# Penn State University
# Great Valley Campus

*Engineering Division*

## Data Specification for
## *Data Warehouse – Part 1*

***Version 1.0***

Date started - 09/15/2023

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

# Table of Contents

# Data Warehouse using PostgreSQL
# by Eric Brown

## INTRODUCTION

A data warehouse is a centralized repository for storing and managing large volumes of structured data from various sources within an organization. Thus, making it an integral part of an information system. Its primary purpose is to support business intelligence (BI) and data analytics activities by providing a structured and optimized environment for data storage, retrieval, and analysis. A data warehouse is typically made of several data sources, some of them remote. Though, data warehouses attempt to minimize their remote query selections.

While data warehouses bear many benefits from their overall data design infrastructure. One challenge data warehouses face happens during the design phase, which involves determining the correct views to materialize. Since, data warehouses have a very large, but finite storage capacity. So, it is not normally possible to store all data inside of them locally. This is why views included in data warehouses have to be chosen in manner that can answer queries of interest in a relatively fast time frame.

## PURPOSE

Overall, the scope of the term project is to compare two different types of information systems. One being a data warehouse and one being a big data application framework called Hadoop. In this project we will focus on the creation and analysis of the data warehouse system using PostgreSQL.

## PROJECT SUMMARY

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

### A. Objectives

[Provide a concise description of the objectives of the document.]

Is to use the data from 2020 election contributions from the Propublica website to answer questions about the trends in election contributions amongst political parties.

### B. Scope

[Briefly describe the scope of the requirements specification.]

*The data requirements include the following:*

- contributions must be for at least two candidates from each party
- contributions must be from at least four states of your choice
- contributions must be from three time periods which are to be loaded separately. These time periods can be weeks, months or quarters.

### C. References

[Identify sources of information used to develop this document, such as IEEE or project documentation.]

- Project 1 Documentation
- Propublica
- PostgreSQL Documentation
- DBeaver Documentation
- KNIME Documentation

### D. Outstanding Issues

[Provide a succinct list of issues or problems which are known to be outstanding with this revision.]

## REQUIREMENTS DEFINITION

### A. Goals

[[Provide a clear list of the expectations of the new application(s), both in terms of what must be improved and what must be retained from the current processes. All detailed requirements should address one or more of these goals]].

To provide a relational database management system (RDBMS) where the historical election data is stored in a star schema. And the necessary method to perform the ETL (Extract, Transform, Load) or CRUD (create, retrieve, update, delete) process of this system will be conducted through standard SQL commands. In conjunction to the use of Knime's as a tool to make the ETL process more efficient and less cumbersome. Knime has a wide variety of connectors and data transformation nodes to automate the ETL process.

Another goal/requirement for this project is to use the freeware open-source relational database management tool called PostgreSQL. This software will be used because it is free to use, and its freeware nature makes it consistent with Hadoop. Which is a tool that is planned to be used in the next phase of the project. Also, PostgreSQL is highly extensible in terms of custom data types, functions, and operators. While also being ACID compliant in terms of data integrity and reliability. And being highly compatible across various types of operating systems. Whether it be used on Windows, macOS, or Linux, PostgreSQL runs in the same manner and provides the same security measures to protect data from unauthorized access across all operating systems.

A third goal of this project is to use PostgreSQL adherence to SQL standards to generate analysis on the historical data given in this project. This analysis would be conducted through different SQL commands and Knime's Report Writer tool ad-on to answer various prescribed business questions about the elections of 2020.

The last goal for this project is to use PostgreSQL as a freeware relational database management tool to stay consistent with the freeware nature of Hadoop. Mainly because the Hadoop software is to be used in the next portion of this project. Which involves using data warehouse systems to analyze big data.

### B. Usability Requirements

[[Specify the requirements associated with ease of use, query capabilities, report layouts, online help and other interfaces to users and/or supervisors]].

***<u>Query Capabilities:</u>***

- PostgreSQL supports SQL (Structured Query Language) and provides a wide range of SQL features for querying and manipulating data. For example, users can perform complex queries on multiple tables and perform subqueries, joins, and aggregations on multiple tables
- PostgreSQL supports the creation and execution of stored procedures and user-defined functions using various procedural languages such as PL/pgSQL, PL/Python, and PL/Java
- PostgreSQL also allows for Window functions. Which is important because window functions do not cause rows to become grouped into a single output row — the rows retain their separate identities. Behind the scenes, the window functions can access more than just the current row of the query result. Which allows for advanced data analysis. As well as reporting by partitioning and ordering data within result sets.
- PostgreSQL also supports various indexing techniques, including B-tree, Hash, GIN, GiST, and SP-GiST, which can enhance a query's performance

***Report Layouts:***

- By itself PostgreSQL does not provide report layout capabilities. However, one could create custom reports by designing SQL queries to extract and format data in a manner that would be shown on a report
- Reporting tools like JasperReports, Tableau, or Power BI can be integrated with PostgreSQL to design and generate reports with advanced layout features
- You can use tools like LaTeX, in conjunction to PostgreSQL to generate formatted documents from database queries if you need complex report layouts
- In conjunction to PostgreSQL, Knime will be used mainly for construction the design for the report segment of this project
- Report Designer is an extension of the KNIME Analytics Platform. It is used to create custom reports and documents based on the data preprocessed through Knime.
- The Report Designer uses concepts of Business Intelligence and Reporting Tools (BIRT) to generate the overall structure of the report through two segments. The Master Page, which contains information for headers, footers, and titles common to every page in the report. And the Layout, which details how tables, charts, images, text, and other items are arranged.
- One could also bind report elements to one's KNIME data. Meaning the content of one's report could be dynamically updated as one's data or analysis changes. This is particularly useful when you have regularly updated data and want to automate the reporting process.

