In [1]:
```
from pyspark.sql import SparkSession
spark = SparkSession.builder \
  .appName('2.1. Google Cloud Storage (CSV) & Spark DataFrames') \
  .getOrCreate()
```

Big Data Technologies

Fraud Detection of CMS Medicare

Team 8

Gayatri Awate 20015754 Shreneel Kala 20015446 Simoni Patani 20011528

A data science model is being built to predict fraud in the medical insurance industry using real-time analysis and classification algorithms. The tool can benefit patients, pharmacies, doctors, and the industry by handling the impact of fraud and increasing credibility. Medicare and Medicaid services have been facing significant fiscal losses due to healthcare fraud. The project aims to build a basic data model, a comprehensive AI model, and a benchmark measurement to develop a market-ready product for fraud detection. Target Value will be is_fraud

In [2]:
```
PartDRawData = "shreneel-bigdata1/Medicare_Part_D_Prescribers_by_Provider_and_Drug_Datas
```

Extracting data from Google data storage buckets

In [3]:
```
from google.cloud import storage

gcs_client = storage.Client()
bucket = gcs_client.bucket('shreneel-bigdata1')

list(bucket.list_blobs(prefix='Medicare_Part_D_Prescribers_by_Provider_and_Drug_Dataset_
```

Out[3]:
```
[<Blob: shreneel-bigdata1, Medicare_Part_D_Prescribers_by_Provider_and_Drug_Dataset_201
6.csv, 1683164667126860>]
```

In [4]:
```
!hdfs dfs -ls 'gs://shreneel-bigdata1/Medicare_Part_D_Prescribers_by_Provider_and_Drug_D
```

```
-rwx------   3 root root 3580761897 2023-05-04 01:44 gs://shreneel-bigdata1/Medicare_Par
t_D_Prescribers_by_Provider_and_Drug_Dataset_2016.csv
```

Creating Spark Dataframe

In [5]:
```
partD_drug_rawdata = spark \
  .read \
  .option ( "inferSchema" , "true" ) \
  .option ( "header" , "true" ) \
  .csv ( "gs://shreneel-bigdata1/Medicare_Part_D_Prescribers_by_Provider_and_Drug_Datase

partD_drug_rawdata.printSchema()
```

```
root
 |-- Prscrbr_NPI: integer (nullable = true)
 |-- Prscrbr_Last_Org_Name: string (nullable = true)
 |-- Prscrbr_First_Name: string (nullable = true)
 |-- Prscrbr_City: string (nullable = true)
 |-- Prscrbr_State_Abrvtn: string (nullable = true)
 |-- Prscrbr_State_FIPS: string (nullable = true)
 |-- Prscrbr_Type: string (nullable = true)
 |-- Prscrbr_Type_Src: string (nullable = true)
 |-- Brnd_Name: string (nullable = true)
 |-- Gnrc_Name: string (nullable = true)
 |-- Tot_Clms: integer (nullable = true)
 |-- Tot_30day_Fills: double (nullable = true)
 |-- Tot_Day_Suply: integer (nullable = true)
 |-- Tot_Drug_Cst: double (nullable = true)
 |-- Tot_Benes: integer (nullable = true)
 |-- GE65_Sprsn_Flag: string (nullable = true)
 |-- GE65_Tot_Clms: integer (nullable = true)
 |-- GE65_Tot_30day_Fills: double (nullable = true)
 |-- GE65_Tot_Drug_Cst: double (nullable = true)
 |-- GE65_Tot_Day_Suply: integer (nullable = true)
 |-- GE65_Bene_Sprsn_Flag: string (nullable = true)
 |-- GE65_Tot_Benes: integer (nullable = true)
```

In [6]:
```
df1=partD_drug_rawdata
```

In [7]:
```
df1
```

Out[7]:
```
DataFrame[Prscrbr_NPI: int, Prscrbr_Last_Org_Name: string, Prscrbr_First_Name: string, P
rscrbr_City: string, Prscrbr_State_Abrvtn: string, Prscrbr_State_FIPS: string, Prscrbr_T
ype: string, Prscrbr_Type_Src: string, Brnd_Name: string, Gnrc_Name: string, Tot_Clms: i
nt, Tot_30day_Fills: double, Tot_Day_Suply: int, Tot_Drug_Cst: double, Tot_Benes: int, G
E65_Sprsn_Flag: string, GE65_Tot_Clms: int, GE65_Tot_30day_Fills: double, GE65_Tot_Drug_
Cst: double, GE65_Tot_Day_Suply: int, GE65_Bene_Sprsn_Flag: string, GE65_Tot_Benes: int]
```

Displaying the number of rows and columns in Dataset 1

In [8]:
```
num_rows = df1.count()
num_cols = len(df1.columns)
print("Number of rows: ", num_rows)
print("Number of columns: ", num_cols)
```

```
Number of rows:  24964300
Number of columns:  22
```

Selecting Essential columns from Dataframe

In [9]:
```
from pyspark.sql.functions import col

partD_Drug_pd1 = df1.select(col("Prscrbr_NPI"), col("Prscrbr_City"), col("Prscrbr_State_
                            col("Prscrbr_Last_Org_Name"), col("Prscrbr_First_Name"), \
                            col("Prscrbr_Type"), col("Brnd_Name"), col("Gnrc_Name"), \
                            col("Tot_Drug_Cst"), col("Tot_Clms"), col("Tot_Day_Suply"))
```

In [10]:
```
partD_pd1 = partD_Drug_pd1
```

In [11]:
```
from pyspark.sql.functions import col
from pyspark.sql.types import StringType



# Select the required columns
```

Loading [MathJax]/extensions/Safe.js

```
partD_Drug_df = partD_pd1.select(col('Prscrbr_NPI'), col('Brnd_Name'), col('Tot_Drug_Cst

# Cast the 'npi' column to 'StringType'
partD_Drug_df = partD_Drug_df.withColumn('Prscrbr_NPI', col('Prscrbr_NPI').cast(StringTy

# Show the resulting DataFrame
partD_Drug_df.show()
```

```
+-----------+--------------------+------------+--------+-------------+-----------------+
|Prscrbr_NPI|           Brnd_Name|Tot_Drug_Cst|Tot_Clms|Tot_Day_Suply|     Prscrbr_Type|
+-----------+--------------------+------------+--------+-------------+-----------------+
| 1003000126|Atorvastatin Calcium|      139.32|      13|          450|Internal Medicine|
| 1003000126|   Ciprofloxacin Hcl|       80.99|      11|           96|Internal Medicine|
| 1003000126| Doxycycline Hyclate|      586.12|      20|          199|Internal Medicine|
| 1003000126|             Eliquis|     6065.02|      17|          510|Internal Medicine|
| 1003000126|          Furosemide|       45.76|      17|          405|Internal Medicine|
| 1003000126|     Hydralazine Hcl|      169.48|      16|          420|Internal Medicine|
| 1003000126|Isosorbide Mononi...|      372.63|      33|         1005|Internal Medicine|
| 1003000126|        Levofloxacin|      222.41|      26|          159|Internal Medicine|
| 1003000126|          Lisinopril|      129.24|      31|          960|Internal Medicine|
| 1003000126| Metoprolol Tartrate|      183.29|      33|         1050|Internal Medicine|
| 1003000126|       Metronidazole|      152.66|      12|          127|Internal Medicine|
| 1003000126| Pantoprazole Sodium|      140.83|      15|          450|Internal Medicine|
| 1003000126|          Prednisone|       59.96|      20|          121|Internal Medicine|
| 1003000126|     Warfarin Sodium|      197.69|      12|          236|Internal Medicine|
| 1003000126|             Xarelto|     12110.2|      34|          943|Internal Medicine|
| 1003000142|Acetaminophen-Cod...|      577.96|      51|         1398|   Anesthesiology|
| 1003000142|   Amitriptyline Hcl|       254.2|      29|          870|   Anesthesiology|
| 1003000142|            Baclofen|     2106.97|     104|         3194|   Anesthesiology|
| 1003000142|             Butrans|    24514.23|      63|         1764|   Anesthesiology|
| 1003000142| Cyclobenzaprine Hcl|       31.88|      11|          300|   Anesthesiology|
+-----------+--------------------+------------+--------+-------------+-----------------+
only showing top 20 rows
```

In [12]:

```
 # Select the required columns
partD_Spec_pd1 = partD_pd1.select(col('Prscrbr_NPI'), col('Prscrbr_Type'))
# Show the resulting DataFrame
partD_Spec_pd1.show()
```

```
+-----------+----------------+
|Prscrbr_NPI|     Prscrbr_Type|
+-----------+----------------+
| 1003000126|Internal Medicine|
| 1003000126|Internal Medicine|
| 1003000126|Internal Medicine|
| 1003000126|Internal Medicine|
| 1003000126|Internal Medicine|
| 1003000126|Internal Medicine|
| 1003000126|Internal Medicine|
| 1003000126|Internal Medicine|
| 1003000126|Internal Medicine|
| 1003000126|Internal Medicine|
| 1003000126|Internal Medicine|
| 1003000126|Internal Medicine|
| 1003000126|Internal Medicine|
| 1003000126|Internal Medicine|
| 1003000126|Internal Medicine|
| 1003000142|   Anesthesiology|
| 1003000142|   Anesthesiology|
| 1003000142|   Anesthesiology|
| 1003000142|   Anesthesiology|
| 1003000142|   Anesthesiology|
+-----------+----------------+
only showing top 20 rows
```

In [13]:

```
partD_Drug_df.head()
```

Out[13]:

```
Row(Prscrbr_NPI='1003000126', Brnd_Name='Atorvastatin Calcium', Tot_Drug_Cst=139.32, Tot
_Clms=13, Tot_Day_Suply=450, Prscrbr_Type='Internal Medicine')
```

Selecting the required columns and Showing the resulting DataFrame

In [14]:

```
# Select the required columns
partD_pd0= partD_pd1.select(col('Prscrbr_NPI'), col('Prscrbr_City'), col('Prscrbr_State_
                               col('Prscrbr_Last_Org_Name'), col('Prscrbr_First_Name'),
                               col('Prscrbr_Type'))

# Show the resulting DataFrame
partD_pd0.show()
```

```
+-----------+------------+--------------------+---------------------+------------------+
-----------------+
|Prscrbr_NPI|Prscrbr_City|Prscrbr_State_Abrvtn|Prscrbr_Last_Org_Name|Prscrbr_First_Name|
Prscrbr_Type|
+-----------+------------+--------------------+---------------------+------------------+
-----------------+
| 1003000126|  Cumberland|                  MD|            Enkeshafi|           Ardalan|
Internal Medicine|
| 1003000126|  Cumberland|                  MD|            Enkeshafi|           Ardalan|
Internal Medicine|
| 1003000126|  Cumberland|                  MD|            Enkeshafi|           Ardalan|
Internal Medicine|
| 1003000126|  Cumberland|                  MD|            Enkeshafi|           Ardalan|
Internal Medicine|
| 1003000126|  Cumberland|                  MD|            Enkeshafi|           Ardalan|
Internal Medicine|
| 1003000126|  Cumberland|                  MD|            Enkeshafi|           Ardalan|
Internal Medicine|
| 1003000126|  Cumberland|                  MD|            Enkeshafi|           Ardalan|
Internal Medicine|
| 1003000126|  Cumberland|                  MD|            Enkeshafi|           Ardalan|
Internal Medicine|
| 1003000126|  Cumberland|                  MD|            Enkeshafi|           Ardalan|
Internal Medicine|
| 1003000126|  Cumberland|                  MD|            Enkeshafi|           Ardalan|
Internal Medicine|
| 1003000126|  Cumberland|                  MD|            Enkeshafi|           Ardalan|
Internal Medicine|
| 1003000126|  Cumberland|                  MD|            Enkeshafi|           Ardalan|
Internal Medicine|
| 1003000126|  Cumberland|                  MD|            Enkeshafi|           Ardalan|
Internal Medicine|
| 1003000126|  Cumberland|                  MD|            Enkeshafi|           Ardalan|
Internal Medicine|
| 1003000126|  Cumberland|                  MD|            Enkeshafi|           Ardalan|
Internal Medicine|
| 1003000142|      Toledo|                  OH|               Khalil|            Rashid|
Anesthesiology|
| 1003000142|      Toledo|                  OH|               Khalil|            Rashid|
Anesthesiology|
| 1003000142|      Toledo|                  OH|               Khalil|            Rashid|
Anesthesiology|
| 1003000142|      Toledo|                  OH|               Khalil|            Rashid|
Anesthesiology|
| 1003000142|      Toledo|                  OH|               Khalil|            Rashid|
Anesthesiology|
+-----------+------------+--------------------+---------------------+------------------+
-----------------+
only showing top 20 rows
```

