US campaign Finance Analytics

By
Harshitha Yentrapragada
Shivani Badinehal
Udveg Jukanti

Problem:

For analysis of expenditure for contesting in an election, most of the new candidates cannot analyze the raw data provided by the FEC(Federal Election Commission).

Solution:

By analyzing the data files of previous years, and generating curated data tables which will indeed create reports that describe overall understandings of expenditures

Data Source

Retrieved from :

US Campaign Finance Data – Federal Election Commission(FEC)

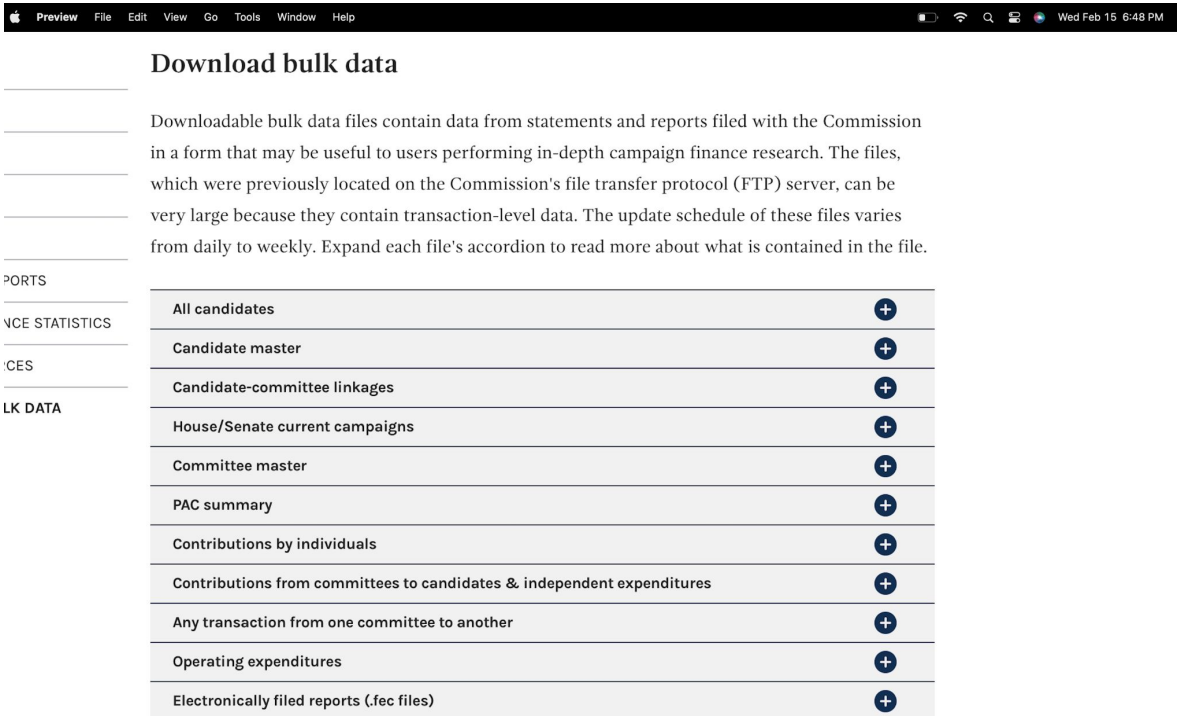
URL:

https://www.fec.gov/data/browse-data/?tab=bulk-data

Description:

- The size of the dataset is approximately 15 gb and has almost 28 million people information.
- This dataset includes info of candidates, committees, Political action committees, House and Senate campaigns, as well as transactions, individual and committee contributions, operations and independent expenditures for US elections.

Files Used


Preview File Edit View Go Tools Window Help
Wed Feb 15 6:48 PM
Download bulk data
Downloadable bulk data files contain data from statements and reports filed with the Commission in a form that may be useful to users performing in-depth campaign finance research. The files, which were previously located on the Commission's file transfer protocol (FTP) server, can be very large because they contain transaction-level data. The update schedule of these files varies from daily to weekly. Expand each file's accordion to read more about what is contained in the file.
PORTS
NCE STATISTICS
CES
LK DATA
All candidates
Candidate master
Candidate-committee linkages
House/Senate current campaigns
Committee master
PAC summary
Contributions by individuals
Contributions from committees to candidates & independent expenditures
Any transaction from one committee to another
Operating expenditures
Electronically filed reports (.fec files)

Fec Data Column Names and Abbreviations

https://www.fec.gov/campaign-finance-data/committee-summary-file-description/

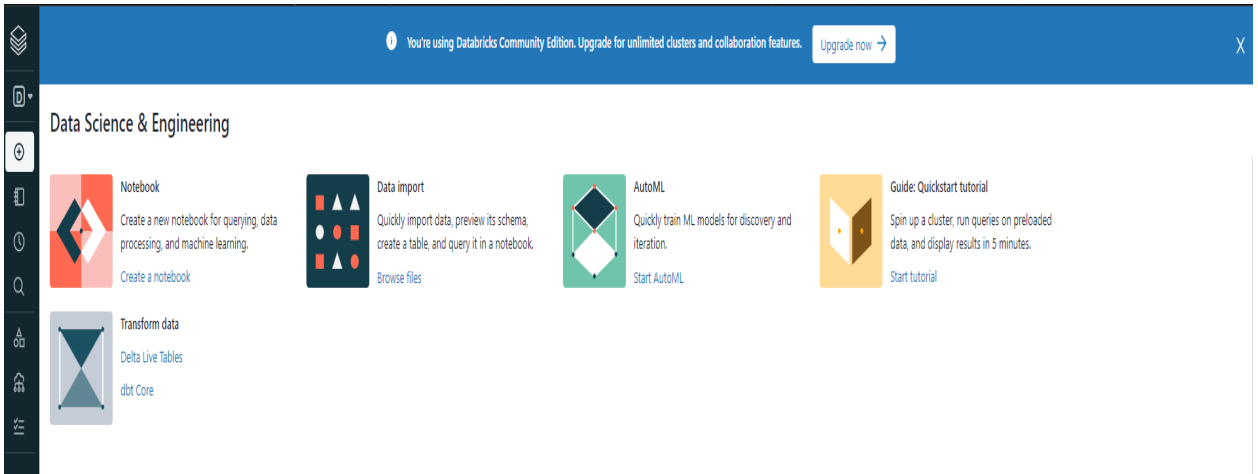
```
'CAND_ID': 'Candidate identification',
'CAND_NAME': 'Candidate name',
'CAND_ICI': 'Incumbent challenger status',
'PTY_CD': 'Party code',
'CAND_PTY_AFFILIATION': 'Party affiliation',
'TTL_RECEIPTS': 'Total receipts',
'TRANS_FROM_AUTH': 'Transfers from authorized committees',
'TTL_DISB': 'Total disbursements',
'TRANS_TO_AUTH': 'Transfers to authorized committees',
'COH_BOP': 'Beginning cash',
'COH_COP': 'Ending cash',
'CAND_CONTRIB': 'Contributions from candidate',
'CAND_LOANS': 'Loans from candidate',
'OTHER_LOANS': 'Other loans',
'CAND_LOAN_REPAY': 'Candidate loan repayments',
'OTHER_LOAN_REPAY': 'Other loan repayments',
'DEBTS_OWED_BY': 'Debts owed by',
'TTL_INDIV_CONTRIB': 'Total individual contributions',
'CAND_OFFICE_ST': 'Candidate state',
'CAND_OFFICE_DISTRICT': 'Candidate district',
'SPEC_ELECTION': 'Special election status',
'PRIM_ELECTION': 'Primary election status',
'RUN_ELECTION': 'Runoff election status',
'GEN_ELECTION': 'General election status',
'GEN_ELECTION_PRECENT': 'General election percentage',
'OTHER_POL_CMTE_CONTRIB': 'Contributions from other political committees',
'POL_PTY_CONTRIB': 'Contributions from party committees',
'CVG_END_DT': 'Coverage end date',
'INDIV_REFUNDS': 'Refunds to individuals',
'CMTE_REFUNDS': 'Refunds to committees'}
```

