Penn State University
Great Valley Campus

Engineering Division

Data Specification for
Texas, New York, and California Election Contributions

Version 1.4

03/01/2024

Modified from template at www.dpw.state.pa.us

Table of Contents

Data Specification for Texas, New York, and California Election Contributions
Afolabi Isiaka

INTRODUCTION

Requirements definition and decomposition are the important first steps in any systems development project. This document provides the recommended structure in which the project scope and requirements for data warehouse and knowledge management development projects are to be defined and documented.

PURPOSE

This document is a template for a project specification document for data warehouse and knowledge management development projects.

PROJECT SUMMARY

This paragraph is used to introduce the following subsections, which can be used for an executive level overview.

A. Objectives

To analyze the effectiveness of data warehouses compared to Hadoop in handling large datasets for election contribution analysis.

B. Scope

This document covers the requirements and architecture for a data warehouse designed to store and analyze election contribution data, excluding the comparison with Hadoop, which will be addressed in a subsequent project.

C. References

Browse individual contributions. FEC.gov. (n.d.).

https://www.fec.gov/data/receipts/individual-contributions/?two_year_transaction_period=2020&min_date=01%2F01%2F2019&max_date=12%2F31%2F2020

Derek Willis, S. W. (2015, August 12). *FEC itemizer*. ProPublica. https://projects.propublica.org/itemizer/presidential-contributors/2020

https://psu.instructure.com/courses/2308718/modules/items/39858246

HDFS architecture guide. Hadoop. (n.d.). https://hadoop.apache.org/docs/r1.2.1/hdfs_design.html#Introduction

D. Outstanding Issues

Determining the optimal schema for the star database model and identifying the most efficient ETL process for real-time data analysis remain outstanding.

REQUIREMENTS DEFINITION

A. Goals

The new data warehouse application aims to provide comprehensive analysis of election contributions, enhance data retrieval speed, and maintain data integrity. It must retain the ability to track contribution trends over time and ensure data accuracy.

B. Usability Requirements

The system should offer a user-friendly interface, allowing users to perform complex queries with ease. It must include customizable report layouts, detailed online help, and intuitive navigation to accommodate various user expertise levels.

C. System Security Requirements

The data warehouse will handle sensitive election contribution data, requiring high-level security measures including encryption, multi-factor authentication, and role-based access controls to safeguard data privacy and integrity.

D. Business Questions

Primary questions include analyzing trends in election contributions, identifying major contributors by demographics, and predicting future contribution patterns. Prioritization is based on stakeholder interest and data availability.

E. Data Requirements

Required data elements include contributor names, contribution amounts, dates, and recipient details. The system should support data archiving and be capable of handling large volumes of data, with a detailed data model and dictionary.

F. Design Constraints

The design must consider compatibility with existing IT infrastructure, scalability to manage data growth, and adherence to election finance laws and data protection regulations.

CONSIDERATIONS

(May include as separate document)

DOCUMENT CHANGE LOG

Change Date	**Version**	CR #	Change Description	Author and Organization
09/22/04	1.0		Initial creation.	Lawrence Leitzel
03/01/24	1.1		Project Summary, Data Dictionary	
03/04/24	1.2		Database Schema and ER diagram	
03/08/24	1.3		ETL diagram and process	
03/10/24	1.4		Reporting Data using knime	
03/23/2024	1.5		Hadoop Implementation- Archeticture Design	
04/19/2024	1.6		Apache Pig Implementation	

2. Architecture Design

2.1 Relational Data Warehouse

The Relational Data Warehouse we wil be using is PostgreSQL. I will be separating my attributes to reflect a dimensional model. When accessing our data we will use Foreign Keys and Primary keys that are related to our dimensions.

Select the business process: Identifying the dimensions is based on the business needs of our organization. For this project I choose to focus on the job information, committees, location and time(dates).

Declare the grain: Each dimension table will be linked to our Fact table using unique identification in the form of Primary Keys and Foreign Keys

Identify the dimensions: Information regarding the location of the donation can be found by the relationship between our tables. For example, we might want to discover which states donate the largest on average? A PostgreSQL query can give us that answer.

Identify the facts: In our facts stable we chose to store our primary keys for each of our dimensions due to the unique properties of primary keys. Doing so means us to access each attribute and the amount of contributions made.

Data Dictionary

Data dictionary for the tables we will be transforming are below:

Contributions Table

Entity	**Description**
filing_id	Identifier for the filing.
linenumber	Line number in the filing document.
flag_orgind	Flag indicating if the contribution is from an individual (IND) or an organization.
org_name	Name of the organization (if applicable).
last_name	Last name of the contributor (if individual).
first_name	First name of the contributor (if individual).
middle_name	Middle name of the contributor (if individual).
prefix	Prefix to the contributor's name (if applicable).
suffix	Suffix to the contributor's name (if applicable).
address_one	Primary address line of the contributor.
address_two	Secondary address line of the contributor (if applicable).
city	City of the contributor's address.
state	State of the contributor's address.
zip	ZIP code of the contributor's address.
employer	Employer of the contributor (if individual).
occupation	Occupation of the contributor (if individual).
amount	Amount of the contribution.
date	Date of the contribution.
aggregate_amount	Aggregate amount contributed by the individual/organization for a specific period or campaign.
memo_code	Code indicating if the entry is a memo entry (details not directly related to a contribution amount).
memo_text	Text providing additional details on the contribution or memo entry.
tran_id	Transaction ID for the contribution.
back_ref_tran_id	Back reference transaction ID, linking related transactions.
back_ref_sched_name	Schedule name for the back-reference transaction.
prigen	Indicates the primary/general election context.
cycle	Election cycle.
fecid	Federal Election Commission ID of the recipient committee.
committee_name	Name of the committee receiving the contribution.