Dropping Duplicates directly as dataset is large and will not affect analysis

In [15]:
```python
partD_catfpd = partD_pd0.drop_duplicates()
```

In [16]:
```python
partD_catfpd.head()
```

Out[16]:
```
Row(Prscrbr_NPI=1184760365, Prscrbr_City='Charlotte', Prscrbr_State_Abrvtn='MI', Prscrbr
_Last_Org_Name='Knowles', Prscrbr_First_Name='Lisa', Prscrbr_Type='Dentist')
```

Renaming columns for readability and seamless future use

In [17]:
```python
# Define the rename dictionary
                = {'Prscrbr_First_Name':'first_name', 'Prscrbr_Last_Org_Name':'last_name', \
```

```
                'Prscrbr_City':'city', 'Prscrbr_State_Abrvtn':'state', 'Prscrbr_Type':'Sp



# Rename the columns
for old_col, new_col in rename_dict.items():
    partD_catfpd = partD_catfpd.withColumnRenamed(old_col, new_col)

# Show the resulting DataFrame
partD_catfpd.show()
```

```
+-----------+------------------+-----+----------+----------+--------------------+
|Prscrbr_NPI|              city|state| last_name|first_name|          Speciality|
+-----------+------------------+-----+----------+----------+--------------------+
| 1154311595|          Harrison|   AR|    Stills|     David|     Family Practice|
| 1154312072|          New York|   NY|   Newland| Jamesetta|  Nurse Practitioner|
| 1154318798|           Cloquet|   MN|   Kosmach|     Lynne|  Nurse Practitioner|
| 1154327534|       Saint Louis|   MO|     Smith|   Kenneth|     General Surgery|
| 1154336956|        Southfield|   MI|      Gork|   Stephen|             Dentist|
| 1154350544|        Pittsburgh|   PA|    Covato|     Lucia|             Dentist|
| 1154354868|          Portland|   OR|  Tenscher|       Max|  Nurse Practitioner|
| 1154356194|         Fairfield|   CT|     Ranno|   Michele|   Internal Medicine|
| 1154356988|            Bourne|   MA|  Langston|      John|Maxillofacial Sur...|
| 1154359040|    Mechanicsville|   VA|     Kelly|     David|     Family Practice|
| 1154361558|         San Diego|   CA|    Dysart|   Jeffrey|     Family Practice|
| 1154362275|           Cullman|   AL|   Windham|   Gregory|     General Surgery|
| 1154365476|Palm Beach Gardens|   FL|    Pinder|     Carol|  Nurse Practitioner|
| 1154372266|         Oak Ridge|   TN|     Greer|      Mark|           Optometry|
| 1154373827|      Myrtle Beach|   SC|    Purvis|    Robert|         Dermatology|
| 1154373850|           Durango|   CO|   Youssef|       Jim|  Orthopedic Surgery|
| 1154379014|             Omaha|   NE|   Janssen|     Misty|     Family Practice|
| 1154395382|             Islip|   NY|Firouztale|    Edward|           Neurology|
| 1154397503|      White Plains|   NY|    Oksman|     Henry|       Ophthalmology|
| 1154398121|       Brownsville|   TX|  Gonzalez|      Juan|     Family Practice|
+-----------+------------------+-----+----------+----------+--------------------+
only showing top 20 rows
```

Creating graph to show number of doctors in each state

In [18]:

```
import seaborn as sns
import matplotlib.pyplot as plt

# Count the number of doctors in each state
state_counts = partD_catfpd.groupBy('state').count().orderBy('state')

# Create a bar chart of the state counts
sns.set(style="darkgrid")
plt.figure(figsize=(15,5))
ax = sns.barplot(x="state", y="count", data=state_counts.toPandas())
ax.set_xticklabels(ax.get_xticklabels(), rotation=90, ha="right")

plt.xlabel("State")
plt.ylabel("Number of Doctors")
plt.title("Bar chart of Number of Doctors in Each State")
plt.show()
```

```
/opt/conda/anaconda/lib/python3.7/site-packages/statsmodels/tools/_testing.py:19: Future
Warning: pandas.util.testing is deprecated. Use the functions in the public API at panda
s.testing instead.
  import pandas.util.testing as tm
<Figure size 1500x500 with 1 Axes>
```

In [19]:

```
partD_catfpd.count()
```

Loading [MathJax]/extensions/Safe.js

Out[19]: 893160

Using agg() function to take sum, mean and max of selected features

In [20]:
```
from pyspark.sql.functions import sum, mean, max

partD_agg=partD_Drug_df.groupBy('Prscrbr_NPI').agg(sum('Tot_Drug_Cst').alias('sum_tot_dr
                              mean('Tot_Drug_Cst').alias('avg_tot_drug_cst'),
                              max('Tot_Drug_Cst').alias('max_tot_drug_cst'),
                              sum('Tot_Clms').alias('sum_tot_clms'),
                              mean('Tot_Clms').alias('avg_tot_clms'),
                              max('Tot_Clms').alias('max_tot_clms'),
                              sum('Tot_Day_Suply').alias('sum_tot_day_suply'),
                              mean('Tot_Day_Suply').alias('avg_tot_day_suply'),
                              max('Tot_Day_Suply').alias('max_tot_day_suply'))
```

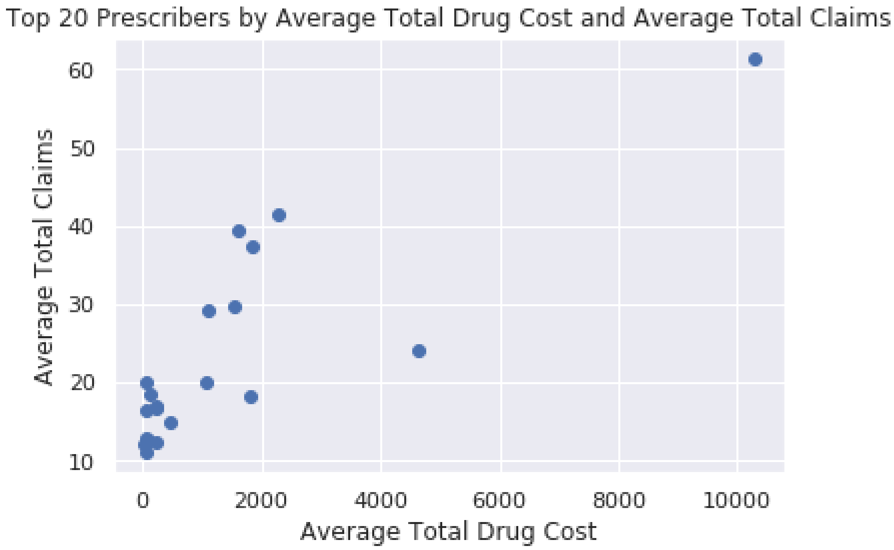
In [21]:
```
partD_agg.show()
```

```
+-----------+-----------------+------------------+-----------------+------------+-------
-----------+------------+-----------------+------------------+-----------------+
|Prscrbr_NPI|  sum_tot_drug_cst|  avg_tot_drug_cst|max_tot_drug_cst|sum_tot_clms|      a
vg_tot_clms|max_tot_clms|sum_tot_day_suply| avg_tot_day_suply|max_tot_day_suply|
+-----------+-----------------+------------------+-----------------+------------+-------
-----------+------------+-----------------+------------------+-----------------+
| 1033388616|        418520.54|        16740.8216|        105879.23|        1135|
45.4|         217|            44265|            1770.6|             9639|
| 1033392337|           703.09|          175.7725|           298.34|         119|
29.75|          62|              747|            186.75|              439|
| 1033394812|           701.25|           116.875|           204.95|         149|24.8333
33333333332|          56|              580| 96.66666666666667|              197|
| 1033406855|          95117.0| 4135.521739130435|         33530.99|        1657| 72.043
47826086956|         532|            49103| 2134.913043478261|            21235|
| 1033424932|2987.7000000000003| 597.5400000000001|          1361.39|         223|
44.6|         130|             6587|            1317.4|             4512|
| 1033430277|           178.55|            178.55|           178.55|          14|
14.0|          14|              480|             480.0|              480|
| 1033438783|229.17000000000002|             76.39|            109.4|          36|
12.0|          14|              147|              49.0|               70|
| 1033461900|            170.4|             170.4|            170.4|          12|
12.0|          12|              360|             360.0|              360|
| 1033493556|          6142.87|361.34529411764703|          4106.29|         311|18.2941
17647058822|          35|             6100| 358.8235294117647|              840|
| 1033498282|           307.07|102.35666666666667|           115.33|          70|23.3333
33333333332|          26|              740|246.66666666666666|              442|
| 1033512579|            99.19|             99.19|            99.19|          12|
12.0|          12|              360|             360.0|              360|
| 1033552682|2005.8000000000002|182.34545454545457|           492.35|         180|16.3636
36363636363|          31|             4263|387.54545454545456|              615|
| 1033555933|          8397.18| 645.9369230769231|          3291.93|         394|30.3076
92307692307|          83|            13630|1048.4615384615386|             3180|
| 1033581269|            98.25|             98.25|            98.25|          14|
14.0|          14|               62|              62.0|               62|
| 1033588405|          1795.98|256.56857142857143|            551.8|         120|17.1428
57142857142|          22|             2415|             345.0|              837|
| 1033664636|          8207.74|           8207.74|          8207.74|          18|
18.0|          18|              630|             630.0|              630|
| 1043200892|           283.79|           141.895|           186.36|          66|
33.0|          38|             2880|            1440.0|             1800|
| 1043202849|        106150.72|8165.4400000000005|         29977.22|         414|31.8461
53846153847|          62|            17294|1330.3076923076924|             2848|
| 1043205974|          7839.57|           522.638|          1468.18|         370|24.6666
66666666668|          73|            10282| 685.4666666666667|             2710|
| 1043215361|250588.47000000003| 5568.632666666667|         60430.96|        3244| 72.088
88888888889|         405|           235469| 5232.644444444444|            33963|
+-----------+-----------------+------------------+-----------------+------------+-------
-----------+------------+-----------------+------------------+-----------------+
only showing top 20 rows
```

Scatterplot depicting Average Drug cose VS. Average total claims

In [22]:

```python
# Extract the top 20 rows from the DataFrame
top_20_rows = partD_agg.limit(20).toPandas()


# Create a scatter plot to visualize the relationship between average total drug cost an
plt.scatter(x=top_20_rows['avg_tot_drug_cst'], y=top_20_rows['avg_tot_clms'])
plt.xlabel('Average Total Drug Cost')
plt.ylabel('Average Total Claims')
plt.title('Top 20 Prescribers by Average Total Drug Cost and Average Total Claims')
plt.show()
```




Joining dataset to Prscrbr_npi using left join

In [23]:
```
from pyspark.sql.functions import col

partD_allpd = partD_agg.join(partD_catfpd, on='Prscrbr_NPI', how='left')
```