Software Tools and Technologies:

- Databricks
- Spark
- Tableau

Data Bricks

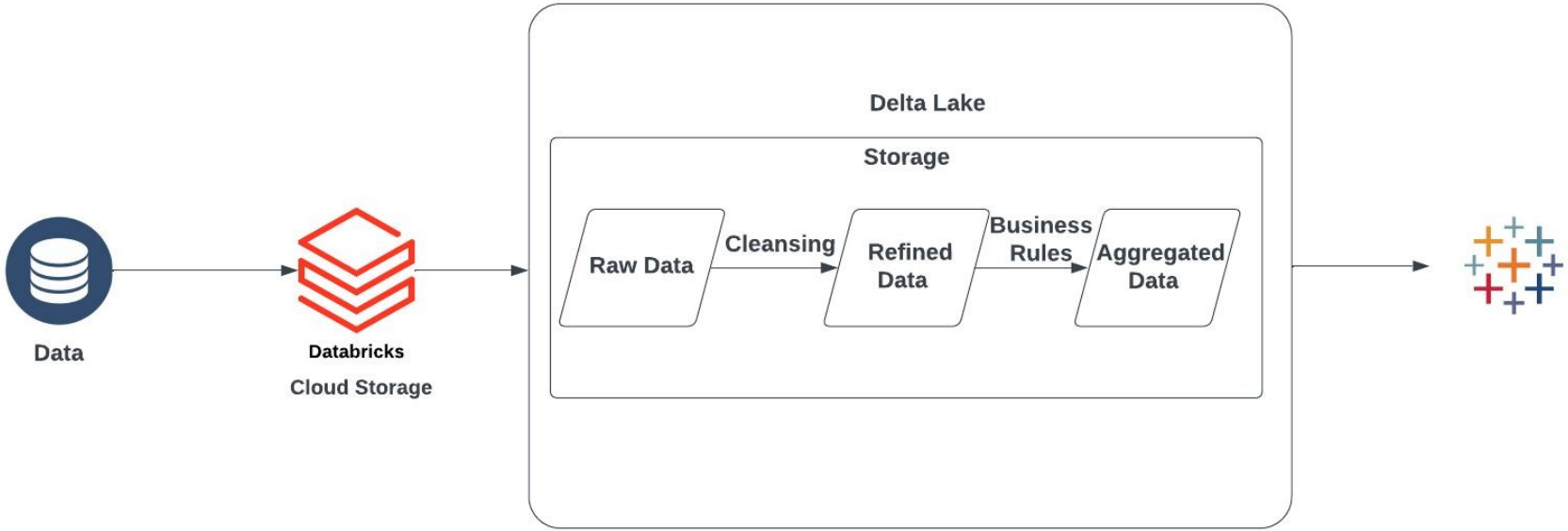
- The AWS, Azure, and Google clouds can all support the use of Databricks to handle enormous volumes of raw, unprocessed data.
- The computing resources and settings that make up a Databricks cluster are used to run data engineering, data science, and data analytics workloads such production ETL pipelines, streaming analytics, ad-hoc analytics, and machine learning.



Parquet

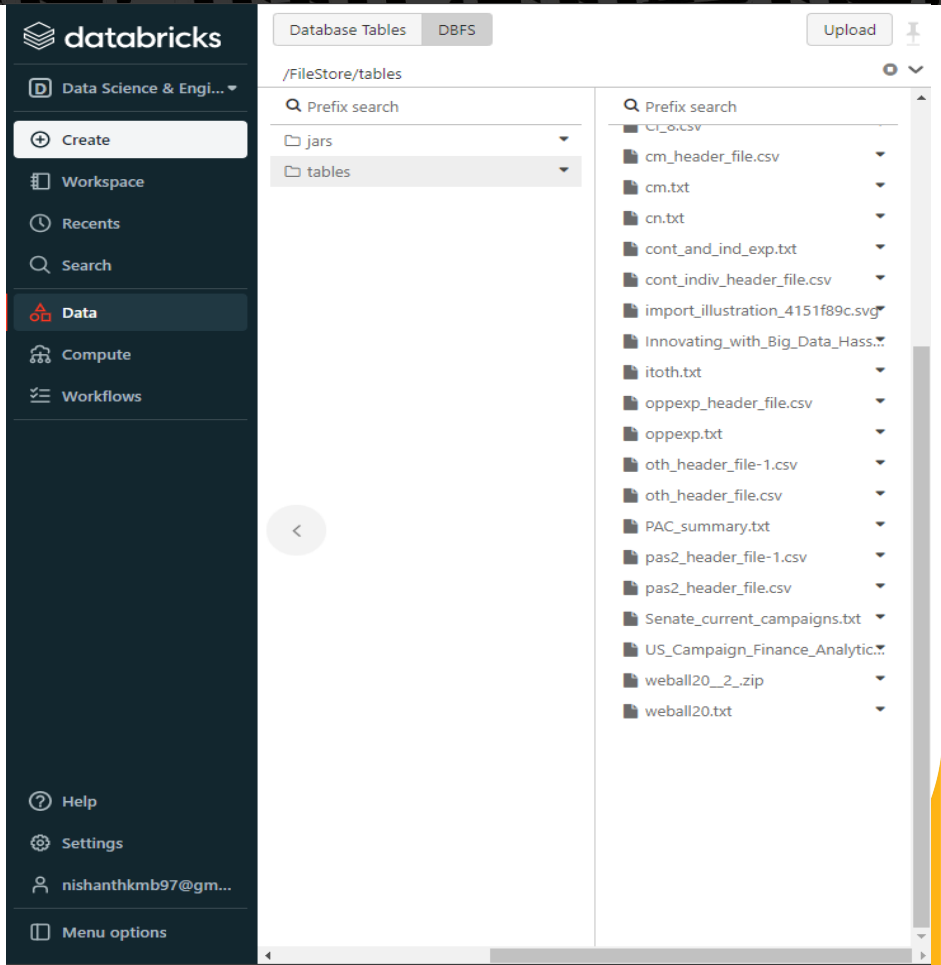
- The very best encoding and compression techniques are supported by Apache Parquet. When used with modern cloud technologies like Amazon Athena, Redshift Spectrum, BigQuery, and Azure Data Lakes, Apache Parquet reduces the cost of data file storage and increases the efficiency of data queries.
- With the ability to read and write Parquet files, Spark SQL maintains the original data's schema automatically. All columns are automatically changed to be nullable when creating Parquet files for compatibility reasons.
- Reading and writing Parquet files is supported by Spark SQL, which automatically preserves the schema of the original data. All columns are automatically altered to be nullable when producing Parquet files due to compatibility concerns.

Architecture



Importing Data

Importing Data Files(.txt) files and .csv(Header files) files into Data Bricks.



