­

Penn State University

Great Valley Campus

Engineering Division

Data Specification for

2020 Election Contributions

Stephanie Watkins sjw6182@psu.edu

Version 1.0

04/30/2021

**Table of Contents**

[Introduction 3](#_Toc66634666)

[Purpose 3](#_Toc66634667)

[Project Summary 3](#_Toc66634668)

[Requirements Definition 4](#_Toc66634669)

[Considerations 5](#_Toc66634670)

[Document Change Log 6](#_Toc66634671)

[2. Architecture Design 7](#_Toc66634672)

[2.1 Relational Data Warehouse 7](#_Toc66634673)

[Design and schema 7](#_Toc66634674)

[Tables Schemas 8](#_Toc66634675)

[Examples of values 26](#_Toc66634676)

[Examples of values 26](#_Toc66634677)

[Examples of values 27](#_Toc66634678)

[Examples of values 27](#_Toc66634679)

[Examples of values 28](#_Toc66634680)

[2.2 Hadoop Implementation 8](#_Toc66634681)

[2.3 Reflective analysis of using a data warehouse vs Hadoop. 8](#_Toc66634682)

[3. Data Preparation 9](#_Toc66634683)

[3.1 Relational Data Warehouse Implementation 9](#_Toc66634684)

[ETL Considerations 9](#_Toc66634685)

[ETL Process Flow with description 9](#_Toc66634686)

[3.2 Hadoop Implementation 15](#_Toc66634687)

[3.3 Reflective analysis of data preparation in relational data warehouse vs Hadoop. 15](#_Toc66634688)

[4. Reporting System 16](#_Toc66634689)

[4.1 Relational Data Warehouse Implementation 16](#_Toc66634690)

[19](#_Toc66634691)

[4.2 Hadoop Implementation 25](#_Toc66634692)

[4.3 Reflective analysis of result in relational data warehouse vs Hadoop. 25](#_Toc66634693)

[Conclusions 25](#_Toc66634694)

[Appendix 26](#_Toc66634695)

# Introduction

Election contributions contain trends over the population that can be discovered to make predictions for future elections. This document provides the analysis of the 2020 election contribution data from the states of Florida (FL), Texas (TX), New York (NY), and California (CA) for the months of May, June, and July of 2020. Through this analysis, the predictive trends in election contributions were discovered by using and comparing two different Data Analytics methodologies: traditional data warehousing and Hadoop implementation.

# Purpose

The purpose of this document is to conduct a data analysis using traditional data warehousing and Hadoop implementation that contains the 2020 election contribution information for the states of FL, TX, NY, and CA over the months of May, June, and July of 2020. From these separate data analysis results, election contribution trends will be determined along with election contribution trends among individuals based on their occupations over time.

# Project Summary

This following briefly outlines the process of identifying trend utilizing the traditional data warehouse.

1. **Objectives**

The objectives of the project involve implementing, populating data, and executing reports on big data solution dealing with the 2020 election contributions in the states of CA, FL, NY, and TX to identify trends utilizing traditional data warehouse and Hadoop techniques.

1. **Scope**

The scope of the project is to study the different architectures of traditional data warehouse vs Hadoop on the same data and hardware. The project analyzes the performance of the two systems through their capabilities of delivering knowledge to management.

1. **References**

Barb, A. (2021), PorstgreSQL Installation [Module 4] Canvas.

<https://psu.instructure.com/courses/2107751/modules>

Barb, A. (2021), Dimensional Modeling [Module 5] Canvas.

<https://psu.instructure.com/courses/2107751/modules>

Barb, A. (2021), Extract Transform Load (ETL) [Module 6 Canvas.

<https://psu.instructure.com/courses/2107751/modules>

Barb, A. (2021), Data Warehouse Reporting [Module 7] Canvas.

<https://psu.instructure.com/courses/2107751/modules>

Barb, A. (2021), Hadoop Architecture [Module 8] Canvas.

<https://psu.instructure.com/courses/2107751/modules>

Barb, A. (2021), Retrieving Data with MapReduce [Module 9] Canvas.

<https://psu.instructure.com/courses/2107751/modules>

Barb, A. (2021), Map Reduce Examples [Module 10] Canvas.

<https://psu.instructure.com/courses/2107751/modules>

Barb, A. (2021), Apache HIVE [Module 12] Canvas.

<https://psu.instructure.com/courses/2107751/modules>

Barb, A. (2021), Examples of Data Processing with Hadoop [Module 13] Canvas.

<https://psu.instructure.com/courses/2107751/modules>

ProPublica. (2020). 2020 Presidential Contribution Data by State [Texas, May-July 2020]

<https://projects.propublica.org/itemizer/presidential-contributors/2020>

ProPublica. (2020). 2020 Presidential Contribution Data by State [California, May-July 2020]

<https://projects.propublica.org/itemizer/presidential-contributors/2020>

ProPublica. (2020). 2020 Presidential Contribution Data by State [Florida, May-July 2020]

<https://projects.propublica.org/itemizer/presidential-contributors/2020>

ProPublica. (2020). 2020 Presidential Contribution Data by State [New York, May-July 2020]

<https://projects.propublica.org/itemizer/presidential-contributors/2020>

# Requirements Definition

1. **Goals**

* Deploy two data analysis systems, insert the data, and execute results
* Discuss and compare the pros and cons of each system
* Select better system architecture based on the data characteristics to be analyzed

1. **Usability Requirements**

The system granularity was defined based on the knowledge required to extract data trends dependent on the summation of total contributions and total counts of contributions within the data. The grain of the data was determined to be committee name, occupation, addresses, contribution date, and aggregated sum and count of contribution transactions. The addresses granularity was limited where the street name of the contributors cannot be found. Additionally, individual identity was protected and not available to be queried in the system such as names and employers of the individuals. Furthermore, dates of transactions cannot be filtered out by singular months or days but is instead displayed as one date in the format of YYYY-MM-DD, and there will be no time format available for the transactions. This allows the grain to be committee donations per committee by day by occupation by contribution amount.

1. **Business Questions**

* What are the top 3 occupations based on the total amounts of contributions for each state
* What are the top 3 occupations based on the count of contributions for each state
* What are the top 3 cities that donated the most times for each state
* What are the top 3 cities that donated the most money for each state
* What are the top 3 cities that donated the most to the top two committees per state

1. **Data Requirements**

The data is pre-processed in the correct scope for the class. The data consists of several csv files that contain election contributions by week. The data must represent contributions from at least four states with at least two separate committee parties. The data must span over at least three time periods, which can be weeks, months, or quarters.

# Considerations

Several considerations should be considered for the project. Due to lack of knowledge for the pre-processing of the data in Knime, redundant case-sensitive values were unable to be removed. This will account for errors in the data results as the same data will be counted twice instead of cumulatively if there are capital letters, special characters, or extra words present. This impacts the resulting graphs and presents noisy data and errors in the plots. During the reporting phase, the lesser of the repeat attributes were removed for simplicity. For example, if “retired” and “RETIRED” appeared on the plot, the majority of the results were kept, and the other attribute was filtered out of the data. Additionally, the results were filtered numerically to achieve better visibility on the graphs. This accounts for errors in the results as the data that did not meet the filter requirements were removed in the analysis. Furthermore, the following items should also be considered:

* Use of Docker, DBeaver, and Knime
* Use of Hadoop and Hadoop Ecosystems
* Granularity of query data
* Future plans for extending or enhancing the software
* Portability requirements
* Scalability, integrity, and efficiency requirements
* Storage capabilities of software
* Normalization requirements
* Business questions
* Accuracy of data and analysis

# Document Change Log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Change Date** | **Version** | **CR #** | **Change Description** | **Author and Organization** |
| 02/17/2021 | 1.0 |  | Initial creation. Added information to introduction, purpose, and project summary and requirements definition. | Stephanie Watkins PSU |
| 02/18/2021 | 1.1 |  | Creation of architecture design and star design. | Stephanie Watkins PSU |
| 02/25/2021 | 2.1 |  | Updated star design and architecture. Began description of ETL process. | Stephanie Watkins PSU |
| 03/01/2021 | 2.2 |  | Identified grain for the dataware house to limit expectations. Updated ETL process description. | Stephanie Watkins  PSU |
| 03/05/2021 | 2.3 |  | Updated the business questions and worked on identifying the considerations. | Stephanie Watkins PSU |
| 03/10/2021 | 3.4 |  | Finalized ETL process description. Added in project considerations and updated business questions. | Stephanie Watkins PSU |
| 3/12/2021 | 3.5 |  | Reviewed and updated project summery, and considerations. Added in the reporting results for the traditional data warehousing. | Stephanie Watkins PSU |
| 3/13/2021 | 3.6 |  | Updated considerations, sections 2.1, 3.1, 4, and 4.1. Added and reviewed conclusions. Added all images. | Stephanie Watkins  PSU |
| 3/14/2021 | 3.7 |  | Updated Appendix and conclusions | Stephanie Watkins  PSU |
| 3/28/2021 | 3.8 |  | Updated granularity of dimensions. Added to section 2.2 | Stephanie Watkins  PSU |
| 4/10/2021 | 3.9 |  | Updated section 2.2, 2.3, and 3.2 | Stephanie Watkins  PSU |
| 4/15/2021 | 4.0 |  | Added section 3.2, updated considerations and references | Stephanie Watkins  PSU |
| 4/20/2021 | 4.1 |  | Updated section 3.2 and added to section | Stephanie Watkins PSU |
| 4/25/2021 | 4.2 |  | Reviewed and revised sections 2.2, 2.3, 3.3, and 3.3 | Stephanie Watkins  PSU |
| 4/28/2021 | 4.3 |  | Created sections 4.2 and started updating plots and tables | Stephanie Watkins  PSU |
| 4/29/2021 | 4.4 |  | Deleted and re-added plots and tables, added section 4.3 | Stephanie Watkins  PSU |
| 4/30/2021 | 4.5 |  | Finalized plots and tables in section 4.4, finalized conclusions and appendix | Stephanie Watkins  PSU |

# 2. Architecture Design

## 2.1 Relational Data Warehouse

In order to solve the business questions, a PostgreSQL data warehouse was created utilizing DBeaver because of its ability to store large data safely. In order to setup the data warehouse, a PostgreSQL image was created within Docker and then a container for the data warehouse was created as well. The use of the Docker container allowed for greater efficiency and more consistent operation because the containers tend to be faster to create and quicker to start. Additionally, these containers are isolated and scalable as new containers can be quickly created if the demand required for the application changed. The container was connected to the data warehouse in DBeaver through setting up a connection through the IP address and port settings. To analyze the data trends, it was determined that the data warehouse, based on the granularity, would require a committee, addresses, contributor information, and contribution date dimension table in additional to a fact table. Each dimension was created and connected to the fact table through the use of foreign keys. The fact table was then created where the foreign keys were assigned. A sample of the foreign key assignment in the data is provided in figure 1 in the Appendix. The relationship between the dimensions and fact table are shown in figure 2. As observed, the fact table contained the primary keys of each dimension, which provided a foreign key relationship as required for a star-schema (Barb, 2021). Additionally, it was recognized that there was a known fact for the aggregated sum and count of the transactions, which were included in the fact table.

### Design and schema

Graphical user interface, diagram

Description automatically generated

Figure : Star Schema

### Tables Schemas

The star schema was conducted to answer the business question based on the amount of data required. Star schemas allow data to be accessed faster due to having less joins and allow for better query performance. The fact table is located in the center of the diagram and contains all facts that can be found in the dimension tables (Barb, 2021). As mentioned, four dimensions were determined to be created to solve the business questions, which covers the occupation, addresses of the contributors, committees who received donation transactions, and the date the transactions were conducted. The fact table referenced the dimension tables through the use of foreign keys, which created relationships and allowed for data retrieval. Further description of the architecture of the dimensions and fact table can be found in tables 1-5 in the Appendix.

## 2.2 Hadoop Implementation

Moving away from traditional relational data warehouses, a data warehouse will now be created utilizing the Hadoop ecosystem Hive. Hadoop is an open-source system that mimics Google’s distributed computing system (Barb, 2021). It allows large amounts of data to be processed faster in a scalable and reliable way. The main components of Hadoop are the Hadoop Distributed File System (HDFS) and Hadoop MapReduce. HDFS is a java-based system that allows for scalability and reliability for large data storage (Barb, 2021). Hive is a data warehouse software built on top of Hadoop that is used to maintain all intermediate data found within the HDFS where the data can be queried utilizing SQL query langue (Barb, 2021). Within Hive, a data warehouse was created that contained a committee table with the columns city, state, zip, employer, occupation, contributionamount, contributiondate, and committee as shown in the Appendix figure 17. The following code was used to make the data warehouse in Hive:

* bl
  + (Used to enter Hive)
* CREATE DATABASE IF NOT EXISTS Project2;
  + (Creates a database named Project 2 if one does not already exist)
* CREATE TABLE committeedata ( city VARCHAR (50), state VARCHAR(20), zip VARCHAR(20), employer VARCHAR(50), occupation VARCHAR(50), contributionamount FLOAT, contributiondate timestamp, committee VARCHAR(50));
  + The above code creates a table with attributes followed by the data type
  + VARCHAR(2) means that the character will have a max length of 20

## 2.3 Reflective analysis of using a data warehouse vs Hadoop.

Comparing the creation of the star schema data warehouse to the Apache Hive data warehouse, Apache Hive was less time consuming. The traditional data warehouse creation required several tables to be created to be able to reference and join the data into a fact table within the star schema. The hive data warehouse only required the creation of one table that could withhold all of the information without the need of other dimension tables. However, it should be noted that Hive is not a full database and inserting, updating or deleting tasks are not supported within Hive as they are in the traditional relational data warehouse. Additionally, the traditional data warehouse is easily understood with SQL supported language. While Hive does support a type of SQL language, it utilizes its own HQL (Hive Query Language) that is not identical to SQL. For large amounts of data, Hive outperforms traditional relational data warehouses when considering time, unstructured data, and cost.

3. Data Preparation

## 3.1 Relational Data Warehouse Implementation

## ETL Considerations

When conducting the ETL (Extracting, Transforming, Loading) process, many factors must be considered. The ETL process is vital to the data warehouse and one of the most important points during its creation (Barb, 2021). The data must be preprocessed and examined for duplicate, missing, invalid, or mismatched data in order to improve the quality and accuracy of the extracted information. The data may be altered by either replacing with another value, altering the format, removing the data, or utilizing a combination of all three techniques. Additionally, the data should be sorted to remove noisy data that is not needed for the business requirements or granularity to provide more free space in the data warehouse and enhance data integrity. Once these cleansing requirements are solidified, they must be implemented in an efficient way to deal with multiple sources and large datasets reliably. Furthermore, the ETL process should also consider how the data may change over time such as the format and change requests for new dimensions or columns.

## ETL Process Flow with description

The data for the contributions for the 2020 presidential election were downloaded for the months of May, June, and July of 2020 for the states of CA, FL, NY, and TX from the ProPublica website. Before data could be added to the data warehouse, the data had to be processed first following the ETL (Extract, Transform, and Load) process. The ETL process was conducted in Knime by first extracting the data by importing one excel file in the excel reader node. Row filtering was then performed on the zip attribute as there were several values that were missing but filled with zero as a place holder. The rule in the row filtering was created to remove the rows where zip was found to have a zero placeholder to eliminate duplicate values. Additionally, several cities were repeated in the data due to case sensitive issues. To solve this, a duplicate row filter was inserted to remove the duplicated city names. Due to the not null constraints in the data warehouse, a missing value node was inserted to remove the rows of data that were found to have missing values in one or more of the following: occupation, amount, zip, date, and committee name. This process ensures that data integrity was kept when new data is introduced into the system. Lastly, a PostgreSQL connector node was added so that Knime could connect to the data warehouse. The extracting process of the ETL is show in in figure 2.

Diagram

Description automatically generated

Figure : Extracting Process

Data transformation was then performed separately for each dimension. Starting with the committee dimension, a column filter was implemented to restrict what columns should be loaded into the data warehouse. In this case, the only attribute required was committee name. This provided all data records only for the different committee names. The committee names were then grouped together to display each committee once. As previously mentioned, a PostgreSQL connector node was used to be able to connect the Knime workflow to the created data warehouse in DBeaver. A query reader node was used to be able to connect to only the committee dimension. In order to prevent data to be added more than once, this process referenced the data warehouse to exclude any uploaded transformed data that was already found in the data warehouse by using the reference row filter node. The committee name attribute was renamed to match the attribute name in the data warehouse so that the data could easily be added. Once the column names matched, the data was added to the data warehouse using a data warehouse writer node. This process was repeated for the occupation where the columns filtered and grouped based on the occupation of the contributors for the occupation dimension as shown in figure 4.

Timeline

Description automatically generated

Figure : Transformation and Loading Process for Committee and Contributor Information Dimensions

The addresses table required three columns to be filtered, which were the city, state, and zip attributes. After these were filtered and grouped, a column combiner had to be used to combine the three attributes together so that the data being transformed and the data read in the query reader node were identical, which allowed all three attributes to be referenced at once. After this step, the combined string was filtered out and the three attributes of city, state, and zip were renamed to match the data warehouse before being written in the data warehouse with the DB writer node as shown in figure 5.

Diagram

Description automatically generated

Figure : Transformation and Loading Process for Addresses Dimension

The contribution date data went through a similar transformation process where the date was filtered out and modified by using the string to date and modify time node to remove the time as the original format was YYYY-MM-DDT00:00. Due to the granularity defined earlier, the time was not needed and removed. The column was then grouped and renamed to match the data warehouse before being written to the data warehouse as shown in figure 6.

Diagram

Description automatically generated

Figure : Transformation and Loading Process for Contribution Date Dimension

During each addition of the data in the four dimensions, serial synthetic primary keys were created within the date warehouse. This process can be observed for the committee table in figure 2 of the Appendix where each committee was assigned a serial number that serves as the primary key for the committee dimension. Each dimension had generated serial primary keys, which became the foreign keys in the fact table.

Once the data was in the proper corresponding dimensions, the fact table required the primary key data from each dimension to be added through the use of several joiner nodes. As mentioned, a foreign key relationship was created within the data warehouse between the dimensions and the fact table, which allowed the joiner nodes to combine data into one table based on the pre-defined relationships. The process for the creation of the contribution date id in the fact table is shown in the first two rows of nodes in figure 8. The first step to transforming the contribution date data into the fact table was to use the DB query reader node to connect the contribution date dimension in the data warehouse with the imported contribution date data that was extracted in Knime. The contribution date from the data import was joined with the contribution date attribute found in the data warehouse by using the joiner node. This join created the contribution date id attribute that was renamed in the column rename node to match the attribute naming in the fact table. It should be noted that in this first joiner node, the selected data required for the fact table must be chosen so that it is included throughout the join process. Since city, state, and zipcode were required to make joins later, these parameters had to be included to be carried throughout the join process when creating the table along with committee name, amount, contribution date, and occupation.

Next, a second joiner node was performed where the committee name from the data warehouse was pulled and combined into a table with the contribution date id as well as the other parameters from the previous joiner node. Then, the contributor information dimension was referenced from the data warehouse and added through a third joiner node into the table to create the contributor information id. Finally, the addresses attributes were referenced from the data warehouse to create the address id for the fact table in the same process with the fourth joiner node. The complete process for creating attributes in the fact table is shown bin figure 8.

Figure : Fact Table Transformation Process

Diagram

Description automatically generated with medium confidenceThe fact table required the summation of the transaction amounts as well as the count of how many transactions occurred because this information could be found throughout all dimensions and were known facts. This was performed through aggregation using the groupby node to be able to sum and count the transactions based on the foreign keys of committeeid, contributiondateid, addressid, and contributorinformationid. The created aggregated columns were renamed to “sum” and “count” to match the data warehouse. The finalized table was written to the data warehouse using the DB writer node where this table was added to the fact table as shown in figure 9. With the Knime process completed, new data was imported and ran through the entire Knime workflow for each excel file. The complete Knime workflow for the ETL process can be found in figure 10.

Diagram

Description automatically generated

Figure : Aggregation Process of Sum/Count and Loading Process of Fact Table

Diagram, map

Description automatically generated

Figure : Complete ETL Process for the Dimensions and Fact Table

## 3.2 Hadoop Implementation

The implementation of ETL within Hadoop was performed using the framework known as MapReduce. MapReduce is a major part of Hadoop in addition with HDFS (Barb, 2021). Utilizing MapReduce allowed the large data files of the state contributions to be processed through clusters in a reliable manner. The first process was to extract the data by creating HDFS folders: one where the java codes would be stored and an input file where the data would be stored. The HDFS folders were added within Hadoop and the data files were added to the input HDFS folder. Before MapReduce could be utilized, the java codes for the MapIt, ReduceIt, and MapReduceIt had to be added and compiled within the java code HDFS folder in order to create the jar file (Barb, 2021). The MapIt java code extracted the desired attributes of city, state, zip, employer, occupation, contribution amount, contribution date, and committee. The ReduceIt java code does not perform any computation in this case, but provides the output of the key (Barb, 2021). The MapReduce java file sets up the parameters of the job and declares the mapper and reducer classes in addition to the class of the output variable and the data input and output (Barb, 2021). Once the java files were added, a jar file was created and the MapReduce was run within the Hadoop home directory. Within the MapReduce framework, the datasets were independently split and processed through the map tasks run through Java. The outputs of the maps were then used as the inputs to the reduce tasks part of the framework. The framework operates by using key value (<key, value>) pairs. Meaning that the input <k1, v1> is used in the map task where the output <k2, v2> is the input for the reduce tasks where the output would then be <k3, v3>. The key value pairs must be serialized by the framework to be writable within the interface (Barb, 2021). At the end of the reduce task, an output file was created that stored the extracted data within the HDFS. The resulting HDFS data from the MapReduce transformation was then moved into the Hive data warehouse using the following command:

* bl
  + Used to enter HIVE
* Load data local inpath ‘/tmp/result\_Project2.csv’ overwrite into table Project2.committeedata

This command moved the created CSV file in the output folder from the map reduce and writes the data into the committeedata table found in the Project2 database.

## 3.3 Reflective analysis of data preparation in relational data warehouse vs Hadoop.

The ETL process proved to be simplified by using Hadoop MapReduce compared to the Knime workflow. The MapReduce proved to be more efficient, which in turn corresponds to a lower cost. All data files were able to be ran at once in MapReduce whereas the Knime workflow had to be ran for each individual data file in order for the data to be extracted, loaded, and transformed. If additional data was required for ETL in Hadoop, the output folder could be removed, and MapReduce could be re-run. HIVE can drop and re-create the table in the database to incorporate this change, or other HQL queries could be used to update new data, which would be more efficient. The further benefits of utilizing Hadoop ETL over traditional ETL is that the raw data, transformed data, staging area, and data warehouse are all in the same program whereas with the above traditional ETL process, the raw data is stored externally and each file is imported into the extraction and transformation process separately (Barb, 2021). The transformed data is then loaded into a separate program into a created data warehouse that contains multiple dimensions tables and a fact table. Overall, the ETL implementation through Hadoop was simpler, more efficient, and was able to be condensed into one table within HIVE. Additionally, the ETL process for big data was more efficient within Hadoop versus the traditional ETL and can handle unstructured data without scalability cost concerns.

4. Reporting System

In order to evaluate trends within the data, further configuration and transformation were required to show visual trends. The data from the data warehouse was read and extracted using the PostgreSQL Connector and DB Query Reader nodes. Each dimension required a DB Query Reader node as shown below. From this point, a domain calculator was needed for the dimensions, excluding the fact table, which allowed the value filter node to read all the data entries from the data warehouse as without this node the number of entries read were limited. The value filter nodes were used on the dimensions as they contained information that could be filtered out if not needed for the analysis. In this case, the data corresponding to the contribution date values that were not from May, June, or July of 2020 were removed for the analysis. Additionally, different names for “retired” were removed while the data that corresponded to the large majority of “retired” was kept. A similar concept was performed with the addresses dimension where some cities were listed multiple times under case sensitive names where the lesser of the city results were removed. The joiner nodes were added to combine the filtered data again based on the foreign key relationship into one table in a similar process as observed above when creating the fact table. For better visualization of data trends, each state was filtered to evaluate separately done by a row-filter process. The entire reporting process in Knime is shown in figure 11.

Diagram

Description automatically generated

## 4.1 Relational Data Warehouse Implementation

Figure : Reporting Process

Once the reporting process was completed, the reporting tools were utilized to create graphs from the traditional data warehouse to reveal trends. The first business question to be answered by using the relational data warehouse was to find the top three occupations that contributed the most to each election. It was decided to observe the trends for each occupation per state for the total sum and total number of contributions. From these results, retired and not employed for occupations were tied for the top donating occupation based on total donations as shown in table 6 based on figures 12, 13, 15, and 16. The top donating occupation based on how many times a donation was made for all states are shown in table 7 based on figures 12, 13, 15, and 16.

Table : Top Three Contributing Occupation Per State by Total Donations

|  |  |  |  |
| --- | --- | --- | --- |
| State | 1st Occupation | 2nd Occupation | 3rd Occupation |
|
| CA | Not Employed | Retired | Attorney |
| FL | Retired | Not Employed | Teacher |
| NY | Not Employed | Attorney | Retired |
| TX | Retired | Owner | President |

The top 3 occupation that contributed the most to the election per state by total count per occupation are summarized below:

Table : Top Three Contributing Occupation per State by Total Count

|  |  |  |  |
| --- | --- | --- | --- |
| State | 1st Occupation | 2nd Occupation | 3rd Occupation |
|
| CA | Retired | Not Employed | Business |
| FL | Retired | Not Employed | Principal |
| NY | Retired | Not Employed | Attorney |
| TX | Retired | Owner | President |

The next business question to be answered was to find the top three cities that donated the most times from each state. The top three cities that donated the most times per state are summarized in table 8 based on figures 12, 13, 15, and 16:

Table : Top Three Contributing Cities per State by Total Count

|  |  |  |  |
| --- | --- | --- | --- |
| State | 1st City | 2nd City | 3rd City |
|
| CA | Lincoln | San Jose | Lompoc |
| FL | The Villages | Jacksonville Beach | Boca Raton |
| NY | New York City | Staten Island | Brooklyn |
| TX | Odessa | Midland | Seminole |

The next business question to be answered was to find the top three cities that donated the most overall from each state. The top three cities that donated the largest amount per state are summarized in table 9 based on figures 12, 13, 15, and 16:

Table : Top Three Contributing Cities Per State by Total Donations

|  |  |  |  |
| --- | --- | --- | --- |
| State | 1st City | 2nd City | 3rd City |
|
| CA | San Francisco | Irvine | San Carlos |
| FL | Sarasota | Ft. Lauderdale | Vero Beach |
| NY | New York City | Brooklyn | Atlantic Beach |
| TX | Odessa | Midland | Seminole |

The final business question was to determine which cities donated the most to the top two committees to be able to identify trends of what political party the cities would be expected to donate to in the future. As shown in figures 14 and 17, the committees that received the most contributions in each state were Donald J. Trump for President, INC and Joe Biden for President. The cities that contributed most to these committees per state are shown below in tables 10 and 11:

Table : Top Three Cities that Contributed to Biden for President by State

|  |  |  |  |
| --- | --- | --- | --- |
| State | 1st City | 2nd City | 3rd City |
| CA | San Francisco | San Carlos | Pacific Palisades |
| FL | Sarasota | Lake Mary | North Miami |
| NY | New York City | Purchase | Brooklyn |
| TX | Austin | Houston | Dallas |

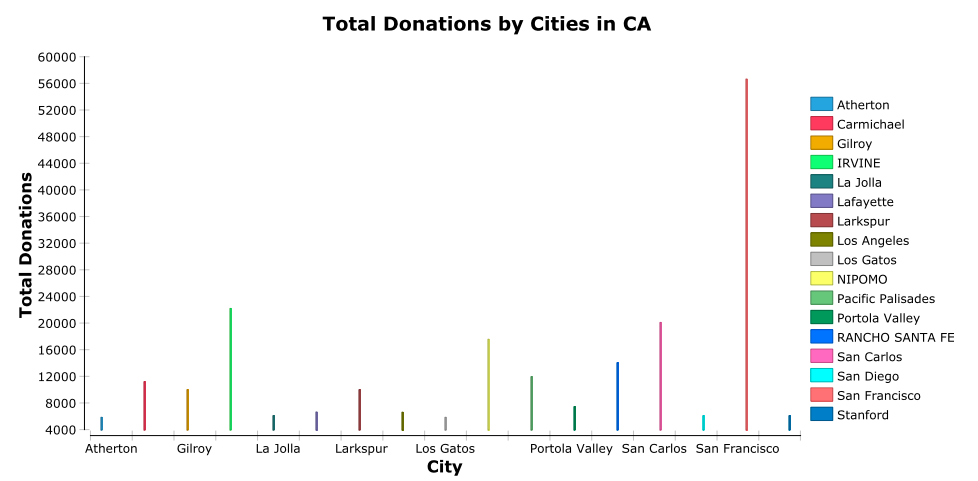
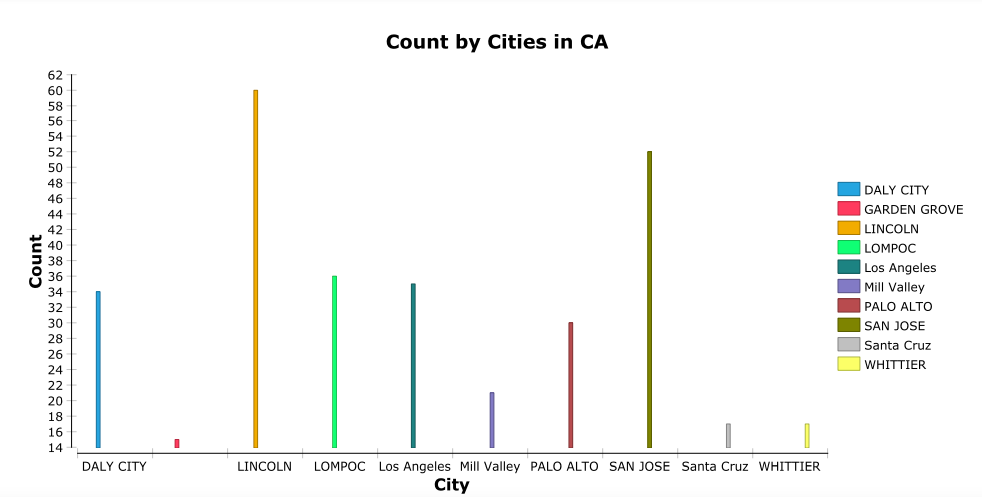
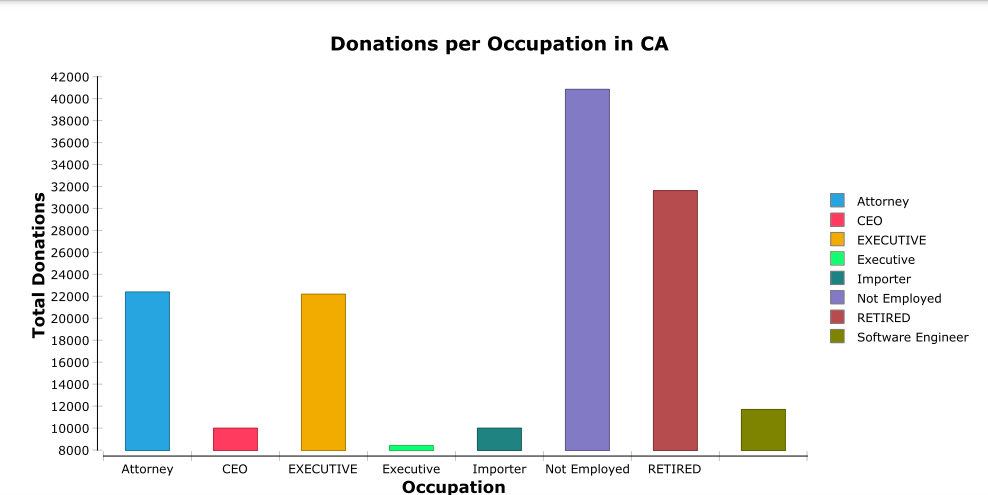
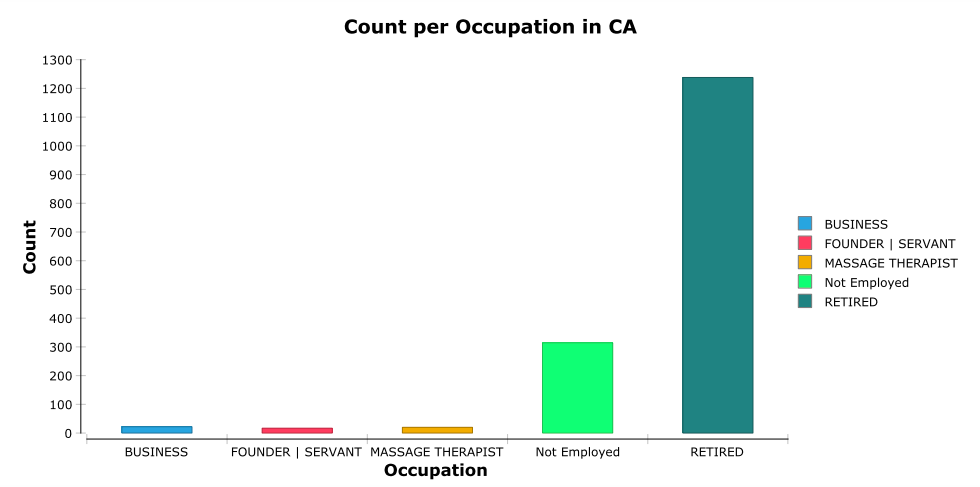
Table :Top Three Cities that Contributed to Donald J. Trump for President, INC by State

|  |  |  |  |
| --- | --- | --- | --- |
| State | 1st City | 2nd City | 3rd City |
| CA | Irvine | Nipomo | Rancho Santa |
| FL | Naples | Boca Grande | Palmetto Bay |
| NY | Brooklyn | Atlantic Beach | Flushing |
| TX | Odessa | Midland | Houston |

It should be noted that the results are dependent on a filter that was determined for each state. California did not include any donation amounts less than $4000 for the determined results, similarly Florida excluded any donations less than $5000, New York excluded donation results that were less than $5000, and ­­Texas excluded donations that were less than $5000. This was performed to be able to visualize the trends better. However, upon performing this filter data and trends were assumed to be lost in the results as they are biased to larger donation amounts.

## 

Figure : Occupation and City Trend Results in CA



## 

Chart, histogram

Description automatically generated

Figure : Occupation and City Results in FL

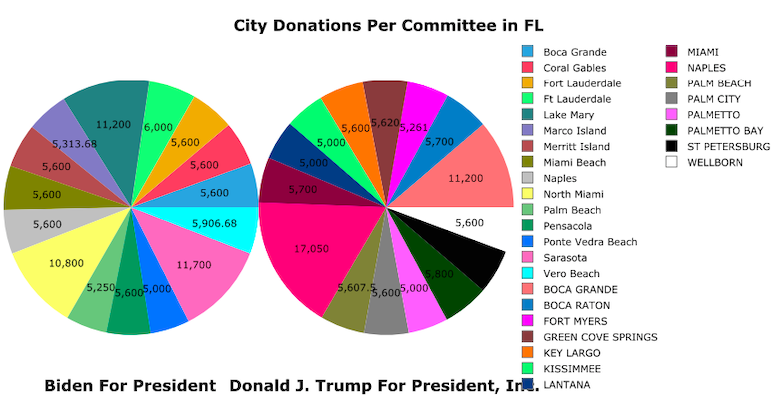
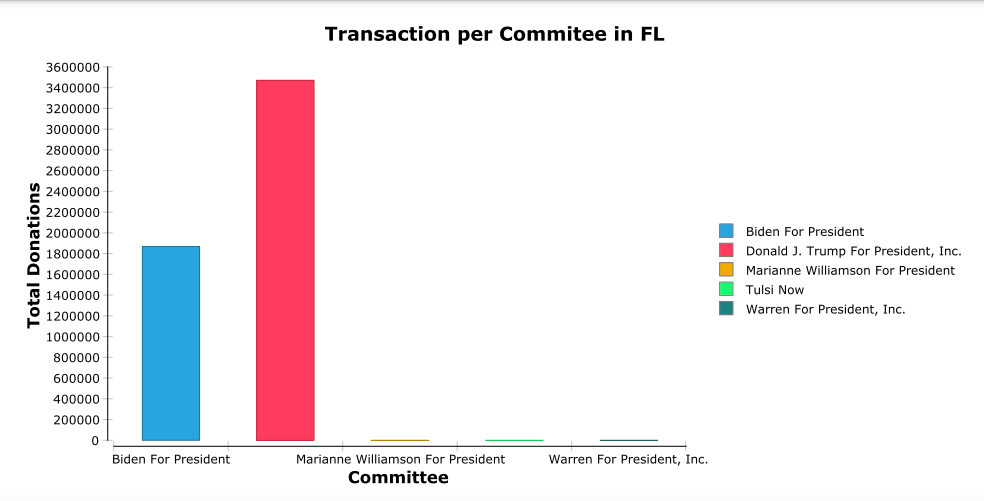
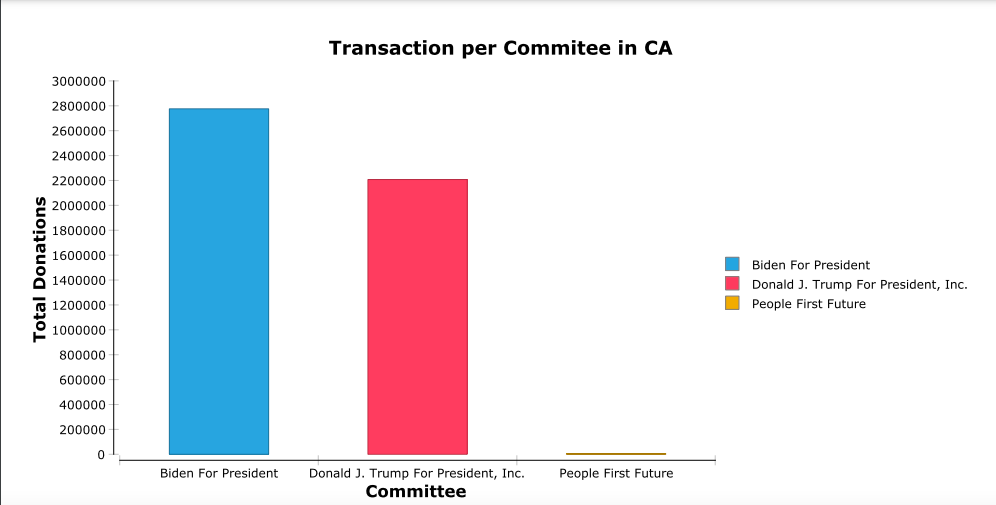
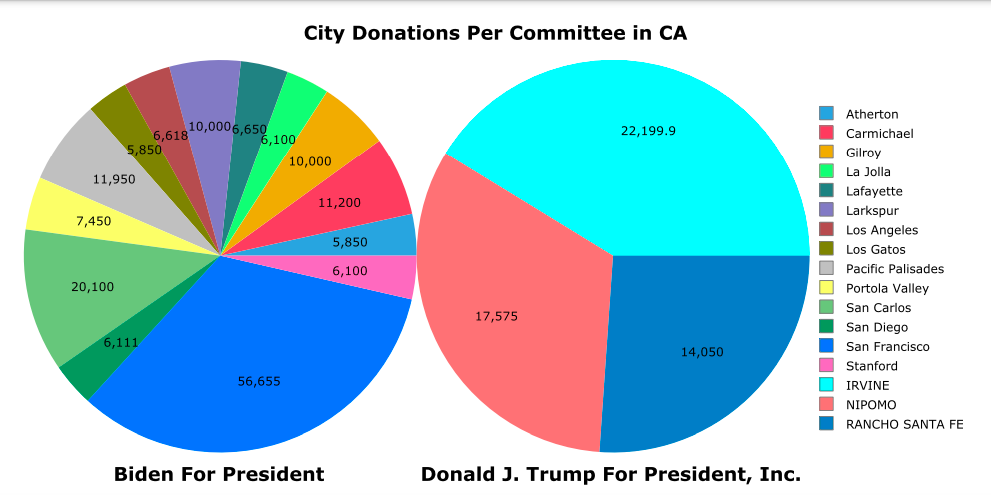
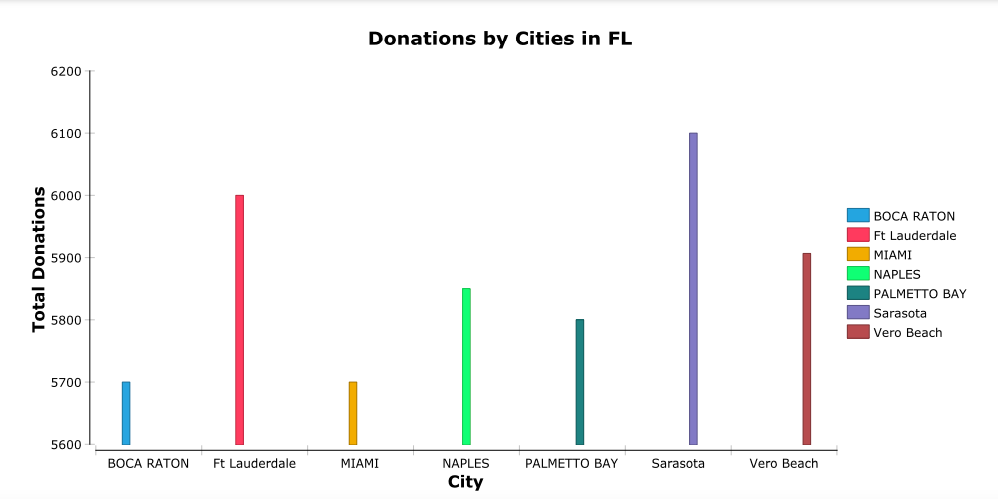
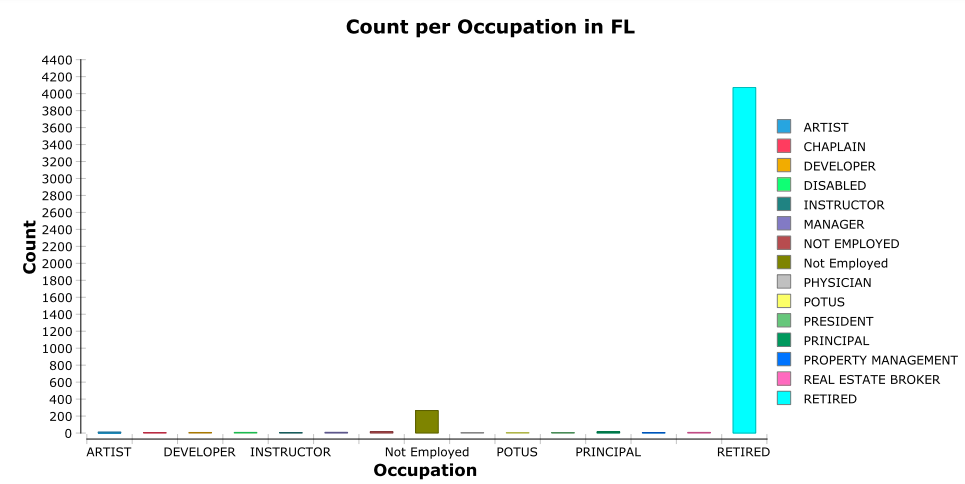
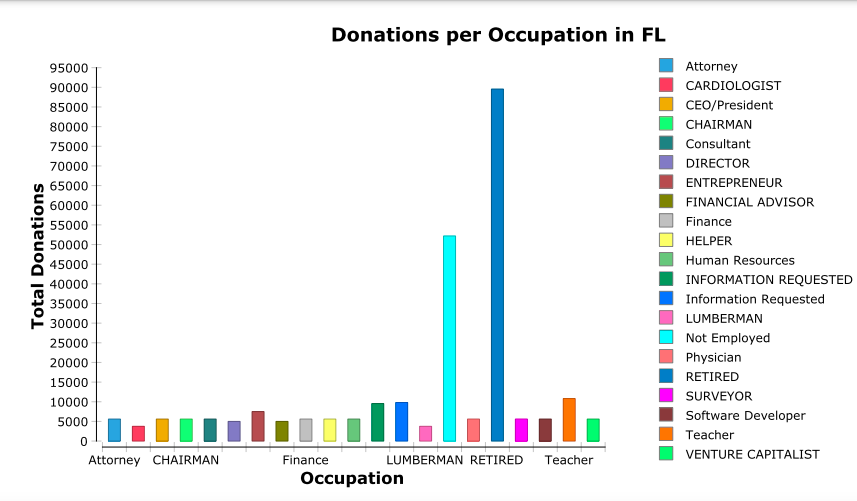


Figure : Committee Results for CA and FL

Figure : Occupation and City Results for NY

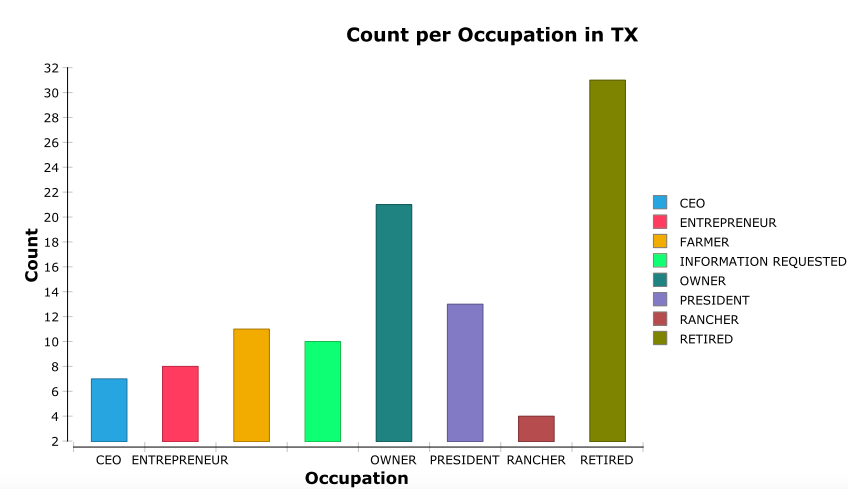
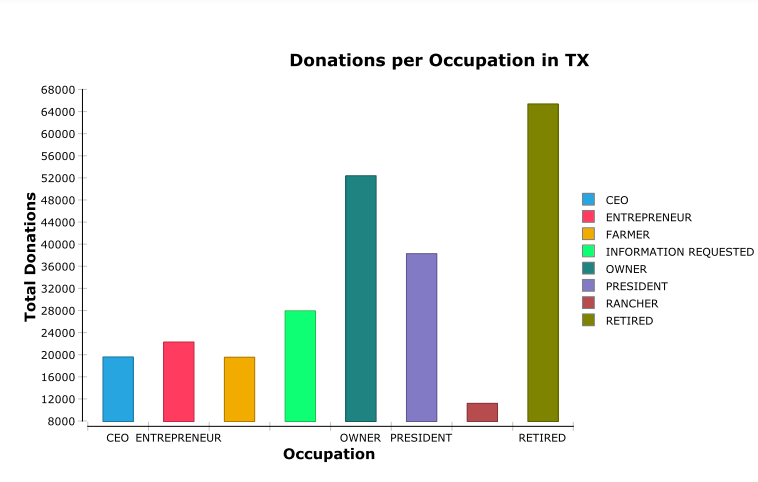
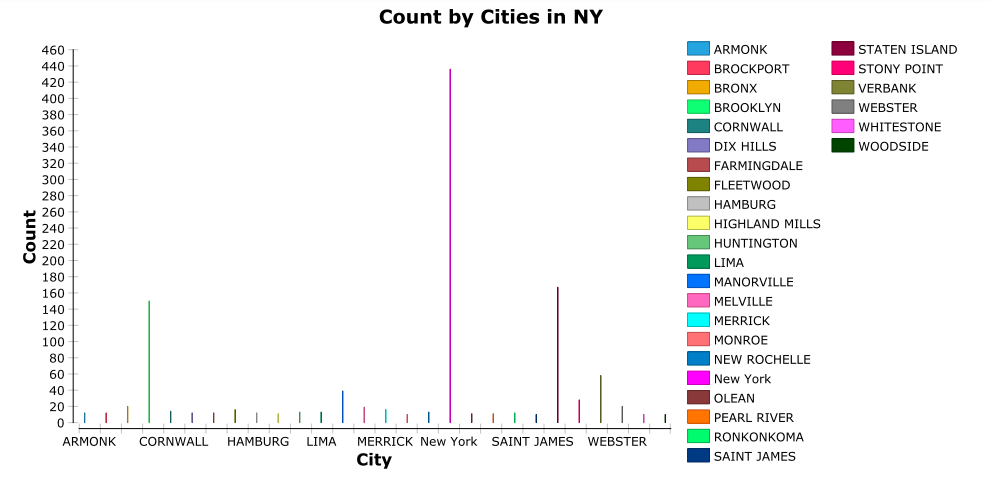
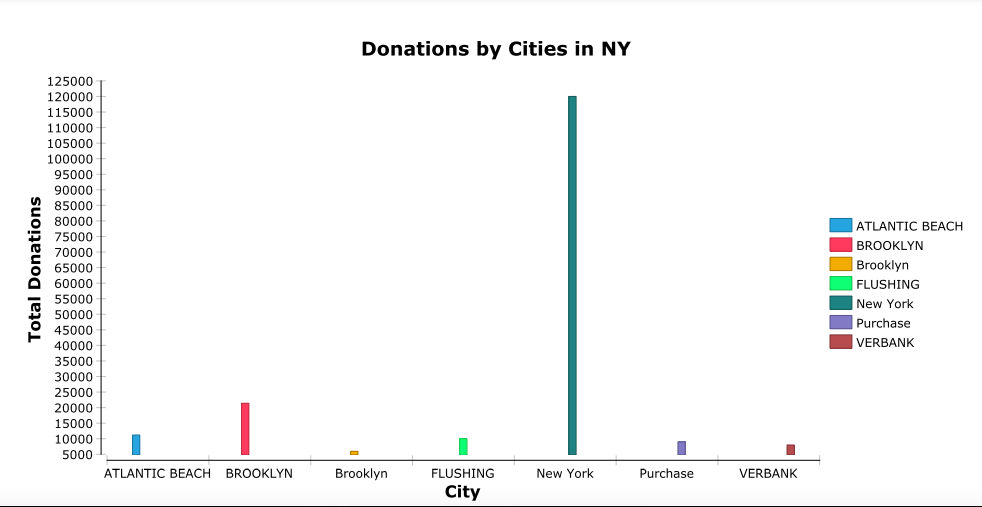
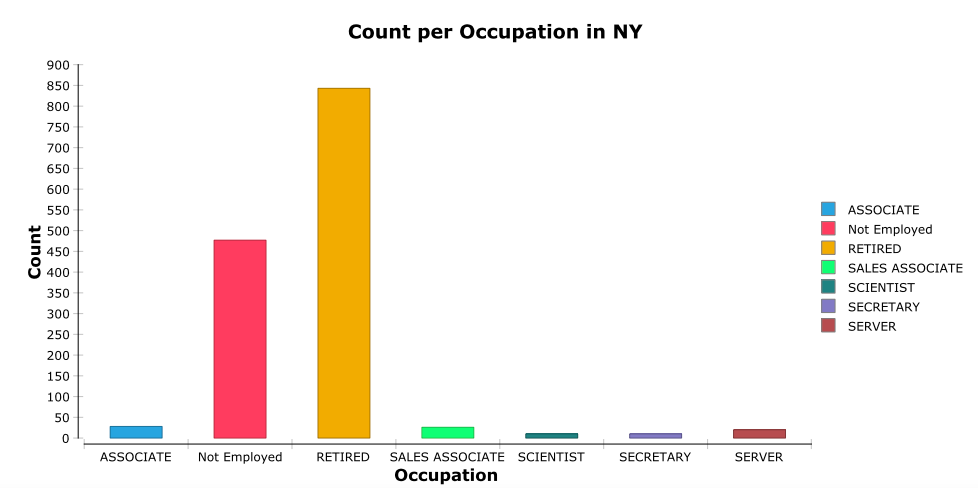
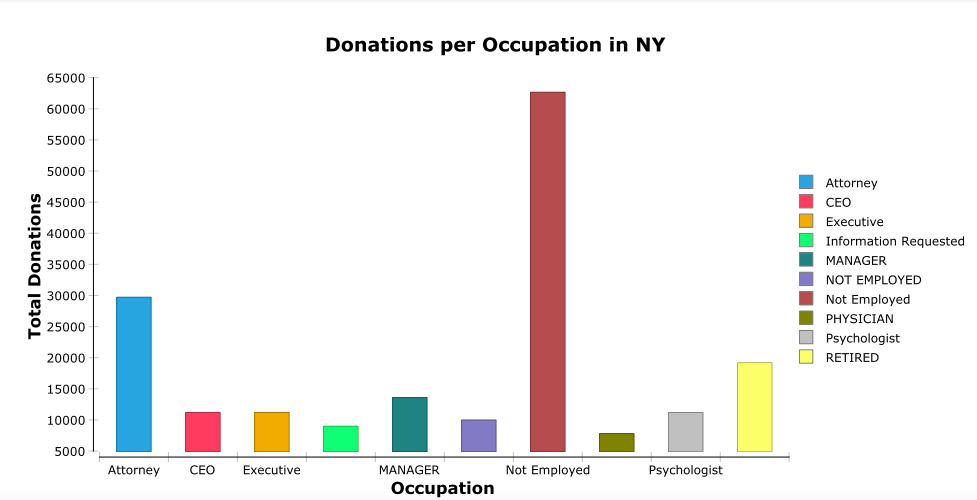
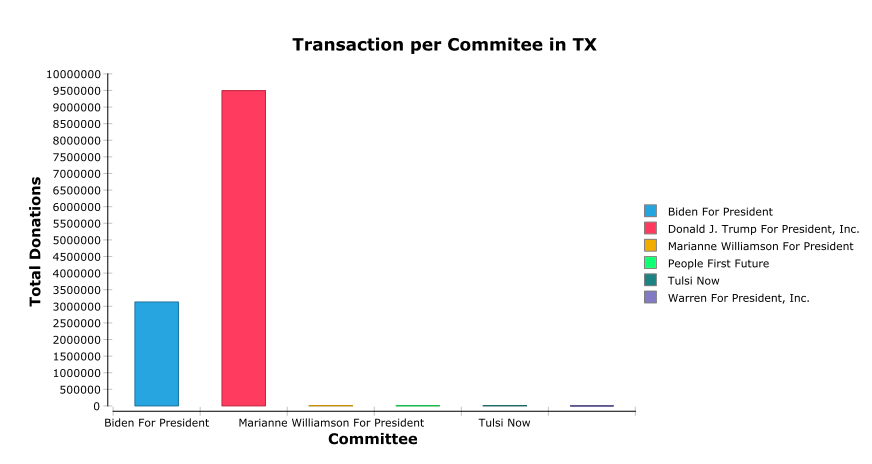
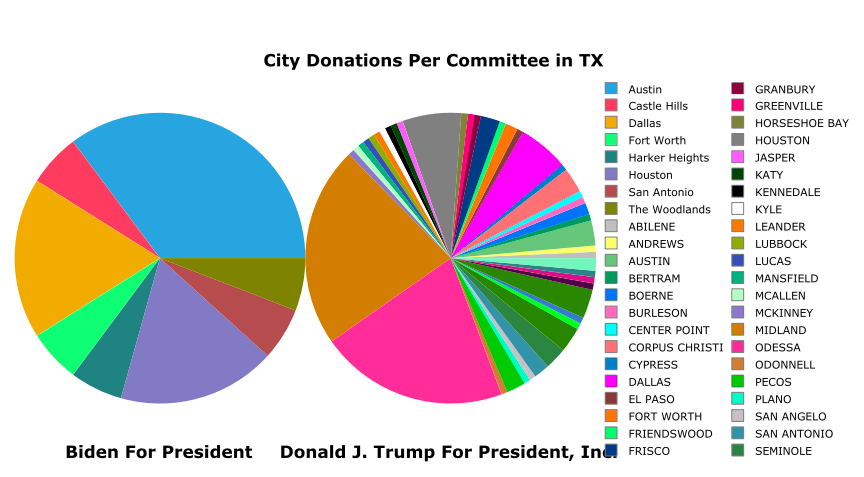
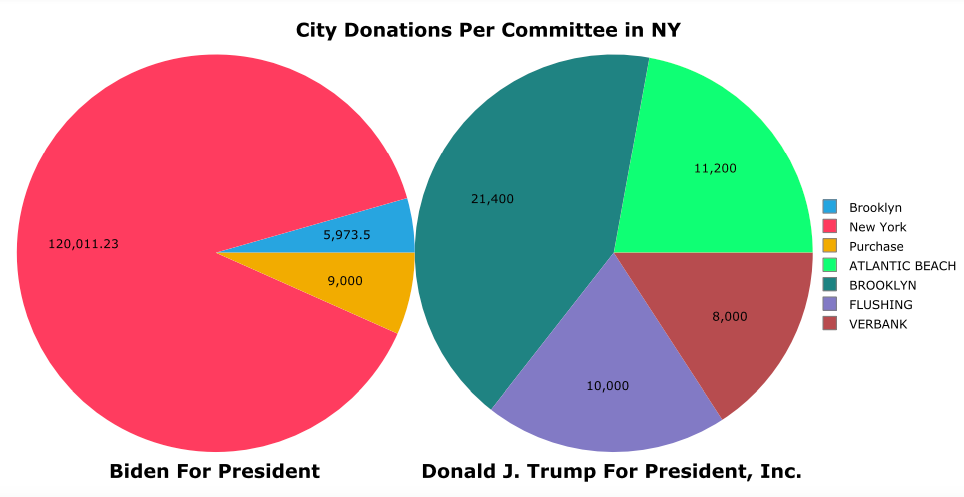
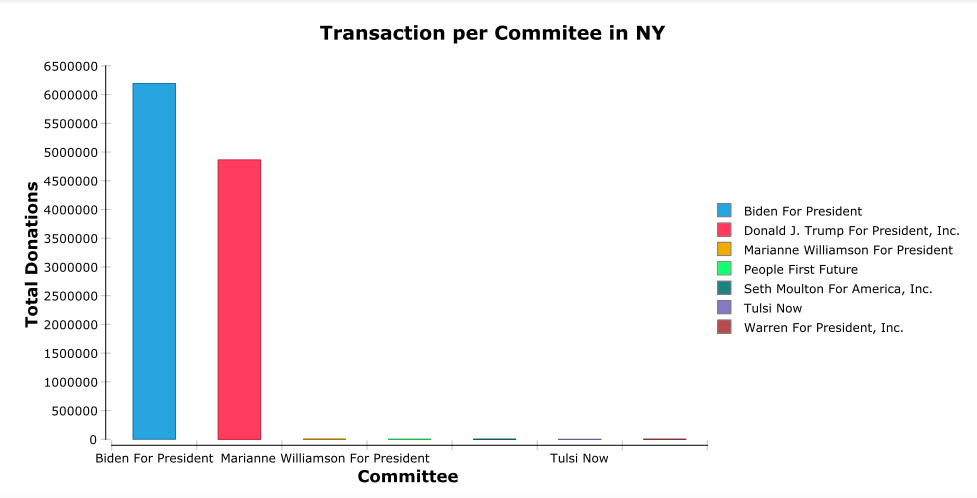


Figure : Occupation and City Results for TX

Figure : Committee Results for NY and TX



## 4.2 Hadoop Implementation

The Hive Database information was imported to Knime through a Hive database connector and database query reader node. From the query reader node, the data was imported into the reporting node in order to create the reports. The Knime workflow is provided in figure 16.

A picture containing scatter chart

Description automatically generated

Figure : Knime Reporting System Utilizing Hive Data

The reporting tools were utilized to create graphs from the Hive database data to reveal trends in order to answer the business questions. The first business question to be answered by using the Hadoop ecosystem through the Apache Hive warehouse was to find the top three occupations that contributed the most to each committee. It was decided to observe the trends for each occupation per state for the total sum of donations and total number of donations (count). From these results, not employed was the top donating occupation based on total donations as shown in table 12 based on figures 12, 13, 15, and 16. The top donating occupation based on how many times a donation was made for all states are shown in table 13 based on figures 17, 18, 19, and 20.

Table 12: Top Three Contributing Occupation Per State by Total Donations

|  |  |  |  |
| --- | --- | --- | --- |
| State | 1st Occupation | 2nd Occupation | 3rd Occupation |
|
| CA | Not Employed | Presidential Candidate | Retired |
| FL | Not Employed | Retired | Attorney |
| NY | Not Employed | Attorney | Retired |
| TX | Not Employed | Attorney | Retired |

The top 3 occupation that contributed the most to the election per state by total count per occupation are summarized below:

Table 13: Top Three Contributing Occupation per State by Total Count

|  |  |  |  |
| --- | --- | --- | --- |
| State | 1st Occupation | 2nd Occupation | 3rd Occupation |
|
| CA | Not Employed | Retired | Attorney |
| FL | Not Employed | Retired | Attorney |
| NY | Not Employed | Attorney | Retired |
| TX | Not Employed | Attorney | Retired |

The next business question to be answered was to find the top three cities that donated the most times from each state. The top three cities that donated the most times per state are summarized in table 14 based on figures 17, 18, 19 and 20.

Table 14: Top Three Contributing Cities per State by Total Count

|  |  |  |  |
| --- | --- | --- | --- |
| State | 1st City | 2nd City | 3rd City |
|
| CA | Los Angeles | San Francisco | San Diego |
| FL | Miami | Tampa | Naples |
| NY | New York | Brooklyn | Bronx |
| TX | Austin | Houston | Dallas |

The next business question to be answered was to find the top three cities that donated the most overall from each state. The top three cities that donated the largest amount per state are summarized in table 15 based on figures 17, 18, 19, and 20:

Table 15: Top Three Contributing Cities Per State by Total Donations

|  |  |  |  |
| --- | --- | --- | --- |
| State | 1st City | 2nd City | 3rd City |
|
| CA | San Francisco | Los Angeles | Berkeley |
| FL | Miami | Sarasota | Naples |
| NY | New York | Brooklyn | Scarsdale |
| TX | Austin | Houston | Dallas |

The final business question was to determine which cities donated the most to the top two committees to be able to identify trends of what political party the cities would be expected to donate to in the future elections. As shown in figures 21 and 22, the committees that received the most contributions in each state were Biden and Bernie for President. The cities that contributed most to these committees per state are shown below in tables 16 and 17:

Table 16: Top Three Cities that Contributed to Biden for President

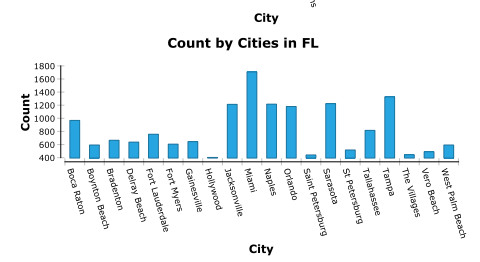
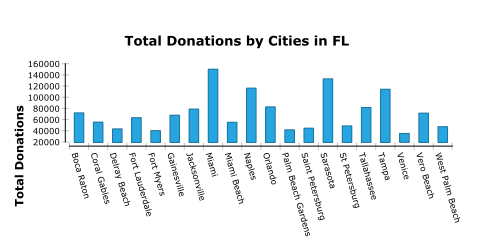
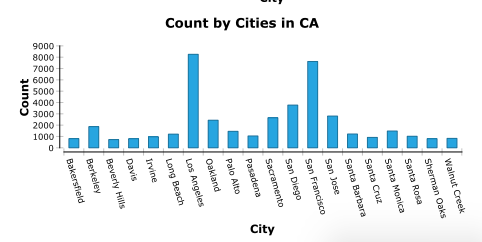
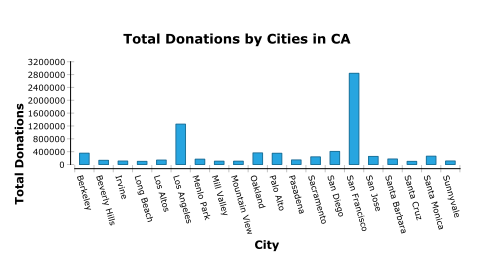
|  |  |  |  |
| --- | --- | --- | --- |
| State | 1st City | 2nd City | 3rd City |
| CA | Los Angles | Alberton | San Francisco |
| FL | Miami | Tampa | Naples |
| NY | New York | Brooklyn | Old Westbury |
| TX | Dallas | Austin | Fort Worth |

Table 17:Top Three Cities that Contributed to Bernie for President

|  |  |  |  |
| --- | --- | --- | --- |
| State | 1st City | 2nd City | 3rd City |
| CA | Los Angles | San Francisco | San Deigo |
| FL | Pennsacola | Coral Gables | Ft. Lauderdale |
| NY | New York | Brooklyn | Old Westbury |
| TX | Dallas | Austin | Houston |

It should be noted that the results are dependent on a filter that was determined for each state. All states were filtered for top 10 results except for the city donations per committee as it oversimplified the data. It should be noted that the filtered data reduces the available trends.

Figure : Total Donations and Counts by Cities per States of FL and CA



f

Figure : Total Donations and Counts by Occupations per States of FL and CA

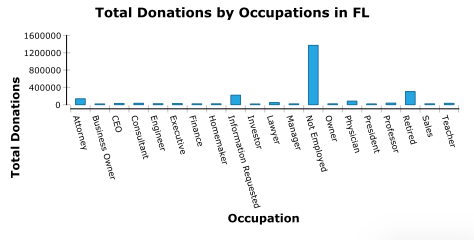
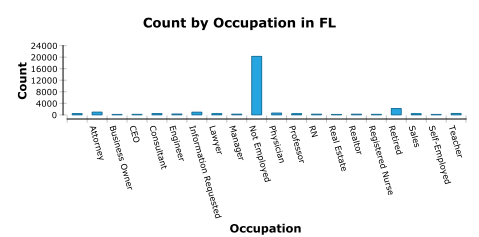
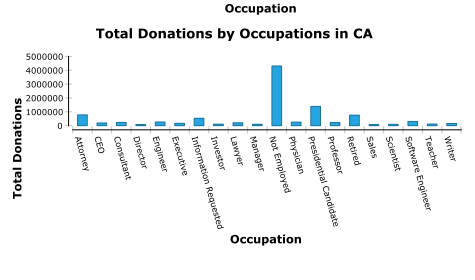
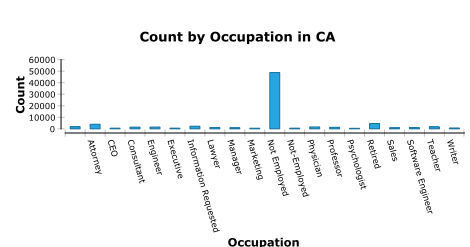


Figure :Total Donations and Counts by Cities per States NY and TX

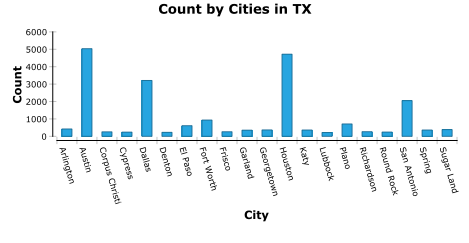
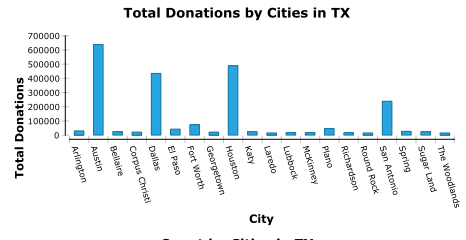
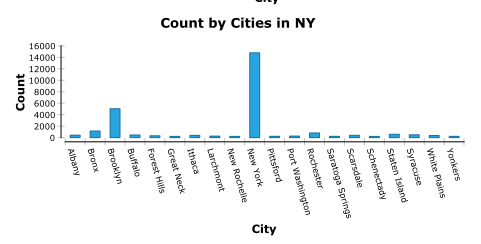
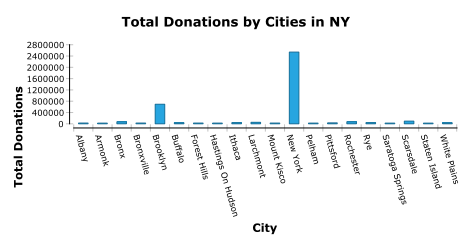


Figure : Total Donations and Counts by Occupations per States

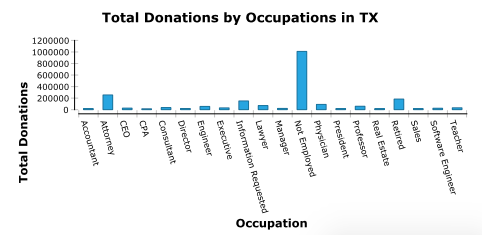
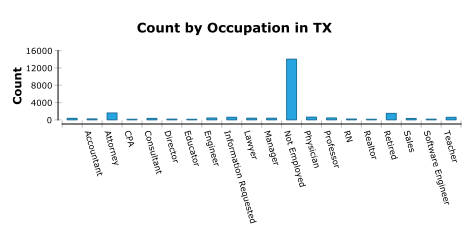
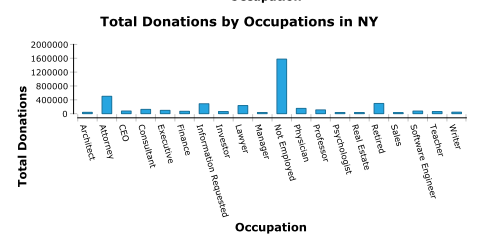
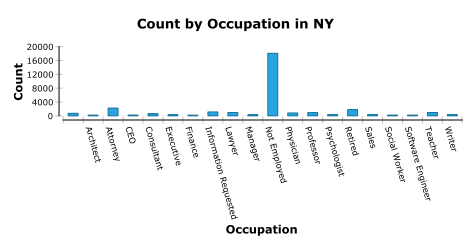


Figure : Transactions per Committee per State

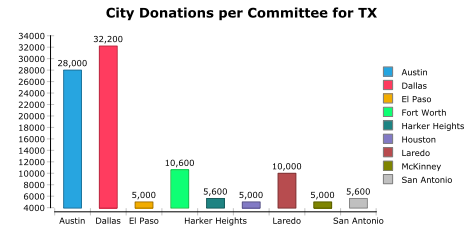
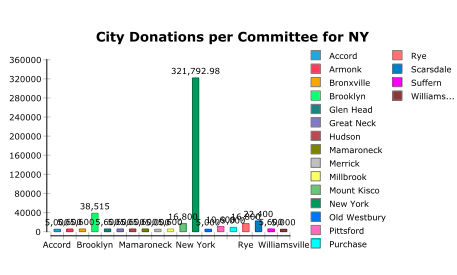
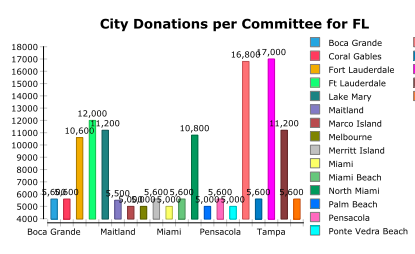
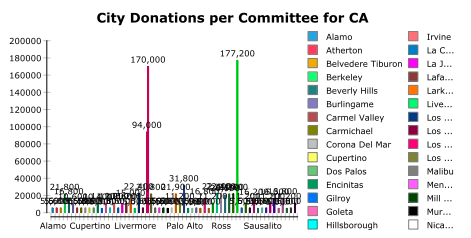
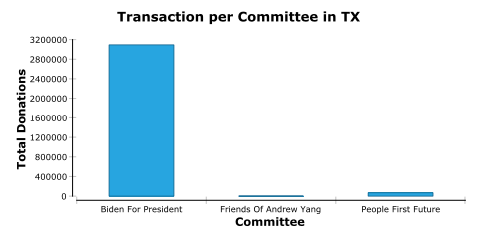
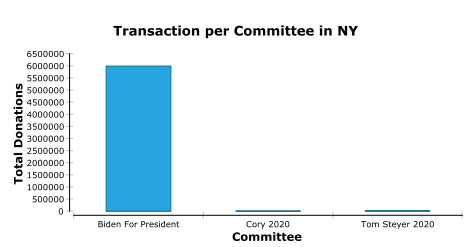
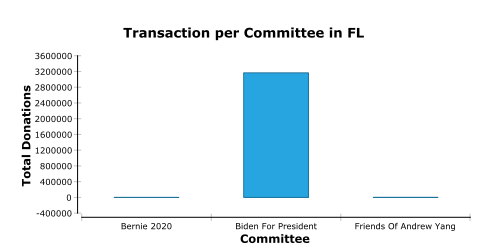
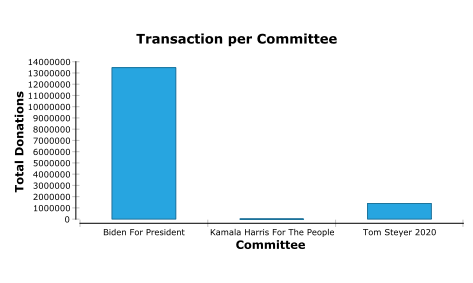


Figure : Transactions per Committee per City

## 4.2 Reflective Analysis of Results in Relational Data Warehouse Vs. Hadoop

When comparing the analysis results of the relational data warehouse compared to the use of the Hadoop open-source system, the Hadoop ecosystem was the preferable model. While filtering was still required for the Hadoop model, the filtering results were the top 10 cities or occupations to allow for readable graphs compared to the traditional data warehouse results where the donation amounts below $5000 had to be ignored to have readable, meaningful data. Additionally, the Hadoop system utilized data cubes to aggregate the count and summations of the contribution amounts, whereas the traditional data warehouse had to have these parameters manually aggregated for each excel file through the use of Knime during the ETL process. Overall, considering that the data for Donald J Trump for President, Inc. was not present in the Hadoop data, the results appeared to be more consistent in the plots when compared to the traditional data warehouse. Direct comparisons are not possible due to the difference in filtering the results as well as missing the data for a prime candidate. The Hadoop plots were overall cleaner in appearance and provided more meaningful, consistent data by being able to sort by the top 10 results for either the city or occupation. Trends are able to be analyzed easier utilizing the data cubes created from the Hive data warehouse. When considering time and costs, it would be expected that Hadoop would win in both of these categories when compared to the traditional data warehouse, and in this case, Hadoop proved to be more efficient. Additionally, Hadoop is able to use unstructured data and has high scalability. Hadoop also was able to eliminate redundant data such as naming conventions within the MapReduce ETL process. This removed noisy data that was present in the traditional data warehouse reporting process, which impacted the data negatively. Overall, Hadoop handled the data more efficiently and provided better results than the traditional data warehouse.

Conclusions

The traditional data warehouse was arranged in a star schema, structured format ensuring the data was consistent through the ETL process. The traditional data warehouse proved time consuming during the ETL process and reinforced that real-time loading, if later necessary, would be impractical due to this delay. Once the ETL process was finalized, it allowed efficient data processing and loading into the data warehouse, which eased the process of combing multiple data files. One of the issues with the ETL processes was that due to the lack of knowledge on how to extract and combine case sensitive data, mismatched data was allowed to enter the data warehouse. When arriving to the reporting analysis, it was apparent that there were multiple records that should have been grouped together that were instead counted separately. There were several occupation and city names that could have been classified as the same record such as “retired teacher” could have been lumped with “retired.” Since data such as this was not preprocessed correctly, the derived trend results are expected to vary from the real trend results. Furthermore, the reporting tool was not able to plot the desired results in the desired manner due to the amount of data. Because of this, the analysis of each state had to be performed separately and even then, did not provide adequate visualization unless the data was further filtered. Further filtering the data resulted in large portions of data being removed, thus removing potential trends that were not detected. The filtered data for the states of CA, FL, NY, and TX over the months of May, June, and July of 2020 all resulted in the retired occupation making more donations as a whole but was split with the unemployed occupation when comparing total donations. This suggests that had the data not been filtered, perhaps a different trend would be observed in either or both of those outcomes especially if the preprocessed data had combined records consistently. Similarly, the top three city contribution amounts per candidate per state may be altered if all donations were considered rather than only considering donations that were greater than $5,000.

In comparison, the Hadoop architecture utilizing the Apache Hive data warehouse schema was a simpler process that allowed for the creation of only one table. Additionally, the ETL process utilizing MapReduce was more efficient as all files were able to be transformed simultaneously where the output was loaded into the Apache Hive data warehouse. The transformation performed within MapReduce did remove the previous issues with the naming conventions of the data that was missed in the traditional ETL process, however; there again existed a lack of knowledge within the ETL process as all records for the committee for Donald J Trump for President, INC were extracted from the data and not included in the final analysis. Furthermore, the results were narrowed within the reporting system to only show the top 20 results for the occupation and city inputs when comparing total donations and count. This was performed differently than the reporting for the traditional data warehouse due to the use of data cubes. A direct comparison of the results cannot be made as the traditional data warehouse results were filtered by excluding donations that were not greater than $5000 as mentioned previously. This filtering process has the potential to skew the results as there may be inaccurate representations for the cities and occupations for each state.

By excluding the issues during the ETL phase, the data warehouse was well suited for the provided data due to the data size and historical nature. The historical nature of the data allowed for the quick discovery of trends within the data. Furthermore, the data warehouse would also allow for efficient further trend analysis that were not evaluated such as discovering which occupations donated the most to each committee, how the contributions changed per committee over a desired time frame based on either the occupation or city donations, and many other possibilities. Nonetheless, the traditional data warehouse is a longstanding standard framework that offers advantages in processing and retrieving data (Barb, 2021). The data warehouse achieved high normalization as each record was atomic and each dimension had one primary key that defined every attribute in the table. The high level of normalization in the dimensions ensures the data is consistent throughout, which allows the user to retrieve combined data from multiple sources efficiently with repeatable results. However, if the business

questions or rules were to change, the data warehouse proves difficult and costly if changes are required in the grain, schema, and data types as it is inflexible and subject orientated.

In conclusion, while the traditional data warehouse did prove to handle the historical data well, Hadoop proved to be more efficient and provide more accurate results. Overall, the process of Hadoop handled the data better in the ETL process and was able to run all 12 files at once compared to the individual uploading and run process in Knime. The reporting time for the Hadoop and traditional database results correlated to the same time, but Hadoop provided more consistent results when comparing the output table. Hadoop did fail in the ETL phase due to the fact that the Donald J Trump for President, INC. data was filtered out due to a lack of knowledge within the java coding of the MapIt phase for the MapReduce system. Nonetheless, Hadoop proved to be the more desired solution between the two processes taking into consideration that there would be an improved decision making process and cost reduction due to the data efficiency.

Appendix

Table : Contributor Information Dimension Architecture

|  |  |  |  |
| --- | --- | --- | --- |
| ***Name of the table*** | contributorinformation |  |  |
| **Description** | This table describes the occupation of the contributors. | | |
| **Attribute** | **Description** | **Type** | Examples of values |
| **id** | Synthetic Key for contributors | Serial | Between 1 and 999999999 |
| **occupation** | Name of occupation | String | Engineer |
| **Primary Key** | id | | |
| **Candidate Keys** (if any) | N/A | | |
| **Foreign Keys** | N/A | | |

Table : Addresses Dimension Architecture

|  |  |  |  |
| --- | --- | --- | --- |
| ***Name of the table*** | addresses |  |  |
| **Description** | This table describes the addresses of the contributors excluding the street. | | |
| **Attribute** | **Description** | **Type** | Examples of values |
| **id** | Synthetic Key for addresses | Serial | Between 1 and 999999999 |
| **city** | City contributor resides | String | Austin |
| **state** | State contributor resides | String | TX |
| **zipcode** | Zip code contributor resides | Integer | 44333 |
| **Primary Key** | id | | |
| **Candidate Keys** (if any) | N/A | | |
| **Foreign Keys** | N/A | | |

Table : Commitee Dimension Architecture

|  |  |  |  |
| --- | --- | --- | --- |
| ***Name of the table*** | committee |  |  |
| **Description** | This table describes the committees that received donations. | | |
| **Attribute** | **Description** | **Type** | Examples of values |
| **id** | Synthetic Key for committees | Serial | Between 1 and 999999999 |
| **committee** | Committee name | String | Biden for President |
| **Primary Key** | id | | |
| **Candidate Keys** (if any) | N/A | | |
| **Foreign Keys** | N/A | | |

Table : Contribution Date Dimension Architecture

|  |  |  |  |
| --- | --- | --- | --- |
| ***Name of the table*** | contributiondate |  |  |
| **Description** | This table describes the date the transaction was made. | | |
| **Attribute** | **Description** | **Type** | Examples of values |
| **id** | Synthetic Key for contribution date | Serial | Between 1 and 999999999 |
| **contributiondate** | Date transaction was made | Date | 2020/06/14 |
| **Primary Key** | id | | |
| **Candidate Keys** (if any) | N/A | | |
| **Foreign Keys** | N/A | | |

Table : Fact Table Architecture

|  |  |  |  |
| --- | --- | --- | --- |
| ***Name of the table*** | fact |  |  |
| **Description** | This table describes the facts that can be found from the other tables. | | |
| **Attribute** | **Description** | **Type** | Examples of values |
| **contributiondateid** | Synthetic Key for id from contributiondate dimension | Serial | Between 1 and 999999999 |
| **committeeid** | Synthetic Key for id from committee dimension | Serial | Between 1 and 999999999 |
| **contibutorinformationid** | Synthetic Key for contribution information from contributorinformation dimension | Serial | Between 1 and 999999999 |
| **addressesid** | Synthetic Key for addresses from addresses dimension | Serial | Between 1 and 999999999 |
| **sum** | Aggregated summation of transaction amounts | Integer | Between 1 and 999999999 |
| **count** | Aggregated count of transactions made | Integer | Between 1 and 999999999 |
| **Primary Key** | **(contributiondate, committeeid, contributorinformationid, addressesid)** | | |
| **Candidate Keys** (if any) | N/A | | |
| **Foreign Keys** | Contributiondate, committeeid, contirbutorinformationid, addressesid | | |

Graphical user interface, application

Description automatically generated

Figure : Foreign Key Addition in Fact Table

A picture containing text, screenshot, black

Description automatically generated

Figure 2: Example of Serial Primary Keys in Committee Dimension

A picture containing text

Description automatically generated

Figure : Hive Database