In [24]:
```
partD_allpd.show()
```

Loading [MathJax]/extensions/Safe.js

```
+-----------+------------------+------------------+----------------+------------+-------
-----------+------------+-----------------+-------------------+-----------------+--------
----+-----+------------+---------+------------------+
|Prscrbr_NPI|  sum_tot_drug_cst|  avg_tot_drug_cst|max_tot_drug_cst|sum_tot_clms|      a
vg_tot_clms|max_tot_clms|sum_tot_day_suply| avg_tot_day_suply|max_tot_day_suply|
city|state|    last_name|first_name|         Speciality|
+-----------+------------------+------------------+----------------+------------+-------
-----------+------------+-----------------+-------------------+-----------------+--------
----+-----+------------+---------+------------------+
| 1003043209|          39676.76|1202.3260606060608|         9329.51|        1170| 35.454
54545454545|         109|            44482| 1347.939393939394|             5262|
Lynn|   MA|        Affel|  Marjorie|    Family Practice|
| 1003072810|214723.17999999996|2618.5753658536582|        21064.12|        3625|44.2073
17073170735|         197|           140816|1717.2682926829268|             7803|  Huntin
gton|   WV|        White|   Leonard|  Internal Medicine|
| 1003094665|            457.97|152.65666666666667|          249.49|          70|23.3333
33333333332|          46|              242| 80.66666666666667|               90|       T
ulsa|   OK|       Kebert|      Cory| Emergency Medicine|
| 1003113671|29403.269999999997|2100.2335714285714|        17090.19|         393|28.0714
28571428573|          65|            12569| 897.7857142857143|             2130|      To
ledo|   OH|        Kyser|   Stephen|Physician Assistant|
| 1003115718|            676.96|           135.392|          333.83|          65|
13.0|          15|             3249|             649.8|             1050|     Lansing|
MI|    Arsenault|    Angela| Nurse Practitioner|
| 1003126392|            501.47|167.15666666666667|          196.42|         104|34.6666
66666666664|          75|              402|             134.0|              238|      Ba
ngor|   PA|      Rinaldi|    Jerome|            Dentist|
| 1003146945|          87275.01| 4593.421578947368|        58106.43|         683| 35.947
36842105263|         104|            23605|1242.3684210526317|             4425|   North
port|   NY|        Moore|    Cheryl| Nurse Practitioner|
| 1003148867|         540061.47|      33753.841875|       269250.22|         603|
37.6875|         142|            20678|          1292.375|             4920|  Greenville
|   NC|Streeter-Moye|     Staci| Nurse Practitioner|
| 1003160532|         109847.27|         5492.3635|        38419.78|         935|
46.75|         176|            49506|            2475.3|            10490|      Dallas|
TX|     Gagliano|     Maria| Nurse Practitioner|
| 1003163924|           6802.99|          3401.495|         5490.38|         118|
59.0|          96|             6673|            3336.5|             6438|    San Jose|
CA|          Yao|      Simo|         Pharmacist|
| 1003164153|156818.74000000008|1522.5120388349521|        26960.17|        3802|  36.91
26213592233|         182|           151473| 1470.611650485437|             7300|     Aug
usta|   GA|       Mccall|    Rachel|Physician Assistant|
| 1003166273|            151.41|            75.705|           99.35|          55|
27.5|          40|              207|             103.5|              170|  Fort Myers|
FL|        Singh|   Reetane|            Dentist|
| 1003179805|1143457.6899999995| 45738.30759999998|        946140.4|         758|
30.32|         139|            30382|           1215.28|             6959|   Baltimore|
MD|         Fang| Christina| Nurse Practitioner|
| 1003216839| 89606.45999999999| 2890.530967741935|         30059.3|        1335|43.0645
16129032256|         160|            42568|1373.1612903225807|             4324|    Wate
rloo|   IA|        Komen|     Evans| Nurse Practitioner|
| 1003228149|          12467.27| 542.0552173913044|          5682.6|         511|22.2173
91304347824|          65|            21169| 920.3913043478261|             3720|      Ed
mond|   OK|      Merrell|  Michelle| Nurse Practitioner|
| 1003229147|3394.9500000000007|242.49642857142862|          701.09|         270|19.2857
14285714285|          34|            14070|            1005.0|             2130|   Galve
ston|   TX|       Pavela|     James|  Internal Medicine|
| 1003229402|19630.130000000005|1033.1647368421054|        11077.33|         526| 27.684
21052631579|          85|            21480|1130.5263157894738|             3596|     Chi
cago|   IL|   Elsharkawy|   Mohamed|    Family Practice|
| 1003245457|12830.899999999996|346.78108108108097|         3451.98|         644|17.4054
05405405407|          41|            26557| 717.7567567567568|             2464|Philadel
phia|   PA|        Tyson|   Vincent| Nurse Practitioner|
| 1003276197| 4266.679999999999|266.66749999999996|          734.14|         349|
```

```
21.8125|          43|                9524|            595.25|              1530|   Frankfort
|   MI|       Coffia|  Michaela| Nurse Practitioner|
| 1003287806|             38.91|              38.91|            38.91|           15|
15.0|          15|                  91|               91.0|               91|    New York|
NY|       Madden|     Kerri|Physician Assistant|
+-----------+----------------+----------------+---------------+-----------+-------
-----------+------------+----------------+-----------------+----------------+--------
----+-----+------------+---------+------------------+
only showing top 20 rows
```

In [25]:
```
partD_allpd.head()
```

Out[25]:
```
Row(Prscrbr_NPI='1003043209', sum_tot_drug_cst=39676.76, avg_tot_drug_cst=1202.326060606
0608, max_tot_drug_cst=9329.51, sum_tot_clms=1170, avg_tot_clms=35.45454545454545, max_t
ot_clms=109, sum_tot_day_suply=44482, avg_tot_day_suply=1347.939393939394, max_tot_day_s
uply=5262, city='Lynn', state='MA', last_name='Affel', first_name='Marjorie', Speciality
='Family Practice')
```

Dataset loading from Google data storage bucket. The entire datasets are incorporating three sections: General Payment, Research Payment and Physician Ownership Details. Key Features: The whole of general installment, Name of medication related the installments.

In [26]:
```
gcs_client = storage.Client()
bucket = gcs_client.bucket('shreneel-bigdata1')

list(bucket.list_blobs(prefix='OP_DTL_GNRL_PGYR2015_P01202023.csv'))
!hdfs dfs -ls 'gs://shreneel-bigdata1/OP_DTL_GNRL_PGYR2015_P01202023.csv'
payment_rawdata = spark \
  .read \
  .option ( "inferSchema" , "true" ) \
  .option ( "header" , "true" ) \
  .csv ( "gs://shreneel-bigdata1/OP_DTL_GNRL_PGYR2015_P01202023.csv" )

payment_rawdata.printSchema()
```

```
-rwx------   3 root root 6409783272 2023-05-04 01:50 gs://shreneel-bigdata1/OP_DTL_GNRL_
PGYR2015_P01202023.csv
root
 |-- Change_Type: string (nullable = true)
 |-- Covered_Recipient_Type: string (nullable = true)
 |-- Teaching_Hospital_CCN: integer (nullable = true)
 |-- Teaching_Hospital_ID: integer (nullable = true)
 |-- Teaching_Hospital_Name: string (nullable = true)
 |-- Physician_Profile_ID: integer (nullable = true)
 |-- Physician_NPI: integer (nullable = true)
 |-- Physician_First_Name: string (nullable = true)
 |-- Physician_Middle_Name: string (nullable = true)
 |-- Physician_Last_Name: string (nullable = true)
 |-- Physician_Name_Suffix: string (nullable = true)
 |-- Recipient_Primary_Business_Street_Address_Line1: string (nullable = true)
 |-- Recipient_Primary_Business_Street_Address_Line2: string (nullable = true)
 |-- Recipient_City: string (nullable = true)
 |-- Recipient_State: string (nullable = true)
 |-- Recipient_Zip_Code: string (nullable = true)
 |-- Recipient_Country: string (nullable = true)
 |-- Recipient_Province: string (nullable = true)
 |-- Recipient_Postal_Code: string (nullable = true)
 |-- Physician_Primary_Type: string (nullable = true)
 |-- Physician_Specialty: string (nullable = true)
 |-- Physician_License_State_code1: string (nullable = true)
 |-- Physician_License_State_code2: string (nullable = true)
 |-- Physician_License_State_code3: string (nullable = true)
 |-- Physician_License_State_code4: string (nullable = true)
 |-- Physician_License_State_code5: string (nullable = true)
 |-- Submitting_Applicable_Manufacturer_or_Applicable_GPO_Name: string (nullable = true)
 |-- Applicable_Manufacturer_or_Applicable_GPO_Making_Payment_ID: string (nullable = tru
e)
 |-- Applicable_Manufacturer_or_Applicable_GPO_Making_Payment_Name: string (nullable = t
rue)
 |-- Applicable_Manufacturer_or_Applicable_GPO_Making_Payment_State: string (nullable =
true)
 |-- Applicable_Manufacturer_or_Applicable_GPO_Making_Payment_Country: string (nullable
= true)
 |-- Total_Amount_of_Payment_USDollars: string (nullable = true)
 |-- Date_of_Payment: string (nullable = true)
 |-- Number_of_Payments_Included_in_Total_Amount: string (nullable = true)
 |-- Form_of_Payment_or_Transfer_of_Value: string (nullable = true)
 |-- Nature_of_Payment_or_Transfer_of_Value: string (nullable = true)
 |-- City_of_Travel: string (nullable = true)
 |-- State_of_Travel: string (nullable = true)
 |-- Country_of_Travel: string (nullable = true)
 |-- Physician_Ownership_Indicator: string (nullable = true)
 |-- Third_Party_Payment_Recipient_Indicator: string (nullable = true)
 |-- Name_of_Third_Party_Entity_Receiving_Payment_or_Transfer_of_Value: string (nullable
= true)
 |-- Charity_Indicator: string (nullable = true)
 |-- Third_Party_Equals_Covered_Recipient_Indicator: string (nullable = true)
 |-- Contextual_Information: string (nullable = true)
 |-- Delay_in_Publication_Indicator: string (nullable = true)
 |-- Record_ID: string (nullable = true)
 |-- Dispute_Status_for_Publication: string (nullable = true)
 |-- Product_Indicator: string (nullable = true)
 |-- Name_of_Associated_Covered_Drug_or_Biological1: string (nullable = true)
 |-- Name_of_Associated_Covered_Drug_or_Biological2: string (nullable = true)
 |-- Name_of_Associated_Covered_Drug_or_Biological3: string (nullable = true)
 |-- Name_of_Associated_Covered_Drug_or_Biological4: string (nullable = true)
 |-- Name_of_Associated_Covered_Drug_or_Biological5: string (nullable = true)
 |-- NDC_of_Associated_Covered_Drug_or_Biological1: string (nullable = true)
 |-- NDC_of_Associated_Covered_Drug_or_Biological2: string (nullable = true)
```

```
 |-- NDC_of_Associated_Covered_Drug_or_Biological3: string (nullable = true)
 |-- NDC_of_Associated_Covered_Drug_or_Biological4: string (nullable = true)
 |-- NDC_of_Associated_Covered_Drug_or_Biological5: string (nullable = true)
 |-- Name_of_Associated_Covered_Device_or_Medical_Supply1: string (nullable = true)
 |-- Name_of_Associated_Covered_Device_or_Medical_Supply2: string (nullable = true)
 |-- Name_of_Associated_Covered_Device_or_Medical_Supply3: string (nullable = true)
 |-- Name_of_Associated_Covered_Device_or_Medical_Supply4: string (nullable = true)
 |-- Name_of_Associated_Covered_Device_or_Medical_Supply5: string (nullable = true)
 |-- Program_Year: string (nullable = true)
 |-- Payment_Publication_Date: string (nullable = true)
```

Selecting essential features from dataset, most importantly Total payemnt column

In [27]:
```
from pyspark.sql.functions import col

payment_fpd = payment_rawdata.select(col('Physician_First_Name'),
                                 col('Physician_Last_Name'),
                                 col('Recipient_City'),
                                 col('Recipient_State'),
                                 col('Total_Amount_of_Payment_USDollars'))
```