Read all the files

- Creating a custom schema for the files which don't have headers

```
from pyspark.sql.types import *
custom_schema_all = StructType([
    StructField("CAND_ID", StringType(), True),
    StructField("CAND_NAME", StringType(), True),
    StructField("CAND_ICI", StringType(), True),
    StructField("PTY_CD", IntegerType(), True),
    StructField("CAND_PTY_AFFILIATION", StringType(), True),
    StructField("TTL_RECEIPTS", DoubleType(), True),
    StructField("TRANS_FROM_AUTH", DoubleType(), True),
    StructField("TTL_DISB", DoubleType(), True),
    StructField("TRANS_TO_AUTH", DoubleType(), True),
    StructField("COH_BOP", DoubleType(), True),
    StructField("COH_COP", DoubleType(), True),
    StructField("CAND_CONTRIB", DoubleType(), True),
    StructField("CAND_LOANS", DoubleType(), True),
    StructField("OTHER_LOANS", DoubleType(), True),
    StructField("CAND_LOAN_REPAY", DoubleType(), True),
    StructField("OTHER_LOAN_REPAY", DoubleType(), True),
    StructField("DEBTS_OWED_BY", DoubleType(), True),
    StructField("TTL_INDIV_CONTRIB", DoubleType(), True),
    StructField("CAND_OFFICE_ST", StringType(), True),
    StructField("CAND_OFFICE_DISTRICT", StringType(), True),
    StructField("SPEC_ELECTION", StringType(), True),
    StructField("PRIM_ELECTION", StringType(), True),
    StructField("RUN_ELECTION", StringType(), True),
    StructField("GEN_ELECTION", StringType(), True),
    StructField("GEN_ELECTION_PRECENT", StringType(), True),
    StructField("OTHER_POL_CMTE_CONTRIB", DoubleType(), True),
    StructField("POL_PTY_CONTRIB", DoubleType(), True),
    StructField("CVG_END_DT", StringType(), True),
    StructField("INDIV_REFUNDS", DoubleType(), True),
    StructField("CMTE_REFUNDS", DoubleType(), True)
])
```

Reading CSV files

```
from pyspark.sql.types import StructType

read_format = 'csv'
load_path = 'dbfs:/FileStore/tables/weball20.txt'


# Load the data from its source.
all = (spark.read.format("csv").schema(custom_schema_all).option("header", True).option("delimiter", "|").load("dbfs:/FileStore/tables/weball20.txt"))
```

```
all.show()
```

```
+---------+--------------------+--------+------+--------------------+------------+---------------+----------+-------------+----------+---------+------------+----------+-----------+
--------------+---------------+------------+----------------+-------------+--------------------+-------------+-------------+------------+------------+--------------------+----
-----------------+---------------+----------+-------------+------------+
|  CAND_ID|           CAND_NAME|CAND_ICI|PTY_CD|CAND_PTY_AFFILIATION|TTL_RECEIPTS|TRANS_FROM_AUTH|  TTL_DISB|TRANS_TO_AUTH|   COH_BOP|  COH_COP|CAND_CONTRIB|CAND_LOANS|OTHER_LOANS|
CAND_LOAN_REPAY|OTHER_LOAN_REPAY|DEBTS_OWED_BY|TTL_INDIV_CONTRIB|CAND_OFFICE_ST|CAND_OFFICE_DISTRICT|SPEC_ELECTION|PRIM_ELECTION|RUN_ELECTION|GEN_ELECTION|GEN_ELECTION_PRECENT|OTHE
R_POL_CMTE_CONTRIB|POL_PTY_CONTRIB|CVG_END_DT|INDIV_REFUNDS|CMTE_REFUNDS|
+---------+--------------------+--------+------+--------------------+------------+---------------+----------+-------------+----------+---------+------------+----------+-----------+
--------------+---------------+------------+----------------+-------------+--------------------+-------------+-------------+------------+------------+--------------------+----
-----------------+---------------+----------+-------------+------------+
|H6AK00045|     YOUNG, DONALD E|       I|     2|                 REP|  1950289.86|      138304.94|1817836.79|          0.0| 116720.12|249173.19|         0.0|       0.0|        0.0|
0.0|             0.0|          0.0|        963416.04|           AK|                  00|         null|         null|        null|        null|                null|             839
094.63|          500.0|12/31/2020|     19609.67|      5500.0|
|H8AK01031| NELSON, THOMAS JOHN|       C|     2|                 REP|         0.0|            0.0|    466.51|          0.0|    466.51|      0.0|         0.0|       0.0|        0.0|
0.0|             0.0|          0.0|             0.0|           AK|                  00|         null|         null|        null|        null|                null|
0.0|            0.0|03/31/2019|          0.0|         0.0|
|H8AK00140|       GALVIN, ALYSE|       C|     3|                 IND|  5253251.54|       60024.76|5162902.93|          0.0|   6245.09|  96593.7|     4371.82|       0.0|        0.0|
0.0|             0.0|          0.0|       4796137.47|           AK|                  00|         null|         null|        null|        null|                null|             340
486.99|        10025.0|12/31/2020|     32594.02|         0.0|
|H0AL01097|     AVERHART, JAMES|       O|     1|                 DEM|    80094.95|            0.0|  78973.24|          0.0|       0.0|  1121.71|         0.0|   29245.0|        0.0|
0.0|             0.0|      29245.0|         50849.95|           AL|                  01|         null|         null|        null|        null|                null|
0.0|            0.0|12/31/2020|          0.0|         0.0|
```

Cleaning The Data

Converting file type and loading

```
write_format = 'parquet'
save_path = '/FinalProject/all'

# Writing the data in binary format into the source folder.
all.write \
  .mode("overwrite")\
  .format(write_format) \
  .save(save_path)
```

```
load_path = '/FileStore/tables/candidate_master_header_file.csv'
cand_master_header = spark \
  .read \
  .format(read_format) \
  .load(load_path , header=True, inferSchema=True)
```

```
cand_master_header.printSchema()

root
 |-- CAND_ID: string (nullable = true)
 |-- CAND_NAME: string (nullable = true)
 |-- CAND_PTY_AFFILIATION: string (nullable = true)
 |-- CAND_ELECTION_YR: string (nullable = true)
 |-- CAND_OFFICE_ST: string (nullable = true)
 |-- CAND_OFFICE: string (nullable = true)
 |-- CAND_OFFICE_DISTRICT: string (nullable = true)
 |-- CAND_ICI: string (nullable = true)
 |-- CAND_STATUS: string (nullable = true)
 |-- CAND_PCC: string (nullable = true)
 |-- CAND_ST1: string (nullable = true)
 |-- CAND_ST2: string (nullable = true)
 |-- CAND_CITY: string (nullable = true)
 |-- CAND_ST: string (nullable = true)
 |-- CAND_ZIP: string (nullable = true)
```

Checking for null values

databricks

Cleaning (Python)

