Data Warehouses with 2020 Election Data

Comparison of PostgreSQL and Hadoop with campaign donation data for New England leading up to the Presidential 2020 election.

Kristopher Nerl

2021

Table of Contents

INTRODUCTION

This SRS document will describe what our Purpose, Project Summary, Definition of Requirements and Considerations are in terms of what we hope to accomplish with our data in two different database environments for comparative purposes. By describing the structure of our tasks, we hope to be able to replicate and improve upon these tasks for ourselves and others reviewing our results. Finally, we will review what knowledge we can glean from these environments and how it can be applied to real-world business-related tasks.

PURPOSE

The purpose of this project is to compare the implementation of PostgreSQL and Hadoop on the campaign donation data for four states in 3 quarters leading up to the Presidential 2020 election.

PROJECT SUMMARY

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

A. Objectives

Implementing, populating data, performing ETL, and executing reports on a Big Data solution to find trends among the 2020 Presidential election donors for Contributions made in Maine Massachusetts, New Hampshire and Vermont for Q1, Q2, and Q3 of 2019.

B. Scope

We will be comparing different implementations of systems (RDBS and document-oriented solutions) able to give the user insight towards their business model. The scope of this project is to analyze the performance of two such systems with different architectures on the same data and analytical goals.

C. References

1) https://www.propublica.org/nerds/introducing-fec-itemizer-a-tool-to-research-federal-election-spending

2) https://psu.instructure.com/courses/2137640/pages/project-1-description?module_item_id=32159585

REQUIREMENTS DEFINITION

A. Goals

- Deploy two databases (PostgreSQL and Hadoop), load data from public sources, and execute reports relevant to business goals.
- Review the pros and cons of each of the two databases.
- Select the best system architecture of the two databases based on the specifics of the analyzed data.

B. Usability Requirements

We build the data warehouse using PostgreSQL, with DBeaver. Knime will be used to perform ETL and generating reports.

The second architecture will be using Hadoop, Pig, and Hive to review the data, while KNIME is used to generate reports.

C. System Security Requirements

To prevent accidental loss and corruption of the data, only high-level users should have write access. Data will not contain sensitive information.
Also, only authorized users will be able access or alter that data through password protected Containerization through Docker.

D. Business Questions

This project will build a database to store data and generate reports to provide insight into election donation campaigns. The primary requirement of this project is to understand the types of individuals that contribute to the elections and the way this population has evolved. In order to address primary business query, the database would address following queries:
1) How are the four states ranked by total donations?
2) What professions have the highest donation amounts among the four states?
3) How are the three quarters ranked by total donations?
4) How many individuals make multiple donations?

E. Data Requirements

Data is provided to us as part of the project. It consists of several Excel files that contain election contributions by week. Data was pre-processed for the scope of this class, especially it was reduced in size. Files include a number of 28 columns and several groupings for each contributor.

The data file includes information about the candidate, contributor's information, collected amount, and profession of the contributor.

F. Design Constraints

Hardware/Software
Web server requirements: Docker will be used for containerization.
Software Interface: PostgreSQL, DBeaver 21, Knime 4.9 are used for the database.

CONSIDERATIONS

See above.

DOCUMENT CHANGE LOG

Change Date	Version	CR #	Change Description	Author and Organization
06/13/2021	1.0	1	Initial creation.	Kristopher Nerl
06/19/2021	2.0	2	Revision to requirements and schema. Addition of ETL.	Kristopher Nerl
07/1/2021	3.0	3	Addition of Reports and Conclusions.	Kristopher Nerl
07/17/2021	4.0	4	Revision to Introduction. Revisions to Architecture Design.	Kristopher Nerl
08/14/2021	5.0	5	Addition of Hadoop Implementation in Architecture Design, ETL and Reporting	Kristopher Nerl

2. ARCHITECTURE DESIGN

2.1 Relational Data Warehouse

Data Dictionary

Following Kimball's model for creating data marts first for the purposes of generating reports and analysis:

1. **Select the business process:** We will first review the raw data attributes that are provided:

Count	Attribute
1	filing_id
2	linenumber
3	flag_orgind
4	org_name
5	last_name
6	first_name
7	middle_name
8	prefix
9	suffix
10	address_one
11	address_two
12	city
13	state
14	Zip

Count	Attribute
15	employer
16	occupation
17	amount
18	date
19	aggregate_amount
20	memo_code
21	memo_text
22	tran_id
23	back_ref_tran_id
24	back_ref_sched_name
25	prigen
26	cycle
27	fecid
28	committee_name

We will aim to reduce these 28 attributes to leave only those relevant to our business questions.

2. **Declare the grain:** We are interested in quarterly results, so we will aggregate our data to share the same date dimensions based on what quarter they occurred in.

3. **Identify the dimensions:** We will review the data table and determine relevant dimensions to our business goals.

Count	Attribute	Notes
1	filing_id	Degenerate dimension
2	linenumber	Redundant dimension to filing_id
3	flag_orgind	Fact (irrelevant)
4	org_name	Dimension to identify organizations donating. (irrelevant)
5	last_name	Dimension to identify last name (irrelevant)
6	first_name	Dimension to identify first name (irrelevant)
7	middle_name	Dimension to identify middle name (irrelevant)
8	prefix	Fact (irrelevant)
9	suffix	Fact (irrelevant)
10	address_one	Fact (irrelevant)
11	address_two	Fact (irrelevant)
12	city	Dimension to identify city of donator (irrelevant)

Count	Attribute	Notes
15	employer	Dimension to identify employer of donator
16	occupation	Dimension to identify occupation of donator
17	amount	Fact
18	date	Dimension to identify date of donation
19	aggregate_amount	Fact
20	memo_code	Fact (irrelevant)
21	memo_text	Fact (irrelevant)
22	tran_id	Degenerate dimension (irrelevant)
23	back_ref_tran_id	Degenerate dimension (irrelevant)
24	back_ref_sched_name	Fact (irrelevant)
25	prigen	Dimension to identify candidate cycle (irrelevant)
26	cycle	Dimension to identify cycle of donation

13	state	Dimension to identify state of donator	27	fecid	Degenerate dimension (irrelevant)
14	zip	Dimension to identify zip of donator	28	committee_name	Dimension to identify candidate being donated to

To accomplish our four business goals and select the business process, we need to focus on states, donations amounts (1), professions (2), date of donation (3), and count of donations (4). These are our dimensions of usable data and will be used to query the data to get more detailed trends. Every dimensional model is composed of one central fact table that has a multi-attribute key and several attributes containing relevant data. Each attribute in the key of the fact table is an aspect of the data that is connected to a dimension table via a foreign key (Cardinality of this foreign key relation is 1-to-N). Each dimension table has an id as a primary key.