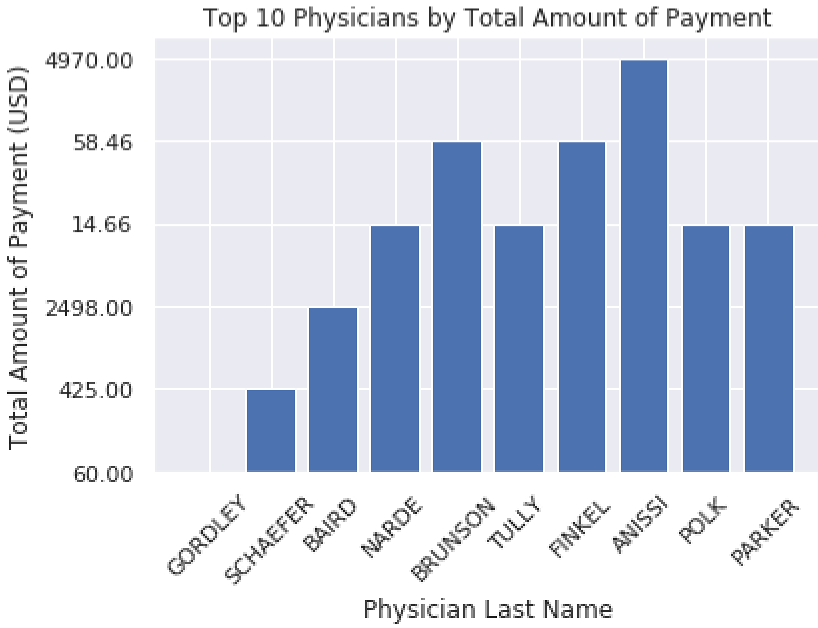
In [28]:
```
payment_fpd.head()
```

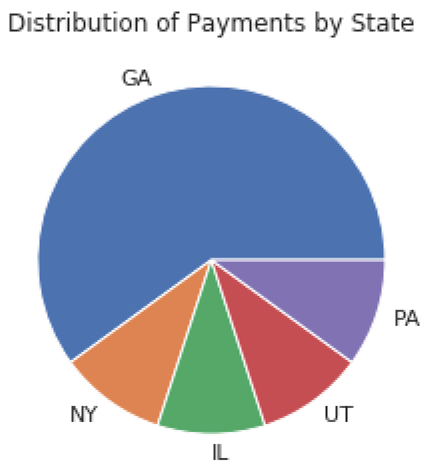
Out[28]:
```
Row(Physician_First_Name='DAVID', Physician_Last_Name='GORDLEY', Recipient_City='SLIPPER
Y ROCK', Recipient_State='PA', Total_Amount_of_Payment_USDollars='60.00')
```

In [ ]:

Bar graph showing total Payment recieved by doctors Pie graph is showing distribution of payments by state

In [29]:
```
import matplotlib.pyplot as plt

# Extract the top 10 rows from the DataFrame
top_10_rows = payment_fpd.limit(10).toPandas()

# Create a bar chart to visualize the total amount of payment for each physician
plt.bar(x=top_10_rows['Physician_Last_Name'], height=top_10_rows['Total_Amount_of_Paymen
plt.xlabel('Physician Last Name')
plt.ylabel('Total Amount of Payment (USD)')
plt.title('Top 10 Physicians by Total Amount of Payment')

# Rotate the x-axis labels by 45 degrees
plt.xticks(rotation=45)

plt.show()

# Create a pie chart to visualize the distribution of payments by state
state_counts = top_10_rows['Recipient_State'].value_counts()
plt.pie(x=state_counts.values, labels=state_counts.index)
plt.title('Distribution of Payments by State')
plt.show()
```




In [30]:
```
payment_fpd.count()
```

Out[30]:
```
11572091
```

Performing grouping and using agg() function to check the total payment sum in USD

In [31]:
```
from pyspark.sql.functions import sum

payment_fpd1 = payment_fpd.groupby(['Physician_First_Name', 'Physician_Last_Name', 'Reci
                         .agg(sum('Total_Amount_of_Payment_USDollars').alias('Total_Amou
```

In [32]:
```
payment_fpd1.show()
```

Loading [MathJax]/extensions/Safe.js

```
+-------------------+------------------+-------------+--------------+---------------
----------------------+
|Physician_First_Name|Physician_Last_Name|Recipient_City|Recipient_State|Total_Amount_of
_Payment_USDollars_sum|
+-------------------+------------------+-------------+--------------+---------------
----------------------+
|               HARRY|                TAMM|      PHOENIX|            AZ|
2793.359999999999|
|              DANIEL|                ROTH|   FORT WAYNE|            IN|
49565.61|
|              DONALD|             SLAPPEY|   BIRMINGHAM|            AL|
97.47000000000001|
|             SHAHEEN|             JAWAHAR|        TRACY|            CA|
654.0300000000001|
|         CHRISTOPHER|            VENDRYES|        MIAMI|            FL|
3468.51|
|                JOHN|      SATTERTHWAITE|   GREENVILLE|            SC|
114.31|
|                JOHN|                JOHN|      KINSTON|            NC|
1839.9800000000002|
|                JOHN|              DAURIA|    BRADENTON|            FL|
338.4|
|             JIENSUP|                 KIM|       COLTON|            CA|
1426.7999999999997|
|              THOMAS|            LOZOWSKI|   TOMS RIVER|            NJ|
2464.85|
|                 RAE|               DAVIS|    LANSDOWNE|            VA|
662.4999999999999|
|              UPLEKH|             PUREWAL|  PHILADELPHIA|            PA|
18067.07|
|              AUDREY|      LEWERENZ-WALSH|    BRADENTON|            FL|
512.06|
|             VINCENT|            THOMPSON|    LANGHORNE|            PA|
1002.0000000000001|
|              ROBERT|            WILLIAMS|          OPP|            AL|
288.4|
|               JAMES|              WRIGHT|  LAKE VILLAGE|            AR|
66.7|
|                 JOB|             MONGARE|       ATHENS|            TX|
781.51|
|              MIKHAL|              MONSON|    CLEVELAND|            OH|
12.38|
|             RICHARD|              HARRIS|       SPARKS|            NV|
30.32|
|              KURTIS|              HOLMES|       FRUITA|            CO|
467.21999999999997|
+-------------------+------------------+-------------+--------------+---------------
----------------------+
only showing top 20 rows
```

Renaming the column names to seamless use with other datasets

In [33]:
```python
from pyspark.sql.functions import col

rename_dict = {'Physician_First_Name':'first_name', 'Physician_Last_Name':'last_name', '

payment_fpd1 = payment_fpd1.select([col(c).alias(rename_dict.get(c, c)) for c in payment
```

In [34]:
```python
payment_fpd1.show()
```

```
+----------+----------+----------------+-----+------------------+
|first_name| last_name|            city|state| Total_Payment_Sum|
+----------+----------+----------------+-----+------------------+
|JACQUELINE| GUERRIERO|     WILKES BARRE|   PA|             17.41|
|   BRADLEY|  BENGTSON|    GRAND RAPIDS|   MI|118854.28000000003|
|    WALTER|     BORIS|    BROWNS MILLS|   NJ|375.45000000000005|
|     PETER|   PANTERA|      FORT MYERS|   FL| 550.0300000000001|
|      PAUL|     SCOTT|          TACOMA|   WA|            142.92|
|     Karin|     Quick|     Minneapolis|   MN|             11.29|
|   CHARLES|   FLOWERS|        COLUMBIA|   SC|1500.6100000000001|
|      MARK|    NADLER|        DANVILLE|   CA|             326.4|
|     HEATH| BROUSSARD|         JACKSON|   TN| 53771.90999999999|
|    STEVEN|      PARK|      CINCINNATI|   OH|            424.86|
|     KEITH|      BUHL|    PHILADELPHIA|   PA|             761.6|
|   ANTHONY|      ROSA|    PHILADELPHIA|   PA|             72.05|
|    Steven| Mardjetko|    Morton Grove|   IL|              82.9|
|    DONALD|      MANN|           HURON|   SD|             84.49|
|     DAVID|    HOOVER|           OMAHA|   NE|             100.0|
|    RONALD|    FISHER|         HANCOCK|   MI|              37.0|
|    JUSTIN|OBERDORFER|       MENOMINEE|   MI|              37.0|
|     ANTON|   PIANTEK|         SHAWANO|   WI|              37.0|
|   RICHARD|  LAGERMAN|       SHOREWOOD|   WI|             34.77|
|    Howard|    Gordon|Briarcliff Manor|   NY|             23.78|
+----------+----------+----------------+-----+------------------+
only showing top 20 rows
```

Sorting the dataset in descending order

In [35]:
```python
from pyspark.sql.functions import desc

payment_fpd2 = payment_fpd1.sort(desc('Total_Payment_Sum'))
```

In [36]:
```python
payment_fpd2.show()
```

```
+----------+---------+-----------------+-----+-------------------+
|first_name|last_name|             city|state|  Total_Payment_Sum|
+----------+---------+-----------------+-----+-------------------+
|      null|     null|           DUARTE|   CA|      3.0654182569E8|
|      null|     null|           BOSTON|   MA|       4.200212036E7|
|     ROGER|  JACKSON|NORTH KANSAS CITY|   MO|       3.450708545E7|
|      null|     null|           Boston|   MA|       2.076668303E7|
|   STEPHEN| BURKHART|      SAN ANTONIO|   TX|1.9421951320000004E7|
|      null|     null|        Rochester|   NY|        1.93059828E7|
|     KEVIN|    FOLEY|          Memphis|   TN| 1.782763144999999E7|
|      null|     null|        Cleveland|   OH|1.4486272639999999E7|
|      YVES|    GOBIN|         New York|   NY|1.2962521479999999E7|
|      null|     null|            PHILA|   PA|       1.157290884E7|
|    RODNEY|    RAABE|          Spokane|   WA|1.0414841879999999E7|
|      null|     null|      Little Rock|   AR|1.0274209679999998E7|
|      null|     null|     PHILADELPHIA|   PA|1.0260102440000001E7|
|      MARK|  HUMAYUN|      LOS ANGELES|   CA|    8314314.300000001|
|   GREGORY|    PEARL|           DALLAS|   TX|   7936496.0200000005|
|      null|     null|          Houston|   TX|            7129585.6|
|      null|     null|      LOS ANGELES|   CA|   6942171.4300000025|
|      null|     null|      Los Angeles|   CA|           6926090.95|
|      null|     null|           DENVER|   CO|    6719416.540000001|
|      null|     null|          HOUSTON|   TX|           6706036.03|
+----------+---------+-----------------+-----+-------------------+
only showing top 20 rows
```

Joining the dataset using left join

In [37]:
```
pay_partD_fpd = partD_allpd.join(payment_fpd2, ['last_name', 'first_name', 'city', 'stat
```