Import Notebook

```
import pyspark
def Shape(dataFrame):
    return (dataFrame.count(), len(dataFrame.columns))
pyspark.sql.dataframe.DataFrame.shape = Shape
print(All_candidates.shape())
```

```
(3981, 30)
```

```
# Distinct checks whether the dataframe has duplicate values
print("distinct count: " + str(All_candidates.distinct().count()))
```

```
distinct count: 3981
```

```
from pyspark.sql.functions import col,isnan, when, count

All_candidates.select([count(when(isnan(i) | col(i).isNull(), i)).alias(i) for i in All_candidates.columns[:10]]).show()
All_candidates.select([count(when(isnan(i) | col(i).isNull(), i)).alias(i) for i in All_candidates.columns[10:20]]).show()
All_candidates.select([count(when(isnan(i) | col(i).isNull(), i)).alias(i) for i in All_candidates.columns[20:30]]).show()
```

```
+-------+---------+--------+------+--------------------+------------+---------------+--------+-------------+-------+
|CAND_ID|CAND_NAME|CAND_ICI|PTY_CD|CAND_PTY_AFFILIATION|TTL_RECEIPTS|TRANS_FROM_AUTH|TTL_DISB|TRANS_TO_AUTH|COH_BOP|
+-------+---------+--------+------+--------------------+------------+---------------+--------+-------------+-------+
|      0|        0|      78|     0|                   1|           0|              0|       0|            0|      0|
+-------+---------+--------+------+--------------------+------------+---------------+--------+-------------+-------+

+-------+------------+----------+-----------+--------------+---------------+-------------+-----------------+--------------+--------------------+
|COH_COP|CAND_CONTRIB|CAND_LOANS|OTHER_LOANS|CAND_LOAN_REPAY|OTHER_LOAN_REPAY|DEBTS_OWED_BY|TTL_INDIV_CONTRIB|CAND_OFFICE_ST|CAND_OFFICE_DISTRICT|
+-------+------------+----------+-----------+--------------+---------------+-------------+-----------------+--------------+--------------------+
|      0|           0|         0|          0|             0|              0|            0|                0|             0|                   0|
+-------+------------+----------+-----------+--------------+---------------+-------------+-----------------+--------------+--------------------+

+-------------+-------------+-------------+-------------+--------------------+---------------------+---------------+---------+-------------+------------+
|SPEC_ELECTION|PRIM_ELECTION|RUN_ELECTION|GEN_ELECTION|GEN_ELECTION_PRECENT|OTHER_POL_CMTE_CONTRIB|POL_PTY_CONTRIB|CVG_END_DT|INDIV_REFUNDS|CMTE_REFUNDS|
+-------------+-------------+-------------+-------------+--------------------+---------------------+---------------+---------+-------------+------------+
|         3981|         3981|         3981|         3981|                3981|                    0|              0|        0|            0|           0|
+-------------+-------------+-------------+-------------+--------------------+---------------------+---------------+---------+-------------+------------+
```

Dropping Null Values

```
# Dropping the columns which have null values
null_columns=("SPEC_ELECTION","PRIM_ELECTION","RUN_ELECTION","GEN_ELECTION","GEN_ELECTION_PRECENT")
All_candidates=All_candidates.drop(*null_columns)
len(All_candidates.columns)
```

```
Out[5]: 25
```

```
display(All_candidates.describe())
```

Table

	summary	CAND_ID	CAND_NAME	CAND_ICI	PTY_CD	CAND_PTY_AFFILIATION	TTL_RECEIPTS	TRANS_FROM_AUTH	TTL_DISB
1	count	3981	3981	3903	3981	3980	3981	3981	3981
2	mean	null	null	null	1.6541070082893745	null	3490903.2603391143	199196.3327103742	3182462.0778698847
3	stddev	null	null	null	0.6416658107355816	null	8.233354043721126E7	5849210.709572181	6.819973962700449E7
4	min	H0AL01055	ANGELA GLASS	C	1	AMP	-674132.5	0.0	-674157.5
5	max	S8WV00143	ZUNKER, TRICIA	O	3	WFP	4.824617973E9	2.7517652812E8	3.848951085E9

Showing all 5 rows.

Replacing the Values

```
TTL_RECEIPTS_correct = when(((All_candidates['TRANS_FROM_AUTH'] != 0.0) & (All_candidates['TRANS_TO_AUTH'] != 0.0)), ((All_candidates['TTL_RECEIPTS'] -
All_candidates['TRANS_FROM_AUTH'])))
All_candidates = All_candidates.withColumn('TTL_RECEIPTS_CORRECTED', TTL_RECEIPTS_correct)
All_candidates.shape()

Out[8]: (3981, 26)
```

```
from pyspark.sql.functions import when
from pyspark.sql.functions import regexp_replace

PAC_file=PAC_file.withColumn('CMTE_DSGN',
      when((PAC_file.CMTE_DSGN == 'A'),regexp_replace(PAC_file.CMTE_DSGN,'A','Authorized by a candidate')) \
     .when((PAC_file.CMTE_DSGN == 'B'),regexp_replace(PAC_file.CMTE_DSGN,'B','Lobbyist/Registrant PAC')) \
     .when((PAC_file.CMTE_DSGN == 'D'),regexp_replace(PAC_file.CMTE_DSGN,'D','Leadership PAC')) \
     .when((PAC_file.CMTE_DSGN == 'J'),regexp_replace(PAC_file.CMTE_DSGN,'J','Joint fundraiser')) \
     .when((PAC_file.CMTE_DSGN == 'P'),regexp_replace(PAC_file.CMTE_DSGN,'P','Principal campaign committee of a candidate')) \
     .when((PAC_file.CMTE_DSGN == 'U'),regexp_replace(PAC_file.CMTE_DSGN,'U','Unauthorized')))

PAC_file.select("CMTE_DSGN").distinct().show()

+--------------------+
|           CMTE_DSGN|
+--------------------+
|        Unauthorized|
|Lobbyist/Registra...|
|                null|
|      Leadership PAC|
|Principal campaig...|
|    Joint fundraiser|
+--------------------+
```

List of files after loading

```
%fs ls "/FinalProject"
```

Table

	path	name	size	modificationTime
1	dbfs:/FinalProject/All_Candidates_Cleaned/	All_Candidates_Cleaned/	0	0
2	dbfs:/FinalProject/Cont_by_indv/	Cont_by_indv/	0	0
3	dbfs:/FinalProject/PAC/	PAC/	0	0
4	dbfs:/FinalProject/all/	all/	0	0
5	dbfs:/FinalProject/cand_master/	cand_master/	0	0
6	dbfs:/FinalProject/ccl/	ccl/	0	0
7	dbfs:/FinalProject/cm/	cm/	0	0

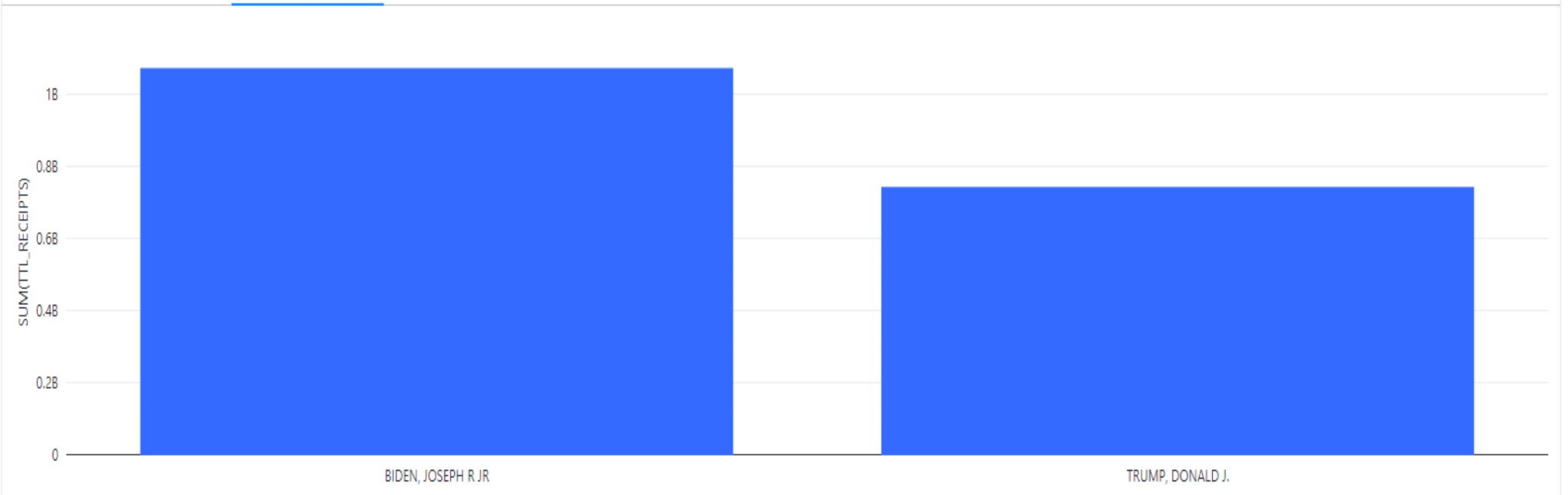
Showing all 13 rows.