For economy of space, we will use only five dimensions (summarized from above):

1. Date of Donation
2. Donator Information
3. Occupation of Donator
4. Committee Donated to
5. Employer of Donator

Date of Donation Table				
Variable	Variable name	Variable type	Values	notes
Date ID number	date_id	Numeric	000001-999999	Not null
Date of Donation	date	Date	04/05/2019	Not null
Day of Donation	day	Numeric	24	Not null
Month of Donation	month	Numeric	05	Not null
Year of Donation	year	Numeric	2019	Not null

Donator Information Table				
Variable	Variable name	Variable type	Values	notes
Donator ID number	donator_id	Numeric	000001-999999	Not null
State of Donator	state	String	MA	Not null
Zip of Donator	zip	Numeric	00001-99999	Not null

Occupation of Donator Table				
Variable	Variable name	Variable type	Values	notes
Occupation ID number	occupation_id	Numeric	000001-999999	Not null
Occupation of Donator	occupation	String	Investor	

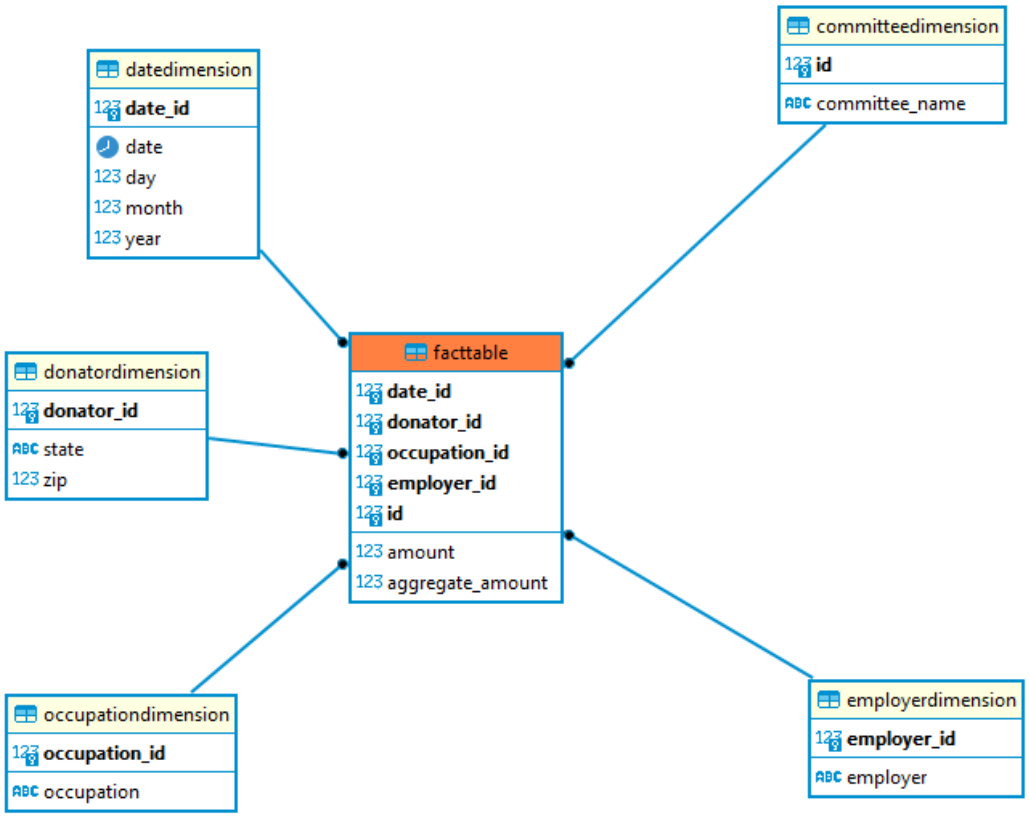
Committee Donated to Table				
Variable	Variable name	Variable type	Values	notes
Committee ID number	id	Numeric	000001-999999	Not null
Committee Name	committee_name	String	Hickenlooper 2020	Not null

Employer Table				
Variable	Variable name	Variable type	Values	notes
Employer ID number	employer_id	Numeric	000001-999999	Not null
Employer of Donator	employer	String	IBM	

Fact Table				
Variable	Variable name	Variable type	Values	notes
Date ID number	date_id	Numeric	000001-999999	Not null
Donator ID number	donator_id	Numeric	000001-999999	Not null
Occupation ID number	occupation_id	Numeric	000001-999999	Not null
Committee ID number	id	Numeric	000001-999999	Not null
Employer ID number	employer_id	Numeric	000001-999999	Not null
Amount of donation	amount	Numeric	200	Not null
Aggregate Amount of donation	aggregate_amount	Numeric	300	Not null

Tables schemas

Our star schema is extremely simple to understand by even end users who might not have much technical knowledge.



Name of the table	datedimension		
Description	This table describes date of donation.		
Attribute	**Description**	**Type**	***Examples of values***
date_id	Synthetic Key	Serial	1
date	Transaction date	Date	04/05/2019
day	Transaction day	Int4	24
month	Transaction month	Int4	05
year	Transaction year	Int4	2019
Primary Key	date_id		
Candidate Key	-		

Candidate Keys (if any)	N/A
Foreign Keys	N/A

Name of the table	donatordimension		
Description	This table describes the details of the donator		
Attribute	**Description**	**Type**	***Examples of values***
donator_id	Synthetic Key	Serial	1
state	State of the donator	Varchar	VT
zip	Zip code	Int4	2108
Primary Key	donator_id		
Candidate Key	-		
Candidate Keys (if any)	N/A		
Foreign Keys	N/A		

Name of the table	occupationdimension		
Description	This table describes the occupation of the donators.		
Attribute	**Description**	**Type**	***Examples of values***
occupation_id	Synthetic Key	Serial	1
occupation	occupation of the donator	Varchar	Investor
Primary Key	occupation_id		
Candidate Key	-		
Candidate Keys (if any)	N/A		
Foreign Keys	N/A		

Name of the table	committeedimension		
Description	This table describes about the name of the committee to which the amount was donated		
Attribute	**Description**	**Type**	***Examples of values***
Id	Synthetic Key	Serial	1
committee_name	Name of the committee	Varchar	Hickenlooper 2020
Primary Key	id		
Candidate Key	-		

Candidate Keys (if any)	N/A
Foreign Keys	N/A

Name of the table	employerdimension		
Description	This table describes about the employer information of the contributors.		
Attribute	**Description**	**Type**	***Examples of values***
employer_id	Synthetic Key	Serial	1
employer	Employer of the contributor	Varchar	IBM
Primary Key	employer_id		
Candidate Key	-		
Candidate Keys (if any)	N/A		
Foreign Keys	N/A		

Name of the table	facttable		
Description	This table describes about the fact table, will give insights about the amount.		
Attribute	**Description**	**Type**	***Examples of values***
date_id	Primary key of datedimension table	Serial	1
donator_id	Primary key of donatordimension table	Serial	2
occupation_id	Primary key of occupationdimension table	Serial	33
employer_id	Primary key of employerdimension table	Serial	33
id	Primary key of committeedimension table	Serial	22
amount	Transaction amount	Float4	589.92
count	Count from datatable	Int4	1
Primary Key	date_id+ donator_id + occupation_id + employer_id + id		
Candidate Key	-		

Candidate Keys (if any)	N/A
Foreign Keys	date_id, donator_id, occupation_id, employer_id, id

SQL Example:

```
CREATE TABLE datedimension (
   date_id serial NOT NULL,
   date date NOT NULL,
   day int4(31) NOT NULL,
   month int4(12) NOT NULL,
   year int4(2021) NOT NULL
);
```

2.2 Hadoop Implementation

We use Hive to create our initial database and table, using relevant entities to our business goals (discussed above).

```
CREATE DATABASE IF NOT EXISTS project;

CREATE TABLE final_project(
     date_ DATE,
     state VARCHAR(20),
     zip INT,
     occupation VARCHAR(20),
     committee_name VARCHAR(20),
     employer VARCHAR(20),
     amount FLOAT,
     aggregate_amount FLOAT
)
ROW FORMAT
DELIMITED FIELDS TERMINATED BY ','
STORED AS TEXTFILE;
```

```
0: jdbc:hive2://localhost:10000> describe final_project;
+-------------------+--------------+----------+
|     col_name      |  data_type   | comment  |
+-------------------+--------------+----------+
| date_             | date         |          |
| state             | varchar(20)  |          |
| zip               | int          |          |
| occupation        | varchar(20)  |          |
| committee_name    | varchar(20)  |          |
| employer          | varchar(20)  |          |
| amount            | float        |          |
| aggregate_amount  | float        |          |
+-------------------+--------------+----------+
8 rows selected (0.224 seconds)
```

2.3 Reflective analysis of using a data warehouse vs Hadoop.

Since our data warehouse required the user to perform ETL, we have inherently introduced a bias of the person initially processing the data. We might be missing great analytical conclusions that were removed in this ETL stage. While the data warehouse will be ACID compliant, it is doubtful that we need this normalization. However, our data is quite small, so performance variations between the two should be minimal.