In [38]:
```
pay_partD_fpd.show()
```

```
+---------+----------+---------------+-----+----------+------------------+------------
------+---------------+-----------+-----------------+-----------+-----------------+-
----------------+---------------+--------------------+------------------+
|last_name|first_name|           city|state|Prscrbr_NPI|  sum_tot_drug_cst|  avg_tot_dr
ug_cst|max_tot_drug_cst|sum_tot_clms|      avg_tot_clms|max_tot_clms|sum_tot_day_suply|
avg_tot_day_suply|max_tot_day_suply|          Speciality| Total_Payment_Sum|
+---------+----------+---------------+-----+----------+------------------+------------
------+---------------+-----------+-----------------+-----------+-----------------+-
----------------+---------------+--------------------+------------------+
|   Abbott|     Laura|   San Francisco|   CA| 1093071367|          16670.63| 5556.876666
666667|         9978.51|          65|21.666666666666668|          35|             6923|2
307.6666666666665|             5003|Student in an Org...|              null|
| Abdullah|   Juveria|     Los Angeles|   CA| 1710229281|             71.55|
71.55|           71.55|          15|              15.0|          15|              450|
450.0|              450|   Internal Medicine|              null|
| Abraksia|     Samir|       Beachwood|   OH| 1275638199|1300524.4200000002|48167.571111
111116|       560482.86|        1018|  37.7037037037037|         144|            37423|
1386.037037037037|             8125| Hematology-Oncology|              null|
|   Abrams|    Rachel|      Santa Cruz|   CA| 1578542171|3147.7999999999997|393.47499999
999997|           712.0|         154|             19.25|          28|             6795|
849.375|             1590|     Family Practice|              null|
|    Abrol| Rajeshwar|         Tomball|   TX| 1447258108| 79439.06999999999| 3782.812857
142857|        19201.27|         685| 32.61904761904762|         133|            30594|
1456.857142857143|             7590|    Gastroenterology|              null|
|   Abuloc|   Timonet|         Garland|   TX| 1548427156| 80965.50999999998|1619.3101999
999997|         19299.3|        1165|              23.3|          80|            62750|
1255.0|             4276|  Nurse Practitioner|              null|
| Acharjee|   Subroto|  Merritt Island|   FL| 1447482070|              98.8|
98.8|            98.8|          12|              12.0|          12|              540|
540.0|              540|Interventional Ca...|              null|
|   Acosta| Christine|        Kingwood|   TX| 1194897231|           6313.72|           6
313.72|         6313.72|          11|              11.0|          11|              450|
450.0|              450|           Optometry|              null|
|  Adamich|    Thomas|Huntington Beach|   CA| 1063579670|            140.65|
140.65|          140.65|          20|              20.0|          20|              190|
190.0|              190|             Dentist|              null|
|    Adams|   Cynthia|         Lubbock|   TX| 1780742536|           3375.85|1125.2833333
333333|         2639.98|          38|12.666666666666666|          15|              901|
300.3333333333333|              330|  Nurse Practitioner|              null|
|    Adams|     David|      Charleston|   SC| 1467566968|         403402.49|57628.927142
857145|       378996.97|         141|20.142857142857142|          32|             3379|
482.7142857142857|              632|     General Surgery|              null|
|    Adams|    Gareth|   The Woodlands|   TX| 1750611158|            621.38|
310.69|          490.53|          45|              22.5|          31|              838|
419.0|              421|        Neurosurgery|              null|
|    Adams|      Lynn|Rockville Centre|   NY| 1396884656|130200.26999999999|14466.696666
666665|       123041.98|         157|17.444444444444443|          43|             3202|3
55.77777777777777|             1285|  Nurse Practitioner|              null|
|    Adams|    Robert|      Wilmington|   NC| 1275693517|121465.91999999995| 3470.454857
142856|         26280.0|         827| 23.62857142857143|          59|            28850|
824.2857142857143|             2370|          Psychiatry|             47.28|
|    Adams|   Stephen|     Chattanooga|   TN| 1598701674|          45233.53| 793.5707017
543859|         8461.12|        1592| 27.92982456140351|          95|            91522|1
605.6491228070176|             5256|     Family Practice|              null|
|    Adams|     Susan|      Florissant|   MO| 1871584474| 558393.5300000003| 3579.445705
128207|        59244.83|       10938| 70.11538461538461|         734|           713079|4
571.0192307692305|            59546|   Internal Medicine|             70.26|
|    Adcox|   Micheal|           Boise|   ID| 1851345193|347232.69999999995| 6313.321818
181817|       138676.64|        2135| 38.81818181818182|         136|           112492|2
045.3090909090909|             9219|          Nephrology|              null|
|Addington|     James|        Columbus|   OH| 1467741082| 834.6800000000001|417.34000000
000003|          484.18|          35|              17.5|          19|             1050|
525.0|              570|           Neurology|              null|
| Adedotun| Oluyemisi|          Eureka|   NV| 1629393632|1221.0000000000002|135.66666666
```

```
666669|          282.24|          155| 17.22222222222222|           31|             5342|
593.5555555555555|            930|  Nurse Practitioner|              null|
|  Adelola| Olubukola|           Logan|    OH| 1952614851|489953.32000000007| 3711.767575
757576|        38844.95|         6814|51.621212121212125|          396|           268571|
2034.628787878788|           16287|      Family Practice|42.400000000000006|
+---------+----------+-----------------+-----+-----------+------------------+------------
------+---------------+-------------+------------------+------------+-----------------+-
-----------------+-----------------+--------------------+-----------------+
only showing top 20 rows
```

Importing the dataset from google data storage. Information on dataset: This database contains a rundown of people and substances that are prohibited from taking an interest in governmentally financed social insurance programs (for example Medicare) because of past medicinal services extortion. We could treat the LEIE dataset as the semi-named information, on the grounds that LEIE is the fraudster-based objective however not a misrepresentation one.

In [39]:
```
gcs_client = storage.Client()
bucket = gcs_client.bucket('shreneel-bigdata1')

list(bucket.list_blobs(prefix='UPDATED.csv'))
!hdfs dfs -ls 'gs://shreneel-bigdata1/UPDATED.csv'
IELE_rawdata = spark \
  .read \
  .option ( "inferSchema" , "true" ) \
  .option ( "header" , "true" ) \
  .csv ( "gs://shreneel-bigdata1/UPDATED.csv" )

IELE_rawdata.printSchema()
```

```
-rwx------   3 root root   13969086 2023-05-04 01:25 gs://shreneel-bigdata1/UPDATED.csv
root
 |-- LASTNAME: string (nullable = true)
 |-- FIRSTNAME: string (nullable = true)
 |-- MIDNAME: string (nullable = true)
 |-- BUSNAME: string (nullable = true)
 |-- GENERAL: string (nullable = true)
 |-- SPECIALTY: string (nullable = true)
 |-- UPIN: string (nullable = true)
 |-- NPI: integer (nullable = true)
 |-- DOB: integer (nullable = true)
 |-- ADDRESS: string (nullable = true)
 |-- CITY: string (nullable = true)
 |-- STATE: string (nullable = true)
 |-- ZIP: string (nullable = true)
 |-- EXCLTYPE: string (nullable = true)
 |-- EXCLDATE: string (nullable = true)
 |-- REINDATE: integer (nullable = true)
 |-- WAIVERDATE: integer (nullable = true)
 |-- WVRSTATE: string (nullable = true)
```

Selecting the column which contains the npi of insurance abusers

In [40]:
```
npifraud_pd0 = IELE_rawdata.select('NPI', 'EXCLTYPE')
npifraud_pd0.show()
```

```
+----------+--------+
|       NPI|EXCLTYPE|
+----------+--------+
|         0|  1128a1|
|1972902351|  1128b8|
|         0|  1128a1|
|         0|  1128b7|
|1922348218|  1128a1|
|         0|  1128b5|
|         0|  1128a1|
|         0|  1128b8|
|         0|  1128a1|
|         0|  1128b8|
|         0|  1128b5|
|         0|  1128a1|
|         0|  1128b8|
|         0|  1128a1|
|         0|  1128a1|
|         0|  1128b4|
|         0|  1128a1|
|         0|  1128b8|
|         0|  1128a1|
|         0|  1128a1|
+----------+--------+
only showing top 20 rows
```

In [41]:
```python
from pyspark.sql.functions import col


npifraud_pd1 = npifraud_pd0.filter(col('NPI') != 0)
```

In [42]:
```python
npifraud_pd1.show()
```

```
+----------+---------+
|       NPI| EXCLTYPE|
+----------+---------+
|1972902351|   1128b8|
|1922348218|   1128a1|
|1942476080|   1128b8|
|1275600959|   1128a1|
|1891731758|   1128b8|
|1265830335|   1128a1|
|1851631543|   1128b7|
|1902198435|   1128a1|
|1073916631|   1128b7|
|1073682936|1128b7   |
|1902166028|   1128b8|
|1992906937|   1128b8|
|1104947944|   1128a1|
|1164669479|   1128a1|
|1043302250|   1128a1|
|1801231436|1128a1   |
|1912011800|   1128b8|
|1780812768|   1128b7|
|1447560867|   1128b8|
|1790963460|   1128b7|
+----------+---------+
only showing top 20 rows
```

In [43]:
```python
rename_dict = {'NPI':'Prscrbr_NPI', 'EXCLTYPE':'is_fraud'}

npi_fraud_pd = npifraud_pd1.select([col(c).alias(rename_dict.get(c, c)) for c in npifrau
```

In [44]:
```
#pip install graphframes
```

In [ ]:
```

```

In [45]:
```
npi_fraud_pd.show()
```

```
+-----------+---------+
|Prscrbr_NPI| is_fraud|
+-----------+---------+
| 1972902351|   1128b8|
| 1922348218|   1128a1|
| 1942476080|   1128b8|
| 1275600959|   1128a1|
| 1891731758|   1128b8|
| 1265830335|   1128a1|
| 1851631543|   1128b7|
| 1902198435|   1128a1|
| 1073916631|   1128b7|
| 1073682936|1128b7   |
| 1902166028|   1128b8|
| 1992906937|   1128b8|
| 1104947944|   1128a1|
| 1164669479|   1128a1|
| 1043302250|   1128a1|
| 1801231436|1128a1   |
| 1912011800|   1128b8|
| 1780812768|   1128b7|
| 1447560867|   1128b8|
| 1790963460|   1128b7|
+-----------+---------+
only showing top 20 rows
```

creating is_fraud and adding 1 to that column so in future can become binary

In [46]:
```
from pyspark.sql.functions import lit

npi_fraud_pd = npi_fraud_pd.withColumn('is_fraud', lit(1))
```

In [47]:
```
npi_fraud_pd.show()
```

```
+-----------+--------+
|Prscrbr_NPI|is_fraud|
+-----------+--------+
| 1972902351|       1|
| 1922348218|       1|
| 1942476080|       1|
| 1275600959|       1|
| 1891731758|       1|
| 1265830335|       1|
| 1851631543|       1|
| 1902198435|       1|
| 1073916631|       1|
| 1073682936|       1|
| 1902166028|       1|
| 1992906937|       1|
| 1104947944|       1|
| 1164669479|       1|
| 1043302250|       1|
| 1801231436|       1|
| 1912011800|       1|
| 1780812768|       1|
| 1447560867|       1|
| 1790963460|       1|
+-----------+--------+
only showing top 20 rows
```

In [48]:
```
print(npi_fraud_pd.dtypes)
```

```
[('Prscrbr_NPI', 'int'), ('is_fraud', 'int')]
```