Data Visualizations

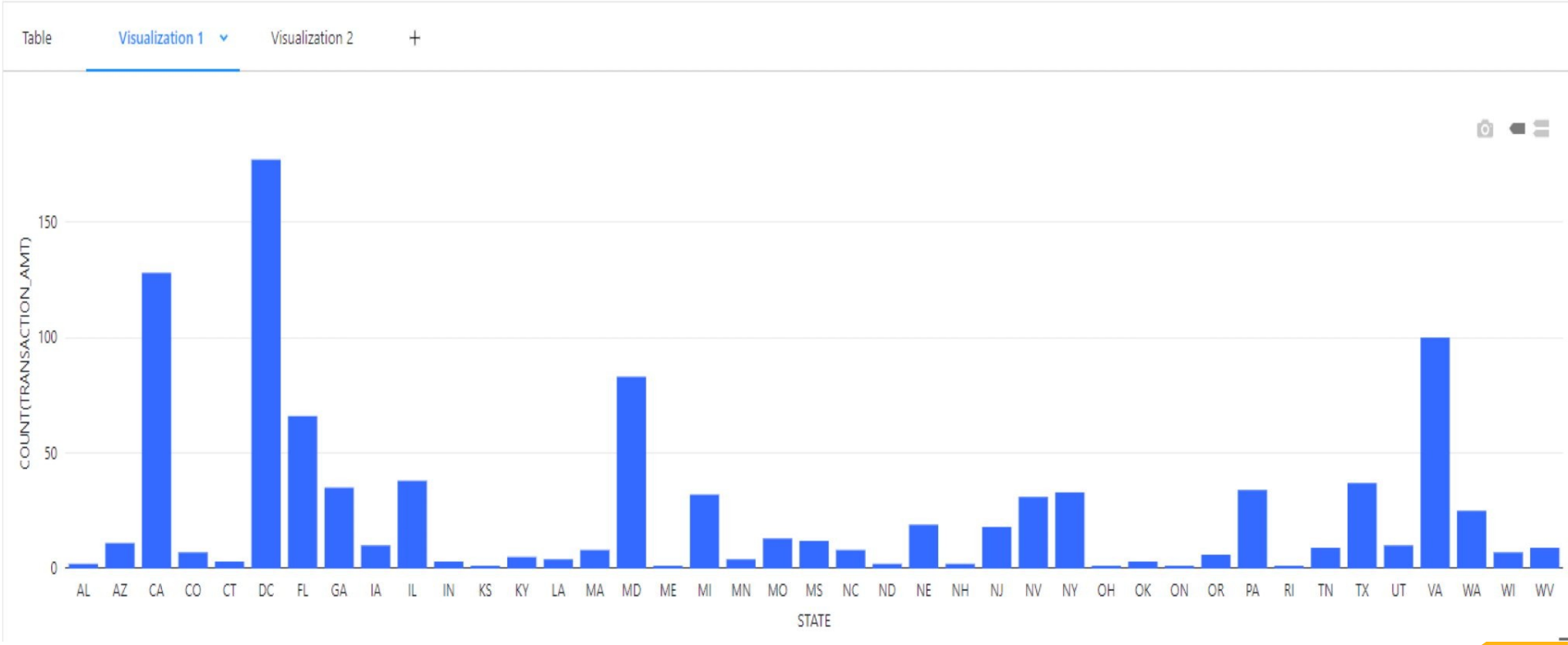
Merging the necessary dataframes and creating tables for visualizing the data.

Visualization For Total Receipts for Each Candidate

- Total Receipts are anything of value (money, goods, services or property) received by a political committee.