Hadoop has the advantages using MapReduce instead of human influenced ETL. Hadoop will define the key values created by the Reducers for our output date, with our necessary business-oriented fields for generating our reports. Its keys are simplified, rather than the complicated and intrinsic variations of keys connecting our warehouse. While both databases are equipped to handle our data in its raw form and we aren't looking to modify the schema at this time, Hadoop will allow us flexibility in the future.

3. Data Preparation

3.1 Relational Data Warehouse Implementation
ETL considerations

The following ETL considerations were considered in creating our Data Warehouse:

- Only necessary data was included.
- Unique values should be prioritized/grouped. Null values were removed.

ETL is performed using KNIME to store the data in the tables from csv files, perform various filters on rows and columns to get the preprocessed data formatted for our data needs.

ETL Process Flow with description

1. Each .csv is combined and converted to a single .xls to be fed into Knime's Excel Reader.

2. Postgre SQL Connector is used to connect to the database.

3. Column Filter is used to keep the relevant dimension columns (Date of Donation, Donator Information, Occupation of Donator, Committee Donated to and Employer).

4. Extract Date and Time creates the values for day, month and year.

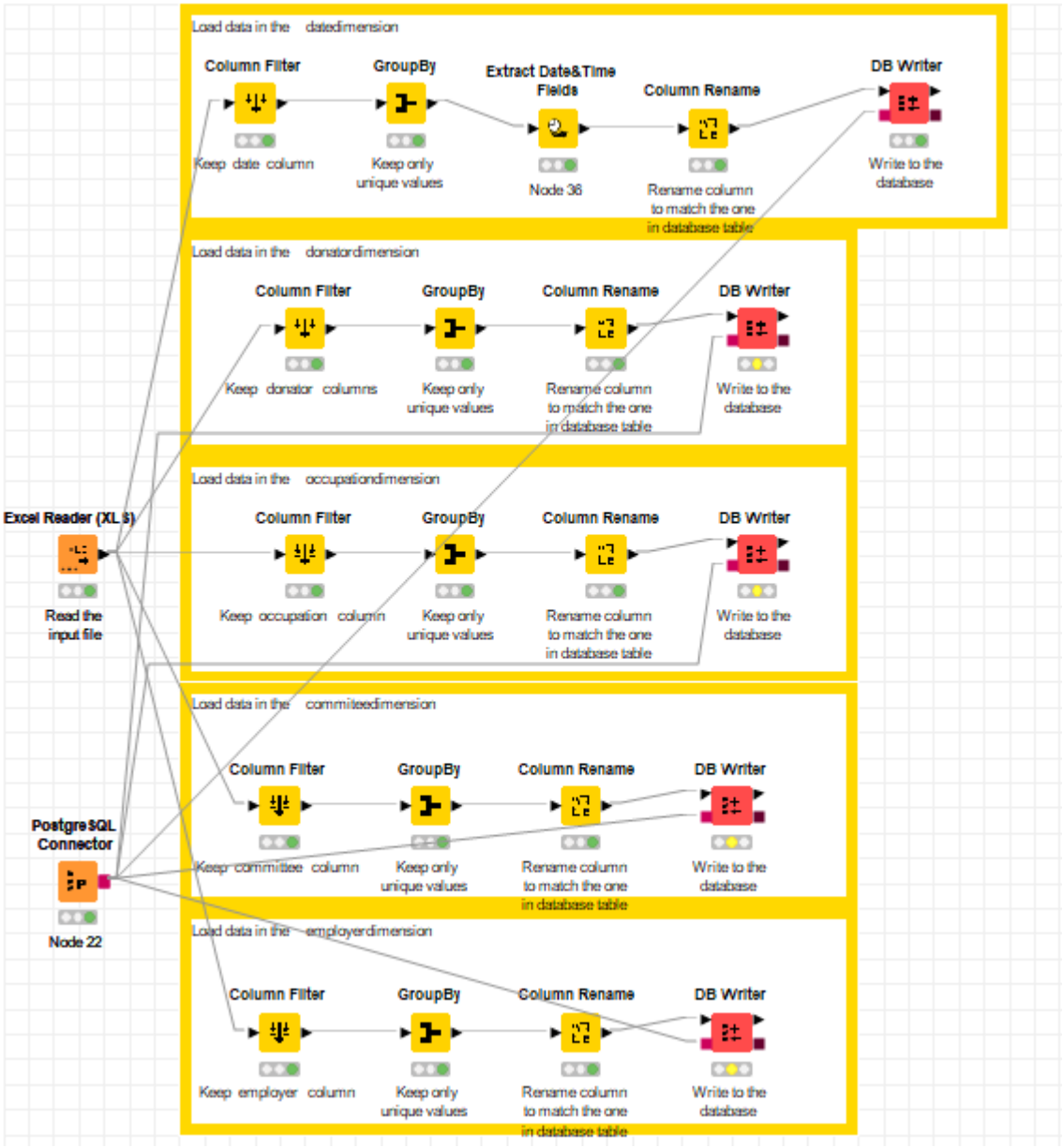
5. Groupby keeps only unique values.

