BIG DATA PROJECT

A Detailed Analysis of Consumer Complaints

MAY 7, 2019

GOKUL KARTHIKEYAN (GXK171030)

TABLE OF CONTENTS

1. PROBLEM DESCRIPTION

A cross sectional dataset comprising of consumer complaints from a wide span of companies are considered for the analysis of this project. It is a challenging task to assess the nature of these complaints and arrive at a suitable solution to achieve utmost customer satisfaction.

To add more value to this dataset, adding economic factors like Income and Education level data will be of great help in uncovering certain insights. The dataset spans across 1million records with 20+ columns. Shown below is the investigation of couple of columns based on a hypothesis and the achieved result.

---

2. RELATED WORK

1. An AWS-EMR Cluster with supporting configuration is created.

2. The customer complaint data is loaded into the s3 bucket using the following commands:

```
- on EMR) cd /mnt/data
- wget --no-check-certificate https://data.consumerfinance.gov/api/views/s6ew-
  h6mp/rows.csv?accessType=DOWNLOAD
- mv 'rows.csv?accessType=DOWNLOAD' cust_complains_all_through_3_31.csv
- aws s3 cp cust_complains_all_through_3_31_tab.csv
  s3://gokulproject/data/cust_complains_all_through_3_31_tab.csv
```

3. Now we create external table in HIVE using the following commands:

```
- CREATE DATABASE cust_complains;
-
- CREATE  EXTERNAL TABLE cust_complains.cust_complaints_all_csv_raw_tst
- (date_received      string,
- product             string,
- sub_product         string,
- issue               string,
- sub_issue           string,
- consumer_complaint_narrative string,
- company_public_response string,
- company             string,
- state               string,
- zip_code            string,
- tags                string,
- consumer_consent_provided string,
- submitted_via       string,
- date_sent_to_company string,
```

```
-     company_response    string,
-     timely_response     string,
-     consumer_disputed   string,
-     complaint_id        string)
-     ROW FORMAT DELIMITED
-     FIELDS TERMINATED BY '\t'
-     STORED AS
-     INPUTFORMAT
-       'com.amazonaws.emr.s3select.hive.S3SelectableTextInputFormat'
-     OUTPUTFORMAT
-       'org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat'
-     LOCATION 's3://big-data-spring-2019/data'
-     TBLPROPERTIES (
-       "s3select.format" = "csv",
-       "s3select.headerInfo" = "ignore"
-     );
-
-
-     CREATE TABLE cust_complains.cust_complaints_all
-     (date_received      string,
-     product             string,
-     sub_product         string,
-     issue               string,
-     sub_issue           string,
-     consumer_complaint_narrative string,
-     company_public_response string,
-     company             string,
-     state               string,
-     zip_code            string,
-     tags                string,
-     consumer_consent_provided string,
-     submitted_via       string,
-     date_sent_to_company string,
-     company_response    string,
-     timely_response     string,
-     consumer_disputed   string,
-     complaint_id        string)
-     ROW FORMAT DELIMITED
-     FIELDS TERMINATED BY ','
-     STORED AS PARQUET
-     TBLPROPERTIES ("parquet.compression"="SNAPPY");
-
-     INSERT INTO cust_complains.cust_complaints_all
-     SELECT * FROM cust_complains.cust_complaints_all_csv_raw_tst;
```

4. Once this data is loaded into the table, we can read this into a dataframe using Spark API.

5. Two additional datasets- IncomeData and EducationData are loaded into the s3 bucket in the location s3://gokulproject/Other_data/

3. DATASET DESCRIPTION

3.1. Income Dataset

- Income Data is collected for numerous states and zipcodes across the united states. This data consists of number of households, Mean Income of household and Median Income.

- The data is read and only certain columns necessary for analysis is selected from the data using the following commands:

In [2]:

```
full_income = spark.read.format("csv")\
.option("header", "true")\
.option("inferSchema", "true")\
.load("s3://gokulproject/Other_data/IncomeDataset.csv")
full_income.printSchema()
root
 |-- id: integer (nullable = true)
 |-- State_Code: integer (nullable = true)
 |-- State_Name: string (nullable = true)
 |-- State_ab: string (nullable = true)
 |-- County: string (nullable = true)
 |-- City: string (nullable = true)
 |-- Place: string (nullable = true)
 |-- Type: string (nullable = true)
 |-- Primary: string (nullable = true)
 |-- Zip_Code: integer (nullable = true)
 |-- Area_Code: string (nullable = true)
 |-- ALand: long (nullable = true)
 |-- AWater: long (nullable = true)
 |-- Lat: double (nullable = true)
 |-- Lon: double (nullable = true)
 |-- Mean_Household_Income: integer (nullable = true)
 |-- Median_HouseHold_Income: integer (nullable = true)
 |-- Stdev: integer (nullable = true)
 |-- NumHouseholds: double (nullable = true)
```

In [3]:

```
income = full_income.select(col("State_ab").alias('Inc_State'), col("Zip_Code
"), col("Mean_Household_Income"), col("Median_Household_Income"), col("NumHou
seholds"))
income = income.withColumn("Inc_Zip", income.Zip_Code.substr(1,3))
income.printSchema()
root
 |-- Inc_State: string (nullable = true)
 |-- Zip_Code: integer (nullable = true)
 |-- Mean_Household_Income: integer (nullable = true)
 |-- Median_Household_Income: integer (nullable = true)
 |-- NumHouseholds: double (nullable = true)
 |-- Inc_Zip: string (nullable = true)
```

In [5]:

```
income.createOrReplaceTempView("IncomeData")
```

In [6]:

```
income_data = spark.sql("select sum(Mean_Household_Income) as Tot_Mean_Income
, \
sum(Median_Household_Income) as Tot_Med_Income, \
SUM(NumHouseholds) as NumHouseholds, \
Inc_State, Inc_Zip \
from IncomeData \
group by Inc_State, Inc_Zip")
income_data.printSchema()
root
 |-- Tot_Mean_Income: long (nullable = true)
 |-- Tot_Med_Income: long (nullable = true)
 |-- NumHouseholds: double (nullable = true)
 |-- Inc_State: string (nullable = true)
 |-- Inc_Zip: string (nullable = true)
```

3.2. Education Dataset

- Education Data is collected for all states in the US and NumStudents across all grades and revenue information across states and total is provided.

- Since most of the complaints is based on credit cards and finance, I consider only relevant columns such as NumStudents above high school, and local state revenue from education.

```
full_education = spark.read.format("csv")\
.option("header", "true")\
```

```
.option("inferSchema", "true")\
.load("s3://gokulproject/Other_data/EducationDataset.csv")
full_education.printSchema()
root
 |-- PRIMARY_KEY: string (nullable = true)
 |-- STATE: string (nullable = true)
 |-- YEAR: integer (nullable = true)
 |-- ENROLL: double (nullable = true)
 |-- TOTAL_REVENUE: double (nullable = true)
 |-- FEDERAL_REVENUE: double (nullable = true)
 |-- STATE_REVENUE: double (nullable = true)
 |-- LOCAL_REVENUE: double (nullable = true)
 |-- TOTAL_EXPENDITURE: double (nullable = true)
 |-- INSTRUCTION_EXPENDITURE: double (nullable = true)
 |-- SUPPORT_SERVICES_EXPENDITURE: double (nullable = true)
 |-- OTHER_EXPENDITURE: double (nullable = true)
 |-- CAPITAL_OUTLAY_EXPENDITURE: double (nullable = true)
 |-- GRADES_PK_G: double (nullable = true)
 |-- GRADES_KG_G: double (nullable = true)
 |-- GRADES_4_G: double (nullable = true)
 |-- GRADES_8_G: double (nullable = true)
 |-- GRADES_12_G: double (nullable = true)
 |-- GRADES_1_8_G: double (nullable = true)
 |-- GRADES_9_12_G: double (nullable = true)
 |-- GRADES_ALL_G: double (nullable = true)
 |-- AVG_MATH_4_SCORE: double (nullable = true)
 |-- AVG_MATH_8_SCORE: double (nullable = true)
 |-- AVG_READING_4_SCORE: double (nullable = true)
 |-- AVG_READING_8_SCORE: double (nullable = true)
```