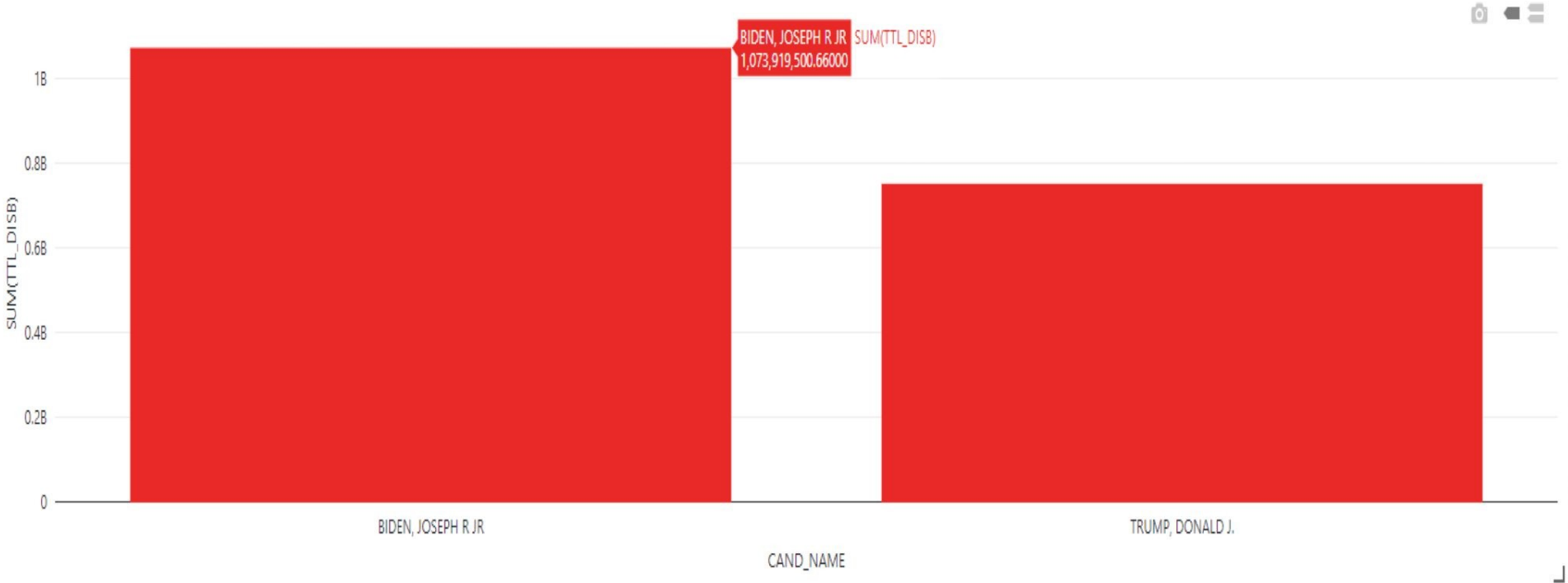


Transaction amount in each state

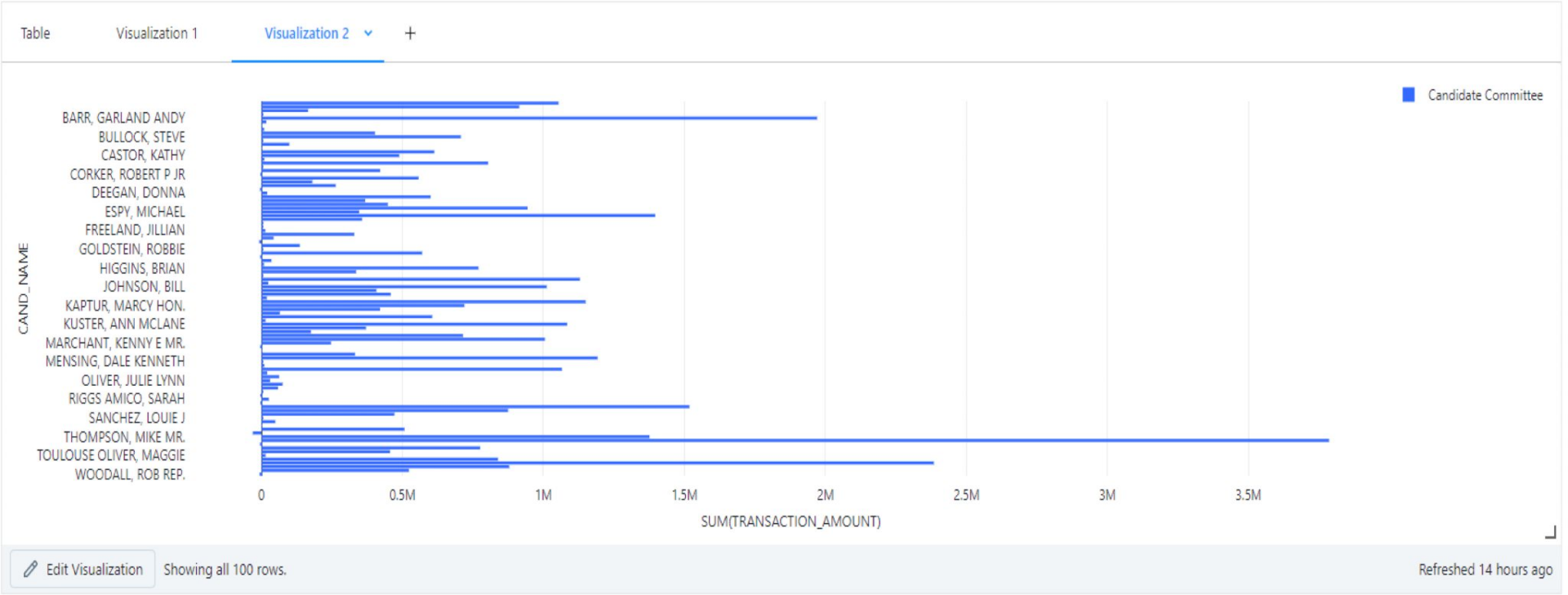


Total disbursements by Candidates

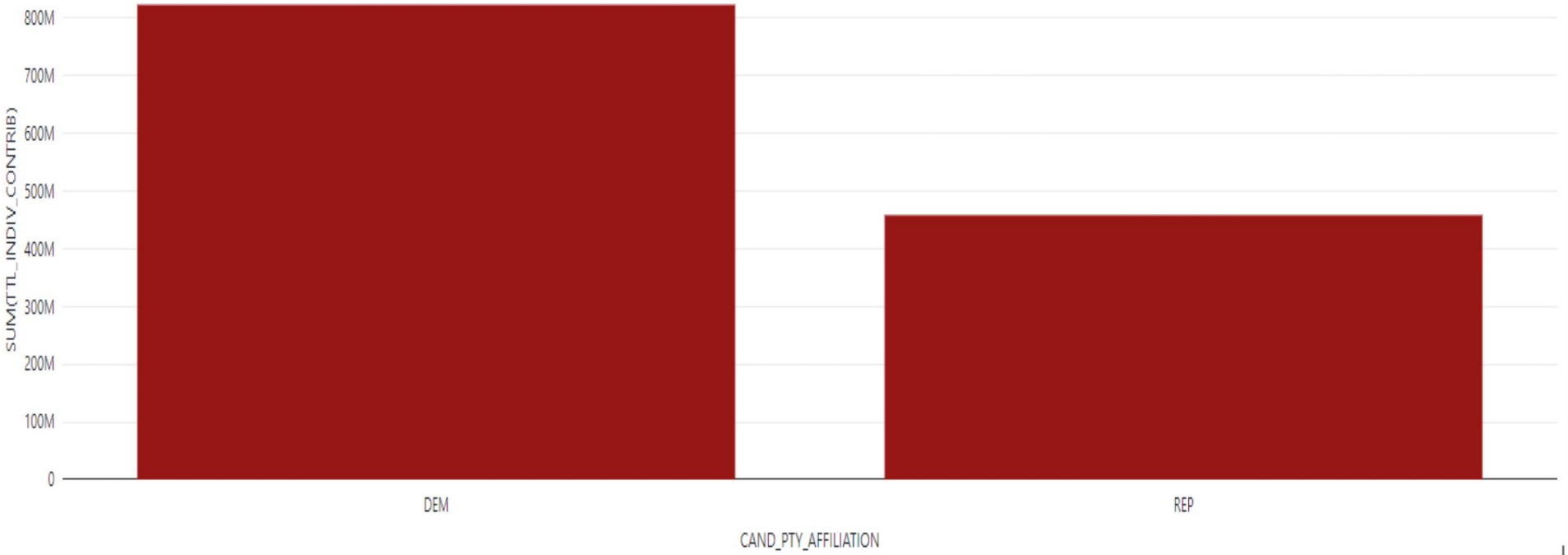
- Disbursement is a border term that covers both expenditure and other kinds of payments(those not made to influence federal election)



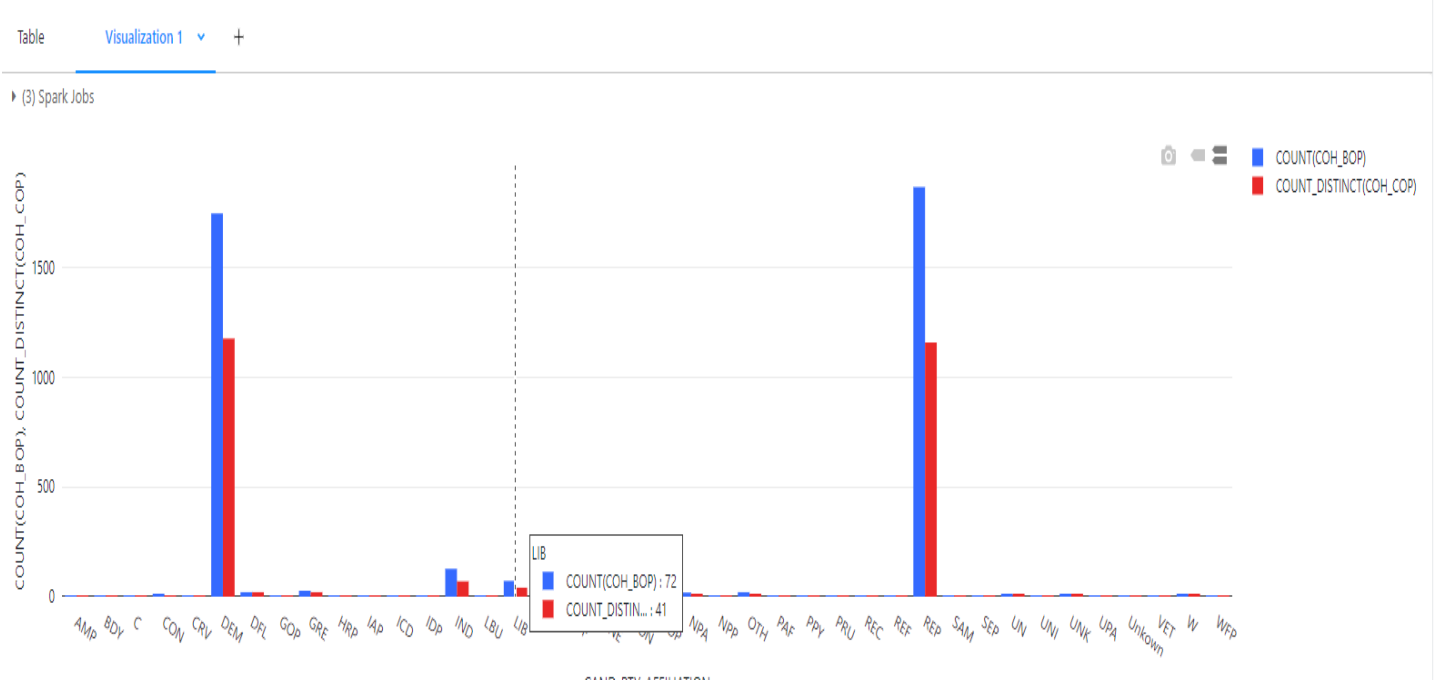
Transaction amount for each candidate contesting