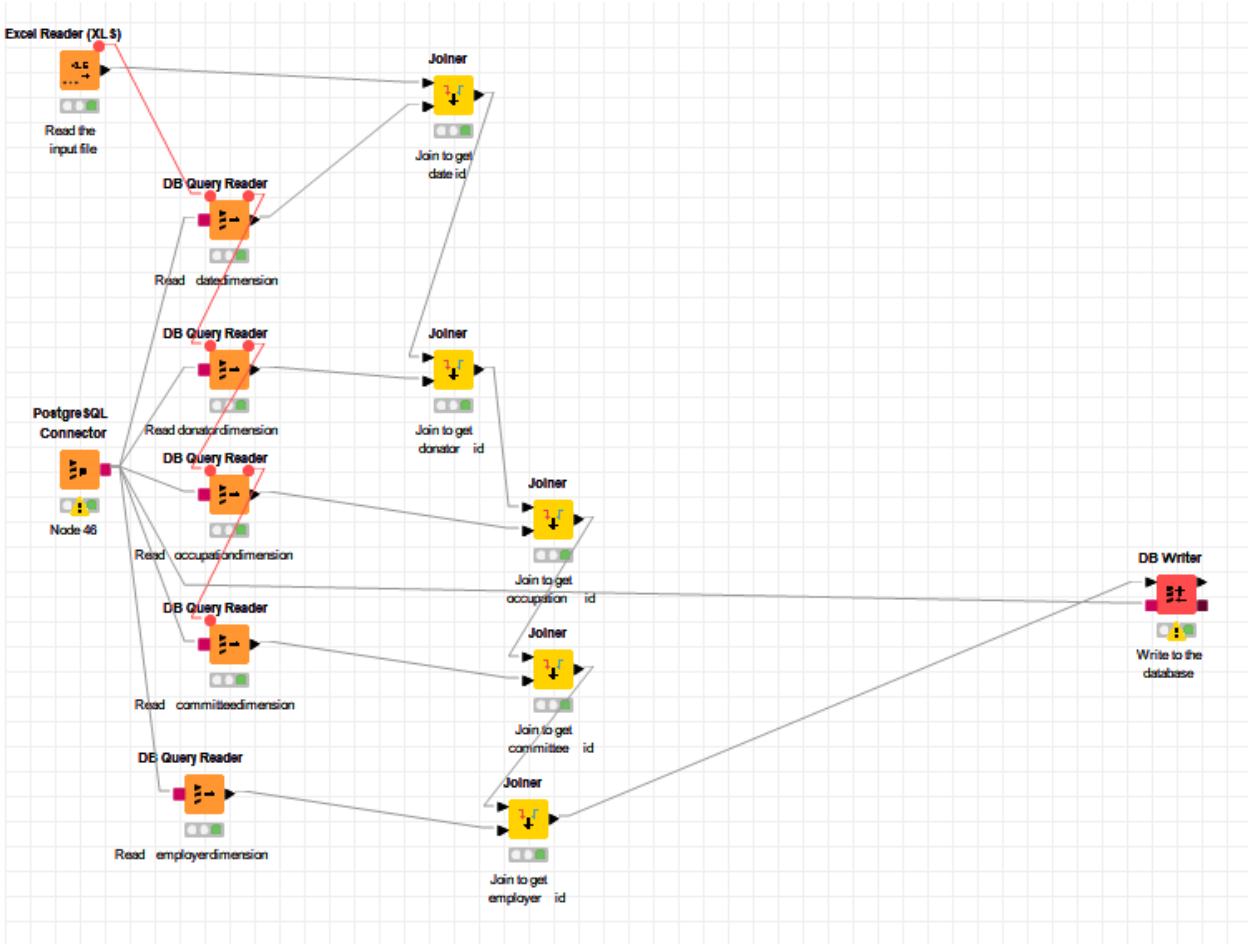
6. Column Rename will fix any naming issues to be congruent with our tables.

7. DB Writer will write the values to the database.

8. We review our database in DBeaver for accuracy prior to creating our fact table.

9. For our fact table, we use a similar process to above, but use the Joiner node to inner join our previously made tables (along with donation amount and aggregate amount).


Load data in the datedimension
Column Filter
GroupBy
Extract Date&Time Fields
Column Rename
DB Writer
Keep date column
Keep only unique values
Node 36
Rename column to match the one in database table
Write to the database
Load data in the donatordimension
Column Filter
GroupBy
Column Rename
DB Writer
Keep donator columns
Keep only unique values
Rename column to match the one in database table
Write to the database
Load data in the occupationdimension
Excel Reader (XLS)
Column Filter
GroupBy
Column Rename
DB Writer
Read the input file
Keep occupation column
Keep only unique values
Rename column to match the one in database table
Write to the database
Load data in the committeedimension
PostgreSQL Connector
Column Filter
GroupBy
Column Rename
DB Writer
Node 22
Keep committee column
Keep only unique values
Rename column to match the one in database table
Write to the database
Load data in the employerdimension
Column Filter
GroupBy
Column Rename
DB Writer
Keep employer column
Keep only unique values
Rename column to match the one in database table
Write to the database



3.2 Hadoop Implementation

We will convert our excel files to .csv, and then upload them to a container as HDFS files. Below is the result of our uploaded csv files to the container.

```
hadoop fs -mkdir /user/root/project # create the project folder
hadoop fs -mkdir /user/root/project/input # create the input folder
hadoop fs -ls /user/root/project/input # check if folders were created
hadoop fs -put ./contributions_q1_2019_MA.csv /user/root/project/input/ALL.csv
hadoop fs -put ./contributions_q2_2019_MA.csv /user/root/project/input/ALL.csv
hadoop fs -put ./contributions_q3_2019_MA.csv /user/root/project/input/ALL.csv
hadoop fs -put ./contributions_q1_2019_NH.csv /user/root/project/input/ALL.csv
hadoop fs -put ./contributions_q2_2019_NH.csv /user/root/project/input/ALL.csv
hadoop fs -put ./contributions_q3_2019_NH.csv /user/root/project/input/ALL.csv
hadoop fs -put ./contributions_q1_2019_VT.csv /user/root/project/input/ALL.csv
hadoop fs -put ./contributions_q2_2019_VT.csv /user/root/project/input/ALL.csv
hadoop fs -put ./contributions_q3_2019_VT.csv /user/root/project/input/ALL.csv
hadoop fs -put ./contributions_q1_2019_ME.csv /user/root/project/input/ALL.csv
```

hadoop fs -put ./contributions_q2_2019_ME.csv /user/root/project/input/ALL.csv
hadoop fs -put ./contributions_q3_2019_ME.csv /user/root/project/input/ALL.csv

Below is our transformation using Pig, using CSV Loader. The following commands rename attributes as applicable, remove headers and missing values, and format our values to their applicable types to match our Hive table above.

DEFINE CSVLoader org.apache.pig.piggybank.storage.CSVLoader(); -- select CSVloader

data = LOAD '/user/root/project/input/*.csv' USING CSVLoader(',') AS (date:chararray, state:chararray, zip:chararray, occupation:chararray, committee_name:chararray, employer:chararray, amount:chararray, aggregate_amount:chararray);

dump data; - view the result

```
(9/19/2019,VT,5765,RETIRED,Donald J. Trump For President, Inc.,RETIRED,37.5)
(9/6/2019,VT,5101,ATTORNEY/AUTHOR,Donald J. Trump For President, Inc.,SMITH VALLIERE PLLC,2800)
(9/6/2019,VT,5101,ATTORNEY/AUTHOR,Donald J. Trump For President, Inc.,SMITH VALLIERE PLLC,2800)
(7/18/2019,VT,5751,RETIRED,Donald J. Trump For President, Inc.,RETIRED,30)
(9/2/2019,VT,5751,RETIRED,Donald J. Trump For President, Inc.,RETIRED,30)
(9/18/2019,VT,5751,RETIRED,Donald J. Trump For President, Inc.,RETIRED,30)
(9/12/2019,VT,5401,CARDIOLOGIST,Donald J. Trump For President, Inc.,U.V.M. MED. CENTER,75)
(9/26/2019,VT,5401,CARDIOLOGIST,Donald J. Trump For President, Inc.,U.V.M. MED. CENTER,150)
(9/4/2019,VT,5456,RETIRED,Donald J. Trump For President, Inc.,RETIRED,187.5)
(7/12/2019,VT,5457,RETIRED,Donald J. Trump For President, Inc.,RETIRED,75)
(7/31/2019,VT,5457,RETIRED,Donald J. Trump For President, Inc.,RETIRED,15)
(8/14/2019,VT,5457,RETIRED,Donald J. Trump For President, Inc.,RETIRED,56.25)
(8/22/2019,VT,5457,RETIRED,Donald J. Trump For President, Inc.,RETIRED,37.5)
(8/28/2019,VT,5457,RETIRED,Donald J. Trump For President, Inc.,RETIRED,22.5)
(9/5/2019,VT,5457,RETIRED,Donald J. Trump For President, Inc.,RETIRED,75)
```