Joining the is_fraud column to the rest of the selected dataset

In [49]:
```
Features_pd1 = pay_partD_fpd.join(npi_fraud_pd, ['Prscrbr_NPI'], how='left')
Features_pd1.show()
```

```
+-----------+--------+----------+--------------+-----+------------------+-----------
------+---------------+-----------+-----------------+-----------+----------------+-
-----------------+---------------+--------------------+------------------+--------+
|Prscrbr_NPI|last_name|first_name|          city|state|  sum_tot_drug_cst|  avg_tot_dr
ug_cst|max_tot_drug_cst|sum_tot_clms|      avg_tot_clms|max_tot_clms|sum_tot_day_suply|
avg_tot_day_suply|max_tot_day_suply|          Speciality| Total_Payment_Sum|is_fraud|
+-----------+--------+----------+--------------+-----+------------------+-----------
------+---------------+-----------+-----------------+-----------+----------------+-
-----------------+---------------+--------------------+------------------+--------+
| 1093071367|   Abbott|     Laura|   San Francisco|   CA|          16670.63| 5556.876666
666667|         9978.51|          65|21.666666666666668|          35|            6923|2
307.6666666666665|             5003|Student in an Org...|              null|    null|
| 1710229281| Abdullah|   Juveria|     Los Angeles|   CA|             71.55|
71.55|           71.55|          15|              15.0|          15|             450|
450.0|              450|   Internal Medicine|              null|    null|
| 1275638199| Abraksia|     Samir|       Beachwood|   OH|1300524.4200000002|48167.571111
111116|       560482.86|        1018|  37.7037037037037|         144|           37423|
1386.037037037037|             8125| Hematology-Oncology|              null|    null|
| 1578542171|   Abrams|    Rachel|      Santa Cruz|   CA|3147.7999999999997|393.47499999
999997|           712.0|         154|             19.25|          28|            6795|
849.375|             1590|     Family Practice|              null|    null|
| 1447258108|    Abrol| Rajeshwar|         Tomball|   TX| 79439.06999999999| 3782.812857
142857|        19201.27|         685| 32.61904761904762|         133|           30594|
1456.857142857143|             7590|    Gastroenterology|              null|    null|
| 1548427156|   Abuloc|   Timonet|         Garland|   TX| 80965.50999999998|1619.3101999
999997|         19299.3|        1165|              23.3|          80|           62750|
1255.0|             4276|  Nurse Practitioner|              null|    null|
| 1447482070| Acharjee|   Subroto|  Merritt Island|   FL|              98.8|
98.8|            98.8|          12|              12.0|          12|             540|
540.0|              540|Interventional Ca...|              null|    null|
| 1194897231|   Acosta| Christine|        Kingwood|   TX|           6313.72|           6
313.72|         6313.72|          11|              11.0|          11|             450|
450.0|              450|           Optometry|              null|    null|
| 1063579670|  Adamich|    Thomas|Huntington Beach|   CA|            140.65|
140.65|          140.65|          20|              20.0|          20|             190|
190.0|              190|             Dentist|              null|    null|
| 1780742536|    Adams|   Cynthia|         Lubbock|   TX|           3375.85|1125.2833333
333333|         2639.98|          38|12.666666666666666|          15|             901|
300.3333333333333|              330|  Nurse Practitioner|              null|    null|
| 1467566968|    Adams|     David|      Charleston|   SC|         403402.49|57628.927142
857145|       378996.97|         141|20.142857142857142|          32|            3379|
482.7142857142857|              632|     General Surgery|              null|    null|
| 1750611158|    Adams|    Gareth|   The Woodlands|   TX|            621.38|
310.69|          490.53|          45|              22.5|          31|             838|
419.0|              421|        Neurosurgery|              null|    null|
| 1396884656|    Adams|      Lynn|Rockville Centre|   NY|130200.26999999999|14466.696666
666665|       123041.98|         157|17.444444444444443|          43|            3202|3
55.77777777777777|             1285|  Nurse Practitioner|              null|    null|
| 1275693517|    Adams|    Robert|      Wilmington|   NC|121465.91999999995| 3470.454857
142856|         26280.0|         827| 23.62857142857143|          59|           28850|
824.2857142857143|             2370|          Psychiatry|             47.28|    null|
| 1598701674|    Adams|   Stephen|     Chattanooga|   TN|          45233.53| 793.5707017
543859|         8461.12|        1592| 27.92982456140351|          95|           91522|1
605.6491228070176|             5256|     Family Practice|              null|    null|
| 1871584474|    Adams|     Susan|      Florissant|   MO| 558393.5300000003| 3579.445705
128207|        59244.83|       10938| 70.11538461538461|         734|          713079|4
571.0192307692305|            59546|   Internal Medicine|             70.26|    null|
| 1851345193|    Adcox|   Micheal|           Boise|   ID|347232.69999999995| 6313.321818
181817|       138676.64|        2135| 38.81818181818182|         136|          112492|2
045.3090909090909|             9219|          Nephrology|              null|    null|
| 1467741082|Addington|     James|        Columbus|   OH| 834.6800000000001|417.34000000
000003|          484.18|          35|              17.5|          19|            1050|
525.0|              570|           Neurology|              null|    null|
| 1629393632| Adedotun| Oluyemisi|          Eureka|   NV|1221.0000000000002|135.66666666
```

```
666669|          282.24|           155| 17.22222222222222|           31|            5342|
593.5555555555555|              930|  Nurse Practitioner|             null|    null|
| 1952614851|  Adelola| Olubukola|          Logan|   OH|489953.32000000007| 3711.767575
757576|        38844.95|          6814|51.621212121212125|          396|          268571|
2034.628787878788|            16287|     Family Practice|42.400000000000006|    null|
+-----------+---------+----------+-----------------+-----+-----------------+------------
------+----------------+------------+-----------------+------------+-----------------+-
-----------------+-----------------+--------------------+-----------------+--------+
only showing top 20 rows
```

In [50]:

```python
Features_pd1.describe().show()
```

```
+-------+------------------+-----------+----------+------+------+------------------+---
---------------+------------------+------------------+------------------+---------------
--+------------------+------------------+------------------+----------------+-----------
------+--------+
|summary|       Prscrbr_NPI|  last_name|first_name|  city| state|  sum_tot_drug_cst|  a
vg_tot_drug_cst|  max_tot_drug_cst|      sum_tot_clms|      avg_tot_clms|     max_tot_cl
ms| sum_tot_day_suply| avg_tot_day_suply| max_tot_day_suply|      Speciality|Total_Payme
nt_Sum|is_fraud|
+-------+------------------+-----------+----------+------+------+------------------+---
---------------+------------------+------------------+------------------+---------------
--+------------------+------------------+------------------+----------------+-----------
------+--------+
|  count|            893173|     893173|    893165|893173|893173|            893173|
893173|            893173|            893173|            893173|           893173|
893173|            893173|            893173|          893173|          120052|    1140
|
|   mean|1.499824340393665E9|       null|      null| 612.0|  null|124095.37574388168|359
5.5598487573534|32727.610132605863|1442.6657903899916|33.172806156611834|142.64137518711
38| 60939.21357004746|1117.2608397149252|6184.5712163265125|            null|769.9191983
473828|     1.0|
| stddev| 2.87863595699132E8|       null|      null|   NaN|  null|342799.07853682246|181
54.489895529456|126480.07757068405|3317.0039345166106| 25.91225632276437|244.16946124279
87|134718.28139898408|1244.4010659555329|10917.456164573054|            null|9035.048215
575853|     0.0|
|    min|        1003000126|     &h's)u| &e'k:(A:i| 00612|    AA|               0.0|
0.0|               0.0|                11|              11.0|              11|
11|              11.0|                11|   Acupuncturist|            0.03|       1|
|    max|        1992999882|Zziwambazza|      Zyra|Zwolle|    ZZ|     1.904894932E7|
4958809.93|     1.014974956E7|            323252| 2266.222222222222|            24845|
4715546|         103890.0|            438319|Vascular Surgery|       1131692.3|
1|
+-------+------------------+-----------+----------+------+------+------------------+---
---------------+------------------+------------------+------------------+---------------
--+------------------+------------------+------------------+----------------+-----------
------+--------+
```

FIlling all na values with 0

In [51]:

```python
Features_pd1 = Features_pd1.fillna(0)
```

In [52]:

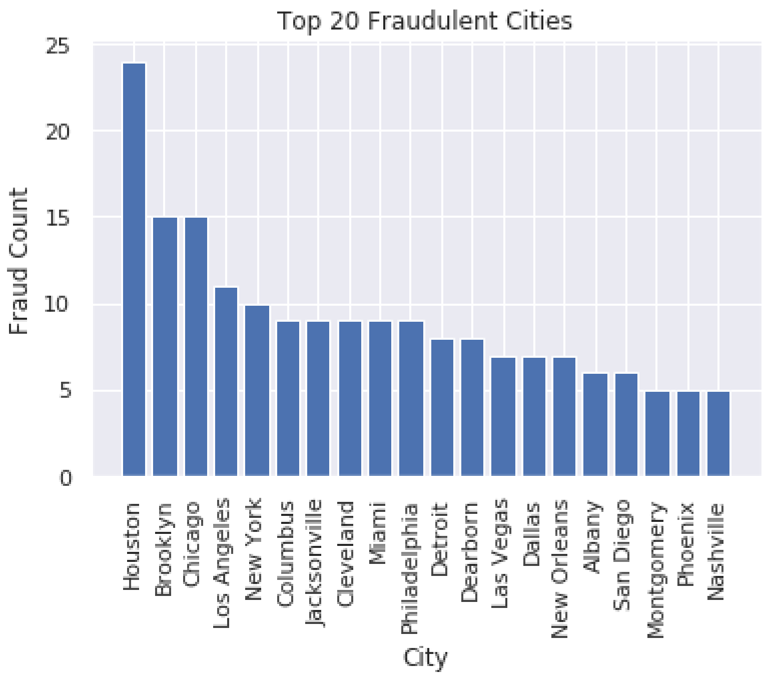
```python
Features_pd1
```

Out[52]:

```
DataFrame[Prscrbr_NPI: string, last_name: string, first_name: string, city: string, stat
e: string, sum_tot_drug_cst: double, avg_tot_drug_cst: double, max_tot_drug_cst: double,
sum_tot_clms: bigint, avg_tot_clms: double, max_tot_clms: int, sum_tot_day_suply: bigin
t, avg_tot_day_suply: double, max_tot_day_suply: int, Speciality: string, Total_Payment_
Sum: double, is_fraud: int]
```

Creating a bar graph to display the cities with the most fraud

In [53]:
```
import matplotlib.pyplot as plt
from pyspark.sql.functions import col

# Filter the fraud data where is_fraud = 1
fraud_cities = Features_pd1.filter(col("is_fraud") == 1).select("city")

# Count the number of fraud occurrences by city
fraud_counts = fraud_cities.groupBy("city").count().orderBy("count", ascending=False).li

# Convert the fraud counts to a Pandas dataframe for plotting
fraud_counts_pd = fraud_counts.toPandas()

# Create a bar plot
plt.bar(fraud_counts_pd["city"], fraud_counts_pd["count"])

# Add labels and title
plt.xlabel("City")
plt.ylabel("Fraud Count")
plt.title("Top 20 Fraudulent Cities")

# Rotate the x-axis labels for better visibility
plt.xticks(rotation=90)

# Show the plot
plt.show()
```




In [54]:
```
from pyspark.sql.functions import col

fraud_count = Features_pd1.filter(col('is_fraud') == 1).count()
```

In [55]:
```
fraud_count
```

Out[55]:
```
1140
```

In [56]:
```
FeaturesAll = Features_pd1
```

Scaling the features

In [57]:
```
from pyspark.sql.functions import log10, col
```

Loading [MathJax]/extensions/Safe.js

```
FeaturesAll = FeaturesAll.withColumn('sum_tot_drug_cst', log10(col('sum_tot_drug_cst') +
FeaturesAll = FeaturesAll.withColumn('sum_tot_clms', log10(col('sum_tot_clms') + 1.0))
FeaturesAll = FeaturesAll.withColumn('sum_tot_day_suply', log10(col('sum_tot_day_suply')
FeaturesAll = FeaturesAll.withColumn('Total_Payment_Sum', log10(col('Total_Payment_Sum')

FeaturesAll = FeaturesAll.withColumn('avg_tot_drug_cst', log10(col('avg_tot_drug_cst') +
FeaturesAll = FeaturesAll.withColumn('avg_tot_clms', log10(col('avg_tot_clms') + 1.0))
FeaturesAll = FeaturesAll.withColumn('avg_tot_day_suply', log10(col('avg_tot_day_suply')

FeaturesAll = FeaturesAll.withColumn('max_tot_drug_cst', log10(col('max_tot_drug_cst') +
FeaturesAll = FeaturesAll.withColumn('max_tot_clms', log10(col('max_tot_clms') + 1.0))
FeaturesAll = FeaturesAll.withColumn('max_tot_day_suply', log10(col('max_tot_day_suply')

FeaturesAll = FeaturesAll.withColumn('claim_max-mean', col('max_tot_clms') - col('avg_to
FeaturesAll = FeaturesAll.withColumn('supply_max-mean', col('max_tot_day_suply') - col('
FeaturesAll = FeaturesAll.withColumn('drug_max-mean', col('max_tot_drug_cst') - col('avg
```

In [58]:
```
FeaturesAll
```

Out[58]:
```
DataFrame[Prscrbr_NPI: string, last_name: string, first_name: string, city: string, stat
e: string, sum_tot_drug_cst: double, avg_tot_drug_cst: double, max_tot_drug_cst: double,
sum_tot_clms: double, avg_tot_clms: double, max_tot_clms: double, sum_tot_day_suply: dou
ble, avg_tot_day_suply: double, max_tot_day_suply: double, Speciality: string, Total_Pay
ment_Sum: double, is_fraud: int, claim_max-mean: double, supply_max-mean: double, drug_m
ax-mean: double]
```

In [59]:
```
from pyspark.sql.functions import col

FeaturesAll = FeaturesAll.withColumn("Prscrbr_NPI", col("Prscrbr_NPI").cast("string"))
```