Tables schemas

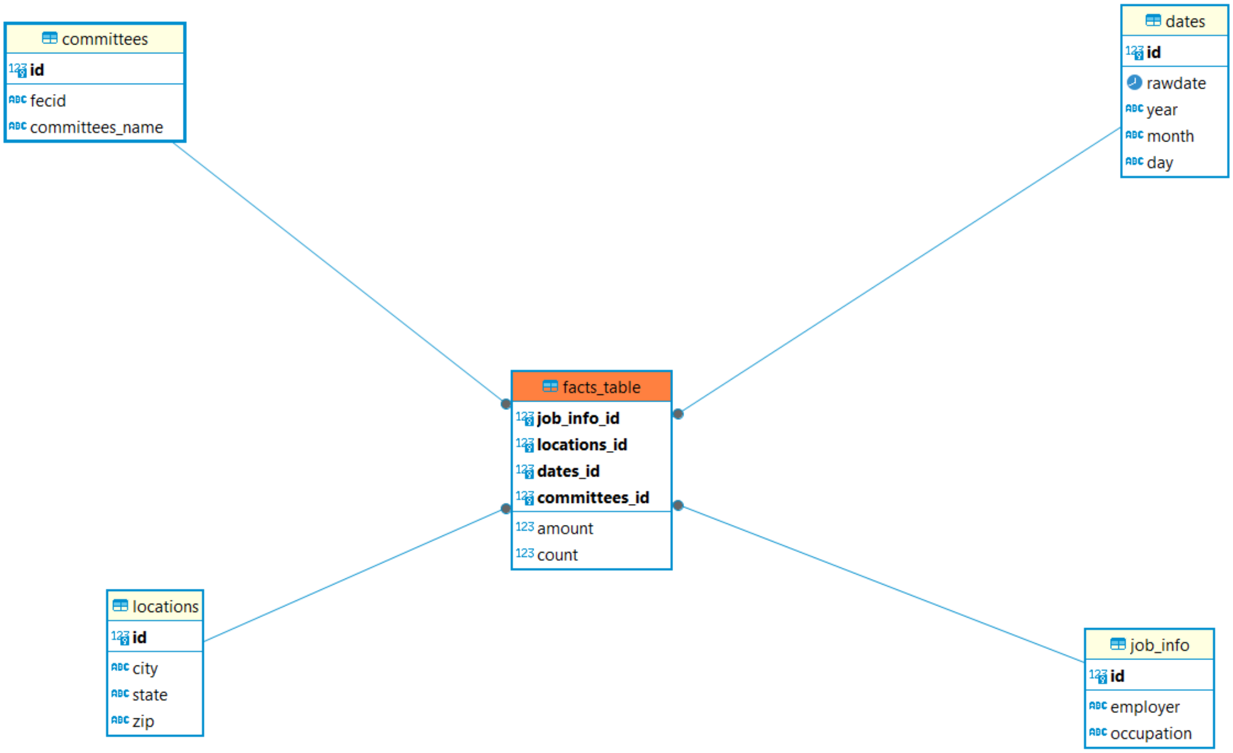
Locations			
Description	Location details of employee		
Attribute	**Description**	**Type**	***Examples of values***
Id	Id of a location	Integer	Between 1 and 999999999
city	Name of location	String	State College
state	State name	String	TX
Zip	Zip Code	Int	16808
Primary Key	id		
Foreign Keys			

Job_info			
Description	Location details of employee		
Attribute	**Description**	**Type**	***Examples of values***
id	Id of a location	Integer	Betweelocatn 1 and 999999999
Employer	Name of employer	String	Wegmans
occupation	Occupation name	String	Engineer
Primary Key	id		
Foreign Keys			

Committees			
Description	Committees that contributed.		
Attribute	**Description**	**Type**	***Examples of values***
id	Id of an committees	Integer	Between 1 and 999999999
fecid	FEC id for committees	String	C00700906
Committees_name	Name of committees	String	Gillibrand 2020
Primary Key	id		
Foreign Keys			

Facts	
Description	Facts Table

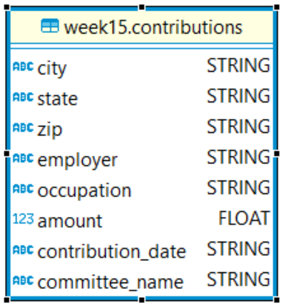
Attribute	Description	Type	***Examples of values***
date_id	Id of donation date	Integer	Between 1 and 999999999
Name_id	Id of a name	Integer	Between 1 and 999999999
Location_Id	Id of a location	Integer	Between 1 and 999999999
Job_info_id	Id of job info	Integer	Between 1 and 999999999
Committee_id	Id of committees	Integer	Between 1 and 999999999
Amount	Amount contributed	Decimal	20.00
Count	Frequency of contributions	interger	5
Primary Key	Combination of job_info_id, locations_id, names_id, dates_id, committees_id		
Foreign Keys	References each id from previous tables: locations, job_info, dates, and committees.		



2.2 Hadoop Implementation

Storing massive data on one computer is likely to be faulty. If the file system on one computer works. There is still a risk associated with our data being contained in one system. The relational data warehouse storage can be distributed into separate systems hence is why the Hadoop Distributed File System (HDFS) is an upgrade to our storage system. HDFS is designed for large amounts of data storage, being it was designed to mimic the Google File storage system.

Where Knime can filter the dataset using various nodes, Apache Pig allows programmers to write complex MapReduce transformations using a simple scripting language called Pig Latin. Apache Pig can be used instead for data cleaning and transformation. After this is done all that is left is to create a table for our database in HiveQL. This database table must remain consistent with our datatypes stated in Pig Latin.



week15.contributions

ABC	city	STRING
ABC	state	STRING
ABC	zip	STRING
ABC	employer	STRING
ABC	occupation	STRING
123	amount	FLOAT
ABC	contribution_date	STRING
ABC	committee_name	STRING

2.3 Reflective analysis of using a data warehouse vs Hadoop.

Our PostgreSQL warehouse used a dimensional model. The dimensional model is composed of one central fact table that has a multi-attribute key and several attributes containing usable data. The main advantage of star schema setup is the retrieval of data with greater security. The database design makes it so that we can get information about anything without having to pull up information that we do not need. We also know where the information is because of our primary keys. However, relying on primary keys could be hazardous if our database gets compromised. Losing one piece of information in our facts table could damage our other dimension tables.

Hadoop database does not use a dimensional model and is straight forward. Hadoop can handle and query multiple rows in seconds and does a lot of the heavy lifting we needed to do in Knime. Hadoop is friendly for people who do not know java. Hadoop can map data wherever it is located on the cluster, and processing tools are readable available for querying and transformation.

3. Data Preparation

3.1 Relational Data Warehouse Implementation

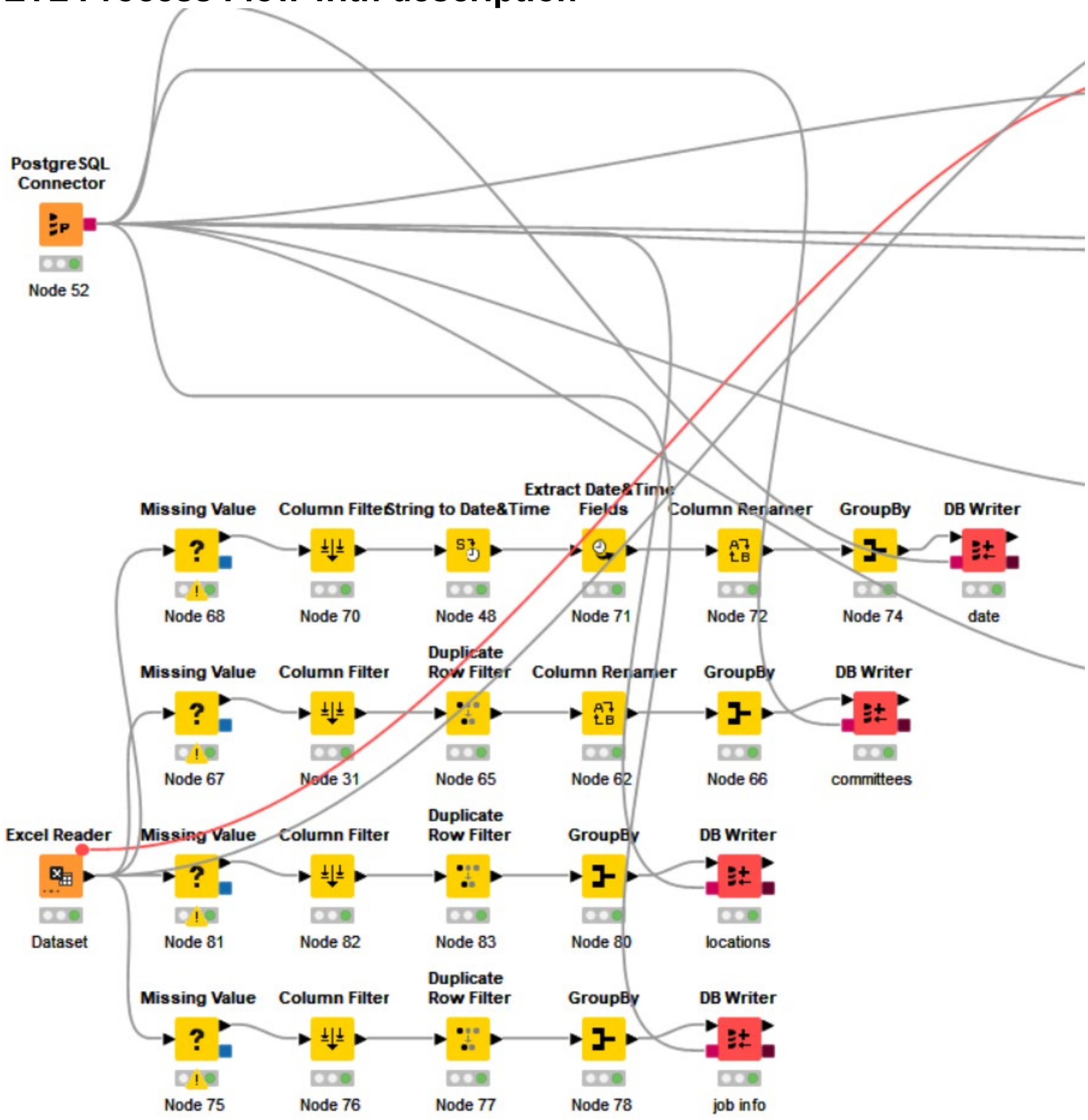
ETL considerations

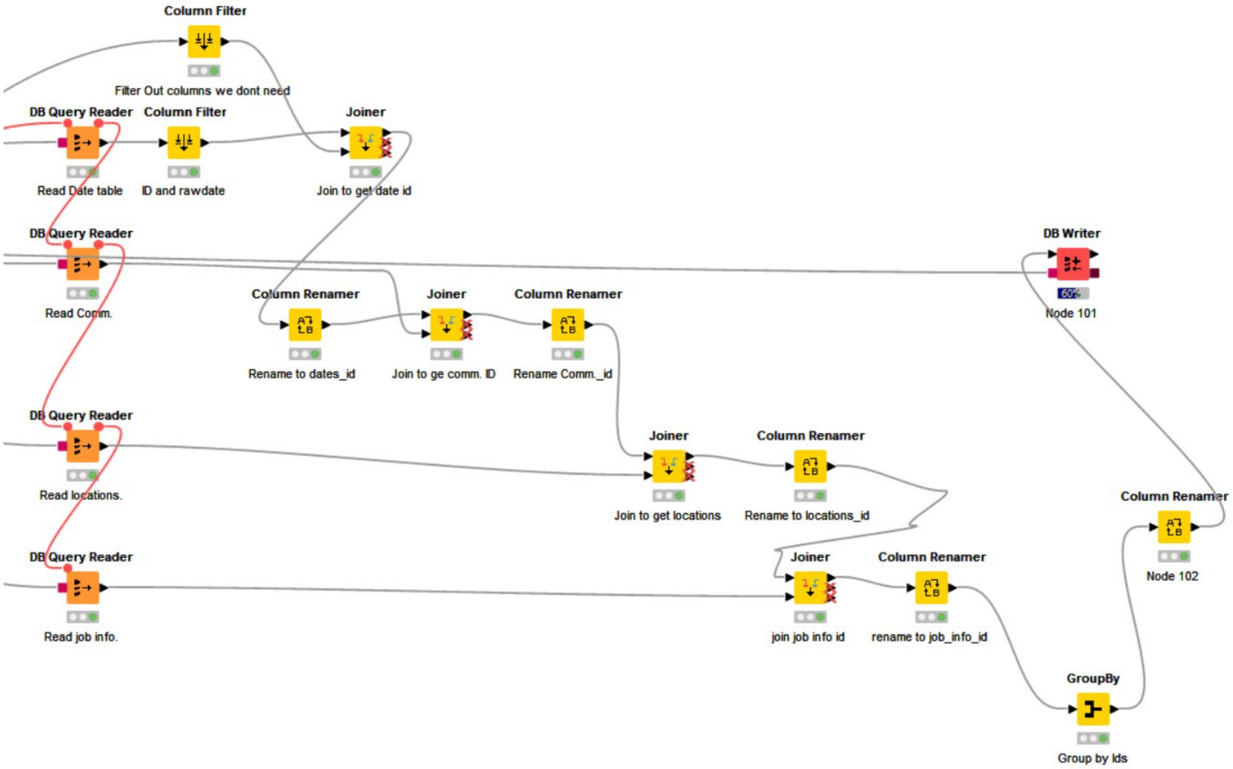
The ETL (Extract, Transform, Load) process involves several critical steps to ensure efficient data handling and integration into a data warehouse. Initially, data extraction focuses on gathering information from diverse sources, like Excel and database systems, employing tools such as KNIME Analytics Platform for efficient data retrieval. Data filtering emphasizes isolating relevant data columns through nodes like the Column Filter in KNIME, streamlining the transition into the Relational Data Warehouse (RDW).

Transformation processes, including data grouping by specific criteria and column renaming for schema consistency, are pivotal. Managing date variations and calculating new values from existing data enhance analysis capabilities. The loading phase entails selecting an appropriate strategy, full or incremental, and effectively writing the data to the RDW tables.

Integrating data from multiple sources and ensuring operation order maintains data integrity. The approach incorporates error handling, performance optimization, workflow automation, and rigorous validation to uphold data security and integrity, underlining the ETL process's significance in bolstering business intelligence and analytics.

ETL Process Flow with description





3.2 Hadoop Implementation

When designing an ETL the steps are like the ETL pipeline we made with Knime. Using a combination of docker and bash we can upload our data into our HDFS. Once we have our data we can use a querying tool called Hive. Hive is an analysis tool that supports ad-hoc queries and data summarization. Hadoop uses MapReduce java scripts to process data, whereas Hive uses a similar process to SQL. Hive tends to be preferred for individuals who do not know java code. In Hive tables are used that are like Relational database management systems—HiveQL.

```
#### code in pig####
-- Define the CSVLoader to read CSV files
DEFINE CSVLoader org.apache.pig.piggybank.storage.CSVLoader();

-- Load data from CSV files located at the specified path with the new schema
contributions = LOAD '/user/root/week15/input/*.csv' USING CSVLoader(',') AS (
   city: chararray,          -- City where the transaction took place
   state: chararray,         -- State where the transaction took place
   zip: chararray,           -- ZIP code
   employer: chararray,      -- Employer of the contributor
   occupation: chararray,    -- Occupation of the contributor
   amount: chararray,        -- Amount of the contribution
   contribution_date: chararray,         -- Date of the contribution
   committee_name: chararray, -- Name of the committee receiving the contribution
   year: chararray,          -- Year of the contribution
   month: chararray,         -- Month of the contribution
   day_of_month: chararray   -- Day of the month of the contribution
);



contributions = FOREACH contributions GENERATE
   REPLACE(city, '^$', 'Undeclared') AS city,
   REPLACE(state, '^$', 'Undeclared') AS state,
   REPLACE(zip, '^$', 'Undeclared') AS zip,
   REPLACE(employer, '^$', 'Undeclared') AS employer,
   REPLACE(occupation, '^$', 'Undeclared') AS occupation,
   (amount IS NULL OR amount == '' ? 'Undeclared' : amount) AS amount,
   REPLACE(contribution_date, '^$', 'Undeclared') AS contribution_date,
   REPLACE(committee_name, '^$', 'Undeclared') AS committee_name,
   REPLACE(year, '^$', 'Undeclared') AS year,
   REPLACE(month, '^$', 'Undeclared') AS month,
   REPLACE(day_of_month, '^$', 'Undeclared') AS day_of_month;

contributions = FOREACH contributions GENERATE
city,
REPLACE(state,',','') as state,
zip,
REPLACE(employer,',','') as employer,
REPLACE(occupation,',','') as occupation,
amount,
contribution_date,
REPLACE(committee_name,',','') as committee_name;



ranked_final_data = rank contributions;
final_data_first_rows = Filter ranked_final_data by ($0 < 10);
```

```
dump final_data_first_rows;

STORE contributions INTO '/user/root/week15/output' USING PigStorage (',');

#### code in pig end####

#### HiveQL####

CREATE DATABASE IF NOT EXISTS week15;SHOW DATABASES;USE week15;

CREATE TABLE contributions (
   city STRING,
   state STRING,
   zip STRING,
   employer STRING,
   occupation STRING,
   amount FLOAT,
   contribution_date STRING,  -- Use backticks to escape the keyword
   committee_name STRING
)
ROW FORMAT DELIMITED
FIELDS TERMINATED BY ','
STORED AS TEXTFILE
```

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

For our ETL into our PostgreSQL database we used Knime. Knime has too many things going on in the ETL, so it is difficult to discover which nodes are causing problems. In addition to this, Knime uses variable ports that control the order in which nodes are executed. Lastly, when handling huge rows of data Knime tends to be slow in the ETL loading process.

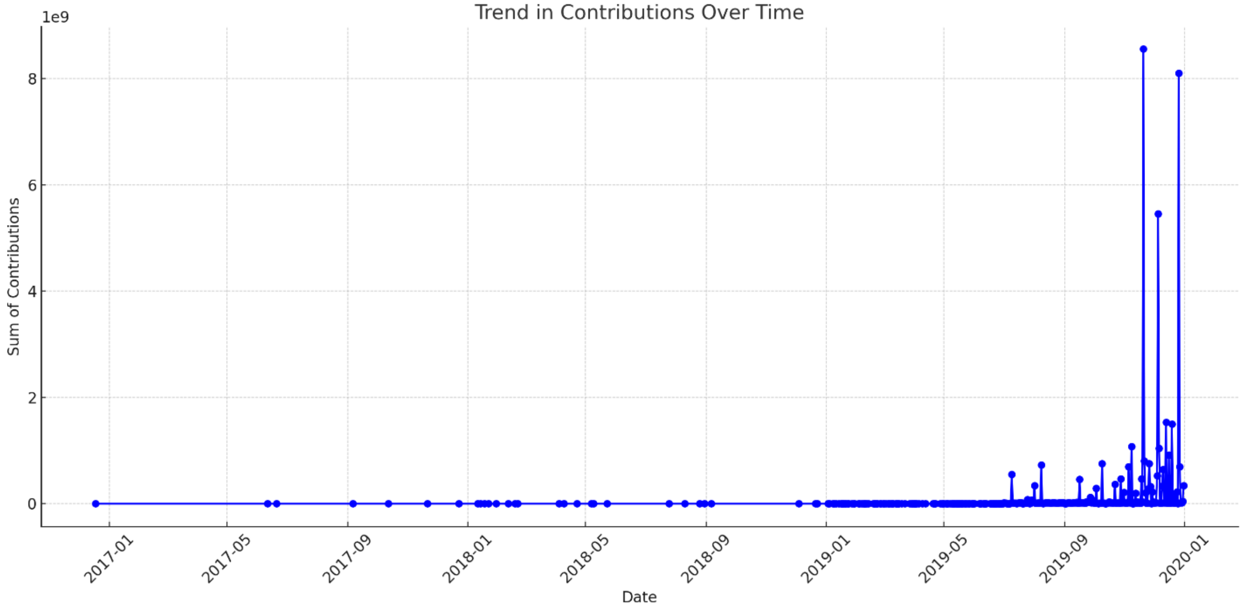
Hadoop has tools that make data make data extraction easier compared to a Relational Database warehouse. This also leads to a simpler design, hence meaning the information can be accessed more easily. In our traditional Datawarehouse we needed to use a star schema which involves using separate tables. In comparison, Hadoop only needs one table for our data.

The size of our data is also large, and Hadoop did a spectacular job in retrieving the data and transforming it into the data we need. Knime world usually takes a long time to filter through are data but Hadoop was able to do it in seconds. However, when it came to printing our data using “dump” it would take a considerable amount of time. Apache Pig for filtering and storage and mainly using HiveQL for querying results.

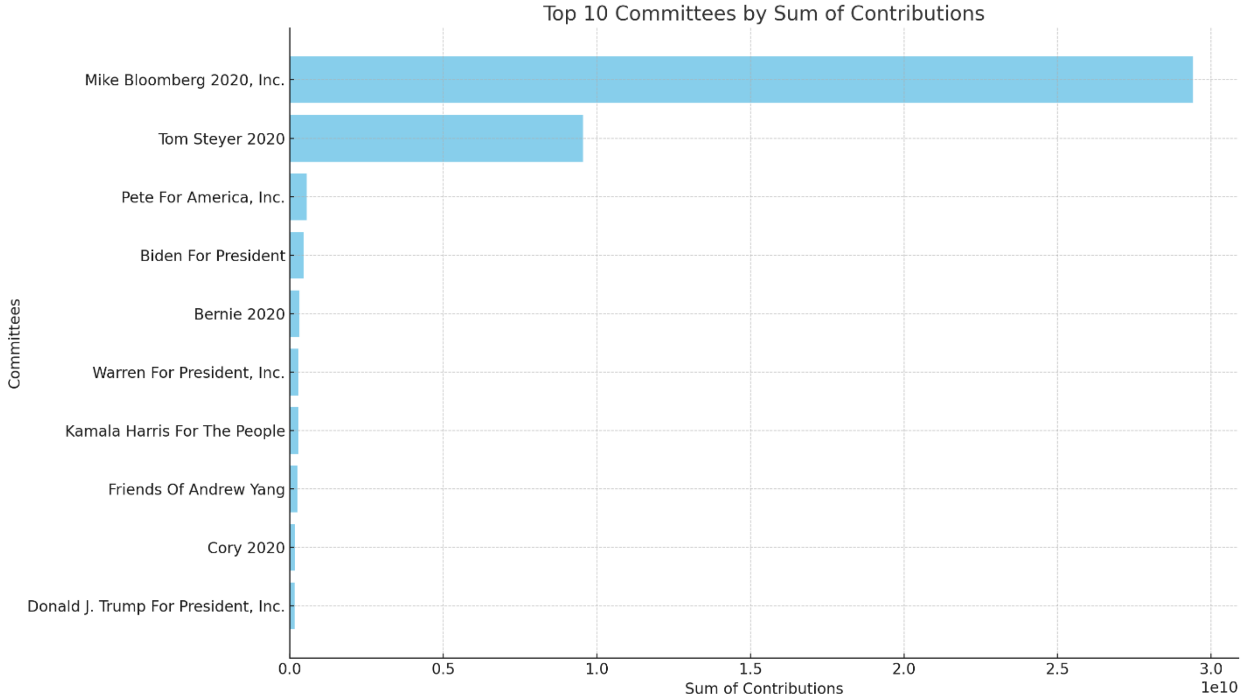
4. Reporting System

Provide screenshot and sample results with discussion on potential knowledge that was elicited.

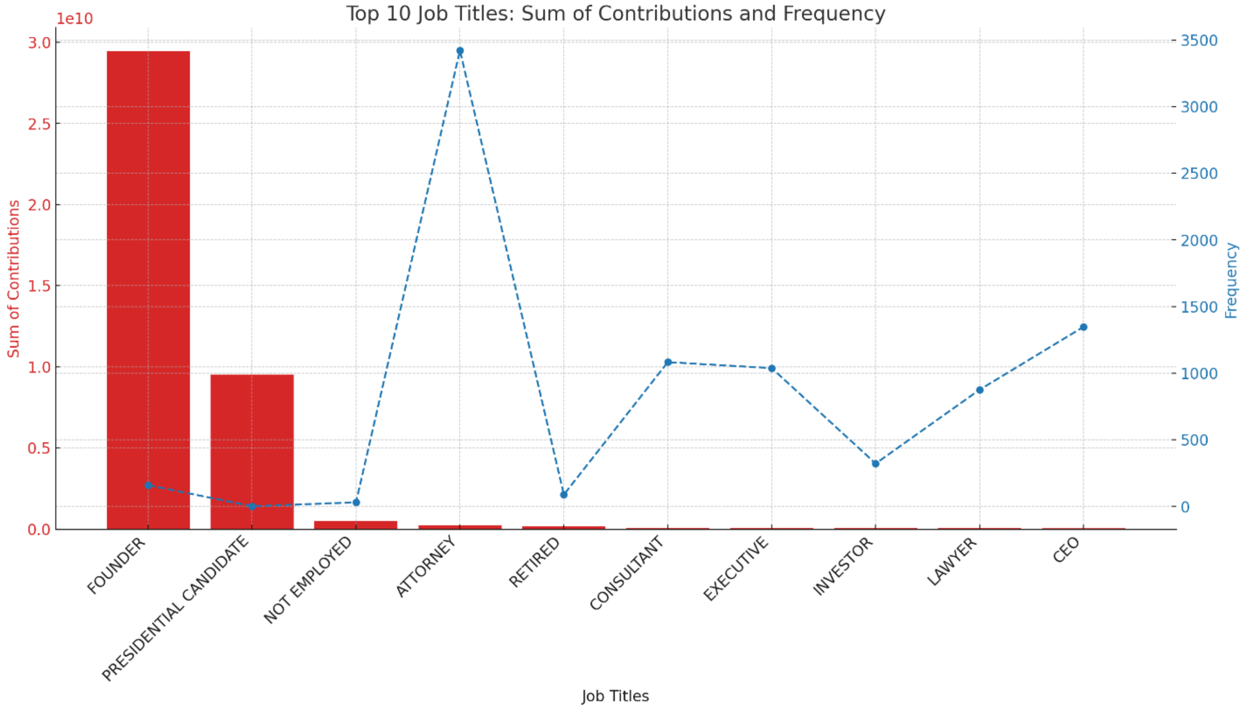
4.1 Relational Data Warehouse Implementation



Following the implementation of the Data Warehouse, an analysis was conducted to observe the distribution of political contributions across New York, Texas, and California. This investigation revealed a notable increase in contributions during October 2019. The temporal analysis aims to understand patterns of financial support within the context of electoral cycles and assess the impact of socio-political events on fundraising activities.



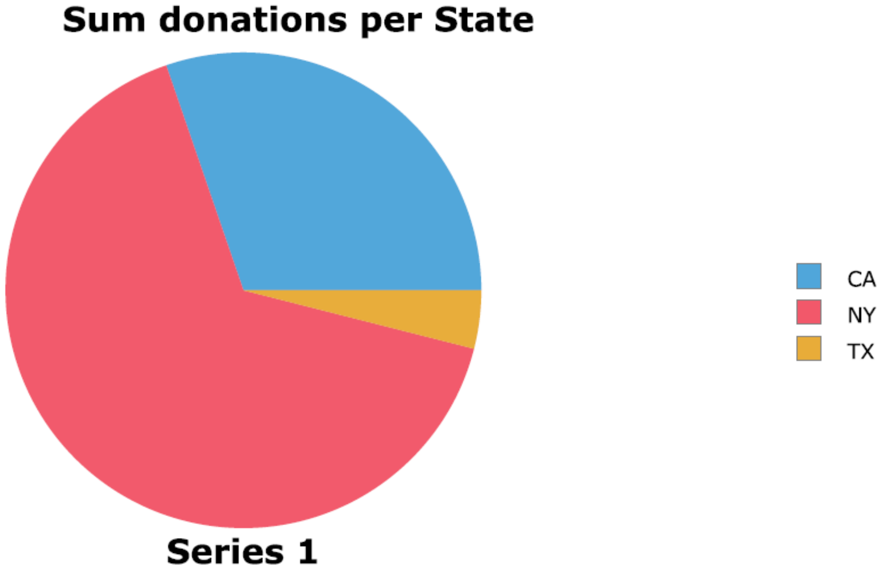
After setting up the Data Warehouse, we decided to check which committees got the most money. We thought Mike Bloomberg's big donations might stand out compared to other committees. This step helps us see how big donors like him influence the flow of money in politics.



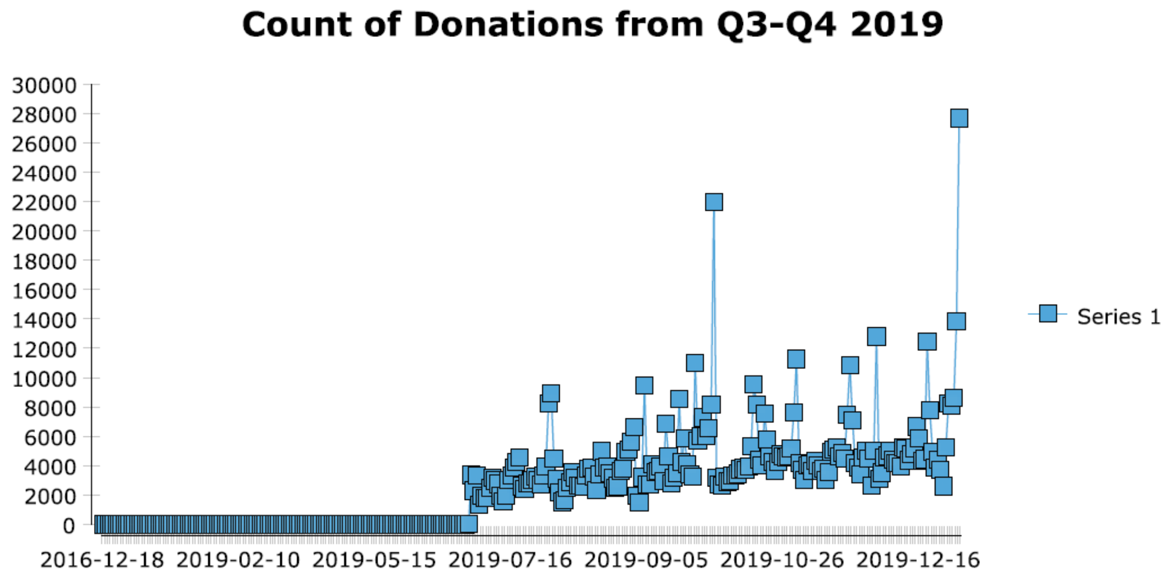
We then investigated the top contributors and their donation patterns. We found that attorneys donate frequently, while founders and presidential candidates contribute the most money overall. Interestingly, non-employed and retired individuals also rank high among contributors, likely due to the older generation's access to various income sources beyond traditional employment. This suggests that significant contributors tend to be older.

4.2 Hadoop Implementation

Knime can be used to visualize the data from our HDFS and can be queried using HiveQL. The data will be reported and a lengthy ETL pipeline would not be needed for reporting. Considerations will need to be made as to char type.



The Pie chart indicates a huge proportion of donations are coming from New York. There is enough evidence to suggest that donations are mainly coming from high income earners in New York. Perhaps Business owners or Lawyers who stand to benefit from law reforms.



Around quarter four is when donations tend to ramp up exponentially. I would argue this is due to it being clearer which candidates might win.

4.3 Reflective analysis of result in relational data warehouse vs Hadoop.

Our data warehouse search, which only begins to reveal a sophisticated understanding of political contributions, revealed the lawyer group to contribute most often and the founder and presidential candidates to give the most money. It also said that many non-employed and retired have explained this behavior by the broadened access of the older generation to alternative income sources. This implies a demographic skew in the direction of older political donors and continues to ask how age and occupation define one's political support.

We also used Knime for our HDFS. The central part of our work was done using Hadoop alone with the nodes we wanted reports from, numbering so few. We were able to derive the frequency of donations with respect to time. I was able to find which states in our data contributed most.

Conclusions

The project specification details the creation of a data warehouse for analyzing election contributions in three states - Texas, New York, and California. The primary objectives of the project include improving data retrieval speed, maintaining data integrity, and analyzing trends in contribution patterns. To ensure the safety of sensitive data, security measures have been prioritized, while usability requirements ensure that the interface is user-friendly. Key considerations involve improving data quality, integrating disparate data sources, and increasing efficiency in data processing through the ETL process (like a data pipeline). The reporting system in the data warehouse enables analysis of temporal distribution, top contributors, and donor patterns. Overall, the project aims to provide comprehensive insights into election contribution dynamics, leveraging advanced data analytics techniques to inform political decision-making effectively.

In summary, the choice between a relational data warehouse and Hadoop depends on the requirements and project scope. Relational databases perform well in settings that prioritize accurate, sophisticated querying and structured data integrity. Hadoop, on the other hand, works better in situations where scalability and cost-effectiveness are critical when there are large amounts of different, unstructured data. Even though each system has unique benefits, combining the two can occasionally result in a complete solution that makes use of Hadoop for large-scale data processing and relational databases for intricate data interactions, optimizing the data management environment for better strategic analysis and decision-making.