data = foreach data generate date as date_, state, zip, occupation, committee_name, employer, amount, aggregate_amount;

data = FILTER data BY date_ != '';

data = FILTER data by date_ != 'date';
data = FILTER data BY state != '';

data = FILTER data by zip != '';

data = FILTER data BY occupation != '';

data = FILTER data BY committee_name != '';

data = FILTER data BY employer != '';

data = FILTER data BY amount != '';

data = FILTER data BY aggregate_amount != '';
ranked_data = rank data;

data_first_rows = Filter ranked_data by ($0 < 10);

dump data_first_rows; -- inspect first 9 rows

```
(1,3/4/2019,MA,2108,Investor,Hickenlooper 2020,Williams Ireland Family Investments,1500)
(2,3/25/2019,MA,1776,Consultant,Hickenlooper 2020,Waterville Consulting,500)
(3,3/4/2019,MA,2142,Not Employed,Hickenlooper 2020,Not Employed,2800)
(4,3/5/2019,MA,2121,Dentist,Hickenlooper 2020,Dr Malkemus Group,2800)
(5,3/4/2019,MA,2138,Healthcare Consultant,Hickenlooper 2020,Matthias J Vinikas,1000)
(6,3/11/2019,MA,2492,Dog Walker,Hickenlooper 2020,Self Employed,500)
(7,3/13/2019,MA,2445,Chief Scientific Officer,Hickenlooper 2020,Vertex Pharmaceuticals,2000)
(8,3/21/2019,MA,2110,Owner,Hickenlooper 2020,Arter Financial Strategies-  Lincoln F,300)
(9,3/25/2019,MA,2462,Biotechnology Executive,Hickenlooper 2020,Q-State Biosciences,500)
```

data = foreach data generate date_, state, (INT)zip, occupation, committee_name, employer, (FLOAT)amount, (FLOAT)aggregate_amount;

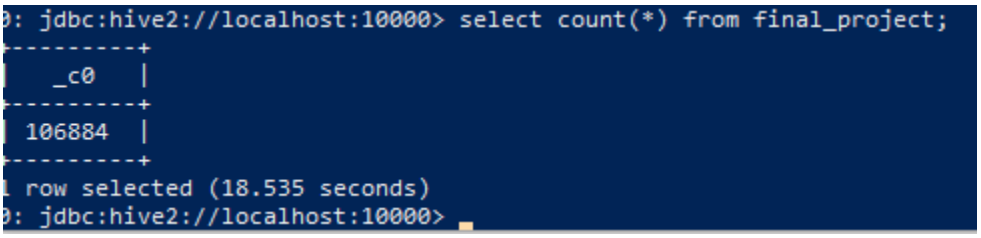
#storing data using PigStorage
STORE data INTO '/user/root/project/output' USING PigStorage (',');

```
root@7ffb7eadf251:/# hadoop fs -ls /user/root/project/output
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/usr/local/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.25.jar!/
.class]
SLF4J: Found binding in [jar:file:/usr/local/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/Stat
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
Found 2 items
-rw-r--r--   1 root supergroup          0 2021-08-10 03:04 /user/root/project/output/_SUCCESS
-rw-r--r--   1 root supergroup    7833434 2021-08-10 03:04 /user/root/project/output/part-m-00000
root@7ffb7eadf251:/#
```

bl

load data local inpath '/tmp/result_project.csv' overwrite into table final_project;

select count(*) from final_project;



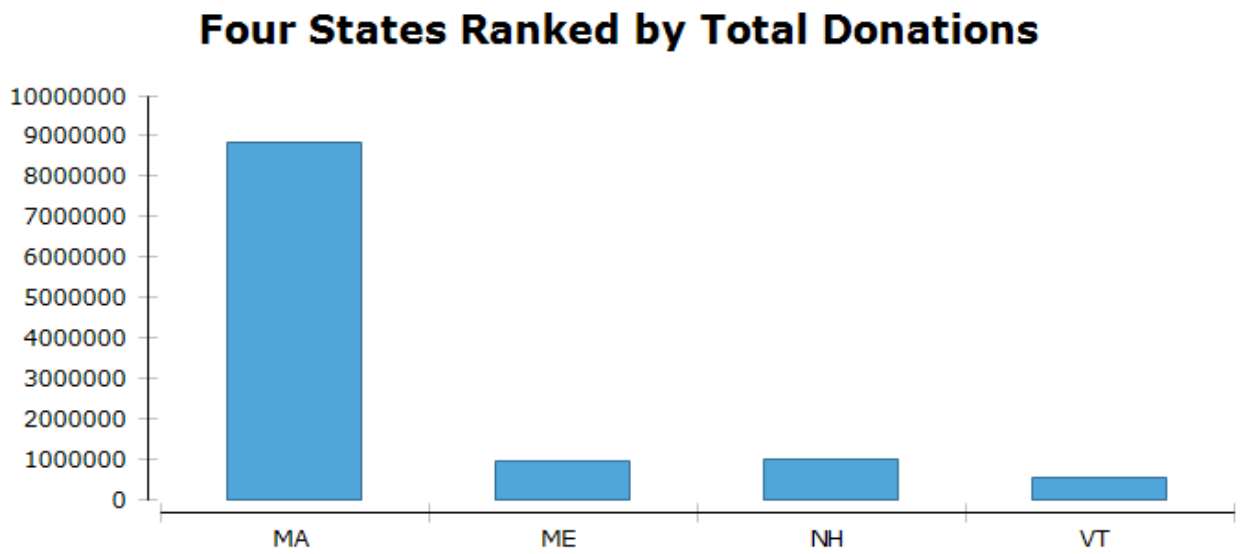
3.3 Reflective analysis of data preparation in relational data warehouse vs Hadoop.

ETL in the relational database inherently requires human input and decision making. It is a tedious process (even) with KNIME, requiring preprocessing decisions at every level. Data issues such as null values and incongruent data exist in these datasets. For example, the user must decide what to do with dates in our input tables that are out frame (Q1, Q2, Q3 of 2019). Or possibly combining similar professions. For this project, these values were not dropped but another user might easily remove these (additional filtering is done at the reporting stage below). Thus, different users could inherently get different results and conclusions that might radically influence the end users we are “selling” to.

However, ETL in Hadoop was processed by Apache Pig, which automates much of the above referenced preprocessing. Pig Latin uses an SQL-like language that can be adapted easily from our data warehouse usage previously. Its flexibility in schema design is an obvious advantage as well. We had 33739 unique rows in our filtered table from our warehouse, while Pig reduced this amount to 10884. This reduces size requirements for hosting data, while also speeding up queries (in addition to Hadoop’s other velocity improvements). One disadvantage however, is the inability to add additional fields not already existing in the data being inputted into Hadoop.

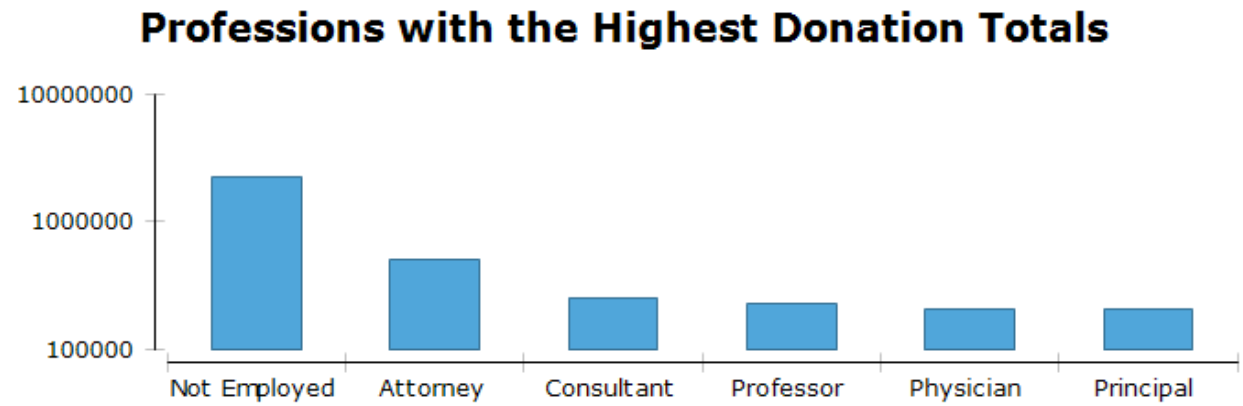
4. Reporting System

1) How are the four states ranked by total donations?



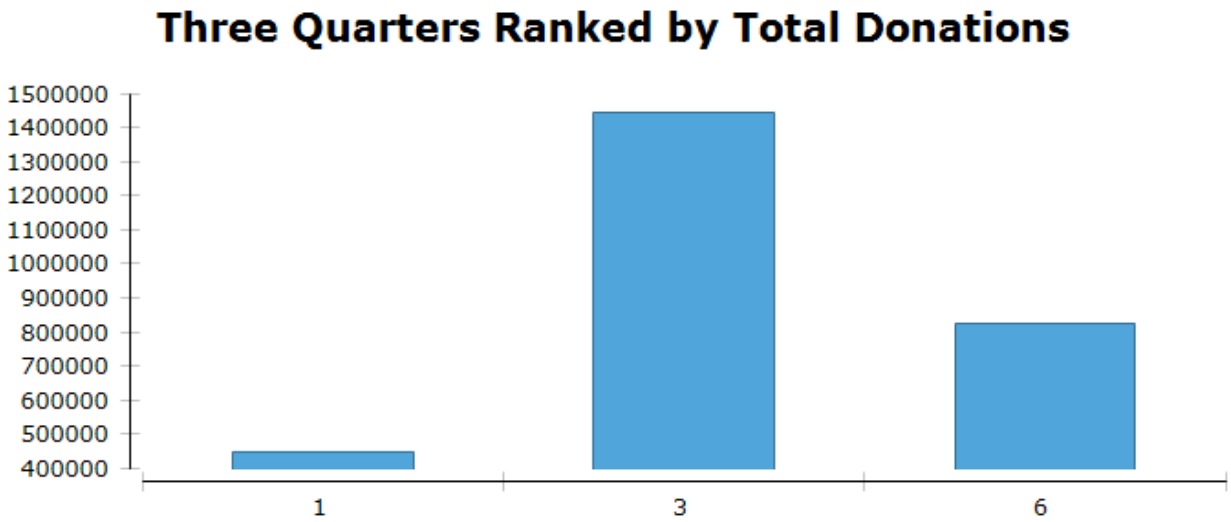
Massachusetts has by far the highest donation total. However, more insight might be gleaned by average donation amounts by state, as Massachusetts might have more donators and wealthier demographics. Regardless, it shows the importance of a campaign focusing on this state.

2) What professions have the highest donation amounts among the four states?



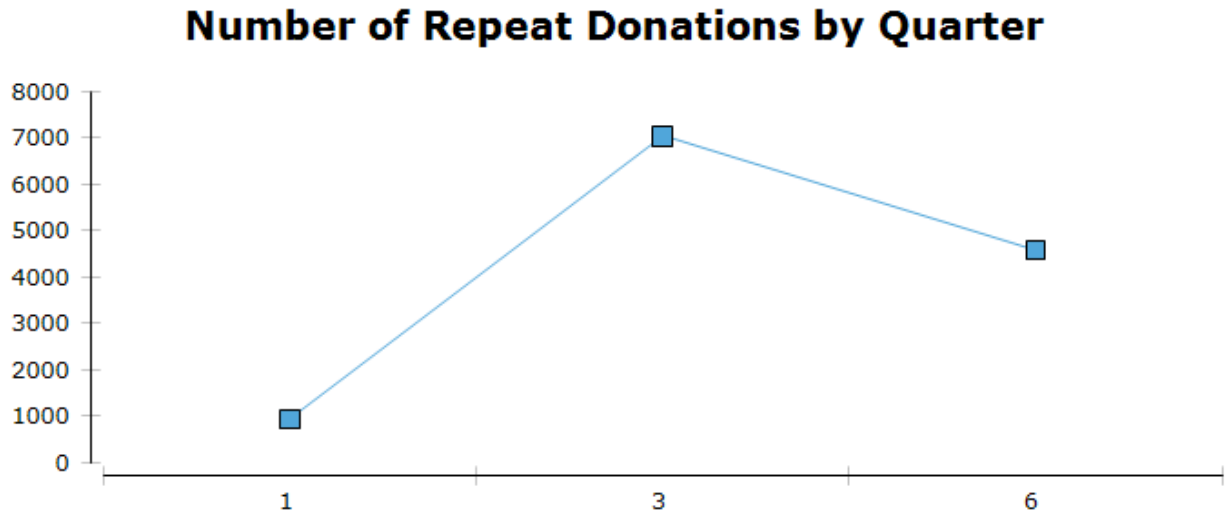
"Not Employed" was by far the highest "profession" listed, although these donators likely have wide-ranging careers. Attorney follows, but the differences among the other top six professions aren't very visible. However, this insight still shows that concentrating campaigns on a single occupation is likely not going to provide a significant benefit to overall donations.

3) How are the three quarters ranked by total donations?



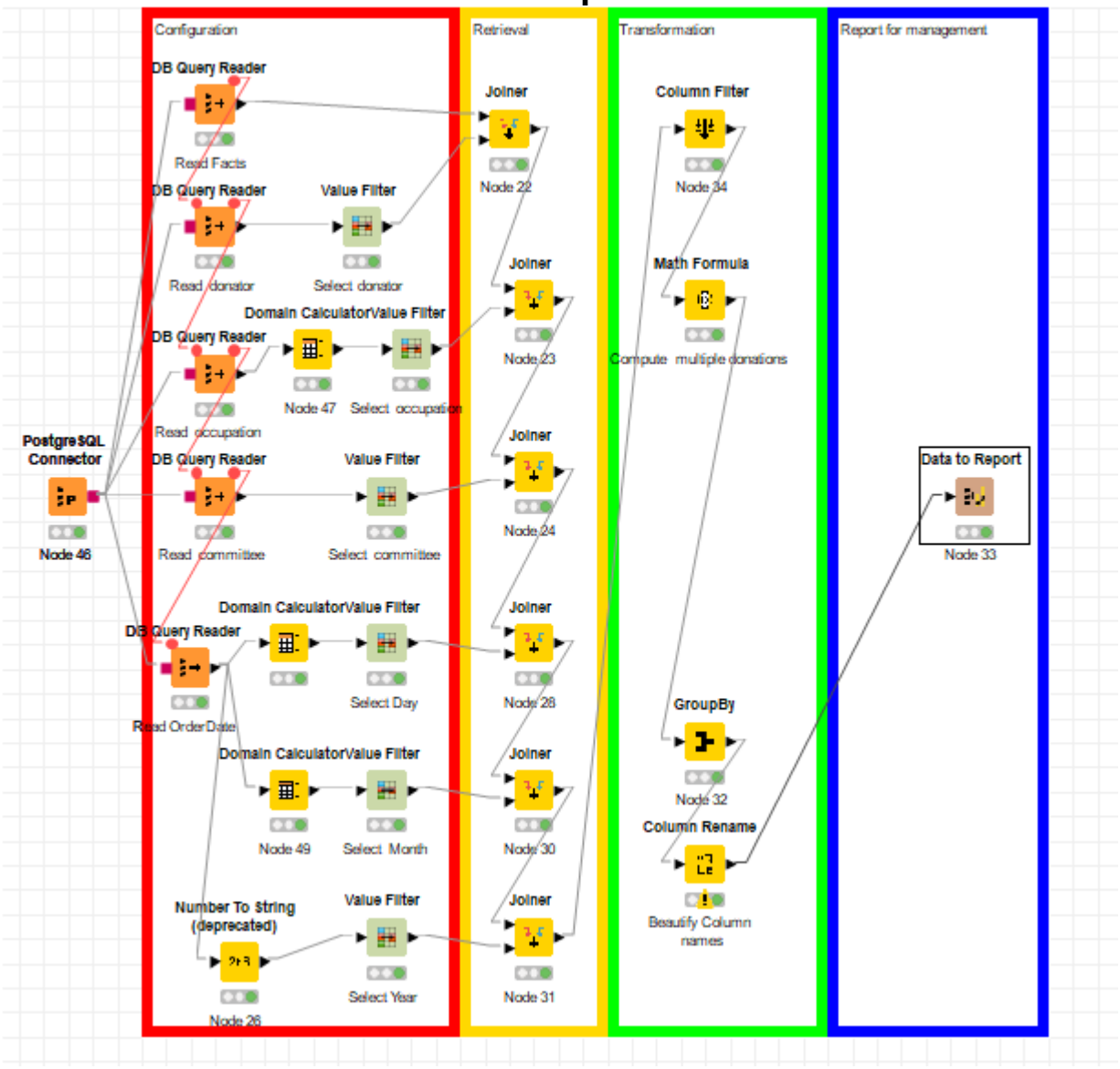
Due to labeling issues on the X axis, 1=Q1, 3=Q2, and 6=Q3. Quarter 2 has by far the most donations. We might review the amount of political advertising spent in these time periods to see if causation exists (outside of this dataset). Regardless, we might then recommend increasing political advertisements in quarters 1 and 3 to gain additional revenue for a candidate.

4) How many individuals make multiple donations?



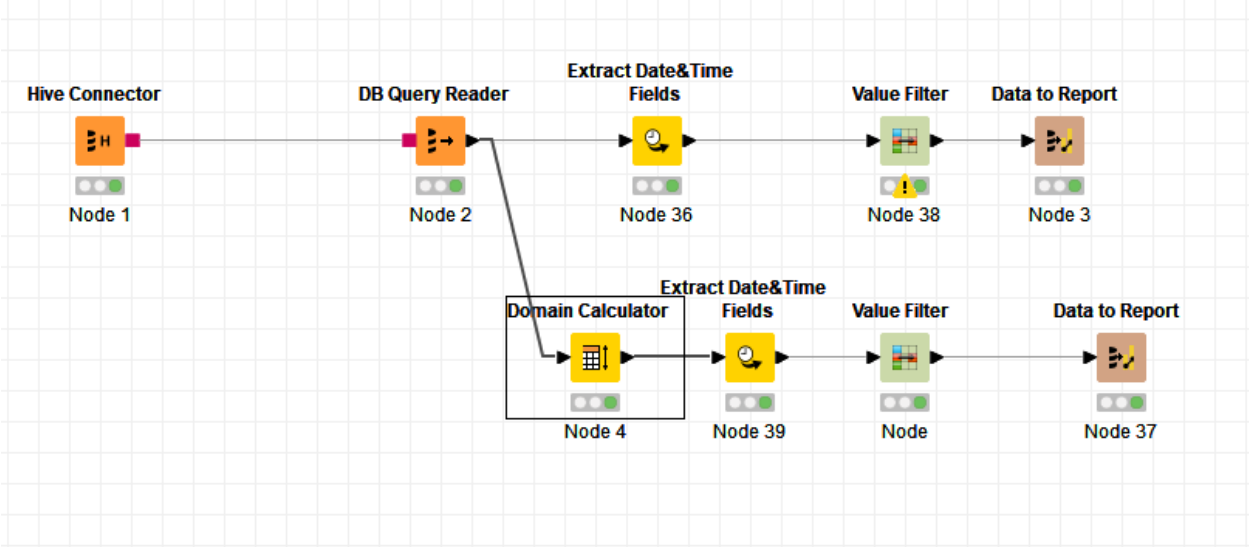
Due to labeling issues on the X axis, 1=Q1, 3=Q2, and 6=Q3. Quarter 2 has the most donators making multiple donations. This is unsurprising since Quarter 2 had the most donations. Out of 33739 unique rows in our filtered table, approximately one third included repeat donators. We might advocate that campaigns concentrate on first time donators to increase donations.

4.1 Relational Data Warehouse Implementation

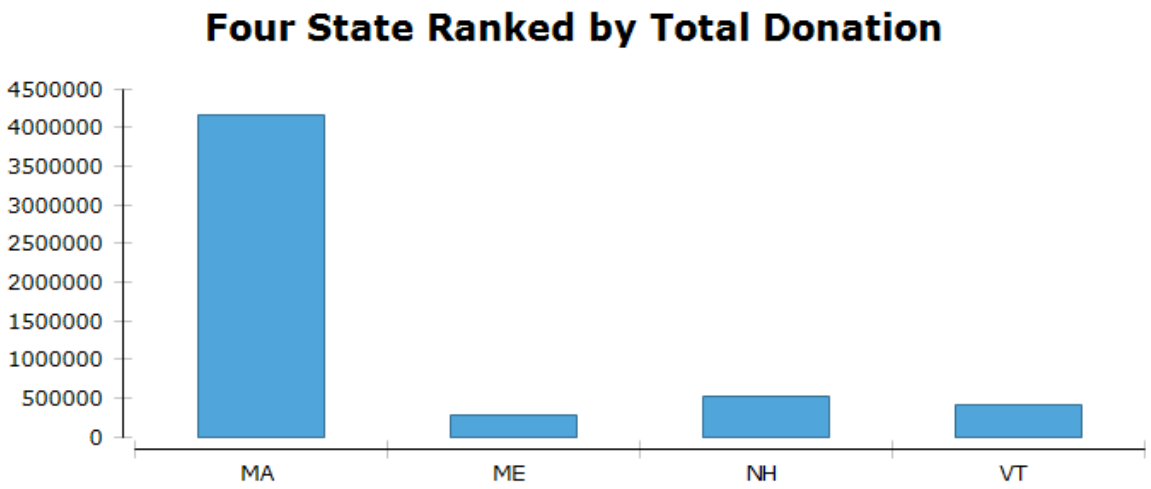


4.2 Hadoop Implementation

We will again use Knime to address our business questions, with this workflow quite a bit simpler than above. Essentially, our Hive Connector brings our database into Knime, with DB Query Reader, with Extract Date & Time (for quarterly data) and value filters added. Our Data Cubes provide for additional aggregation in creating our reports below.

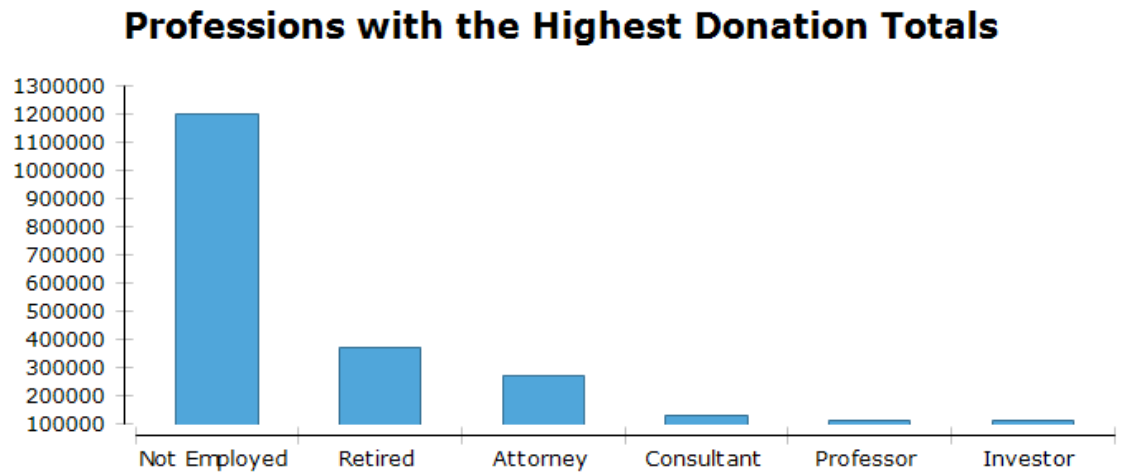


1) How are the four states ranked by total donations?



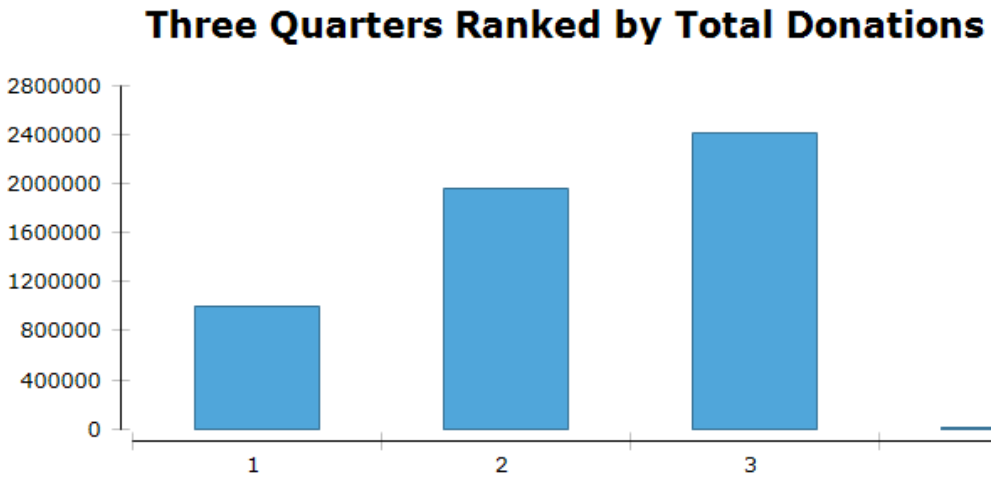
Massachusetts again has by far the highest donation total. No noticeable differences exist between this result and the one produced by the data warehouse.

2) What professions have the highest donation amounts among the four states?



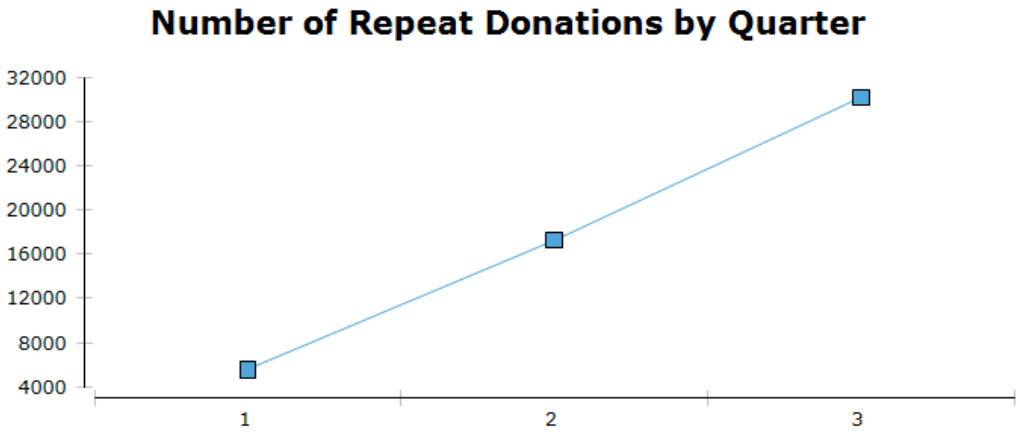
"Not employed" is still the highest "profession" listed, however, we now see "Retired" come into the fold. Hadoop's ETL has radically altered both what professions are filtered to the top, as well as their sum totals. However, the conclusions we might gather from this is that Attorneys are still a leading candidate for donation, followed still by Consultants and Professors.

3) How are the three quarters ranked by total donations?



What a difference Hadoop makes! This time, Quarter 3 has the highest amount of donations, followed gradually by quarters 2 and 1. This is likely due to differences in preprocessing. For example, negative donations would have been processed differently in the two reports.

4) How many individuals make multiple donations?



Like above, Hadoop has radically altered our outputs. I again imagine some preprocessing needs to happen to see what errant values might exist, especially around quarter 3.

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

KNIME allows us to read the relational database from DBeaver and provides us with visually stimulating and easy-to-read reports. However, because of previously mentioned ETL user decisions, along with decisions also made at the report level, a level of bias exists to our results. Our first two business questions appear unaffected by the different processing natures (likely due to our large scale). I would like to know more about why my results were so skewed in business questions 3 and 4. I tend to believe this is due to ETL, rather than input error, as I reloaded the data into Hadoop and received the same results.

For practical purposes, the same unanswered or new business questions exist in both databases. Assuming practically accurate results, Hadoop's speed and ease of use cannot be understated here.

Conclusions

The data warehouse we created successfully answers our business questions and provides the user insights into the election datasets. Our reporting and analysis has created even more questions to be asked and answered by the data. Additional data and political science analysis not available at our input would be helpful as an addition to our warehouse, and in answering these questions even as a business understanding reference.

In conclusion, the amount of steps (and troubleshooting) needed to have our warehouse working for our business needs caused a much longer implementation period (time and

resources) from a developer’s standpoint when compared to achieving the same goals in Hadoop.