Categorizing the features into categorical and numerical for easier analysis

In [60]:
```
from pyspark.sql.types import StringType

categorical_features = ['Prscrbr_NPI', 'last_name', 'Speciality', 'first_name', 'city',

for feature in categorical_features:
    FeaturesAll = FeaturesAll.withColumn(feature, FeaturesAll[feature].cast(StringType()
```

In [61]:
```
numerical_features = ['sum_tot_drug_cst', 'avg_tot_drug_cst','Total_Payment_Sum',
       'max_tot_drug_cst', 'sum_tot_clms',
       'avg_tot_clms', 'max_tot_clms',
       'sum_tot_day_suply', 'avg_tot_day_suply', 'max_tot_day_suply',
    'claim_max-mean','supply_max-mean', 'drug_max-mean']
```

assigning target name to is_fraud column

In [62]:
```
target = ['is_fraud']
```

In [63]:
```
allvars = categorical_features + numerical_features + target
```

In [64]:
```
y = FeaturesAll.select("is_fraud").rdd.flatMap(lambda x: x).collect()
X = FeaturesAll.select([col(c) for c in allvars if c != 'is_fraud'])
```

Using 100% of the data (Scaling out) to train and test into 80:20 ratio

In [65]:
```
from pyspark.ml.feature import VectorAssembler
from pyspark.sql.functions import col
from pyspark.sql.types import DoubleType
from pyspark.ml.tuning import TrainValidationSplit
```

```
# select the numerical columns from the original dataframe
numerical_features = ['sum_tot_drug_cst', 'avg_tot_drug_cst','Total_Payment_Sum',
       'max_tot_drug_cst', 'sum_tot_clms',
       'avg_tot_clms', 'max_tot_clms',
       'sum_tot_day_suply', 'avg_tot_day_suply', 'max_tot_day_suply',
    'claim_max-mean','supply_max-mean', 'drug_max-mean']
X = FeaturesAll.select(numerical_features)

# convert numerical columns to double type
for feature in numerical_features:
    X = X.withColumn(feature, col(feature).cast(DoubleType()))

# combine features into a single vector column
vectorAssembler = VectorAssembler(inputCols=X.columns, outputCol="features_vec")
X = vectorAssembler.transform(X)

# split the data into train and validation sets
train, test = X.randomSplit([0.8, 0.2], seed=0)

# select the correct columns for input and output
X_train = train.select(X.columns)
X_valid = test.select(X.columns)
y_train = train.select("Total_Payment_Sum")
y_valid = test.select("Total_Payment_Sum")

print(X_train.count(), len(X_train.columns))
print(X_valid.count(), len(X_valid.columns))
```

```
714253 14
178920 14
```

Handling the null values

In [66]:

```
from pyspark.sql.functions import col

# fill null values in numerical columns with 0
for feature in numerical_features:
    X_train = X_train.withColumn(feature, col(feature).cast("double"))
    X_valid = X_valid.withColumn(feature, col(feature).cast("double"))
    X_train = X_train.na.fill(0, [feature])
    X_valid = X_valid.na.fill(0, [feature])

# fill null values in categorical columns with 'NA'
for feature in categorical_features:
    if feature in X_train.columns:
        X_train = X_train.na.fill('NA', [feature])
    if feature in X_valid.columns:
        X_valid = X_valid.na.fill('NA', [feature])
```

In [67]:

```
from pyspark.sql.functions import col

X_train.select([col(col_name).cast("double").alias(col_name) for col_name in numerical_f
```

Out[67]:
```
[('sum_tot_drug_cst', 'double'),
 ('avg_tot_drug_cst', 'double'),
 ('Total_Payment_Sum', 'double'),
 ('max_tot_drug_cst', 'double'),
 ('sum_tot_clms', 'double'),
 ('avg_tot_clms', 'double'),
 ('max_tot_clms', 'double'),
 ('sum_tot_day_suply', 'double'),
 ('avg_tot_day_suply', 'double'),
 ('max_tot_day_suply', 'double'),
 ('claim_max-mean', 'double'),
 ('supply_max-mean', 'double'),
 ('drug_max-mean', 'double')]
```

In [68]:
```
from pyspark.sql.functions import rand

df_len = FeaturesAll.count()
train_len = int(df_len * 0.8)

df_train = FeaturesAll.orderBy(rand()).limit(train_len)
df_valid = FeaturesAll.orderBy(rand()).exceptAll(df_train)

print(df_train.count())
print(df_valid.count())
```

```
714538
178635
```

In [69]:
```
df_train.printSchema()
```

```
root
 |-- Prscrbr_NPI: string (nullable = true)
 |-- last_name: string (nullable = true)
 |-- first_name: string (nullable = true)
 |-- city: string (nullable = true)
 |-- state: string (nullable = true)
 |-- sum_tot_drug_cst: double (nullable = true)
 |-- avg_tot_drug_cst: double (nullable = true)
 |-- max_tot_drug_cst: double (nullable = true)
 |-- sum_tot_clms: double (nullable = true)
 |-- avg_tot_clms: double (nullable = true)
 |-- max_tot_clms: double (nullable = true)
 |-- sum_tot_day_suply: double (nullable = true)
 |-- avg_tot_day_suply: double (nullable = true)
 |-- max_tot_day_suply: double (nullable = true)
 |-- Speciality: string (nullable = true)
 |-- Total_Payment_Sum: double (nullable = true)
 |-- is_fraud: integer (nullable = true)
 |-- claim_max-mean: double (nullable = true)
 |-- supply_max-mean: double (nullable = true)
 |-- drug_max-mean: double (nullable = true)
```

In [70]:
```
from pyspark.sql.functions import col

partD_drug_train = partD_Drug_df.join(df_train.select('Prscrbr_NPI', 'is_fraud'), on=['P
partD_drug_all = partD_Drug_df.join(FeaturesAll.select('Prscrbr_NPI', 'is_fraud'), on=['
```

Displaying tottal fraud in entire dataset

In [71]:
```
print(partD_drug_train.filter(col("is_fraud") == 1).count())
```

```
46754
```

In [ ]:

In [72]:

```
# Total records in train set
print("Total records in train set : ")
print(partD_drug_train.count())

# Total Fraud in train set
print("Total Fraud in train set : ")
print(partD_drug_train.filter("is_fraud == 1").count())

# Show DataFrame
partD_drug_train.show()
```

Loading [MathJax]/extensions/Safe.js

```
Total records in train set :
19949515
Total Fraud in train set :
46706
+----------+------------------+-----------+--------+------------+---------------+--------+
|Prscrbr_NPI|         Brnd_Name|Tot_Drug_Cst|Tot_Clms|Tot_Day_Suply|   Prscrbr_Type|is_fraud|
+----------+------------------+-----------+--------+------------+---------------+--------+
| 1003017906|Acetaminophen-Cod...|     415.07|      12|         280|Family Practice|       0|
| 1003017906|         Acyclovir|     795.45|      39|        1159|Family Practice|       0|
| 1003017906|  Alendronate Sodium|     175.21|      15|         868|Family Practice|       0|
| 1003017906|       Allopurinol|     683.83|      51|        2607|Family Practice|       0|
| 1003017906|        Alprazolam|     305.93|      30|         790|Family Practice|       0|
| 1003017906|     Amiodarone Hcl|      45.69|      12|         360|Family Practice|       0|
| 1003017906|   Amitriptyline Hcl|      195.5|      20|         600|Family Practice|       0|
| 1003017906| Amlodipine Besylate|    1151.16|     162|       10493|Family Practice|       0|
| 1003017906|       Amoxicillin|      60.29|      13|          91|Family Practice|       0|
| 1003017906|Amoxicillin-Clavu...|     202.48|      15|         158|Family Practice|       0|
| 1003017906|Aspirin-Dipyridam...|    8427.78|      13|         750|Family Practice|       0|
| 1003017906|          Atenolol|     450.93|      48|        3480|Family Practice|       0|
| 1003017906|Atorvastatin Calcium|    1840.58|      96|        5621|Family Practice|       0|
| 1003017906|      Azithromycin|     356.93|      54|         270|Family Practice|       0|
| 1003017906|          Baclofen|     124.87|      12|         345|Family Practice|       0|
| 1003017906|     Benazepril Hcl|     157.82|      16|        1440|Family Practice|       0|
| 1003017906|    Bupropion Hcl Sr|     251.92|      15|         690|Family Practice|       0|
| 1003017906|      Bupropion Xl|    1323.25|      48|        2220|Family Practice|       0|
| 1003017906|        Carvedilol|     254.22|      34|        1995|Family Practice|       0|
| 1003017906|     Chlorthalidone|     1978.0|      48|        2820|Family Practice|       0|
+----------+------------------+-----------+--------+------------+---------------+--------+
only showing top 20 rows
```

Using the next line to scale down the size of database and use it by filter the 'fraction' parameter. ANalysis done on 100%, 50%, 20% and 5%

In [73]:
```
partD_drug_train_20= partD_drug_train.sample(fraction=0.05, seed=42)
```

Using vector assembler and splitting the data train and test data

In [74]:
```
feature_cols = ['Tot_Drug_Cst', 'Tot_Clms', 'Tot_Day_Suply']

# Create a vector assembler to assemble the features into a vector
assembler = VectorAssembler(inputCols=feature_cols, outputCol="features")

# Apply the vector assembler to the training data
train_data = assembler.transform(partD_drug_train_20).select("features", "is_fraud")

# Split the data into training and test sets
train_set, test_set = train_data.randomSplit([0.7, 0.3], seed=12345)
```

In [75]:
```
num_train_data = train_set.count()
print("Number of data in train_set:", num_train_data)
```

```
Number of data in train_set: 697534
```

In [76]:
```
################################################
```

In [77]:
```
 import time
```

Running Logistic Regression

In [80]:
```
from pyspark.ml.classification import LogisticRegression
from pyspark.ml.evaluation import MulticlassClassificationEvaluator,BinaryClassification

# Create a logistic regression model with default parameters
lr = LogisticRegression(featuresCol='features', labelCol='is_fraud')

# Train the model using the training set
start = time.time()
lr_model = lr.fit(train_set)
end = time.time()
print(f"Time to train logistic regression model: {end - start:.4f} seconds")

# Make predictions on the test set
start = time.time()
predictions = lr_model.transform(test_set)
end = time.time()
print(f"Time to make predictions on test set: {end - start:.4f} seconds")

# Evaluate the model using binary classification metrics
binary_evaluator = BinaryClassificationEvaluator(labelCol='is_fraud')
accuracy = binary_evaluator.evaluate(predictions)

# Evaluate the model using F1 score
multi_evaluator = MulticlassClassificationEvaluator(labelCol='is_fraud', metricName='f1'
f1_score = multi_evaluator.evaluate(predictions)

print(f"Accuracy: {accuracy:.4f}")
print(f"F1 Score: {f1_score:.4f}")
```

```
Time to train logistic regression model: 220.3205 seconds
Time to make predictions on test set: 0.0326 seconds
Accuracy: 0.5937
F1 Score: 0.9965
```

Running Naive Bayes on data

In [81]:
```
from pyspark.ml.classification import NaiveBayes
from pyspark.ml.evaluation import BinaryClassificationEvaluator, MulticlassClassificatio

# Create a Naive Bayes model with default parameters
```

```python
nb = NaiveBayes(featuresCol='features', labelCol='is_fraud')

# Train the model using the training set
start = time.time()
nb_model = nb.fit(train_set)
end = time.time()
print(f"Time to train Naive Bayes model: {end - start:.4f} seconds")

# Make predictions on the test set
start = time.time()
predictions = nb_model.transform(test_set)
end = time.time()
print(f"Time to make predictions on test set: {end - start:.4f} seconds")

# Evaluate the model using binary classification metrics
binary_evaluator = BinaryClassificationEvaluator(labelCol='is_fraud')
accuracy = binary_evaluator.evaluate(predictions)

# Evaluate the model using F1 score
multi_evaluator = MulticlassClassificationEvaluator(labelCol='is_fraud', metricName='f1'
f1_score = multi_evaluator.evaluate(predictions)

print(f"Accuracy: {accuracy:.4f}")
print(f"F1 Score: {f1_score:.4f}")
```

```
Time to train Naive Bayes model: 161.3878 seconds
Time to make predictions on test set: 0.0361 seconds
Accuracy: 0.4770
F1 Score: 0.8752
```