***Online Help:***

- PostgreSQL has extensive online documentation available on its official website. This documentation covers installation, configuration, SQL reference, and more
- One tool that could be used as help is PostgreSQL's pg_dump utility. It could be used to generate a SQL script containing the schema and data of a given database, which can serve as documentation to one's database
- The KNIME Analytics Platform has many online resources for troubleshooting, learning, and community engagement. Here are some of the common online support resources for KNIME: KNIME Forum, KNIME

Documentation, and KNIME Hub. As well as KNIME Blog and KNIME events/webinars

**<u>Interfaces to Users:</u>**

- PostgreSQL provides a command-line tool called psql that allows users to interact with the database using SQL commands through the command line
- There are several GUI tools available, such as pgAdmin and DBeaver that provide a user-friendly interface for interacting with a PostgreSQL database
- Developers can integrate PostgreSQL into applications using programming languages like Python, Java, and PHP by using appropriate libraries and drivers
- Web applications can be built using frameworks like Django, Ruby on Rails, or Node.js, with PostgreSQL as the backend database
- Users can connect to PostgreSQL databases through ODBC (Open Database Connectivity) or JDBC (Java Database Connectivity) drivers, enabling database interaction from various applications
- PostgreSQL has both web and mobile accessibility through REST API
- KNIME has various user interfaces for users to interact with the platform as well. These interfaces include KNIME’s desktop interface, webportal, executor, and REST API

**<u>Additional Requirement to ease computer memory:</u>**

1.) On a Windows computer, if one sets docker with WSL, one needs to set some reasonable resource constraints on what WSL2 can actually use.

2.) Create a .wslconfig file (no extension) with the following:

- memory=4GB

***IMPORTANT: no more than half of the ram in your computer, minimum 2.5 GB***

- processors=2

***IMPORTANT: no more than half the virtual processors you have***

3.) To confirm these optimization changes, one should Open Windows PowerShell with admin rights, restart WSL2 by typing, *Restart-Service LxssManager*

## C. System Security Requirements

[[Provide details of the security classification of the data handled by the system, special handling required for the data, and the types and levels of protection and control required for user access to the data]].

A Docker image runs a PostgreSQL database server by:
(1) pulling the base of a Linux distribution such as Ubuntu
(2) modifying it to include the PostgreSQL database and all other dependencies
(3) making a new image out of that can be pushed to the Docker Hub

Once Docker creates the image, it can be used to create a container. Then the container is created in the user's file system and a read-write layer is added to the image. Also, the creation of a docker file can allow a network interface to be created to allow the local host to be connected through a user's port. This way an IP address is attached to the container and PostgreSQL database can run through the command line. In our project the local host will be through port 49168 on the user's local machine.

***In terms of Third-Party Extensions:***
Security measures will be in place based on the initial connection to DBeaver through the following authorization.
***Host:*** localhost
***Port:*** *49168*
***Database/Schema:*** student
***User name:*** student
***Password:*** student

## D. Business Questions

[[Identify the business questions that should be answered by the software product. Include any prioritization of these requirements]].

1.) Which campaigning party had the highest contributions in terms of dollars between the four states of California, New Jersey, Florida, and Arizona

2.) Which campaigning party had the lowest contributions in terms of dollars amongst the four states of California, New Jersey, Florida, and Arizona

3.) Which state had the highest contributions to the top two campaigning parties

4.) Contributions made in California, New Jersey, Florida, and Arizona aggregated monthly in Q1 during the year of 2019

5.) Which company made the most contributions in terms of dollars to the top campaigning party

6.) What was the overall demographic makeup of the contributors of each of the top campaigning parties

## E. Data Requirements

[[Identify the data elements and logical data groupings that will be required to answer the business questions above. Include any archiving requirements and the projected level of effort. This section may be supported by a data model and a data dictionary which should be included in an appendix]].

***Typically, there are four steps in developing a data warehouse:***

1. ***Select the business process:*** *This step is completed by converting the database schema into to a set of dimensional models for each discrete business processes.*

   The main question for this project would be to provide information regarding contribution totals for each of the campaigning parties during different time periods of the 2020 election.

2. ***Declare the grain:*** *This step identifies exactly what each record in the fact table represents. For example, we may be interested in daily sales figures.*

   For this project, we will choose monthly aggregation totals. This implies that we need to aggregate all our data sharing the same dimensions in the same month.

