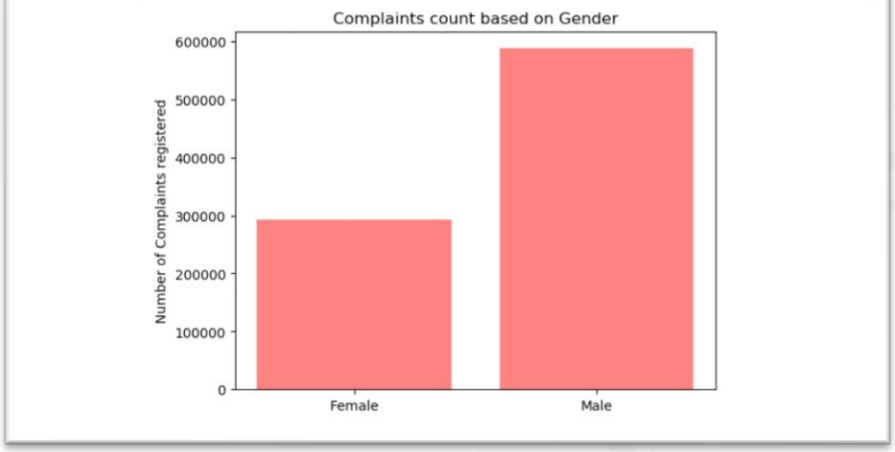
```
+------+------+
|gender| count|
+------+------+
|Female|292229|
|  Male|587685|
+------+------+
```

```
In [15]: import matplotlib.pyplot as plt; plt.rcdefaults()
         import numpy as np
         import matplotlib.pyplot as plt

         gender = ('Female', 'Male')
         y_pos = np.arange(len(objects))
         count = [292229,587685]

         plt.bar(y_pos, count, align='center', alpha=0.5, color='red')
         plt.xticks(y_pos, objects)
         plt.ylabel('Number of Complaints registered')
         plt.title('Complaints count based on Gender')

         plt.show()
```



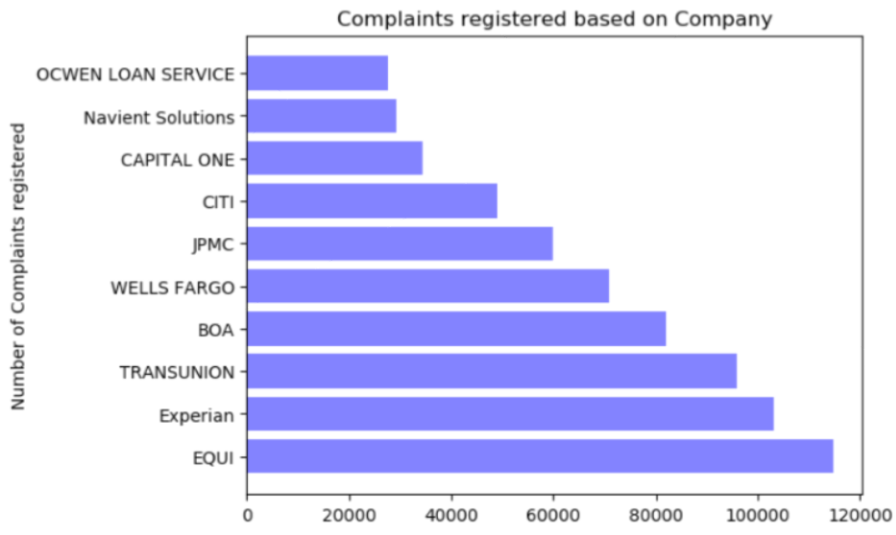
- Visualization of company and number of complaints registered under it.

```
In [33]: import matplotlib.pyplot as plt; plt.rcdefaults()
         import numpy as np
         import matplotlib.pyplot as plt

         objects = ('EQUI', 'Experian','TRANSUNION','BOA','WELLS FARGO','JPMC','CITI', \
                    'CAPITAL ONE', 'Navient Solutions','OCWEN LOAN SERVICE')
         y_pos = np.arange(len(objects))
         count = [114721,103061,95807,81910,70750,60030,48925,34463,29227,27731]

         plt.barh(y_pos, count, align='center', alpha=0.5, color="blue")
         plt.yticks(y_pos, objects)
         plt.ylabel('Number of Complaints registered')
         plt.title('Complaints registered based on Company')

         plt.show()
```



Conclusion

The consumer data set gives great insights about the companies and products getting maximum number of complaints. After completion of the analysis, it is seen that Equifax and the Experian are the two companies which are facing major issues and complaints. They should improve on their products and services to reduce the number of complaints in the future as it may diminish sales of the products in the future.
It is also observed that people with lower income file lot of complaints as compared to people with high income and also male file more complaints than female. Based on this we can put up our resources towards the areas with high complaint rate in order to resolve issues more efficiently.

References

- https://www.kaggle.com/muonneutrino/us-census-demographic-data/downloads/us-census-demographic-data.zip/3#acs2015_census_tract_data.csv
- https://www.kaggle.com/wenruliu/adult-income-dataset
- https://spark.apache.org/docs/latest/api/python/index.html