Running Gradient Boosting Trees Classifier

In [82]:

```python
from pyspark.ml.classification import GBTClassifier
from pyspark.ml.evaluation import BinaryClassificationEvaluator, MulticlassClassificatio

# Create a GBTClassifier model with default parameters
gbt = GBTClassifier(featuresCol='features', labelCol='is_fraud')

# Train the model using the training set
start = time.time()
gbt_model = gbt.fit(train_set)
end = time.time()
print(f"Time to train GBTClassifier model: {end - start:.4f} seconds")

# Make predictions on the test set
start = time.time()
predictions = gbt_model.transform(test_set)
end = time.time()
print(f"Time to make predictions on test set: {end - start:.4f} seconds")

# Evaluate the model using binary classification metrics
binary_evaluator = BinaryClassificationEvaluator(labelCol='is_fraud')
accuracy = binary_evaluator.evaluate(predictions)

# Evaluate the model using F1 score
multi_evaluator = MulticlassClassificationEvaluator(labelCol='is_fraud', metricName='f1'
f1_score = multi_evaluator.evaluate(predictions)

print(f"Accuracy: {accuracy:.4f}")
print(f"F1 Score: {f1_score:.4f}")
```

```
Time to train GBTClassifier model: 285.3160 seconds
Time to make predictions on test set: 0.0272 seconds
Accuracy: 0.6036
F1 Score: 0.9964
```

Using Random Forest Classifier

In [ ]:

```
from pyspark.ml.classification import RandomForestClassifier
from pyspark.ml.evaluation import BinaryClassificationEvaluator, MulticlassClassificatio

# Create a RandomForestClassifier model with default parameters
rf = RandomForestClassifier(featuresCol='features', labelCol='is_fraud')

# Train the model using the training set
start = time.time()
rf_model = rf.fit(train_set)
end = time.time()
print(f"Time to train RandomForestClassifier model: {end - start:.4f} seconds")

# Make predictions on the test set
start = time.time()
predictions = rf_model.transform(test_set)
end = time.time()
print(f"Time to make predictions on test set: {end - start:.4f} seconds")

# Evaluate the model using binary classification metrics
binary_evaluator = BinaryClassificationEvaluator(labelCol='is_fraud')
accuracy = binary_evaluator.evaluate(predictions)

# Evaluate the model using F1 score
multi_evaluator = MulticlassClassificationEvaluator(labelCol='is_fraud', metricName='f1'
f1_score = multi_evaluator.evaluate(predictions)

print(f"Accuracy: {accuracy:.4f}")
print(f"F1 Score: {f1_score:.4f}")
```

```
Time to train RandomForestClassifier model: 324.4908 seconds
Time to make predictions on test set: 0.0276 seconds
Accuracy: 0.5000
F1 Score: 0.9966
```

Running Decision Tree Classifier

In [83]:

```
from pyspark.ml.classification import DecisionTreeClassifier


# Split the data into training and test sets
start = time.time()
train_set, test_set = train_data.randomSplit([0.7, 0.3], seed=12345)
end = time.time()
print(f"Time to split data into training and test sets: {end - start:.4f} seconds")

# Create a DecisionTreeClassifier model
dt = DecisionTreeClassifier(
    featuresCol='features',
    labelCol='is_fraud',
    maxDepth=5,
    maxBins=32,
    minInstancesPerNode=1,
    impurity='gini'
)

# Train the model using the training set
        time()
```

Loading [MathJax]/extensions/Safe.js

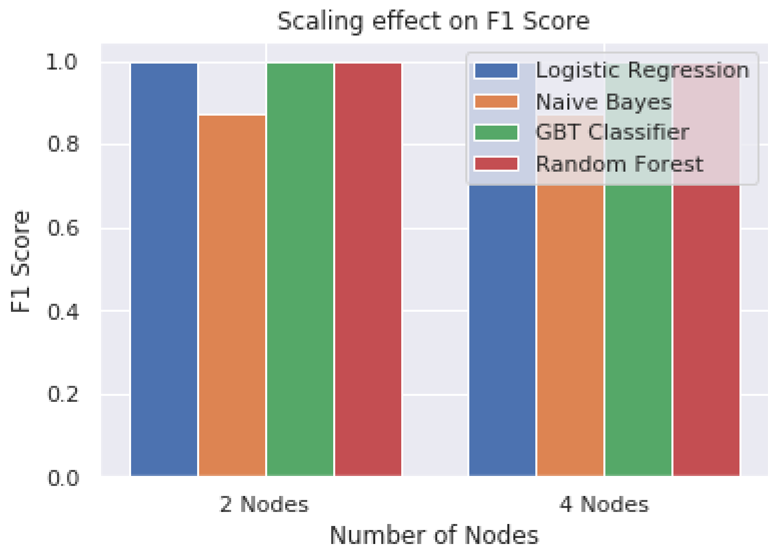
```
dt_model = dt.fit(train_set)
end = time.time()
print(f"Time to train DecisionTreeClassifier model: {end - start:.4f} seconds")

# Make predictions on the test set
start = time.time()
predictions = dt_model.transform(test_set)
end = time.time()
print(f"Time to make predictions on test set: {end - start:.4f} seconds")

# Evaluate the model using binary classification metrics
evaluator = BinaryClassificationEvaluator(labelCol='is_fraud')
accuracy = evaluator.evaluate(predictions)

print(f"Accuracy: {accuracy:.4f}")
```

```
Time to split data into training and test sets: 0.0152 seconds
Time to train DecisionTreeClassifier model: 174.7545 seconds
Time to make predictions on test set: 0.0292 seconds
Accuracy: 0.5000
```

Creating a graph to display the effect of F1 score on Scaling

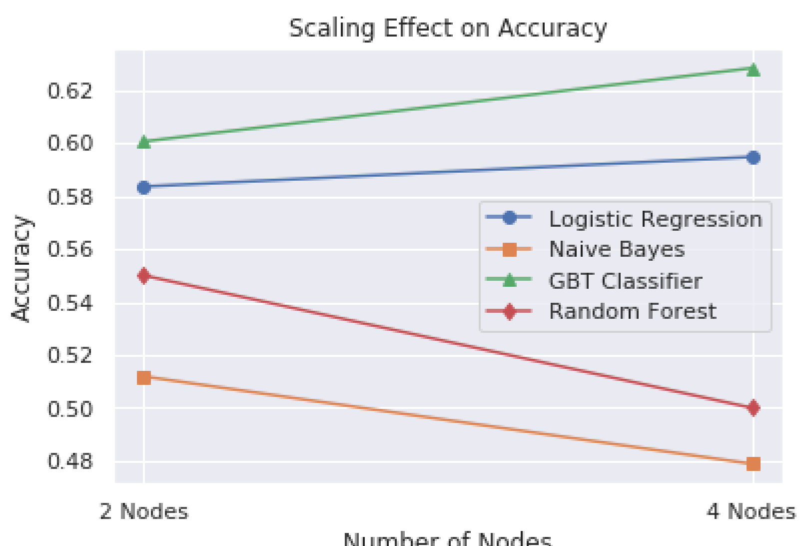
In [84]:

```
import matplotlib.pyplot as plt
import numpy as np

# Define the x-axis points
x = ['2 Nodes', '4 Nodes']

# Define the y-axis values for each bar
logistic_regression = [0.9965, 0.9964]
naive_bayes = [0.8734, 0.8725]
gbt_classifier = [0.9965, 0.9965]
random_forest = [0.9966, 0.9965]

# Set the width of each bar
bar_width = 0.2

# Create an array to position the bars on the x-axis
bar_positions = np.arange(len(x))

# Create the bar plots for each line
plt.bar(bar_positions - 1.5*bar_width, logistic_regression, width=bar_width, label='Logi
plt.bar(bar_positions - 0.5*bar_width, naive_bayes, width=bar_width, label='Naive Bayes'
plt.bar(bar_positions + 0.5*bar_width, gbt_classifier, width=bar_width, label='GBT Class
plt.bar(bar_positions + 1.5*bar_width, random_forest, width=bar_width, label='Random For

# Add labels and title to the graph
plt.xlabel('Number of Nodes')
plt.ylabel('F1 Score')
plt.title('Scaling effect on F1 Score')

# Add tick labels to the x-axis
plt.xticks(bar_positions, x)

# Add a legend to the graph
plt.legend()

# Display the graph
plt.show()
```



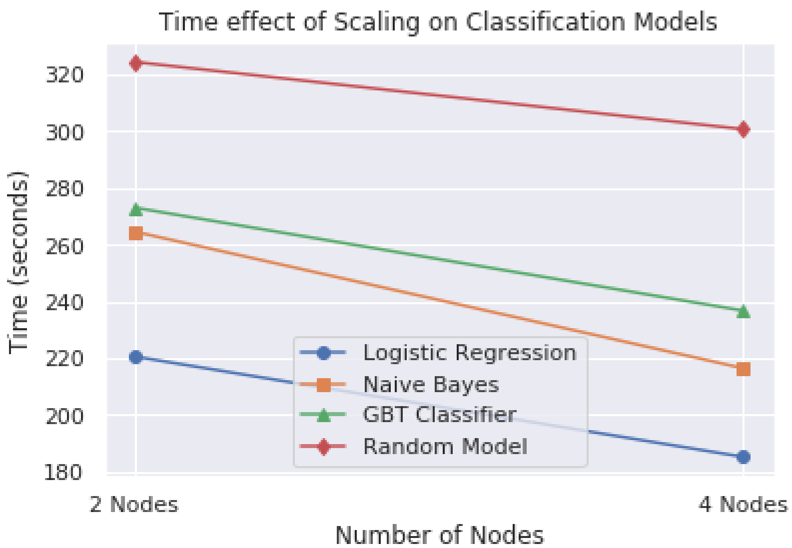

Creating a graph to display the effect of Accuracy on Scaling

In [85]:
```
# Define the x-axis points
x = ['2 Nodes', '4 Nodes']

# Define the y-axis values for each line
logistic_regression = [0.5835, 0.5947]
naive_bayes = [0.5119, 0.4789]
gbt_classifier = [0.6005, 0.6282]
random_forest = [0.5500, 0.5000]

# Plot the lines on the graph with markers for each data point
plt.plot(x, logistic_regression, marker='o', label='Logistic Regression')
plt.plot(x, naive_bayes, marker='s', label='Naive Bayes')
plt.plot(x, gbt_classifier, marker='^', label='GBT Classifier')
plt.plot(x, random_forest, marker='d', label='Random Forest')

# Add labels and title to the graph
plt.xlabel('Number of Nodes')
plt.ylabel('Accuracy')
plt.title('Scaling Effect on Accuracy')

# Add a legend to the graph
plt.legend()

# Display the graph
plt.show()
```



Creating a graph to display the effect of Time on Scaling and using various algorithms

In [86]:
```
# Define the x-axis points
x = ['2 Nodes', '4 Nodes']

# Define the y-axis values for each line
logistic_regression = [220.5698, 185.1697]
naive_bayes = [264.6157, 216.4411]
gbt_classifier = [273.0909, 236.7622]
random_model = [324.4908, 300.7622]

# Plot the lines on the graph with markers for each data point
plt.plot(x, logistic_regression, marker='o', label='Logistic Regression')
plt.plot(x, naive_bayes, marker='s', label='Naive Bayes')
plt.plot(x, gbt_classifier, marker='^', label='GBT Classifier')
plt.plot(x, random_model, marker='d', label='Random Model')

# Add labels and title to the graph
plt.xlabel('Number of Nodes')
plt.ylabel('Time (seconds)')
plt.title('Time effect of Scaling on Classification Models')

# Add a legend to the graph
plt.legend()

# Display the graph
plt.show()
```




In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Loading [MathJax]/extensions/Safe.js

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: