Penn State University

Great Valley Campus

Engineering Division

Data Specification for

Presidential Election Contributions 2020

Final Version

[03-24-2024]

**Table of Contents**

[Introduction 3](#_Toc457330103)

[Purpose 3](#_Toc457330104)

[Project Summary 3](#_Toc457330105)

[Requirements Definition 4](#_Toc457330106)

[Considerations 4](#_Toc457330107)

[Document Change Log 4](#_Toc457330108)

[Architecture Design 5](#_Toc457330109)

[Entity Relationship Diagram 5](#_Toc457330110)

[Data Dictionary 5](#_Toc457330111)

[Tables schemas 5](#_Toc457330112)

[Examples of values 5](#_Toc457330113)

[ETL Process 6](#_Toc457330114)

[ETL considerations 6](#_Toc457330115)

[ETL Process Flow with description 6](#_Toc457330116)

[Reporting System 7](#_Toc457330117)

[Conclusions 7](#_Toc457330118)

Presidential Election Contributions 2020

[Krishna Chaithanya Reddy Atla-kma6396@psu.edu, Abhishek Reddy Tetali-aft5636@psu.edu]

# Introduction

This project aims to create a data warehouse for Presidential Election Contributions 2020 data using the KNIME platform to mine the data and answer business questions.

**Data Warehouse (KNIME):**

The KNIME Data Warehouse is a reliable and powerful tool for managing data, designed to support business intelligence and analytics efforts seamlessly. It serves as a solid foundation for analyzing structured data and making informed decisions. Inside the warehouse are extensive collections of historical data, carefully organized and waiting to be explored. By following the schema-on-write approach, this strong solution guarantees effectiveness and precision in processing large datasets, turning the search for valuable insights into a rewarding experience.

# Purpose

The purpose of this document is to a template for a project specification document for data warehouse and knowledge management development projects.

# Project Summary

1. **Objectives**

Exploring this exciting Big Data project allows us to delve into a detailed analysis of the 2020 Presidential Contribution Data from different states in the USA. The process involves carefully Extracting, Transforming, and Loading the data into a data warehouse using the KNIME Analytics Platform. Next, we move on to using the powerful KNIME Reports API to create engaging reports and visualizations for better business intelligence. These insights will reveal the intricate trends present in the data, giving us a deep understanding of the contributions during the significant 2020 Presidential election.

1. **Scope**

We're going to dive deep into the data on contributions by the State, looking closely at the total amount given, the number of contributions, and the average amount donated. We'll also look at the political affiliations of the donors and gather some key demographic information. This project is scheduled to produce a wide range of results within a set period. This will include a carefully organized data warehouse holding the processed data. We'll also create interactive reports and visualizations to showcase the complex patterns influencing the 2020 Presidential Election contributions in different states.

1. **References**

The data that is being used in this project is taken from the [2020 Presidential Contribution Data by State](https://projects.propublica.org/itemizer/presidential-contributors/2020) referred by the program instructor. In this analysis, I am considering contributions made in Delaware, Hauwai, Texas, and Missouri for January and February 2020.

1. **Outstanding Issues**

There are some inconsistencies in the data like some columns have missing values and the formats of some columns like date should be consistent.

# Requirements Definition

1. **Goals**

The main goal of this Data warehouse is to act as a storage space for past data, enabling thorough examination to reveal trends and patterns within the data.

1. **Usability Requirements**

The platform should support various file formats and should be able to integrate other software tools like databases and reporting tools.

1. **System Security Requirements**

The data that is being used in this project is available for public use. So, the system security requirements are very basic, like the data warehouse will have username and password and the data is hosted on local system.

1. **Business Questions**
2. To find the candidate popular among the students.
3. To find who between the two(Biden, Trump) got highest contributions in January
4. Identifying the predominant occupation among contributors
5. Which committee has received the highest funds overall?
6. Which state raised the highest funds on March 31, 2020?
7. Which city of Texas has more contribution.

1. **Data Requirements:**

2020 presidential contributions made in California, New York, Texas, and Missouri for January and February 2020. Identify which columns are useful to answer the business questions and create fact tables and dimension tables.

1. **Design Constraints**

I used star schema to store the data.

# Considerations

(May include as separate document)

# Document Change Log

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Change Date** | **Version** | **CR #** | **Change Description** | **Author and Organization** |
| 03/24/2024 | 1.0 |  | Business and Data Understanding | Krishna, Abhishek |
| 04/06/2024 | 2.0 |  | ETL and Report Generation using KNIME | Krishna, Abhishek |
| 04/07/2024 | 3.0 |  | Formatting | Krishna, Abhishek |
| 04/26/2024 | 4.0 |  | Project-2 | Krishna, Abhishek |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

# 2. Architecture Design

|  |  |  |
| --- | --- | --- |
| filing\_id | N/A |  |
| linenumber | N/A |  |
| flag\_orgind | dimension | Contributors' Table |
| org\_name | dimension | Contributors' Table |
| last\_name | N/A |  |
| first\_name | N/A |  |
| middle\_name | N/A |  |
| prefix | N/A |  |
| suffix | N/A |  |
| address\_one | N/A |  |
| address\_two | N/A |  |
| city | dimension | Location Table |
| state | dimension | Location Table |
| zip | dimension | Location Table |
| employer | dimension | Contributors' Table |
| occupation | dimension | Contributors' Table |
| Amount | Fact | Fact Table |
| count | Fact | Fact Table |
| date | dimension | Date table |
| aggregate\_amount | N/A |  |
| memo\_code | N/A |  |
| memo\_text | N/A |  |
| tran\_id | N/A |  |
| back\_ref\_tran\_id | N/A |  |
| back\_ref\_sched\_name | N/A |  |
| prigen | N/A |  |
| cycle | dimension | candidate table |
| fecid | dimension | candidate table |
| committee\_name | dimension | candidate table |

## 2.1 Relational Data Warehouse

### Data Dictionary

A Data Dictionary is a document that describes the basic organization of a database. Typically, a data dictionary will contain a list of variables in the database as well as the assigned variable names and a description of each type of variable. The data dictionary should also include the values accepted for each variable and any helpful comment such as important exclusions and skip patterns. The data dictionary is used primarily for data analysis.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Table** | | | | |
| Variable | Variable name | Variable type | Values | notes |
| city | city | String | Thousand Oaks | N/A |
| state | state | String | TX | N/A |
| zip | zip | Integer | 19355 | N/A |
| flag\_orgind | flag\_orgind | String | IND | N/A |
| org\_name | org\_name | String | First Foundation Bank, NULL | N/A |
| employer | employer | String | Cal Poly Pomona | N/A |
| occupation | occupation | String | Student | N/A |
| amount | amount | Integer | 5 | N/A |
| date | date | Date | 07-01-2020 | N/A |
| cycle | cycle | Integer | 2020 | N/A |
| fecid | fecid | String | C00693234 | N/A |
| committee\_name | committee\_name | String | Warren for president, Inc. | N/A |

**Defining Grain:**

For this project on data warehousing, I have carefully specified the granularity. The level of detail I have chosen includes flag\_orgind, org\_name, employer, occupation, cycle, fecid, committee\_name, city, state, zip, amount, and date. Granularity, or the level of detail, in the context of data warehousing refers to how finely data is gathered and examined. By selecting this specific level, it suggests a concentration on analyzing and presenting data accurately, potentially revealing subtle patterns or trends within specific organizations, sectors, or regions. Analyzing data in fine-grained detail can be useful for identifying specific patterns in campaign finance within certain regions or industries. Nonetheless, it's important to acknowledge that this level of specificity may not always be appropriate for every analytical task. It might limit the ability to draw broader conclusions or apply findings to larger datasets or trends.

Daily Amount is the grain of the fact table:

|  |  |  |
| --- | --- | --- |
| amount | Fact\_table | Fact |
| count | Fact\_table | Fact |
| Falg\_orgind | Contributor\_table | Dimensional |
| employer | Contributor\_table | Dimensional |
| occupation | Contributor\_table | Dimensional |
| Date | Date\_table | Dimensional |
| Month | Date\_table | Dimensional |
| Year | Date\_table | Dimensional |
| Date\_date | Date\_table | Dimensional |
| Fecid | Candidate\_table | Dimensional |
| Cycle | Candidate\_table | Dimensional |
| Committee\_name | Candidate\_table | Dimensional |
| City | Location\_table | Dimensional |
| State | Location\_table | Dimensional |
| Zip | Location\_table | Dimensional |

### Tables schemas

Describe the physical schema of the data warehouse. Use the steps in the lesson and explain.

|  |  |  |  |
| --- | --- | --- | --- |
| ***Candidate\_Table*** |  |  |  |
| **Description** | This table details a political party | | |
| **Attribute** | **Description** | **Type** | Examples of values |
| **fecid** | FEC ID of the political party | Varchar | C00693234 |
| **cycle** | Election Year | Integer | 2020 |
| **Committee\_name** | Name of the political party | Varchar | Warren for president, Inc. |
| **Candidate\_ID** | Unique ID of the table | Serial | 1 |
| **Primary Key** | ID | | |
| **Foreign Keys** |  | | |
| **SQL Query** | -- public.candidate\_table definition  -- Drop table  -- DROP TABLE candidate\_table;  **CREATE** **TABLE** candidate\_table (  candidate\_id **int4** **DEFAULT** **nextval**(**'candidate\_table\_id\_seq'**::**regclass**) **NOT** **NULL**,  fecid **varchar** **NULL**,  **"cycle"** **int4** **NULL**,  committee\_name **varchar** **NULL**,  **CONSTRAINT** candidate\_table\_pkey **PRIMARY** **KEY** (candidate\_id)  ); | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Location\_Table*** |  |  |  |
| **Description** | This table details the location of contributor | | |
| **Attribute** | **Description** | **Type** | Examples of values |
| **City** | City of Contributor | String | Thousand Oaks |
| **State** | State of Contributor | String | NY |
| **Zip** | Zip code of Contributor | Integer | 19355 |
| **Location\_ID** | Unique ID of the table | Serial | 1 |
| **Primary Key** | Location\_ID | | |
| **Foreign Keys** |  | | |
| **SQL Query** | -- public.location\_table definition  -- Drop table  -- DROP TABLE location\_table;  **CREATE** **TABLE** location\_table (  location\_id **int4** **DEFAULT** **nextval**(**'location\_table\_id\_seq'**::**regclass**) **NOT** **NULL**,  city **varchar** **NULL**,  state **varchar** **NULL**,  zip **int4** **NULL**,  **CONSTRAINT** location\_table\_pkey **PRIMARY** **KEY** (location\_id)  ); | | |
|  |  | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Date\_Table*** |  |  |  |
| **Description** | This table details of Contribution date | | |
| **Attribute** | **Description** | **Type** | Examples of values |
| **Date** | Date of Contribution | String | 02 |
| **Month** | Month of Contribution | String | 01 |
| **Year** | Year of Contribution | Integer | 2020 |
| **Date\_ID** | Unique ID of the table | Serial | 1 |
| **Primary Key** | Date\_ID | | |
| **Foreign Keys** |  | | |
| **SQL Query** | -- public.date\_table definition  -- Drop table  -- DROP TABLE date\_table;  **CREATE** **TABLE** date\_table (  date\_id **int4** **DEFAULT** **nextval**(**'date\_table\_id\_seq'**::**regclass**) **NOT** **NULL**,  **"date"** **int4** **NULL**,  **"month"** **int4** **NULL**,  **"year"** **int4** **NULL**,  date\_date **date** **NULL**,  **CONSTRAINT** date\_table\_pkey **PRIMARY** **KEY** (date\_id)  ); | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Fact\_Table*** |  |  |  |
| **Description** | The fact table | | |
| **Attribute** | **Description** | **Type** | Examples of values |
| **Amount** | Amount of money contributed | Integer | 5 |
| **Contributor\_id** | Unique ID of the contributor\_table | Integer | 1 |
| **Candidate\_id** | Unique ID of the candidate\_table | Integer | 1 |
| **Location\_id** | Unique ID of the location\_table | Integer | 1 |
| **count** | Count of number of trasactions | integer | 4 |
| **amount** | Total amount of money | integer | 10000 |
| **Date\_id** | Unique ID of the date\_table Integer 1 | | |
| **Primary Key** | Date\_id+Location\_id+Election\_id+contributor\_id | | |
| **Foreign Key** | Date\_id, Location\_id, Election\_id, contributor\_id | | |
| **SQL Query** | -- public.fact\_table definition  CREATE TABLE public.fact\_table (  amount INT NULL,  count INT NULL,  contributor\_id INT NOT NULL,  election\_id INT NOT NULL,  location\_id INT NOT NULL,  date\_id INT NOT NULL,  CONSTRAINT fact\_table\_pk PRIMARY KEY (contributor\_id, location\_id, election\_id, date\_id),  FOREIGN KEY (date\_id) REFERENCES public.date\_table(id),  FOREIGN KEY (location\_id) REFERENCES public.location\_table(id),  FOREIGN KEY (contributor\_id) REFERENCES public.contributor\_table(id),  FOREIGN KEY (election\_id) REFERENCES public.candidate\_table(id)  ); | | |

|  |  |  |  |
| --- | --- | --- | --- |
| ***Contributor\_Table*** |  |  |  |
| **Description** | This table details the contributor. | | |
| **Attribute** | **Description** | **Type** | Examples of values |
| **Flag\_orgind** | Flags whether the contribution was made by an individual or an organization. | String | IND |
| **Employer** | Contributor’s employer | String | VISION BUILDERS USA |
| **Occupation** | Occupation | Integer | 3D ARCHITECTURAL ARTIST |
| **Contributor\_id** | Unique ID of the table | Serial | 1 |
| **Primary Key** | Contributor\_ID | | |
| **Foreign Keys** |  | | |
| **SQL Query** | -- public.contributor\_table definition  -- Drop table  -- DROP TABLE contributor\_table;  **CREATE** **TABLE** contributor\_table (  contributor\_id **int4** **DEFAULT** **nextval**(**'contributor\_table\_id\_seq'**::**regclass**) **NOT** **NULL**,  flag\_orgind **varchar** **NULL**,  employer **varchar** **NULL**,  occupation **varchar** **NULL**,  **CONSTRAINT** contributor\_table\_pkey **PRIMARY** **KEY** (contributor\_id)  ); | | |

## Star Schema

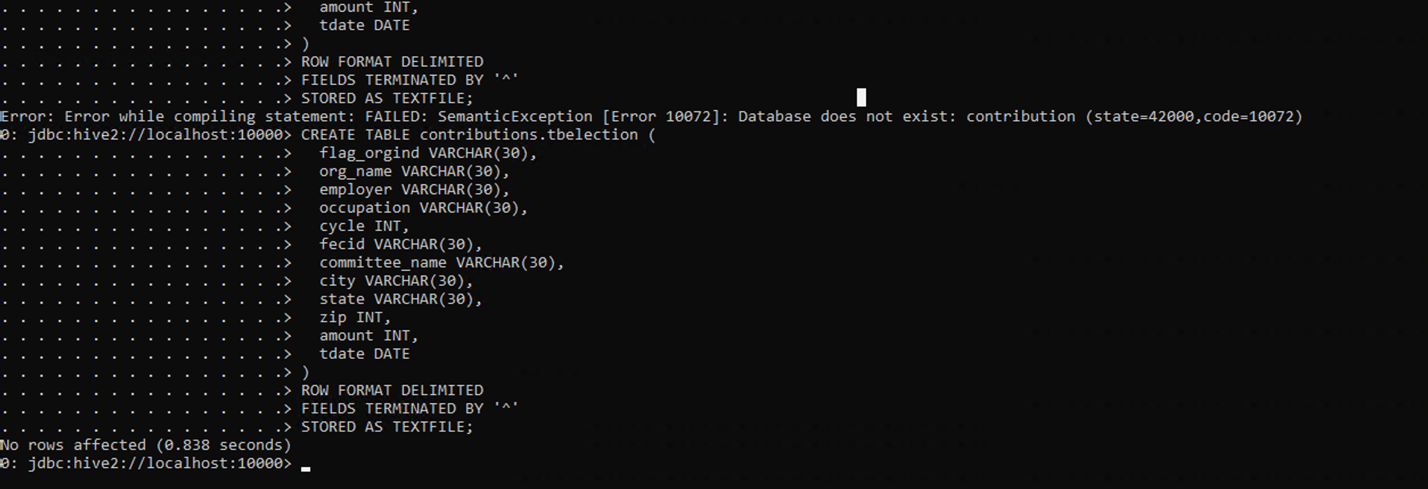
A screenshot of a computer

Description automatically generated

## 2.2 Hadoop Implementation

To be completed with project number 2

We already have the business queries above, based on those business queries I’m selecting the required and necessary column. In Hadoop implementation, all the CSV files of all the four different states are loaded into the Hadoop containers. I have created the database schema in the Hive using the following HiveQL code.

  
  
Here I have used ‘^’ as the delimiter while storing the data into the Hive database.

And the schema is created, we can see it in the following image:

A screen shot of a computer

Description automatically generated

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

|  |  |  |
| --- | --- | --- |
| **Feature** | **Data Warehouse** | **Hadoop** |
| **Data Type** | Data warehousing in relational data warehouse is mainly designed for structured data, which usually means data arranged in tables with predetermined relationships. | Hadoop is designed to handle big data which includes not only structured data but also unstructured and semi-structured data. It does not force schema, allowing it to store and process data like as text, images, and log files. |
| **Data Processing** | Data warehousing is specifically optimized to efficiently manage intricate queries that require combining data from multiple tables. | Hadoop processes the data using a distributed computing model, and MapReduce, allowing for the processing of large volumes of data in parallel across a large number of nodes. It can be effective for tasks such as indexing, searching, and analyzing large datasets. |
| **Query Language** | SQL is used as the main query language in data warehousing. | Hadoop uses Hadoop Query Language (HiveQL) in Hive which is similar to SQL, and Pig Latin in Pig, for querying data. But, Hadoop also supports other languages and models through different processing engines like Apache Spark (using Scala, Python, or Java). |
| **Scalability** | Data warehousing often faces limitations in horizontal scalability, which means it is only able to efficiently handle a certain amount of data and processing power. | Hadoop is highly scalable as it is designed to easily add more nodes to the cluster for increase inn capacity. This allows it to handle petabytes and even exabytes of data without a decrease in processing efficiency. |
| **Fault Tolerance** | Data warehouses usually provide decent fault tolerance, while Hadoop is well known for its strong fault tolerance capabilities. | Hadoop's distributed file system, is designed with high fault tolerance. Data is replicated in all multiple nodes, ensuring that the system can recover from the failure of a node without data losing. |
| **Cost** | Setting up a data warehouse can be costly in the beginning due to the need for purchasing hardware and software licenses. | Hadoop's is an open-source framework that can be more cost-effective as it runs on commodity hardware and does not need expensive specialized systems. Maintenance and scaling costs can be lower compared to traditional data warehousing solutions. |
| **Use Cases** | Data warehousing is ideal for analyzing data in real-time, creating reports, and making informed decisions with structured data. | Hadoop is suited for batch processing large volumes of data, data mining, processing log files, and performing complex transformations and analyses on data that does not fit well into relational models, such as unstructured data or raw, detailed data from various sources. |

3. Data Preparation

## 3.1 Relational Data Warehouse Implementation

## ETL considerations

Extracting, transforming, and loading (ETL) is a crucial step in the world of data storage and business analysis. The process includes extracting data from different sources, reorganizing it to fit a specific format, and then transferring it to a specific destination. When creating an ETL process, there are several important factors to keep in mind.

1. **Data Quality**: One major concern is the quality of the data being extracted. It is essential to guarantee that the data is accurate, consistent, and complete. This requires tasks like cleaning, analyzing, and verifying the information to uphold its integrity.
2. **Improving Performance**: To ensure fast processing of large amounts of data, it's essential to optimize ETL processes. This can be achieved through techniques like partitioning, parallel processing, and caching, which improve performance.
3. **Enhancing Scalability:** With the continuous growth of data, it's important to design ETL procedures that can scale effectively. This often involves using distributed architectures such as Hadoop and organizing the ETL process to work on clusters of computers.
4. **Data Security:** It's crucial to protect data throughout the extraction, transformation, and loading phases, especially when dealing with sensitive information. Methods like data encryption, access controls, and thorough auditing practices are implemented to boost data security.
5. **Data lineage and documentation**: It's important to keep thorough records of the ETL process. This documentation should cover where the data comes from, what transformations are made to it, and where it ends up. Having this information on hand is useful for troubleshooting, debugging, and making sure the data stays consistent and accurate.
6. **Error handling and monitoring:** Dealing with errors and keeping an eye on things during the ETL process is also key. Errors can happen, so it's important to have good error-handling mechanisms in place. Logging errors, setting up alerts, and having plans for retrying failed operations all help keep things running smoothly.
7. **Maintenance and testing:** Regular testing and maintenance are essential for ensuring accuracy and reliability in ETL procedures due to their complexity and dependencies. Techniques like unit testing, integration testing, and regression testing are used to validate the ETL process.

## ETL Process Flow with description

A diagram of a network

Description automatically generated with medium confidence

I stored all the CSV files in a designated folder and utilized a CSV reader node to read them collectively. Next, I filtered the columns based on the specified data architecture requirements.

To maintain consistency, I employed a case converter to transform all data entries into uppercase format.

I merged the "employer" and "organization" columns into a unified "employer" column for streamlined data management.

Addressing missing values in each column was a priority, ensuring data integrity and completeness.

The date column underwent conversion from string format to the appropriate date datatype.

Extracting new columns from the date information, such as day of month, month, and year, was carried out to enrich the dataset.

Renaming columns and adjusting their datatypes was done to align with the predefined database schema.

Each dimension table's columns were meticulously filtered, duplicates were eliminated, and columns were sorted to match the database schema specifications. This meticulous process ensures data consistency and adherence to the schema requirements.

After completing the preprocessing steps, I proceeded to write all the dimension tables to the database utilizing the DBwriter and DBconnector nodes.

Subsequently, I generated the fact table by joining the data obtained from the column rename node with the corresponding dates from each dimension table in the database. This process allowed me to acquire the primary keys of each table and store them into the fact table, ensuring alignment with the granularity specified in the data architecture.

|  |  |  |
| --- | --- | --- |
| **Table Name** | **Records** | **Attributes** |
| All csv Files data | 577,405 | 28 |
| Contributor\_table | 143,247 | 4 |
| Date\_table | 96,356 | 5 |
| Candidate\_table | 82,444 | 4 |
| Location\_table | 147,418 | 4 |
| Fact\_table | 653,919 | 7 |

## 3.2 Hadoop Implementation

To be completed with project number 2

During the implementation of Hadoop, the twelve CSV files of all the states are loaded into the Hadoop containers using code similar to the following Hadoop docker code. To do that, first zip all the 12 CSV files and then send this zip to the Hadoop container.

**docker cp your\_data.zip container\_id:/tmp/your\_data.zip**

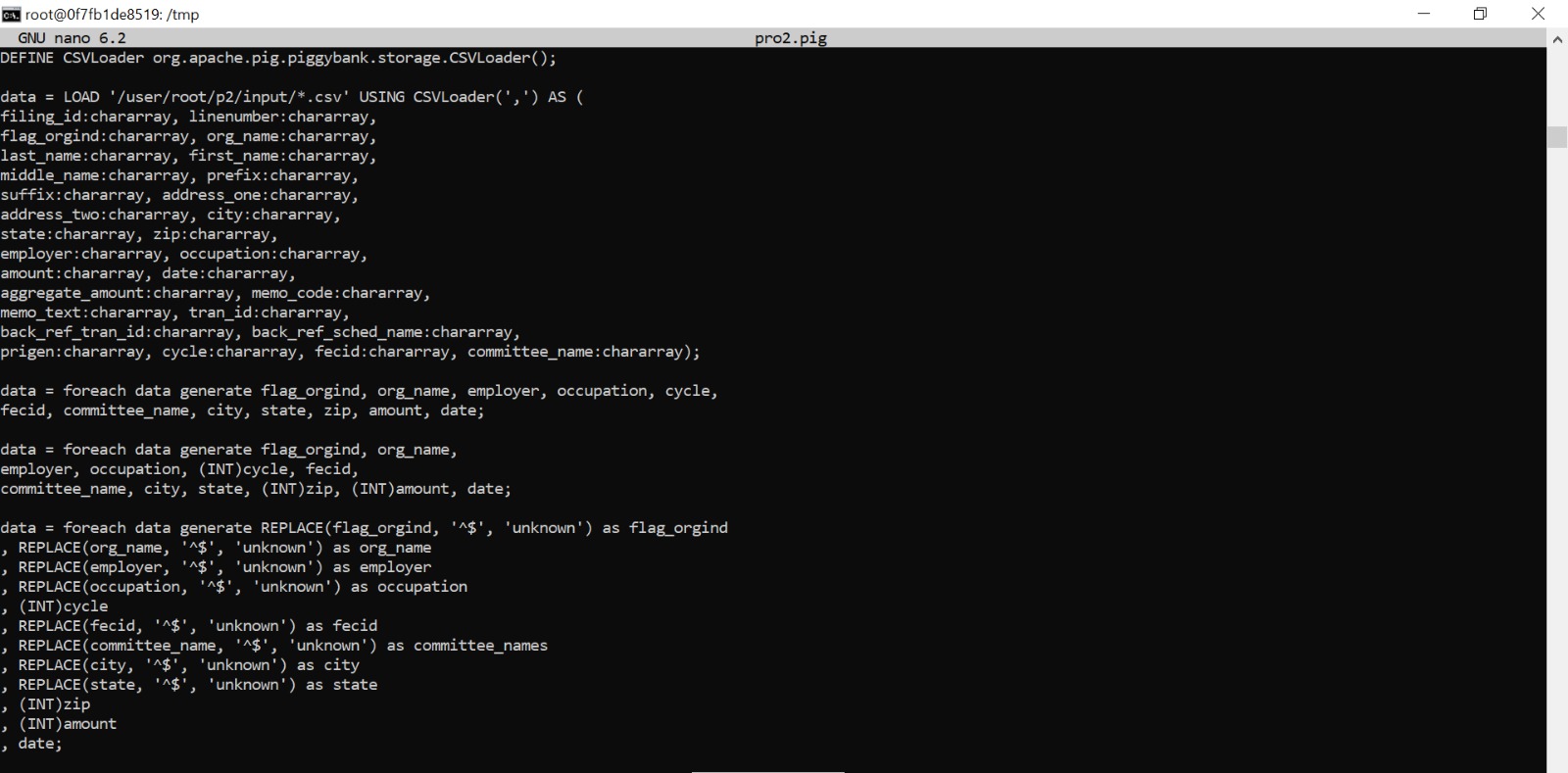
After that loading the zip file into the docker container, is the next step is to unzip the files into a specific path, here I’m unzipping it into the tmp folder.

Now I have moved all the unzipped csv files into the input folder using the following command format.

**hadoop fs -put ./source of csv file /destination of csv file.**

Sample:

**hadoop fs -put ./contributions\_m1\_2020\_CT.csv /user/root/p2/input/contributions\_m1\_2020\_CT.csv**

Once all the files are loaded into the docker container input path, the next step is to preprocess the data using Pig. In order to do that, we had made a Pig Script as follows:  
  
  
  
A black screen with white text

Description automatically generated  
  
  
After doing all the process using pig, then the data is stored into Hive database, the total number of records in the Hive Database is **99,240**.

A black screen with white text

Description automatically generated

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

For my Data Warehouse project, I used Dbeaver and KNIME for handling data. With KNIME, I imported files and performed ETL operations like extracting, cleaning, and transforming data. Then, in Dbeaver, I set up a database schema by creating dimension tables and a fact table. By linking the dimension tables to the fact table using foreign keys, I established a well-organized star schema. After setting up the star schema, the next step was to load the data into the database tables. This task was efficiently carried out using KNIME. By using the PostgreSQL connector, I created a seamless connection between Dbeaver and KNIME. The DBQuery Reader and DBWriter nodes in KNIME made it easy to read and write data to the Dbeaver database. All required preprocessing steps, such as converting data types, managing missing values, extracting features, and other transformations, were completed in KNIME just before saving the data in the Dbeaver database. After finishing the ETL processes, the clean data was saved in the Dbeaver database for further analysis. Using KNIME's features, I created analytics and visualizations that were crucial in developing informative reports and dashboards for our Big Data project. KNIME provides different reporting tools and methods, enabling me to design personalized reports and interactive dashboards that effectively convey the data insights.

For the Hadoop project, I have used Hive, Pig and KNIME to process, store and visualize the data. Using Hive, I have created the database schema and table to store the data. Using my pig script file I have performed all the ETL steps like, data extraction from the zip file, loading all the data into the pig environment for processing, selecting the required columns, handling null values or missing values, type conversion and all other preprocessing steps are completed using the Pig code file. Once the ETL process is completed, then result is stored into the temporary CSV file and then this file data is moved into the hive table using HiveQL query. Then the data is stored into the Hive database, and I have used KNIME to generate the visualizations which can be used to generate reports and visualizations for projects.

4. Reporting System

## 4.1 Relational Data Warehouse Implementation

I have established a connection between the Dbeaver database and KNIME using the PostgreSQL Connector node on KNIME. Once the connection is established using the DB query reader, I have extracted the data from the Dbeaver database and loaded into the KNIME platform. Using Bar chart and Pie chart nodes, the data visualizations are generated.

1. . The election contribution by students towards each candidate

1, To find the candidate popular among the students

A graph with different colored bars

Description automatically generated

The graph shows that Bernie has highest contributions from students in Texas, Delaware, Hawaii, Missouri. Which indicates he could be popular among students from these states.

2. The contributions towards Biden and Trump in January

1. To find who between the two got highest contributions in January.

A graph of a bar chart

Description automatically generated

Presidential candidate had slightly more contributed amount than Trump in January 2020

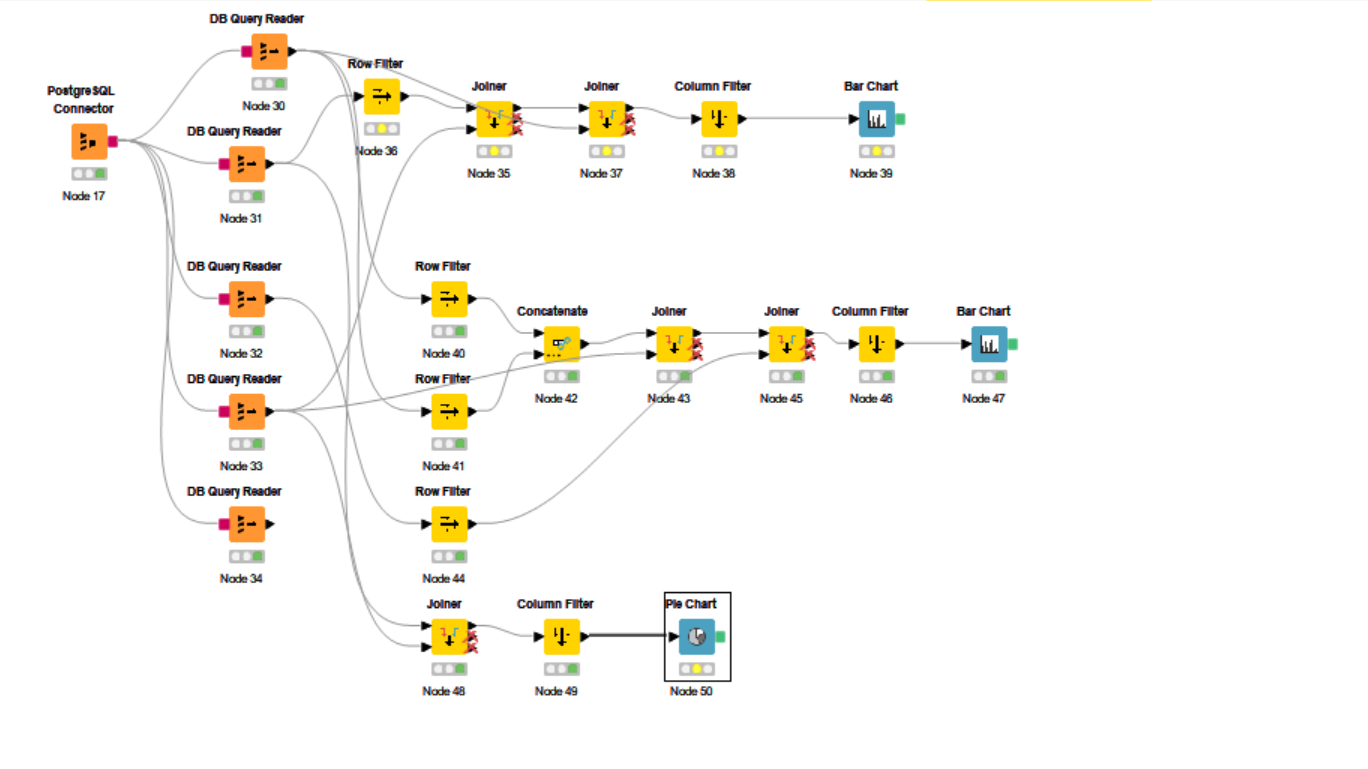
3. Analyzing the distribution of contributors based on occupation

3. Identifying the predominant occupation among contributors

A pie chart with different colored circles

Description automatically generated

I replaced the missing values in the occupation column with "Other". This suggests that there are a substantial number of missing entries in the occupation column. Interestingly, it appears that unemployed individuals (those not employed or retired) are significant contributors, if not the primary contributors.



## 4.2 Hadoop Implementation

To be completed with project number 2

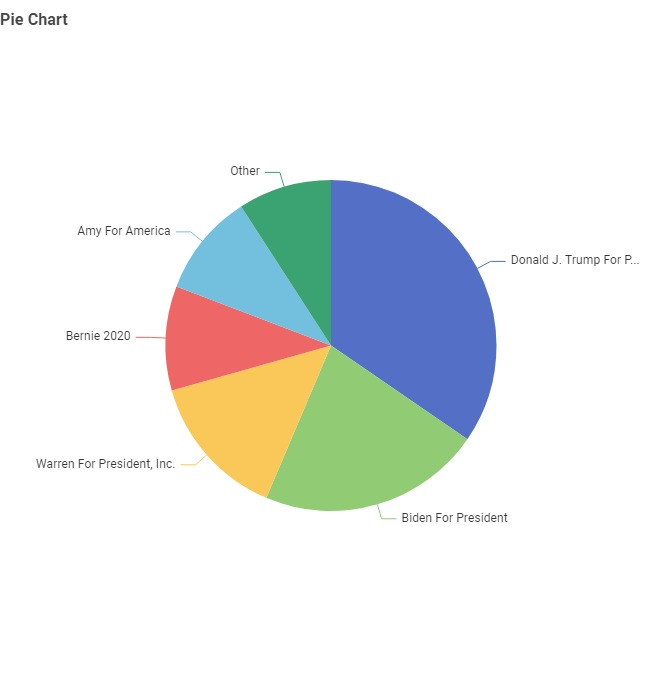
Again, I have made a connection between the Hive database and KNIME using the Hive Connector node on KNIME. After the connection is established using the DB query reader, I have extracted the data from the Hive database and loaded it into KNIME. By using the Data to Report node, I have generated the data visualization reports as follows.  
  
A computer screen shot of a diagram

Description automatically generated

1. To find the candidate popular among the students.  
     
   A graph of different colored bars

   Description automatically generated  
     
   We can clearly see from the plot that the candidate of Bernie2020 has the lead and is the 1st popular candidate than the candidate of Friends of Andrew Yang by some difference.
2. To find who between the two(Biden, Trump) got highest contributions in January   
     
   A graph of a bar chart

   Description automatically generated  
     
   We can say from the plot that Donald J. Trump For President is leading between the two candidates.
3. Identifying the occupation that has raised highest funds among contributors.  
   A pie chart with different colored circles

   Description automatically generated  
     
   As the plot shows that people with occupation as other had made the highest contribution.
4. Which committee has received the highest funds overall?   
     
     
   Again we can see that in all the committees also Donald J. Trump has got more funds than any other committee.
5. Which state raised the highest funds on Jan 31, 2020?   
   A bar chart with different colored squares

   Description automatically generated  
     
   As the plot shows TX is the state that has raised the highest amount of funds on that day.
6. Contribution by each city of Texas  
     
   A pie chart with different colored circles

   Description automatically generated  
     
   We can see that the others contribution has dominated the contribution of all the remaining cites together.

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

In project 1, please enter your reflective findings. Final version to be be completed with project number 2.

In the Data warehouse project, KNIME was used for all the ETL and Reporting tasks, it is a powerful tool that can be used for data preprocessing, transformation, and analysis. Dbeaver is a database tool that allows users to interact with relational databases. By using KNIME and Dbeaver together for a relational data warehouse helped me in performing data analysis and generate insights using SQL queries and visualizations easily.

While in the Hadoop project, the systems are designed for storing and processing large volumes of data in a distributed environment. I have used Hive (SQL-like query language) to process data stored in the HDFS. Also used Pig Latin to write data processing pipelines in a high-level language and run them on Hadoop. Then I have used KNIME and generated the reports using a node called Data cubes. After this it is same as we do in relational data warehouse.

By using both relational data warehouse and Hadoop I got similar results, but I have faced some issues in storing the data into the Hive using with delimiters, in process I have missed some records.

When I have used relational Data warehouse, I got more records at granularity level. Whereas in the Hadoop project I got less records. I believe that I have made some mistakes while handling data with pig.

Bu all the graphs and the trends looks very accurate by using both the relational Data warehouse and Hadoop project.

Conclusions

Overall conclusions of the project. In project 2, add a reflective analysis of the advantages and disadvantages of the two implementations.

To conclude about the data warehouse project which was done using KNIME & Dbeaver and the Hadoop project using Hive, Pig, & KNIME to find any observations in the election contributions, I think that both approaches have their own plus and minus.

While the project1 might look as if it is very simple and easy to work. The Data transformation we do Knime should be worked on very carefully. As we were using the tools it a bit more easier than the project when compared. But when working on the project2 it was a bit more complex to understand the commands but once we have an idea about the commands that should be used it becomes more easier than the project1.

Advantages of the data warehouse approach is that it is easy to setup and use, efficient for smaller datasets and we get to have good user interface. But it also has some disadvantages like it may struggle with very large datasets, and also not good for unstructured or semi-structured data.