In [8]:

```
education = full_education.select(col('STATE').alias('State'), col('STATE_REV
ENUE').alias('State_Educ_Revenue'), col('GRADES_ALL_G').alias('NumStudents'))
education.printSchema()
root
 |-- State: string (nullable = true)
 |-- State_Educ_Revenue: double (nullable = true)
 |-- NumStudents: double (nullable = true)
```

In [9]:

```
education.createOrReplaceTempView("EducationData")
```

In [10]:

```
educ_data = spark.sql("select State, sum(NumStudents) as NumStudents, \
sum(State_Educ_Revenue) as Tot_State_Educ_Revenue \
from EducationData group by State")
```

```
educ_data.printSchema()
root
 |-- State: string (nullable = true)
 |-- NumStudents: double (nullable = true)
 |-- Tot_State_Educ_Revenue: double (nullable = true)
```

In [11]:

```
educ_data.show()
```

Coverting education – State into abbreviation. [Texas -> TX]

```
states = spark.read.format("csv")\
.option("header", "true")\
.option("inferSchema", "true")\
.load("s3://gokulproject/Other_data/states.csv")
states.printSchema()
root
 |-- State: string (nullable = true)
 |-- Abbreviation: string (nullable = true)
```

In [13]:

```
from pyspark.sql.functions import lower, upper
states = states.withColumn('State', upper(col('State')))
```

In [14]:

```
states.show()
+--------------------+------------+
|               State|Abbreviation|
+--------------------+------------+
|             ALABAMA|          AL|
|              ALASKA|          AK|
|             ARIZONA|          AZ|
|            ARKANSAS|          AR|
|          CALIFORNIA|          CA|
|            COLORADO|          CO|
|         CONNECTICUT|          CT|
|            DELAWARE|          DE|
|DISTRICT_OF_COLUMBIA|          DC|
|             FLORIDA|          FL|
|             GEORGIA|          GA|
|              HAWAII|          HI|
|               IDAHO|          ID|
|            ILLINOIS|          IL|
|             INDIANA|          IN|
```

```
|                IOWA|          IA|
|              KANSAS|          KS|
|            KENTUCKY|          KY|
|           LOUISIANA|          LA|
|               MAINE|          ME|
+--------------------+------------+
only showing top 20 rows
```

In [15]:

```
EducData_final = educ_data.join(states,"State", how='inner')
EducData_final = EducData_final.drop('State')
EducData_final = EducData_final.withColumnRenamed("Abbreviation", "state")
EducData_final.show()
+-----------+----------------------+-----+
|NumStudents|Tot_State_Educ_Revenue|state|
+-----------+----------------------+-----+
| 1.547483E7|            7.0745597E7|   SC|
|1.9460842E7|           1.10117605E8|   WI|
|  1673688.0|                    0.0|   DC|
|3.9643588E7|           1.88960285E8|   PA|
|4.5100706E7|           1.79809932E8|   IL|
| 1.864524E7|            9.8826436E7|   MD|
|  5784834.0|            2.5820605E7|   ID|
|2.0061151E7|            8.3661928E7|   MO|
|5.9646734E7|           4.46777506E8|   NY|
|  3354134.0|            1.5196139E7|   MT|
|  3.58826E7|           2.32956123E8|   MI|
|5.5298947E7|           2.18933074E8|   FL|
|1.1815272E7|            6.2709544E7|   OR|
|3.0015324E7|           1.54477379E8|   NC|
|  2910185.0|            2.6742962E7|   AK|
|1.6038502E7|            6.8490629E7|   LA|
|  4428573.0|            2.1842658E7|   ME|
| 2.787006E7|           1.96224231E8|   NJ|
|  1.40915E7|            5.8306836E7|   OK|
|2.6046432E7|           1.22374713E8|   VA|
+-----------+----------------------+-----+
only showing top 20 rows
```

3.3. Consumer Dataset

- The consumer complaint dataset is already processed and stored as a Parquet file in the S3 bucket.

```
spark.sql("use cust_complains")
spark.sql("show tables").show()
+--------------+--------------------+-----------+
|      database|           tableName|isTemporary|
+--------------+--------------------+-----------+
|cust_complains| cust_complaints_all|      false|
|cust_complains|cust_complaints_a...|      false|
|              |       educationdata|       true|
|              |          incomedata|       true|
+--------------+--------------------+-----------+
```

In [18]:

```
customer = spark.sql("select * from cust_complaints_all")
customer.printSchema()
root
 |-- date_received: string (nullable = true)
 |-- product: string (nullable = true)
 |-- sub_product: string (nullable = true)
 |-- issue: string (nullable = true)
 |-- sub_issue: string (nullable = true)
 |-- consumer_complaint_narrative: string (nullable = true)
 |-- company_public_response: string (nullable = true)
 |-- company: string (nullable = true)
 |-- state: string (nullable = true)
 |-- zip_code: string (nullable = true)
 |-- tags: string (nullable = true)
 |-- consumer_consent_provided: string (nullable = true)
 |-- submitted_via: string (nullable = true)
 |-- date_sent_to_company: string (nullable = true)
 |-- company_response: string (nullable = true)
 |-- timely_response: string (nullable = true)
 |-- consumer_disputed: string (nullable = true)
 |-- complaint_id: string (nullable = true)
```

In [19]:

```
customer = customer.join(EducData_final, "state", how="left")
```

```
customer.printSchema()
root
 |-- state: string (nullable = true)
 |-- date_received: string (nullable = true)
 |-- product: string (nullable = true)
 |-- sub_product: string (nullable = true)
 |-- issue: string (nullable = true)
 |-- sub_issue: string (nullable = true)
 |-- consumer_complaint_narrative: string (nullable = true)
 |-- company_public_response: string (nullable = true)
 |-- company: string (nullable = true)
 |-- zip_code: string (nullable = true)
 |-- tags: string (nullable = true)
 |-- consumer_consent_provided: string (nullable = true)
 |-- submitted_via: string (nullable = true)
 |-- date_sent_to_company: string (nullable = true)
 |-- company_response: string (nullable = true)
 |-- timely_response: string (nullable = true)
 |-- consumer_disputed: string (nullable = true)
 |-- complaint_id: string (nullable = true)
 |-- NumStudents: double (nullable = true)
 |-- Tot_State_Educ_Revenue: double (nullable = true)
```

---

4. PRE-PROCESSING DATA

4.1. Merging Datasets

- Merging Education data with complaints: The education data was collected based on states and it did not contain state abbreviation (TX). Thus, a new file to map the full form to abbreviations was joined and then dataset was combined with the consumer complaint dataset by state.

```
customer = spark.sql("select * from cust_complaints_all")
customer.printSchema()
root
```

```
 |-- date_received: string (nullable = true)
 |-- product: string (nullable = true)
 |-- sub_product: string (nullable = true)
 |-- issue: string (nullable = true)
 |-- sub_issue: string (nullable = true)
 |-- consumer_complaint_narrative: string (nullable = true)
 |-- company_public_response: string (nullable = true)
 |-- company: string (nullable = true)
 |-- state: string (nullable = true)
 |-- zip_code: string (nullable = true)
 |-- tags: string (nullable = true)
 |-- consumer_consent_provided: string (nullable = true)
 |-- submitted_via: string (nullable = true)
 |-- date_sent_to_company: string (nullable = true)
 |-- company_response: string (nullable = true)
 |-- timely_response: string (nullable = true)
 |-- consumer_disputed: string (nullable = true)
 |-- complaint_id: string (nullable = true)
```

In [19]:

```
customer = customer.join(EducData_final, "state", how="left")
customer.printSchema()
root
 |-- state: string (nullable = true)
 |-- date_received: string (nullable = true)
 |-- product: string (nullable = true)
 |-- sub_product: string (nullable = true)
 |-- issue: string (nullable = true)
 |-- sub_issue: string (nullable = true)
 |-- consumer_complaint_narrative: string (nullable = true)
 |-- company_public_response: string (nullable = true)
 |-- company: string (nullable = true)
 |-- zip_code: string (nullable = true)
 |-- tags: string (nullable = true)
 |-- consumer_consent_provided: string (nullable = true)
 |-- submitted_via: string (nullable = true)
 |-- date_sent_to_company: string (nullable = true)
 |-- company_response: string (nullable = true)
 |-- timely_response: string (nullable = true)
 |-- consumer_disputed: string (nullable = true)
 |-- complaint_id: string (nullable = true)
 |-- NumStudents: double (nullable = true)
 |-- Tot_State_Educ_Revenue: double (nullable = true)
```

- Merging Income data with customer complaints: This was a crucial process, as the zip-code in the consumer complaint dataset was in the format 752XX whereas the income dataset contained full zipcode. Thus, the first 3 letters were extracted as a column – zip and datasets were merged on the zip column.

```
customer = customer.withColumn("zip", customer.zip_code.substr(1,3))
customer.printSchema()
root
 |-- state: string (nullable = true)
 |-- date_received: string (nullable = true)
 |-- product: string (nullable = true)
 |-- sub_product: string (nullable = true)
 |-- issue: string (nullable = true)
 |-- sub_issue: string (nullable = true)
 |-- consumer_complaint_narrative: string (nullable = true)
 |-- company_public_response: string (nullable = true)
 |-- company: string (nullable = true)
 |-- zip_code: string (nullable = true)
 |-- tags: string (nullable = true)
 |-- consumer_consent_provided: string (nullable = true)
 |-- submitted_via: string (nullable = true)
 |-- date_sent_to_company: string (nullable = true)
 |-- company_response: string (nullable = true)
 |-- timely_response: string (nullable = true)
 |-- consumer_disputed: string (nullable = true)
 |-- complaint_id: string (nullable = true)
 |-- NumStudents: double (nullable = true)
 |-- Tot_State_Educ_Revenue: double (nullable = true)
 |-- zip: string (nullable = true)
```

In [21]:

```
#Merging Income with State and Zip
customer = customer.join(income_data,(customer.state==income_data.Inc_State)&
(customer.zip==income_data.Inc_Zip),
                          how="left")
```

In [22]:

```
customer = customer.drop('Inc_State').drop('Inc_Zip')
```

```
customer.printSchema()
root
 |-- state: string (nullable = true)
 |-- date_received: string (nullable = true)
 |-- product: string (nullable = true)
 |-- sub_product: string (nullable = true)
 |-- issue: string (nullable = true)
 |-- sub_issue: string (nullable = true)
 |-- consumer_complaint_narrative: string (nullable = true)
 |-- company_public_response: string (nullable = true)
 |-- company: string (nullable = true)
 |-- zip_code: string (nullable = true)
 |-- tags: string (nullable = true)
 |-- consumer_consent_provided: string (nullable = true)
 |-- submitted_via: string (nullable = true)
 |-- date_sent_to_company: string (nullable = true)
 |-- company_response: string (nullable = true)
 |-- timely_response: string (nullable = true)
 |-- consumer_disputed: string (nullable = true)
 |-- complaint_id: string (nullable = true)
 |-- NumStudents: double (nullable = true)
 |-- Tot_State_Educ_Revenue: double (nullable = true)
 |-- zip: string (nullable = true)
 |-- Tot_Mean_Income: long (nullable = true)
 |-- Tot_Med_Income: long (nullable = true)
 |-- NumHouseholds: double (nullable = true)
```

4.2. Missing Data

- The merged dataset comprises of a lot of missing data across all columns. And missing data exists in three forms – as na, as null, and as “ ”. We need to impute appropriate values for all missing data before further analysis.

- For all categorical columns or string columns, null values are replaced with “NoRecord” and other numerical columns the nulls are replaced with 0.

```
data_filled = data.na.fill(0)
```

In [4]:

```
items = data_filled.columns
print("Number of missing values in each column in the dataset is: "+'\n')
```

```
newcols =[]
for item in items:
    count = data_filled.filter((data_filled[item] == "") | data_filled[item].
isNull()
                                | isnan(data_filled[item])).count()
    print(" " +item+ "  --------->  " +str(count))
    if(count > 0):
        newcols.append(item)
Number of missing values in each column in the dataset is:

 state  --------->  18321
 date_received  --------->  0
 product  --------->  0
 sub_product  --------->  235166
 issue  --------->  0
 sub_issue  --------->  527631
 consumer_complaint_narrative  --------->  878057
 company_public_response  --------->  820539
 company  --------->  0
 zip_code  --------->  111426
 tags  --------->  1084568
 consumer_consent_provided  --------->  23095
 submitted_via  --------->  0
 date_sent_to_company  --------->  0
 company_response  --------->  6
 timely_response  --------->  0
 consumer_disputed  --------->  0
 complaint_id  --------->  0
 NumStudents  --------->  0
 Tot_State_Educ_Revenue  --------->  0
 zip  --------->  111426
 Tot_Mean_Income  --------->  0
 Tot_Med_Income  --------->  0
 NumHouseholds  --------->  0
```

Missing Values for string =""

In [5]:

```
#Imputing missing values that are not na
#newcols.remove('zip_code')
for item in newcols:
    data_filled = data_filled.withColumn(item, regexp_replace(item, '^$', 'No
Record'))
```

In [6]:

```
item = data_filled.columns
for item in items:
    data_filled = data_filled.withColumn(item, regexp_replace(item, 'null', '
NoRecord'))
```

In [7]:

```
#checking after all missing values are imputed
items = data_filled.columns
print("Number of missing values in each column in the dataset is: "+'\n')
for item in items:
    count = data_filled.filter((data_filled[item] == "") | data_filled[item].
isNull() | isnan(data_filled[item])).count()
    print(" " +item+ "  -------->  " +str(count))
```

```
Number of missing values in each column in the dataset is:

 state  -------->  0
 date_received  -------->  0
 product  -------->  0
 sub_product  -------->  0
 issue  -------->  0
 sub_issue  -------->  0
 consumer_complaint_narrative  -------->  0
 company_public_response  -------->  0
 company  -------->  0
 zip_code  -------->  0
 tags  -------->  0
 consumer_consent_provided  -------->  0
 submitted_via  -------->  0
 date_sent_to_company  -------->  0
 company_response  -------->  0
 timely_response  -------->  0
 consumer_disputed  -------->  0
 complaint_id  -------->  0
 NumStudents  -------->  0
 Tot_State_Educ_Revenue  -------->  0
 zip  -------->  0
 Tot_Mean_Income  -------->  0
 Tot_Med_Income  -------->  0
 NumHouseholds  -------->  0
```