Total Individual Contributions made by candidates

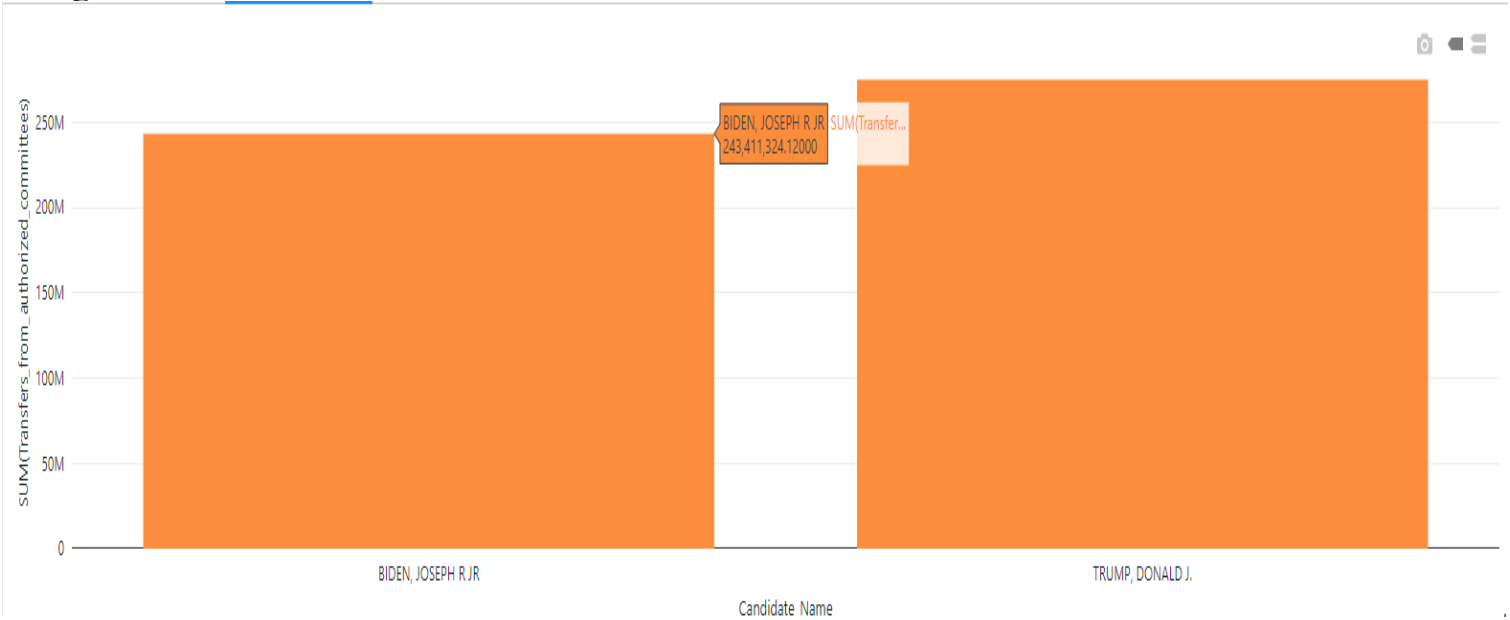


Visualization of COH_BOP and COH_COP for each party

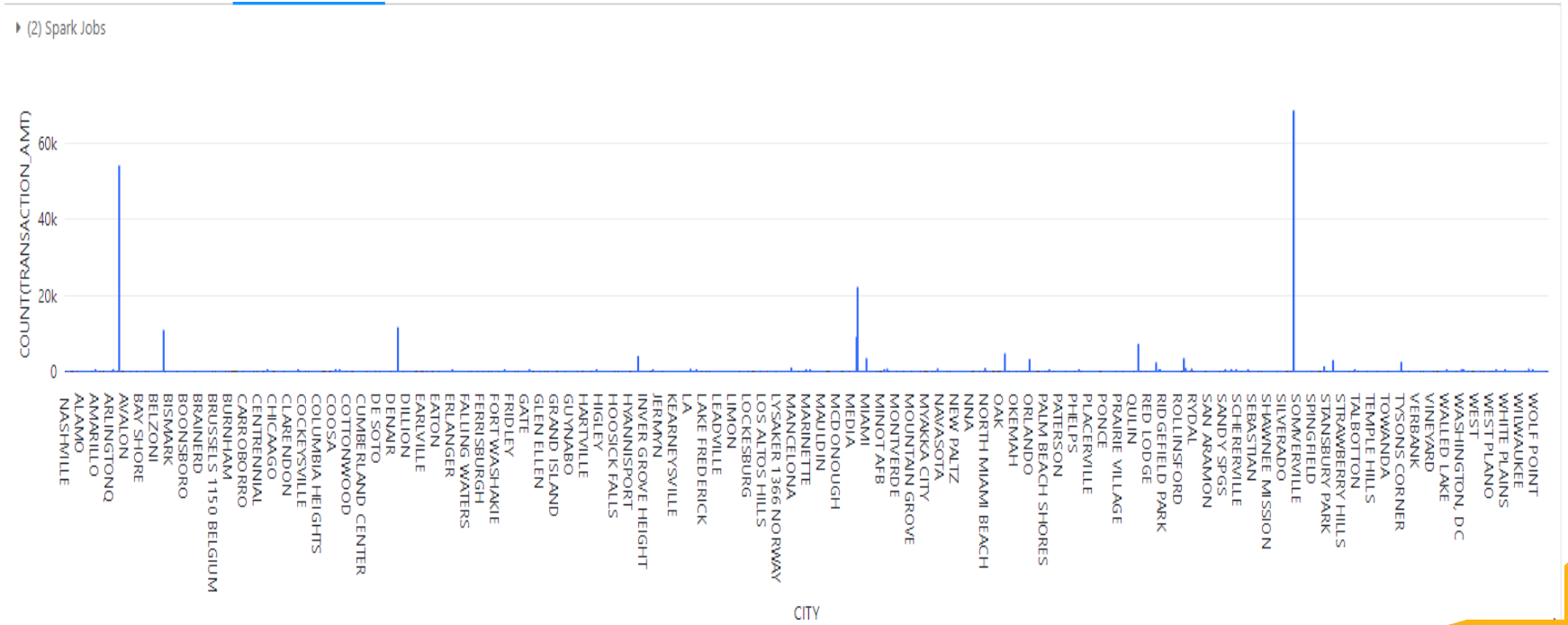


Visualization for transfers from authorized committees to the candidates:

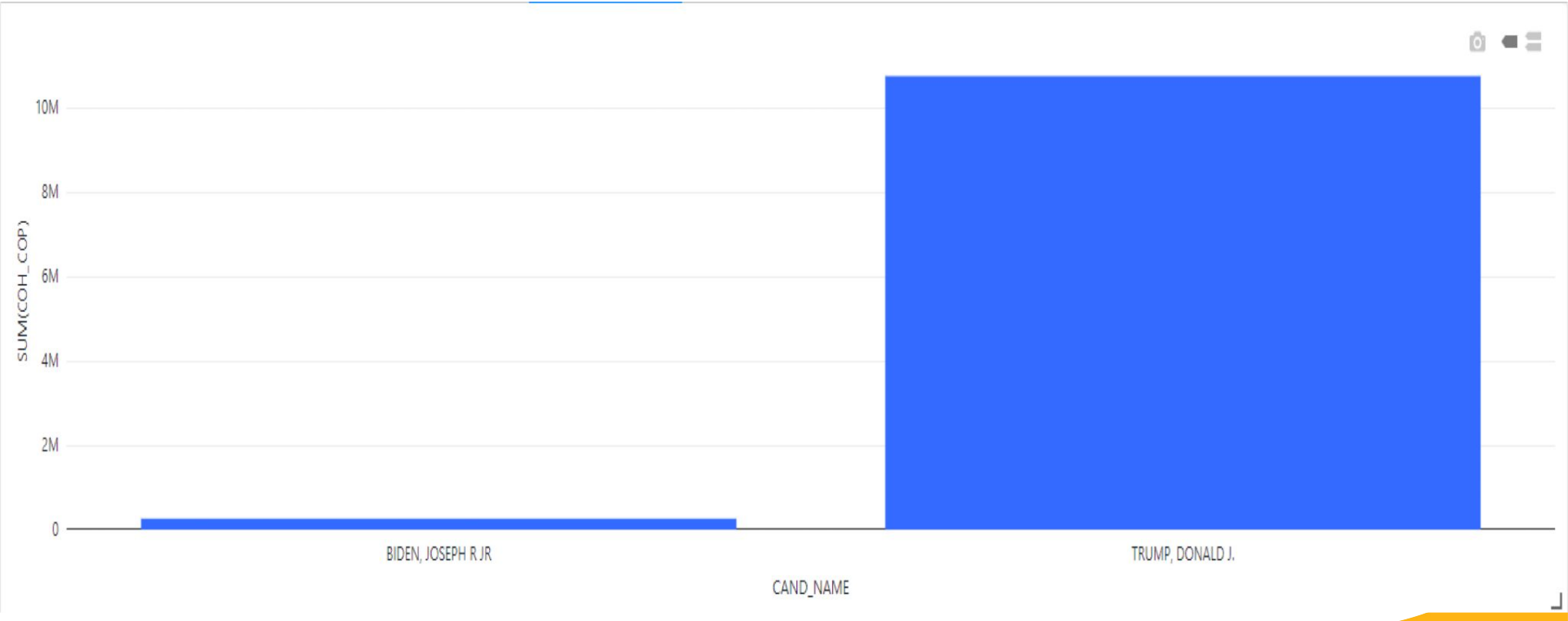
- In general, funds may be transferred between authorized committees of the same candidate (for example, from a previous campaign to a current campaign committee) without limit as long as the committee making the transfer has no net debts outstanding.



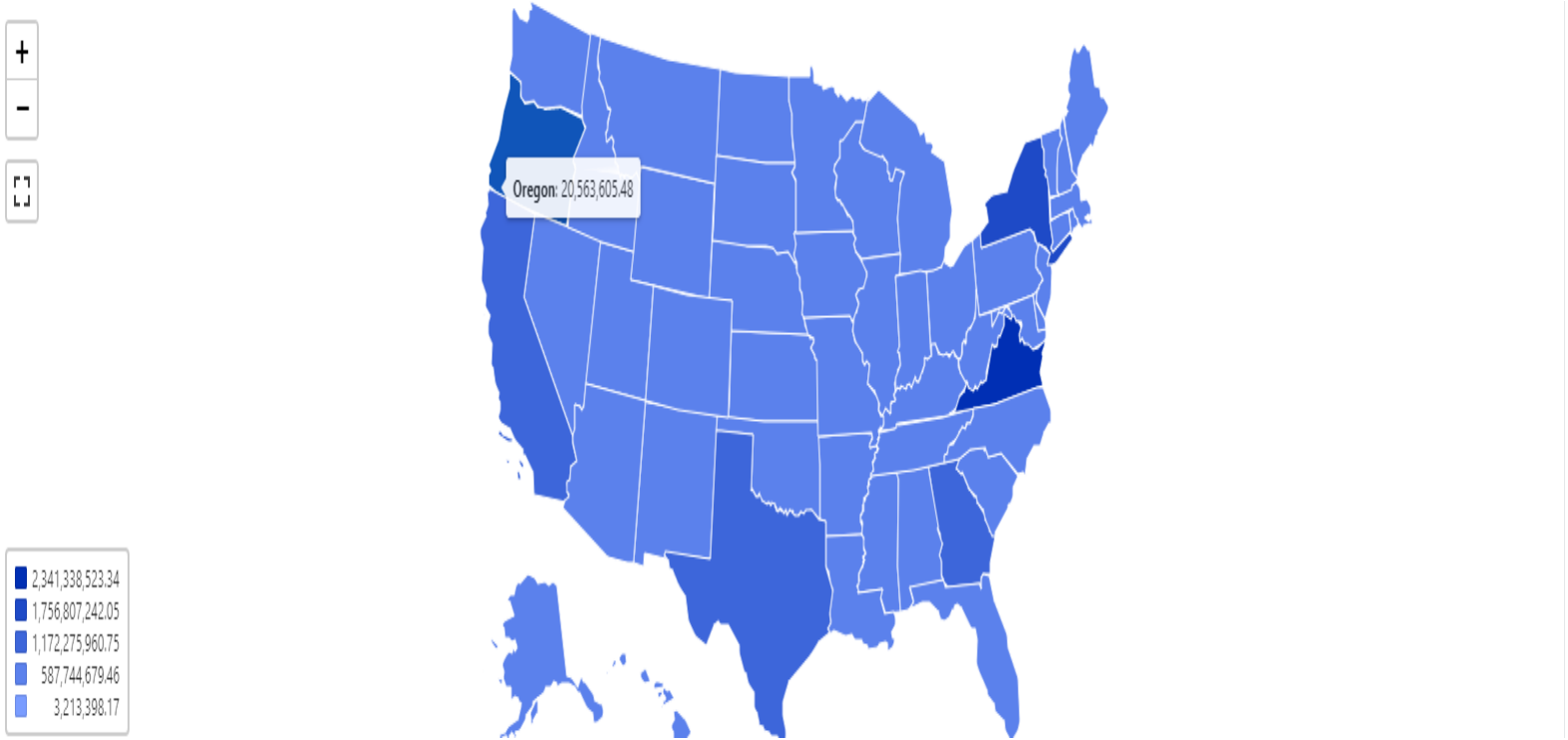
Visualization of Transaction amount in each City



Visualization for COH for the candidates



Visualization of Transaction amount by state for campaigning




Thank
you