4.3. Unique Values

- The number of distinct values in all columns is measured and displayed. We get an idea of how many companies, products and issues we are dealing with.

```
#Counting unique values in a dataset - all columns:
items = data.columns
#items.remove('InvoiceDate')
print("Count of distinct values in each column is: "+'\n')

for item in items:
    count_unique = data.select(col(item)).distinct().count()
    print(" " +item+ "  --------------->" +str(count_unique))
Count of distinct values in each column is:

 state  --------------->64
 date_received  --------------->2685
 product  --------------->18
 sub_product  --------------->77
 issue  --------------->167
 sub_issue  --------------->219
 consumer_complaint_narrative  --------------->362028
 company_public_response  --------------->11
 company  --------------->5253
 zip_code  --------------->22460
 tags  --------------->4
 consumer_consent_provided  --------------->6
 submitted_via  --------------->6
 date_sent_to_company  --------------->2634
 company_response  --------------->9
 timely_response  --------------->2
 consumer_disputed  --------------->3
 complaint_id  --------------->1256552
 NumStudents  --------------->52
 Tot_State_Educ_Revenue  --------------->51
 zip  --------------->999
 Tot_Mean_Income  --------------->854
 Tot_Med_Income  --------------->854
 NumHouseholds  --------------->854
```

4.4. Describing Columns

- All columns and its stats are displayed using the describe command. The distribution of values such as max, min and mean can be noted to get a clear understanding of data

```
items = ['date_received', 'zip_code', 'date_sent_to_company', 'complaint_id']
items2 = ['NumStudents', 'Tot_State_Educ_Revenue',
        'Tot_Mean_Income', 'Tot_Med_Income', 'NumHouseholds']
data.select(items).describe().show()
data.select(items2).describe().show()
+-------+-------------+------------------+--------------------+--------------
---+
|summary|date_received|          zip_code|date_sent_to_company|     complaint
_id|
+-------+-------------+------------------+--------------------+--------------
---+
|  count|      1256552|           1256552|             1256552|          1256
552|
|   mean|         null| 51110.26429484044|                null|1903487.285669
833|
| stddev|         null|30976.412374808933|                null|956531.1977919
687|
|    min|   01/01/2012|             (1352|          01/01/2013|
1|
|    max|   12/31/2018|          NoRecord|          12/31/2018|            99
999|
+-------+-------------+------------------+--------------------+--------------
---+

+-------+--------------------+----------------------+-----------------+------
-----------+-----------------+
|summary|         NumStudents|Tot_State_Educ_Revenue|  Tot_Mean_Income|   Tot
_Med_Income|    NumHouseholds|
+-------+--------------------+----------------------+-----------------+------
-----------+-----------------+
|  count|             1256552|               1256552|          1256552|
1256552|          1256552|
|   mean|4.980687304670399 4E7|    2.61329454332151E8|5298669.044306165|672730
0.299024632|51246.28715877263|
```

```
| stddev|  3.93070148470954E7|  2.3300158740997577E8|4801526.808827681|615724
6.418178633|86179.87624320471|
|    min|                 0.0|                   0.0|                0|
0|              0.0|
|    max|          9.7652535E7|           9.8826436E7|           993821|
993998|    9974.76366156|
+-------+--------------------+----------------------+----------------+------
-----------+-----------------+
```

5. DESCRIPTIVE ANALYTICS

5.1. Common Product and Issues in Complaints

- **Hypothesis:** Finding top 5 companies and identifying where they fail. Which product receives maximum complaints and what issues are most prominent among the product.

- **Result:** In an example of Equifax, INC. which is the company with most complaints, the most complaints are from products such as credit reporting, debt collection. And the issues are dominantly about incorrect information.

```
df1 = data.groupby("company").agg(func.count(lit(1)).alias("Num_Complaints"))
\
.sort('Num_Complaints', ascending=False).limit(20)
```

In [10]:

```
df1.createOrReplaceTempView("df1")
```

In [11]:

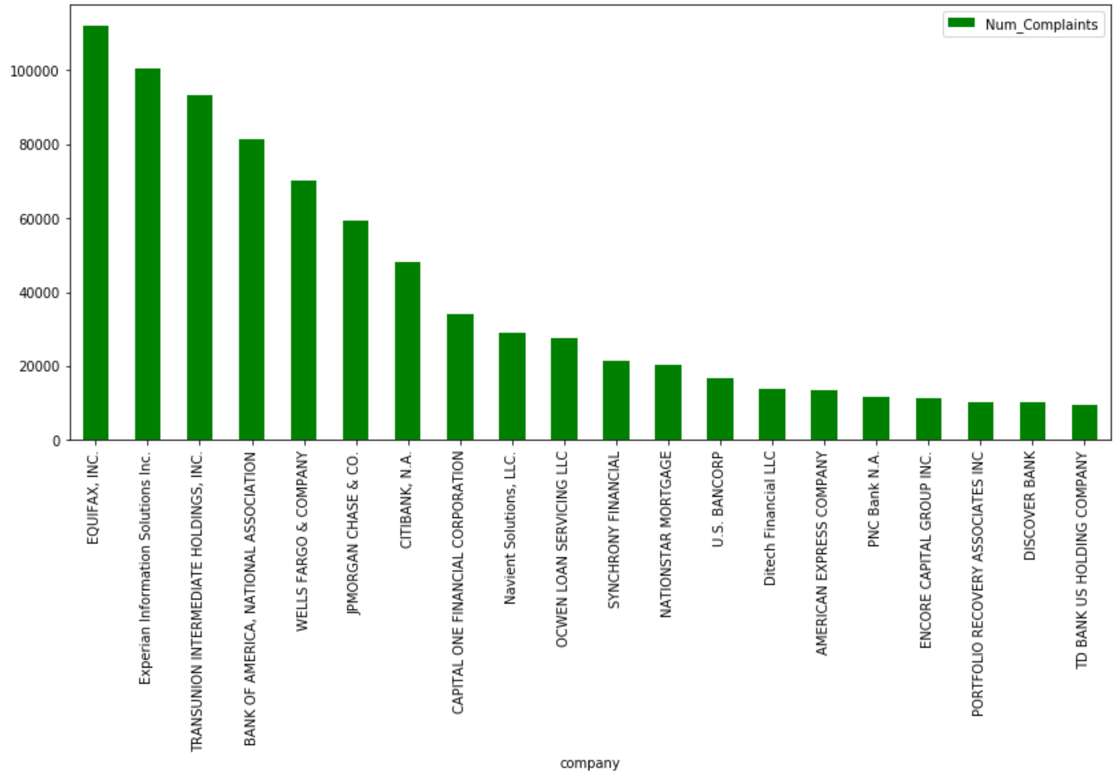
```
%%sql -q -n 20 -o pd_df1
SELECT * FROM df1
```

In [21]:

```
%%local
%matplotlib inline
import matplotlib as plt
pd_df1.plot.bar(x='company', y='Num_Complaints', color='green', figsize=(14,6
))
```

Out[21]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa93c0c9128>
```



In [28]:

```
equifax_data= data.where("company=='EQUIFAX, INC.'")
```

In [29]:

```
equifax_data.groupby('product')\
.agg(func.count(lit(1)).alias("num-complaints"))\
.sort('num-complaints', ascending=False).show(3)
+--------------------+--------------+
|             product|num-complaints|
+--------------------+--------------+
|Credit reporting,...|         62197|
|    Credit reporting|         48128|
|     Debt collection|          1520|
+--------------------+--------------+
only showing top 3 rows
```

In [69]:

```
equifax_data.groupby('issue', 'sub_issue')\
.agg(func.count(lit(1)).alias("num-complaints"))\
.sort('num-complaints', ascending=False).show(3, False)
+---------------------------------------+-----------------------------------+-
-------------+
|issue                                  |sub_issue                          |n
um-complaints|
```

```
+-----------------------------------+----------------------------------+-
-------------+
|Incorrect information on your report  |Information belongs to someone else|1
5996         |
|Incorrect information on credit report|Account status                    |1
2037         |
|Incorrect information on credit report|Information is not mine           |1
1017         |
+-----------------------------------+----------------------------------+-
-------------+
```

5.2. Complaints by Geography

- **Hypothesis:** Identifying complaints in each state and investigating on fixing complaints from top states by zip-code

- **Result:** Top 3 states, CA, FL and TX were analyzed in detail at the zipcode level to get an idea about where complaints arise from.

```
data_filtered = data.where("zip_code != 'NoRecord' and state != 'NoRecord'")
                                                                    In [31]:
data_filtered.groupby("state").agg(func.count(lit(1)).alias("Num_Complaints")
)\
.sort('Num_Complaints', ascending= False).show(5, False)
+-----+--------------+
|state|Num_Complaints|
+-----+--------------+
|CA   |166694        |
|FL   |115219        |
|TX   |98578         |
|NY   |78760         |
|GA   |62062         |
+-----+--------------+
only showing top 5 rows
```

Creating a visualization

```
                                                                    In [32]:
df2 = data_filtered.groupby("state").agg(func.count(lit(1)).alias("Num_Compla
ints")).sort('state')
df2.createOrReplaceTempView("df2")
```

In [33]:

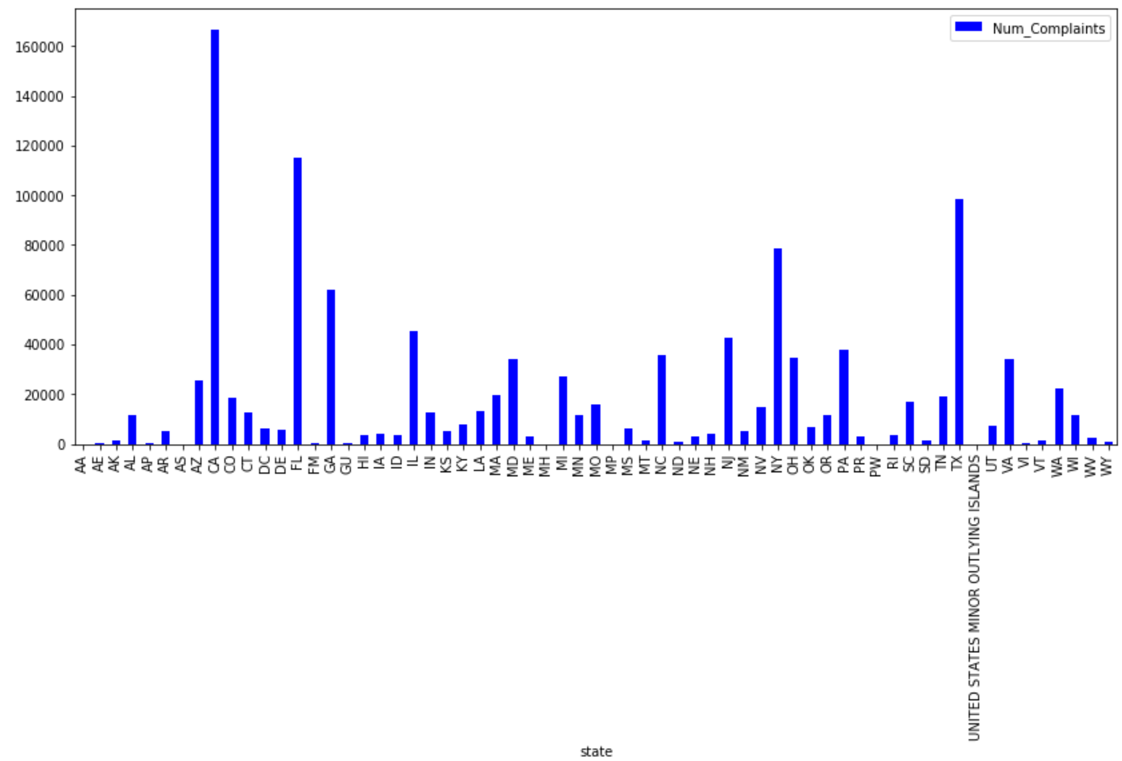
```
%%sql -q -n 100 -o pd_df2
SELECT * FROM df2
```

In [35]:

```
%%local
%matplotlib inline
import matplotlib as plt
pd_df2.plot.bar(x='state', y='Num_Complaints', color='blue', figsize=(14,6))
```

Out[35]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fa92fa79fd0>
```



Now, we know that States like CA, FL, and TX have more complaints, lets deep dive into these 3 at a zip_code level

In [36]:

```
top = ['CA','FL','TX']

for item in top:
    print("Number of complaints by Zipcode in "+item+" ")
    data_filtered.filter(data_filtered['state']==item)\
    .groupby("zip_code")\
    .agg(func.count(lit(1)).alias("Num_Complaints"))\
    .sort('Num_Complaints', ascending=False)\
    .show(5, False)
```

```
Number of complaints by Zipcode in CA
```

```
+--------+-------------+
|zip_code|Num_Complaints|
+--------+-------------+
|945XX   |4059         |
|900XX   |4012         |
|926XX   |2646         |
|917XX   |2306         |
|921XX   |2288         |
+--------+-------------+
only showing top 5 rows

Number of complaints by Zipcode in FL
+--------+-------------+
|zip_code|Num_Complaints|
+--------+-------------+
|330XX   |5228         |
|331XX   |5156         |
|334XX   |4435         |
|333XX   |3280         |
|322XX   |2523         |
+--------+-------------+
only showing top 5 rows

Number of complaints by Zipcode in TX
+--------+-------------+
|zip_code|Num_Complaints|
+--------+-------------+
|770XX   |5801         |
|750XX   |4812         |
|752XX   |2691         |
|760XX   |2279         |
|774XX   |2226         |
+--------+-------------+
only showing top 5 rows
```

5.3. Issues on top products

- **Hypothesis:** To identify what are the most dominating issues in the top 3 products that receive many complaints.

- **Result:** The 3 products – Mortgage, Debt Collection and Credit Card has issues such as Loan modification, debt not owed and Billing disputes as the key issue respectively.

```
data.groupby("product")\
.agg(func.count(lit(1)).alias("Num_complaints"))\
.sort('Num_Complaints', ascending=False).show(5, False)
+---------------------------------------------------------------------------
+--------------+
|product
|Num_complaints|
+---------------------------------------------------------------------------
+--------------+
|Mortgage
|275730        |
|Debt collection
|240429        |
|Credit reporting, credit repair services, or other personal consumer reports
|213842        |
|Credit reporting
|140432        |
|Credit card
|89190         |
+---------------------------------------------------------------------------
+--------------+
only showing top 5 rows
```

In [74]:

```
top = ['Mortgage','Debt collection','Credit card']

for item in top:
    print("Top issues in "+item+" ")
    data.filter(data['product']==item)\
    .groupby("issue")\
    .agg(func.count(lit(1)).alias("Num_Complaints"))\
    .sort('Num_Complaints', ascending=False)\
    .show(5, False)
Top issues in Mortgage
+----------------------------------------+--------------+
```

```
|issue                                    |Num_Complaints|
+-----------------------------------------+--------------+
|Loan modification,collection,foreclosure|112311        |
|Loan servicing, payments, escrow account|77333         |
|Trouble during payment process          |21861         |
|Struggling to pay mortgage              |17970         |
|Application, originator, mortgage broker|17229         |
+-----------------------------------------+--------------+
only showing top 5 rows

Top issues in Debt collection
+--------------------------------------+--------------+
|issue                                 |Num_Complaints|
+--------------------------------------+--------------+
|Cont'd attempts collect debt not owed|60687         |
|Attempts to collect debt not owed    |40842         |
|Communication tactics                |34912         |
|Disclosure verification of debt      |30800         |
|Written notification about debt      |22844         |
+--------------------------------------+--------------+
only showing top 5 rows

Top issues in Credit card
+--------------------------------------+--------------+
|issue                                 |Num_Complaints|
+--------------------------------------+--------------+
|Billing disputes                      |15136         |
|Other                                 |9353          |
|Identity theft / Fraud / Embezzlement|8481          |
|Closing/Cancelling account            |6389          |
|APR or interest rate                  |5506          |
+--------------------------------------+--------------+
only showing top 5 rows
```

5.4. Source of complaints for Timely Response

- **Hypothesis:** For companies, that provide timely response, what is a promising source of complaint.

- **Result:** On counting complaints received via each source in the submitted_via column, it is seen that complaints from Web is clearly dominating in providing a timely response, followed by referrals.

```
data.select('submitted_via').distinct().show()
+-------------+
|submitted_via|
+-------------+
|        Phone|
|          Fax|
|        Email|
|     Referral|
|  Postal mail|
|          Web|
+-------------+
```

In [72]:

```
data = data.withColumn("is_phone", expr("submitted_via == 'Phone'"))
data = data.withColumn("is_fax", expr("submitted_via == 'Fax'"))
data = data.withColumn("is_email", expr("submitted_via == 'Email'"))
data = data.withColumn("is_referral", expr("submitted_via == 'Referral'"))
data = data.withColumn("is_postalmail", expr("submitted_via == 'Postal mail'"
))
data = data.withColumn("is_web", expr("submitted_via == 'Web'"))
```

In [73]:

```
data_timely = data.where("timely_response = 'Yes'")
```

In [74]:

```
data_timely\
.groupby("company")\
.agg(sum(data_timely['is_phone'].cast(IntegerType())).alias("phone"),
     sum(data_timely['is_fax'].cast(IntegerType())).alias("fax"),
     sum(data_timely['is_email'].cast(IntegerType())).alias("email"),
     sum(data_timely['is_referral'].cast(IntegerType())).alias("referral"),
     sum(data_timely['is_postalmail'].cast(IntegerType())).alias("postalmail"
),
     sum(data_timely['is_web'].cast(IntegerType())).alias("web"))\
.sort('phone','fax','email','referral',
      'postalmail','web', ascending=False).show(5)
+--------------------+-----+----+-----+--------+----------+-----+
```

```
|            company|phone| fax|email|referral|postalmail|  web|
+--------------------+-----+----+-----+--------+----------+-----+
|BANK OF AMERICA, ...| 6171|1301|   44|   24934|      4224|43089|
|WELLS FARGO & COM...| 6072|1195|   31|   19312|      3450|36820|
|JPMORGAN CHASE & CO.| 4574| 923|   25|   16020|      3033|34661|
|      CITIBANK, N.A.| 3317| 619|   14|    7627|      2868|33519|
|Experian Informat...| 3176|1975|   17|    4394|      8802|82213|
+--------------------+-----+----+-----+--------+----------+-----+
only showing top 5 rows
```

Visualization

In [36]:

```
data.createOrReplaceTempView("CustData")
df4 = spark.sql("select submitted_via, count(complaint_id) as Num_Complaints from CustData group by submitted_via")
df4.show()
+-------------+--------------+
|submitted_via|Num_Complaints|
+-------------+--------------+
|        Phone|         74956|
|          Fax|         18659|
|        Email|           382|
|     Referral|        171390|
|  Postal mail|         66734|
|          Web|        924431|
+-------------+--------------+
```

In [37]:

```
df4.createOrReplaceTempView("df4")
```

In [38]:

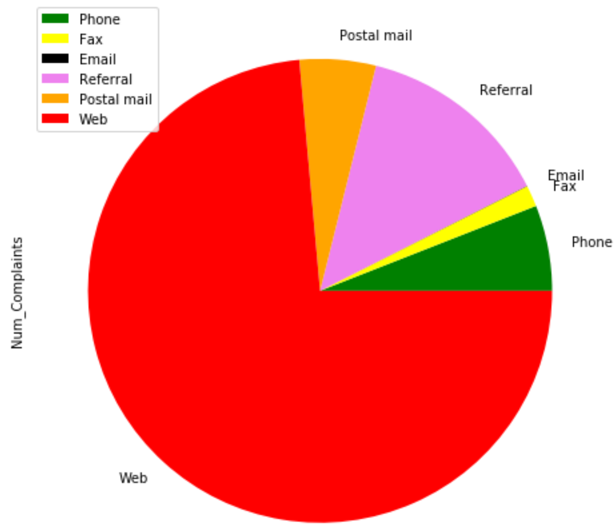
```
%%sql -q -n 100 -o pd_df4
SELECT * FROM df4
```

In [118]:

```
%%local
%matplotlib inline
import matplotlib as plt
ax = pd_df4.plot.pie(labels=pd_df4['submitted_via'], y='Num_Complaints',
                colors=['green', 'yellow', 'black', 'violet', 'orange', 'red'
], figsize=(15,8))
ax.legend(loc="upper left")
```

Out[118]:

<matplotlib.legend.Legend at 0x7fa92b1fdd30>



5.5. Income affecting Complaints

- **Hypothesis:** Testing the idea that people who earn less (middle class income) tend to complain more about issues than the elitist class.

- **Result:** On segmenting the data based on buckets of 50, 50k-100k, 100k-150k and <200k, it is noted that people belong to the bucket 50k to 100k tend to file more complaints proving the hypothesis.

```
data.select('Tot_Mean_Income').describe().show()
+-------+-----------------+
|summary|  Tot_Mean_Income|
+-------+-----------------+
|  count|          1256552|
|   mean|5298669.044306165|
| stddev|4801526.808827681|
|    min|                0|
|    max|           993821|
+-------+-----------------+
```

In [17]:

```
data_2 = data.filter("Tot_Mean_Income != 0")
```

```
data_2 = data_2.withColumn("Tot_Mean_Income_int", data_2['Tot_Mean_Income'].c
ast(IntegerType()))
data_2.select('Tot_Mean_Income_int').describe().show()
+-------+-------------------+
|summary|Tot_Mean_Income_int|
+-------+-------------------+
|  count|            1049566|
|   mean|   6343625.06498972|
| stddev|  4579572.236572872|
|    min|              35339|
|    max|           19118905|
+-------+-------------------+
```

In [18]:

```
data_2 = data_2.withColumn("Less50k", expr("Tot_Mean_Income_int < 50000"))
data_2 = data_2.withColumn("Bet50to100k", expr("Tot_Mean_Income_int >= 50000
AND Tot_Mean_Income_int < 100000"))
data_2 = data_2.withColumn("Bet100to150k", expr("Tot_Mean_Income_int >= 10000
0 AND Tot_Mean_Income_int <150000"))
data_2 = data_2.withColumn("Bet150to200k", expr("Tot_Mean_Income_int >= 15000
0 AND Tot_Mean_Income_int <200000"))
```

In [19]:

```
items = ['Less50k','Bet50to100k','Bet100to150k','Bet150to200k']
for item in items:
    cnt = data_2.where(item).count()
    print("Number of complaints in "+item+" population: "+str(cnt))
Number of complaints in Less50k population: 13
Number of complaints in Bet50to100k population: 644
Number of complaints in Bet100to150k population: 80
Number of complaints in Bet150to200k population: 283
```

5.6. Issues delaying timely response

- **Hypothesis:** Of the companies that failed to provide timely response, what is the product and issue that's causing it?

- **Result:** In the filtered dataset, the top issues are related to debt collection and the products include bank account (wells Fargo), Credit reporting (Equifax) and Mortgage (Bank of America).

```
data_notimely= data.where("timely_response=='No'")
```

In [97]:

```
data_notimely.groupby('issue')\
.agg(func.count(lit(1)).alias('Num_complaints'))\
.sort('Num_complaints', ascending=False)\
.show(5, False)
+----------------------------------------+--------------+
|issue                                   |Num_complaints|
+----------------------------------------+--------------+
|Cont'd attempts collect debt not owed   |4131          |
|Communication tactics                   |2452          |
|Loan modification,collection,foreclosure|2155          |
|Disclosure verification of debt         |2035          |
|Incorrect information on your report    |1734          |
+----------------------------------------+--------------+
only showing top 5 rows
```

In [98]:

```
data_notimely.groupby('company', 'product')\
.agg(func.count(lit(1)).alias('Num_complaints'))\
.sort('Num_complaints', ascending=False)\
.show(3)
+--------------------+--------------------+--------------+
|             company|             product|Num_complaints|
+--------------------+--------------------+--------------+
|WELLS FARGO & COM...|Bank account or s...|          1530|
|       EQUIFAX, INC.|Credit reporting,...|          1512|
|BANK OF AMERICA, ...|            Mortgage|          1173|
+--------------------+--------------------+--------------+
only showing top 3 rows
```

5.7. Education and complaints correlation

- **Hypothesis:** If the complaints are silly and not relevant, I would expect majority of complaints to negatively correlate with educational information.

- **Result:** We can see that highly educated states with more high school graduates and state education revenue have filed more complaints. This indicates that complaints are quite relevant and fixing them is crucial to consumer retention.

```
data = data.withColumn('NumStudents', data.NumStudents.cast(DecimalType(18, 2
)))
data = data.withColumn('Tot_State_Educ_Revenue', data.Tot_State_Educ_Revenue.
cast(DecimalType(18, 2)))
data = data.withColumn('complaint_id', data.complaint_id.cast(IntegerType()))
```

In [105]:

```
data_educ = data.where("Tot_State_Educ_Revenue != 0")
```

In [106]:

```
data_educ.groupby('state')\
.agg(first("Tot_State_Educ_Revenue").alias('States-Educ_Revenue'),
     first("NumStudents").alias("NumberOfStudents"),
    func.count(func.lit(1)).alias("num_complaints"))\
.sort('States-Educ_Revenue','NumberOfStudents', ascending=False).show()
+-----+-------------------+----------------+--------------+
|state|States-Educ_Revenue|NumberOfStudents|num_complaints|
+-----+-------------------+----------------+--------------+
|   CA|       774232107.00|    127943053.00|        172954|
|   NY|       446777506.00|     59646734.00|         84384|
|   TX|       362816451.00|     97652535.00|        105251|
|   MI|       232956123.00|     35882600.00|         29754|
|   FL|       218933074.00|     55298947.00|        123289|
|   NJ|       196224231.00|     27870060.00|         46988|
|   PA|       188960285.00|     39643588.00|         43325|
|   OH|       188741219.00|     40008831.00|         37720|
|   IL|       179809932.00|     45100706.00|         48653|
|   NC|       154477379.00|     30015324.00|         38794|
|   GA|       153062090.00|     33651759.00|         65512|
|   WA|       144118417.00|     22506488.00|         24012|
|   MN|       130631274.00|     18710320.00|         13142|
|   IN|       129311835.00|     22473056.00|         14223|
|   VA|       122374713.00|     26046432.00|         36853|
|   MA|       117629806.00|     21054625.00|         22386|
|   WI|       110117605.00|     19460842.00|         13000|
```

```
|   MD|        98826436.00|     18645240.00|         35995|
|   MO|        83661928.00|     20061151.00|         17637|
|   AL|        79704806.00|     16436994.00|         14173|
+-----+-------------------+----------------+--------------+
```

5.8. Delay in sending complaints & timely response

- **Hypothesis:** Since we have date_received and date_sent_to_company, it's safe to assume that some complaints take long time to resolve. Hence, I chose to measure if it affects timely response.

- **Result:** Difference in dates were measured. It was noted that though there is significant delay in receiving complaints, bigger firms like Equifax, BOA are able to provide timely response. However small firms struggle to do so.

```
data = data.withColumn("date_received", to_date(unix_timestamp(col("date_received"), "M/dd/yyyy").cast("timestamp")))
data = data.withColumn("date_sent_to_company", to_date(unix_timestamp(col("date_sent_to_company"), "M/dd/yyyy").cast("timestamp")))
data = data.withColumn("Delay", datediff(col('date_sent_to_company'), col('date_received')))
```

In [195]:

```
data_delay.select('Delay', 'company', 'timely_response')\
.where("timely_response=='No'")\
.sort('Delay', ascending=False).show()

data_delay.select('Delay', 'company', 'timely_response')\
.where("timely_response=='Yes'")\
.sort('Delay', ascending=False).show()
```

```
+-----+--------------------+---------------+
|Delay|             company|timely_response|
+-----+--------------------+---------------+
| 1133|SMS Check Recover...|             No|
| 1106|  Westhill Financial|             No|
|  909|Clayton Holdings LLC|             No|
|  832|Advanced Recovery...|             No|
|  643|Elite Financial S...|             No|
|  588|Phillips, Reinhar...|             No|
|  587|Phillips, Reinhar...|             No|
|  585|Phillips, Reinhar...|             No|
|  573|  Kadent Corporation|             No|
|  572|Evans Law Associa...|             No|
```

```
|  564|Phillips, Reinhar...|             No|
|  551|Phillips, Reinhar...|             No|
|  536|Phillips, Reinhar...|             No|
|  536|Phillips, Reinhar...|             No|
|  531|Phillips, Reinhar...|             No|
|  525|Phillips, Reinhar...|             No|
+-----+--------------------+---------------+

+-----+--------------------+---------------+
|Delay|             company|timely_response|
+-----+--------------------+---------------+
| 1962|    BANK OF THE WEST|            Yes|
| 1754|       EQUIFAX, INC.|            Yes|
| 1753|Experian Informat...|            Yes|
| 1613|NETSPEND CORPORATION|            Yes|
| 1601|       EQUIFAX, INC.|            Yes|
| 1553|    BB&T CORPORATION|            Yes|
| 1421|National Manageme...|            Yes|
| 1395|Merchants Credit ...|            Yes|
| 1365|HSBC NORTH AMERIC...|            Yes|
| 1270|      CITIBANK, N.A.|            Yes|
| 1188| SYNCHRONY FINANCIAL|            Yes|
| 1019|Jim Bottin Enterp...|            Yes|
|  999|Sortis Financial,...|            Yes|
|  993|AMERICAN EXPRESS ...|            Yes|
|  959|Red Cedar Service...|            Yes|
|  855|  CAC Financial Corp|            Yes|
|  838|        CREDICO. INC|            Yes|
|  833|CCS Financial Ser...|            Yes|
|  825|CCS Financial Ser...|            Yes|
|  783| NATIONSTAR MORTGAGE|            Yes|
+-----+--------------------+---------------+
only showing top 20 rows
```

5.9. Month level Analysis

- **Hypothesis:** Data was analyzed at a month level to investigate the existence of any seasonal patterns and analyze them based on real world events.

- **Result:** There seems to be no pattern in the data. However, March month has received the most complaints on average.

```
#Creating a column called Month

data = data.withColumn("date_sent_to_company", to_date(unix_timestamp(col("da
te_sent_to_company"), "M/dd/yyyy").cast("timestamp")))
data = data.withColumn("Month", date_format(col("date_sent_to_company"), 'MMM
MM'))
data.select("Month").show(5)
+--------+
|   Month|
+--------+
|November|
|November|
|November|
|November|
|November|
+--------+
only showing top 5 rows
```

In [68]:

```
df9 = data.groupby("Month")\
.agg(func.count(lit(1)).alias("Num_Complaints"))\
.sort(to_date(col('Month'), 'MMMM'))
df9.show()
+---------+--------------+
|    Month|Num_Complaints|
+---------+--------------+
|  January|        111990|
| February|        111496|
|    March|        125247|
|    April|        100115|
|      May|         98876|
|     June|         98342|
|     July|        102666|
|   August|        106438|
|September|        106341|
|  October|        104961|
```

```
| November|         93313|
| December|         96767|
+---------+--------------+
```

In [87]:

```
df9.createOrReplaceTempView("df9")
```

In [88]:

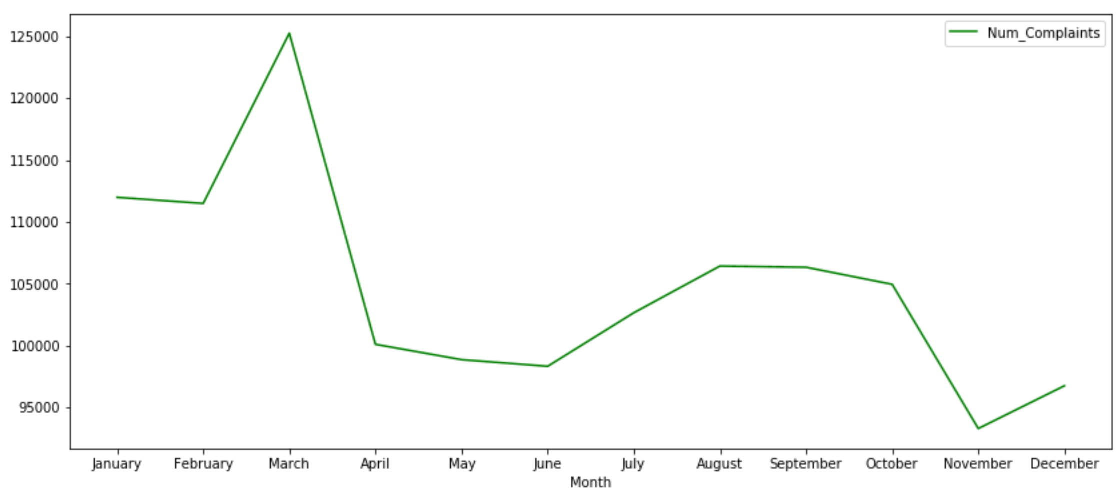
```
%%sql -q -n 20 -o pd_df9
SELECT * FROM df9
```

In [92]:

```
%%local
%matplotlib inline
import matplotlib as plt
ax = pd_df9.plot.line(x= 'Month', y='Num_Complaints', color='green', figsize=
(14,6))
ax.set_xticks(pd_df9.index)
ax.set_xticklabels(pd_df9.Month)
```

Out[92]:

```
[Text(0, 0, 'January'),
 Text(0, 0, 'February'),
 Text(0, 0, 'March'),
 Text(0, 0, 'April'),
 Text(0, 0, 'May'),
 Text(0, 0, 'June'),
 Text(0, 0, 'July'),
 Text(0, 0, 'August'),
 Text(0, 0, 'September'),
 Text(0, 0, 'October'),
 Text(0, 0, 'November'),
 Text(0, 0, 'December')]
```



5.10. Product complaint Segmentation

- **Hypothesis:** Our dataset comprised of 18 products in total, what is the distribution of these products like?

- **Result:** Each product is grouped, and number of complaints are recorded and visualized as shown below. Mortgage, Debt collection and Credit Reporting cover major territories.

```
df10 = data.groupby('product')\
.agg(func.count(lit(1)).alias("Num_Complaints"))
```

In [94]:

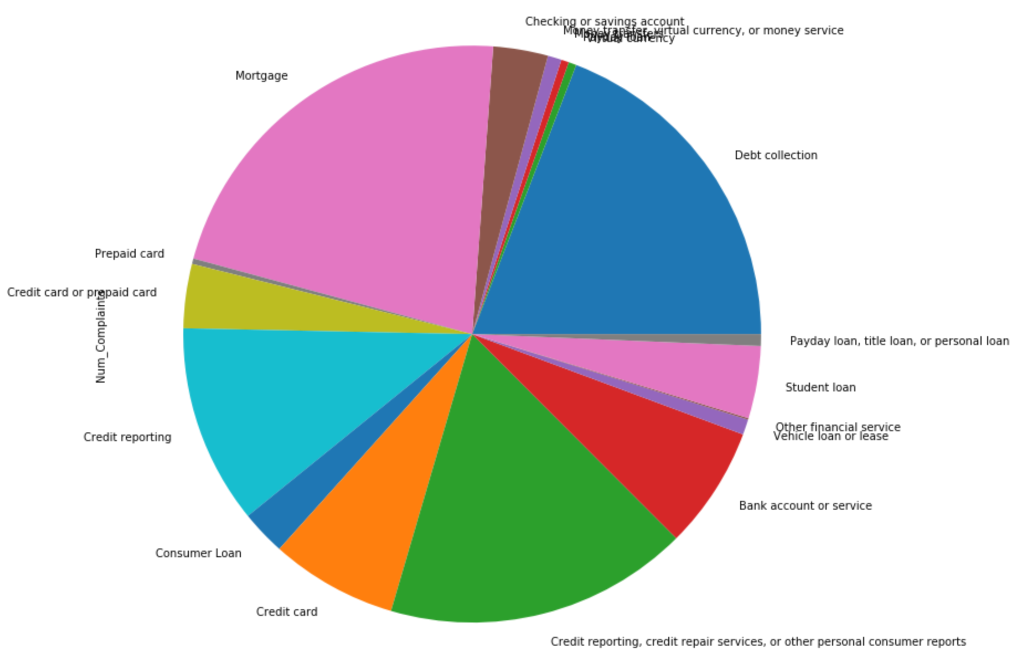
```
df10.createOrReplaceTempView("df10")
```

In [95]:

```
%%sql -q -n 20 -o pd_df10
SELECT * FROM df10
```

In [113]:

```
%%local
%matplotlib inline
import matplotlib as plt ax = pd_df10.plot.pie(labels=pd_df10['product'], y='
Num_Complaints', figsize=(22,12), legend=False)
```



6. CONCLUSION

- The dataset posed several challenges and it was filled with insights and left room for numerous analytical thought points.

- With key indicators like timely_response, issues, products and company I was able to find insights on how to improve customer satisfaction and reduce complaints.

- A summary of key takeaways from this dataset is that:

  1. Most companies, face complaints regarding erroneous report, incorrect information and communication tactics.

  2. Some key areas to improve on the product line would be on Mortgage, Debt, and Credit reporting.

  3. Smaller firms are struggling to provide timely response when there is a delay in receiving complaints. This should be prioritized to pass information faster to tier 2 firms.

  4. Highly educated states also face abundance of issues in relating to credit card and other products. Thus, the idea of automation to remove human errors from the job is a favorably efficient idea.

  5. People from the average income bucket contribute to high margin of complaints, this is just an indication of nature of complaints to be critical and at the same time made of trivial errors.

  6. There is no seasonal patter at the month level, however March month on average receives a greater number of complaints. This could be further investigated by each company's acquisition and customer data to uncover more insights.

7. REFERENCES

1. **Income Dataset** [https://www.kaggle.com/goldenoakresearch/us-household-income-stats-geo-locations#kaggle_income.csv]

2. **Education Dataset** [https://www.kaggle.com/noriuk/us-education-datasets-unification-project#states_all.csv]

3. **Pandas Visualization** [https://pandas.pydata.org/pandasdocs/version/0.23/generated/pandas.DataFrame.plot.html]

4. **Spark Manipulations [** https://spark.apache.org/docs/latest/sql-getting-started.html **]**

5. **Handling Null Values [** https://stackoverflow.com/questions/48059640/replace-all-null-with-spaces-before-writing-data-out-in-the-spark-scala **]**

6. **General Queries [** stackoveflow **]**