3. ***Identify the dimensions:*** *This step identifies those aspects of the facts that are interesting for the analytical process.*

   **filing_id** = Normally, we would use this as a degenerate dimension but it is not aggregrated sequentially, so it is irrelevant data

   **tran_id** = We can use this as a degenerate dimension

   **linenumber** = This a dimension that is irrelevant, since it has information that has no importance to our business objectives/findings

   **flag_orgind** = This a dimension that is irrelevant, since it has information that has no importance to our business objectives/findings

   **date** = This is a good dimension that define the grain of our data

**cycle** = Could be used a dimension to define time period, but it is a static data point

**employer** = This is a good dimension if we want to identify which employer are active in giving campaign contributions

**occupation** = This is a good dimension if we want to identify which occupation are active in giving campaign contributions

**committee_name** = This is an important dimension because it identifies which candidates a given contributor supports in the 2020 elections

**org_name** = This is an irrelevant dimension since it has no data list inside of it

**address_one** = This is a dimension that I will consider irrelevant in our case because we want to look only at trends and not at predicting individual behavior

**address_two** = This is a dimension that I will consider irrelevant in our case because we want to look only at trends and not at predicting individual behavior

**last_name** = This is a dimension that I will consider irrelevant in our case because we want to look only at trends and not at predicting individual behavior

**first_name** = This is a dimension that I will consider irrelevant in our case because we want to look only at trends and not at predicting individual behavior

**middle_name** = This is a dimension that I will consider irrelevant in our case because we want to look only at trends and not at predicting individual behavior

**prefix** = This is a dimension that could provide information regarding which gender contributed the most during the 2020 elections. But I would still consider it irrelevant for our business questions established previously. Also, there are many null values in this dimension

**suffix** = This is a dimension that could provide information regarding which profession contributed the most during the 2020 elections. But I would still consider it irrelevant for our business questions established previously. Mainly because most of the values in this dimension are null values

**state** = This is a good dimension if we want to identify which part of the country supports a particular type of candidate

**zip** = This is a good dimension if we want to delve deeper into identifying which part of the country supports a particular type of candidate

**city =** This is a good dimension if we want to delve deeper into identifying which part of the country supports a particular type of candidate

**amount** = This is not a dimension but rather a fact

**aggregate_amount** = This is not a dimension but rather a fact. But I would probably not want to use this aggregate total because it aggregates the amount field in a different manner than what we need for our business questions

**memo_code** = This is a dimension that is irrelevant, since it has information that has no importance to our business objectives/findings

**memo_text** = This is a dimension that is irrelevant, since it has information that has no importance to our business objectives/findings

**back_ref_tran_id** = This is a dimension that has similar information to another dimension in our database, tran_id. So, I would consider it not necessary to include as another dimension in our data warehouse

**back_ref_sched_name** = This a dimension that is irrelevant, since it has information that has no importance to our business objectives/findings

**prigen** = This a dimension that is irrelevant, since it has information that has no importance to our business objectives/findings

**fecid** = This a dimension that is irrelevant, since it has information that has no importance to our business objectives/findings

*In conclusion we should create a star schema with eight dimensions:*

- Tran_id
- Date
- Employer
- Occupation
- Committee_name
- State
- Zip
- City

For each of these dimensions we will create a table containing an id and a description field.

4. ***Identify the facts:*** *This step identifies what actionable data we need to store.*

**Amount** = is the main fact that we are trying to create our analysis around; total monetary contributions that were made to each political party

## F. Design Constraints

[[Document any design constraints that should be taken into consideration during the system design phase]].

There must be at least three tables loaded into the PostgreSQL database. Create a star schema with eight dimensions:

- Tran_id
- Date
- Employer
- Occupation
- Committee_name
- State
- Zip
- City

## CONSIDERATIONS

(May include as separate document)

## DOCUMENT CHANGE LOG

| Change Date | Version | CR # | Change Description | Author and Organization |
|---|---|---|---|---|
| 09/24/23 | 1.0 | | Initial creation | Eric Brown |
| 10/05/2023 | 2.0 | | Add more table details about data warehouse | Eric Brown |
| 10/11/2023 | 3.0 | | Add schema details about data warehouse | Eric Brown |
| 10/20/2023 | 4.0 | | Add information about ETL process | Eric Brown |
| 10/14/2023 | 5.0 | | Add information regarding the report design and discoveries learned | Eric Brown |
| 10/15/2023 | 6.0 | | Final edits for Project 1 submission | Eric Brown |

# 2. Architecture Design

## 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.

The relational database was made up of four tables that had different contribution information inside of them. The four tables were based on the first quarter contribution results amongst four states. These states were Arizona, New Jersey, California, and Florida. All four tables were organized in the same way and listed below was generally how they were structured.

| **Contribution Table – Data Dictionary** | | | | |
|---|---|---|---|---|
| ***Variable*** | ***Variable name*** | ***Variable type*** | ***Values*** | ***notes*** |
| Filling Id | filing_id | String | SA17A, SA18, SB28A | Can not be null |
| Line number | linenumber | String | SA17A, SA18, SB28A | Can not be null |
| Organization type | flag_orgind | String | IND | |
| Organization Name | org_name | String | ADP, 1144 Summit Ave, LLC, Atlantic Agri Imports, LLC | Has mostly null values |
| Contributor Last name | fname | String | Doe | Can not be null |
| Contributor First name | lname | String | John | Can not be null |
| Contributor middle name | middle_name | String | Bruce | |
| Professional abbreviations | prefix | String | Dr., Mrs, Mr, Judge | |

| **Contribution Table – Data Dictionary** | | | | |
|---|---|---|---|---|
| Professional & Personal abbreviations | suffix | String | III, Jr., JR., M.D. | |
| Address | address_one | String | 12 Larson Rd | |
| Second Part of Address | address_two | String | Unit 45, Suite 100 | |
| City of Contributor | city | String | Phoenix, Los Angeles, Newark, Orlando | |
| State of Contributor | state | String | California, New Jersey, Florida, Arizona | must have a California, New Jersey, Arizona, or Florida state String value |
| Zipcode of Contributor | zip | String | 85327 | |
| Employer of Contributor | employer | String | Stagecoach Digital, City University Of New York- John Jay, etc.. | |
| Occupation of Contributor | occupation | String | Engineer, Mechanic, Software programmer, etc… | |
| Dollar amount of contribution | amount | Numeric | 2800, 1000, 26 | |
| Date of contribution | date | Date/Time | 3/6/2019, 3/5/2019, 3/31/2019, etc… | |

| **Contribution Table – Data Dictionary** | | | | |
|---|---|---|---|---|
| Aggregated Amount contributed by contributor | aggregate_amount | Numeric | 2800, 262.5 5000 | |
| Memo code | memo_code | String | Null, X | Mostly null values |
| Memo text | memo text | String | * Earmarked Contribution through ACTBLUE on 01/13/2019, SEE REDESIGNATION | |
| Transaction Id of contribution | tran_id | Integer | 819053, 815124, 815077 | |
| Additional transaction Info | back_ref_tran_id | String | 1688351E, 169662E, 1691257E | |
| Campaign Contribution Year Id | prigen | String | P2020, G2020, Blank | |
| Campaign Year | cycle | Date | 2020 | Must be for year 2020 |
| Id of Committee | fecid | String | C00698258, C00694455 | |
| Committee Name | committee_name | String | Kamala Harris For The People, Bernie 2020, Friends Of John Delaney | |

## *Tables schemas*

Provide a description of the physical schema of the data warehouse. Use the steps in the lesson and explain.

| | | | |
|---|---|---|---|
| *Name of the table* | State | | |
| Description | This table describes information regarding each state of the contributors. | | |
| Attribute | Description | Type | *Examples of values* |
| id | Id of each unique state | Integer | Between 1 and 12000 |
| name | Name of state | String | California, New Jersey, Arizona, Florida |
| Primary Key | id | | |
| Foreign Keys | NA | | |

| | | | |
|---|---|---|---|
| *Name of the table* | City | | |
| Description | This table describes information regarding each city of the contributors. | | |
| Attribute | Description | Type | *Examples of values* |
| id | Id of each unique city | Integer | Between 1 and 12000 |
| city | Name of the city | String | Los Angeles, Key Largo, Tempe, Montclair |
| Primary Key | id | | |
| Foreign Keys | NA | | |

| | | | |
|---|---|---|---|
| *Name of the table* | Zipcode | | |
| Description | This table describes information regarding each zipcode of the contributors. | | |
| Attribute | Description | Type | *Examples of values* |
| id | Id of each unique zipcode | Integer | Between 1 and 12000 |
| zip | Zipcode of the city | Integer | 33095, 7950, 85266, 90274 |
| Primary Key | id | | |
| Foreign Keys | NA | | |

| | | | |
|---|---|---|---|
| *Name of the table* | Employer | | |
| Description | This table describes information regarding each employer of the contributors. | | |
| Attribute | Description | Type | *Examples of values* |
| id | Id of each unique employer | Integer | Between 1 and 12000 |
| employer | Employer name of each contributor | String | Town Of Prescott, Self Employed, Whole foods, Not Employed |
| Primary Key | id | | |
| Foreign Keys | NA | | |

| | | | |
|---|---|---|---|
| *Name of the table* | Occupation | | |
| Description | This table describes information regarding each occupation of the contributors. | | |
| Attribute | Description | Type | *Examples of values* |

| | | | |
|---|---|---|---|
| id | Id of each unique occupation | Integer | Between 1 and 12000 |
| employer_title | Occupation title of each contributor | String | Sales, Planner, Attorney, Investor |
| Primary Key | id | | |
| Foreign Keys | NA | | |

| | | | |
|---|---|---|---|
| *Name of the table* | Date | | |
| Description | This table displays all the contribution dates made by contributors. | | |
| Attribute | Description | Type | *Examples of values* |
| id | Id of each unique employer | Integer | Between 1 and 12000 |
| date | Contribution transaction date | Date | 9/30/2019, 10/1/2019, 11/4/2019 |
| year | year portion of the date value | | 2019 |
| quarter | quarter value of the year in which the date was in | | 1 to 4 |
| month | month portion of the date value | | 1 to 12<br>1 = January<br>12 = December |
| weekofyear | week value of the date | | ranging from 1 to 52 |
| dayofweek | day value of the date value | | ranging from 1 to 7<br>1 being equal to Monday & 2 being equal to Tuesday, etc.. |

| | |
|---|---|
| Primary Key | id |
| Foreign Keys | NA |

| *Name of the table* | Transactions | | |
|---|---|---|---|
| Description | This table displays all the contribution transactions made by contributors. | | |
| Attribute | Description | Type | *Examples of values* |
| id | Id of each unique contribution transaction | Integer | Between 1 and 12000 |
| tran_id | Contribution transaction reference | Integer | 1764024, 1964608, 454961 |
| Primary Key | id | | |
| Foreign Keys | NA | | |

| *Name of the table* | Committee_Name | | |
|---|---|---|---|
| Description | This table describes information regarding each committee in which the contributors made contributions. | | |
| Attribute | Description | Type | *Examples of values* |
| id | Id of each unique committee name | Integer | Between 1 and 12000 |
| committee_name | Names of the politicians that contributions were made to in the 2020 elections | String | Kamala Harris For The People, Bernie 2020, Friends Of John Delaney |
| Primary Key | id | | |
| Foreign Keys | NA | | |

| *Name of the Fact table* | AmountFacts | | | |
|---|---|---|---|---|
| Description | This table will store the fact variable - contribution amount in the data warehouse | | | |
| Attribute | Description | Type | *Examples of values* | *Notes* |
| stateid | State id | Integer | 10 | From the dimension table |
| cityid | City id | Integer | 150 | From the dimension table |
| zipcodeid | Zipcode id | Integer | 500 | From the dimension table |
| employerid | Employer id | Integer | 1000 | From the dimension table |
| occupationid | Occupation id | Integer | 25 | From the dimension table |
| dateid | Date id | Integer | 45 | From the dimension table |
| transactionid | Transactions id | Integer | 1200 | From the dimension table |
| committeeid | Committee id | Integer | 60 | From the dimension table |
| amount | Contribution amount by each contributor | Numeric(6,2) | -2838.81,<br>800.90,<br>600 | Already available |
| | | | | |
| Primary Key | Combination of stateid, cityid, zipcodeid, employerid, occupationid, dateid, transactionid, committeeid | | | |

| | |
|---|---|
| Foreign Keys | stateid REFERENCES id in State table<br>cityid REFERENCES id in City table<br>zipcodeid REFERENCES id in Zipcode table<br>employerid REFERENCES id in Employer table<br>occupationid REFERENCES id in Occupation table<br>dateid REFERENCES id in Date table<br>transactionid REFERENCES id in Transactions table<br>committeid REFERENCES id in Committee_Name table |

Please try use at least two types of input formats (SQl and text)

## 2.2 Hadoop Implementation

To be completed with project number 2

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

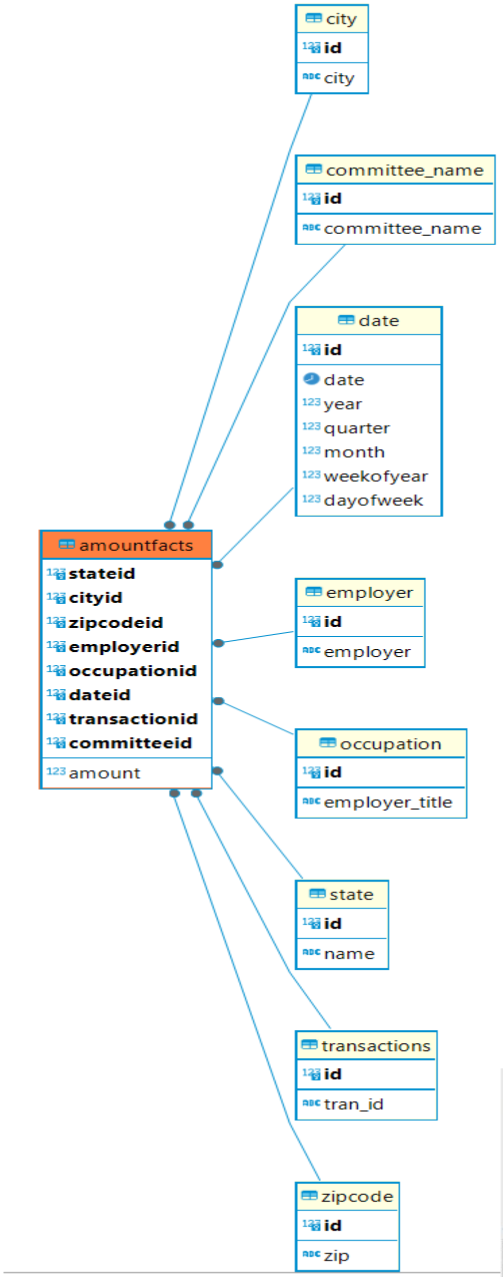
In project 1, please enter your reflective findings. Final version to be be completed with project number 2.

My reflective findings during the design stage of this project include some discoveries of data types which PostgreSQL allows in its data management system. These values included Boolean types, Character Types (char, varchar, and text), Numeric types (such as integer and floating-point numbers), Temporal types (such as date, time, timestamp), and serial types. I found these findings interesting because I had the assumption that all data management systems used similar data types. And that was the case overall, but there were still some features that I was not aware of that PostgreSQL provided. One of these features included the data type serial.

Before completing this project, I was used to creating auto incrementing columns in tables through AUTO_INCREMENT or using the Identity modifier to increment a column. However, in PostgreSQL the serial data type is used to automatically increment a column value, so it preforms like a primary key. It is able to do this through the use of a regex expression that uses an auto generated sequence to increase the column's value sequentially.

I also found the use of indexing relevant in the design phase of data warehouses too. Since, I learned that indexing plays a crucial role in enhancing the performance and efficiency of database operations. Especially when one is designing tables in a data warehouse to handle data from large datasets. In this project I mainly used indexes on columns that served as primary keys or had unique constraints. This way PostgreSQL's data management system could have fast retrieval times for specific rows in these columns.

## Indexing Values in tables:



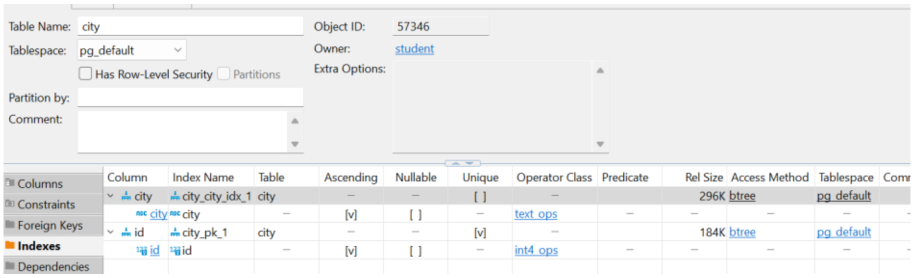


Table Name: city
Tablespace: pg_default
□ Has Row-Level Security □ Partitions
Partition by:
Comment:
Object ID: 57346
Owner: student
Extra Options:

| Column | Index Name | Table | Ascending | Nullable | Unique | Operator Class | Predicate | Rel Size | Access Method | Tablespace | Comm |
|---|---|---|---|---|---|---|---|---|---|---|---|
| city | city_city_idx_1 | city | – | – | [ ] | – | | 296K | btree | pg_default | |
| city | city | – | [v] | [ ] | – | text_ops | – | – | – | – | - |
| id | city_pk_1 | city | – | – | [v] | – | | 184K | btree | pg_default | |
| id | id | – | [v] | [ ] | – | int4_ops | – | – | – | – | - |

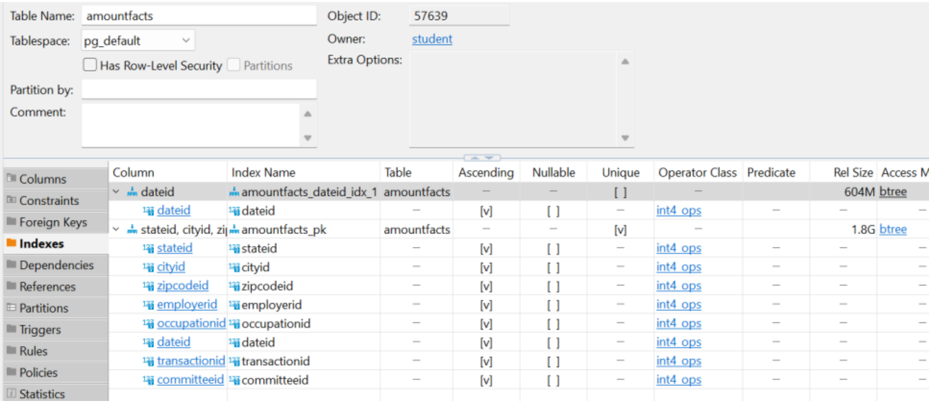


Table Name: amountfacts
Tablespace: pg_default
□ Has Row-Level Security □ Partitions
Partition by:
Comment:
Object ID: 57639
Owner: student
Extra Options:

| Column | Index Name | Table | Ascending | Nullable | Unique | Operator Class | Predicate | Rel Size | Access M |
|---|---|---|---|---|---|---|---|---|---|
| dateid | amountfacts_dateid_idx_1 | amountfacts | – | – | [ ] | – | | 604M | btree |
| dateid | dateid | – | [v] | [ ] | – | int4_ops | – | – | – |
| stateid, cityid, zi | amountfacts_pk | amountfacts | – | – | [v] | – | | 1.8G | btree |
| stateid | stateid | – | [v] | [ ] | – | int4_ops | – | – | – |
| cityid | cityid | – | [v] | [ ] | – | int4_ops | – | – | – |
| zipcodeid | zipcodeid | – | [v] | [ ] | – | int4_ops | – | – | – |
| employerid | employerid | – | [v] | [ ] | – | int4_ops | – | – | – |
| occupationid | occupationid | – | [v] | [ ] | – | int4_ops | – | – | – |
| dateid | dateid | – | [v] | [ ] | – | int4_ops | – | – | – |
| transactionid | transactionid | – | [v] | [ ] | – | int4_ops | – | – | – |
| committeeid | committeeid | – | [v] | [ ] | – | int4_ops | – | – | – |

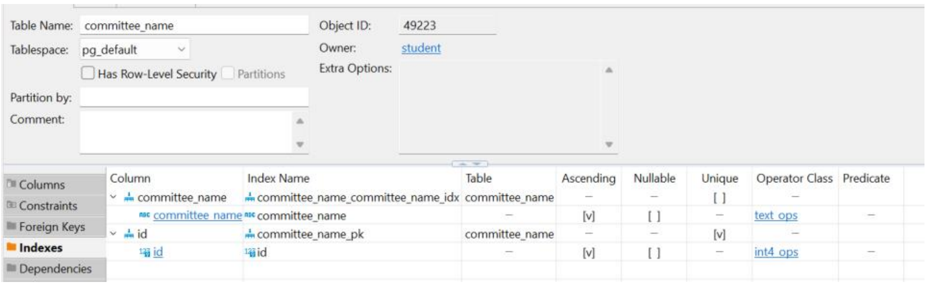


Table Name: committee_name
Tablespace: pg_default
□ Has Row-Level Security □ Partitions
Partition by:
Comment:
Object ID: 49223
Owner: student
Extra Options:

| Column | Index Name | Table | Ascending | Nullable | Unique | Operator Class | Predicate |
|---|---|---|---|---|---|---|---|
| committee_name | committee_name_committee_name_idx | committee_name | – | – | [ ] | – | |
| committee_name | committee_name | – | [v] | [ ] | – | text_ops | – |
| id | committee_name_pk | committee_name | – | – | [v] | – | |
| id | id | – | [v] | [ ] | – | int4_ops | – |

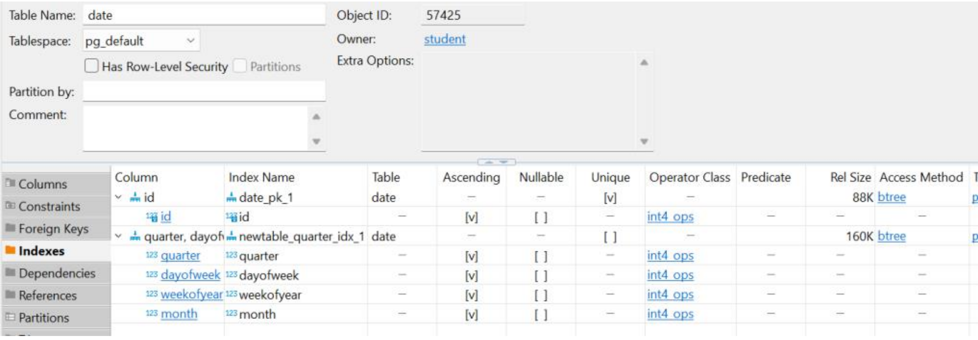


Table Name: date
Tablespace: pg_default
□ Has Row-Level Security □ Partitions
Partition by:
Comment:
Object ID: 57425
Owner: student
Extra Options:

| Column | Index Name | Table | Ascending | Nullable | Unique | Operator Class | Predicate | Rel Size | Access Method | T |
|---|---|---|---|---|---|---|---|---|---|---|
| id | date_pk_1 | date | – | – | [v] | – | | 88K | btree | p |
| id | id | – | [v] | [ ] | – | int4_ops | – | – | – | |
| quarter, dayofv | newtable_quarter_idx_1 | date | – | – | [ ] | – | | 160K | btree | p |
| quarter | quarter | – | [v] | [ ] | – | int4_ops | – | – | – | |
| dayofweek | dayofweek | – | [v] | [ ] | – | int4_ops | – | – | – | |
| weekofyear | weekofyear | – | [v] | [ ] | – | int4_ops | – | – | – | |
| month | month | – | [v] | [ ] | – | int4_ops | – | – | – | |

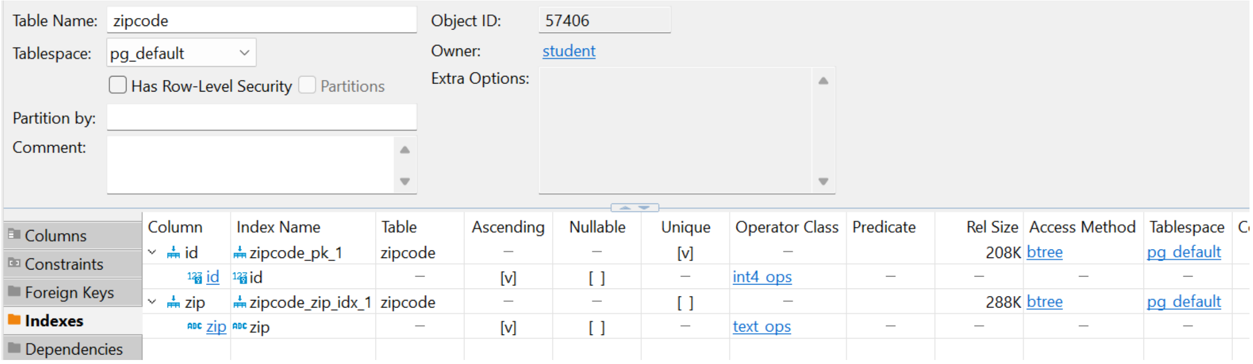


Table Name: zipcode
Tablespace: pg_default
□ Has Row-Level Security □ Partitions
Partition by:
Comment:

Object ID: 57406
Owner: student
Extra Options:

| Column | Index Name | Table | Ascending | Nullable | Unique | Operator Class | Predicate | Rel Size | Access Method | Tablespace | C |
|---|---|---|---|---|---|---|---|---|---|---|---|
| id | zipcode_pk_1 | zipcode | – | – | [v] | – | | 208K | btree | pg_default | |
| id | id | – | [v] | [ ] | – | int4_ops | – | – | – | – | |
| zip | zipcode_zip_idx_1 | zipcode | – | – | [ ] | – | | 288K | btree | pg_default | |
| zip | zip | – | [v] | [ ] | – | text_ops | – | – | – | – | |

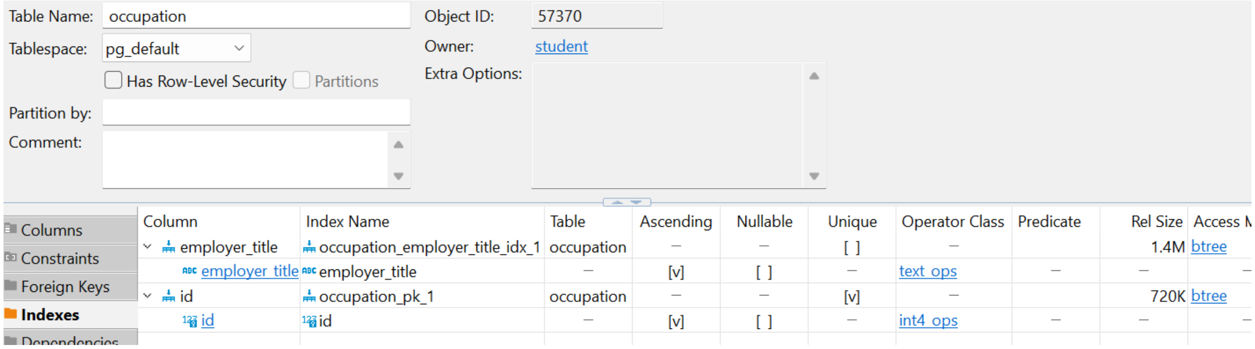


Table Name: occupation
Tablespace: pg_default
□ Has Row-Level Security □ Partitions
Partition by:
Comment:

Object ID: 57370
Owner: student
Extra Options:

| Column | Index Name | Table | Ascending | Nullable | Unique | Operator Class | Predicate | Rel Size | Access M |
|---|---|---|---|---|---|---|---|---|---|
| employer_title | occupation_employer_title_idx_1 | occupation | – | – | [ ] | – | | 1.4M | btree |
| employer_title | employer_title | – | [v] | [ ] | – | text_ops | – | – | – |
| id | occupation_pk_1 | occupation | – | – | [v] | – | | 720K | btree |
| id | id | – | [v] | [ ] | – | int4_ops | – | – | – |

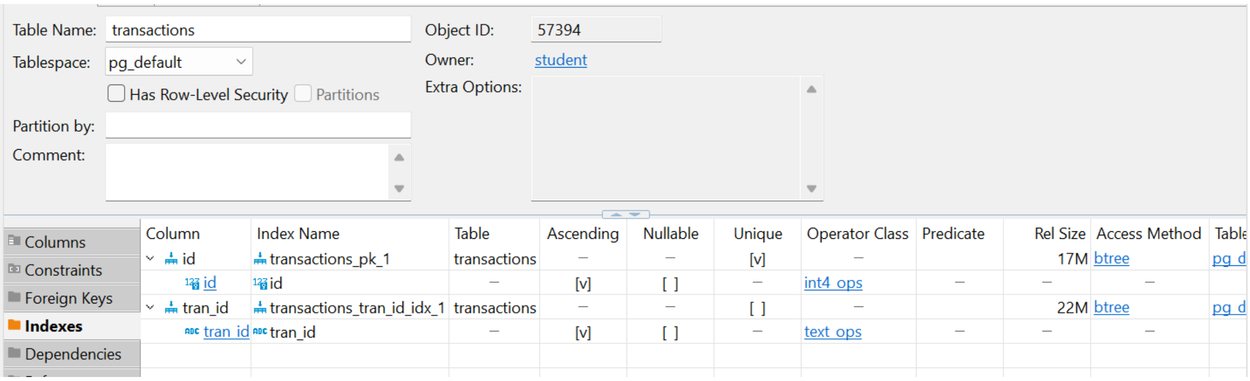


Table Name: transactions
Tablespace: pg_default
□ Has Row-Level Security □ Partitions
Partition by:
Comment:

Object ID: 57394
Owner: student
Extra Options:

| Column | Index Name | Table | Ascending | Nullable | Unique | Operator Class | Predicate | Rel Size | Access Method | Table |
|---|---|---|---|---|---|---|---|---|---|---|
| id | transactions_pk_1 | transactions | – | – | [v] | – | | 17M | btree | pg_d |
| id | id | – | [v] | [ ] | – | int4_ops | – | – | – | |
| tran_id | transactions_tran_id_idx_1 | transactions | – | – | [ ] | – | | 22M | btree | pg_d |
| tran_id | tran_id | – | [v] | [ ] | – | text_ops | – | – | – | |

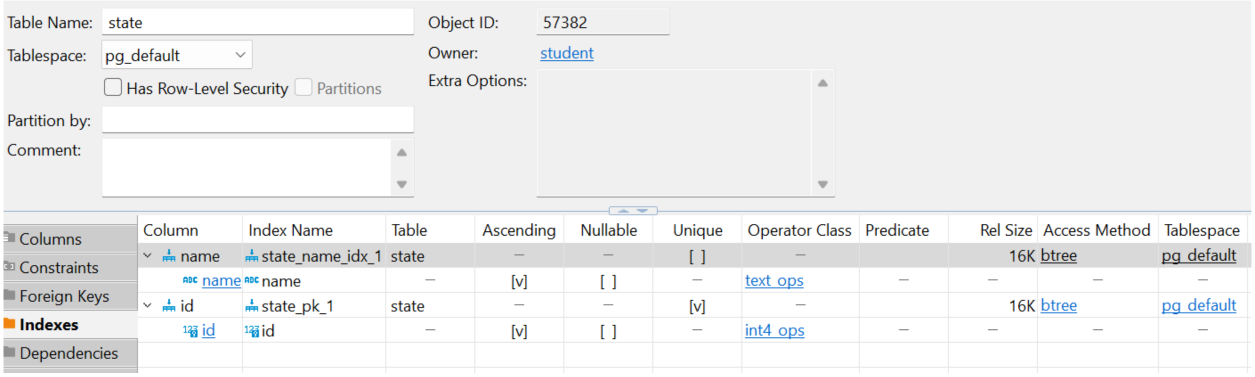
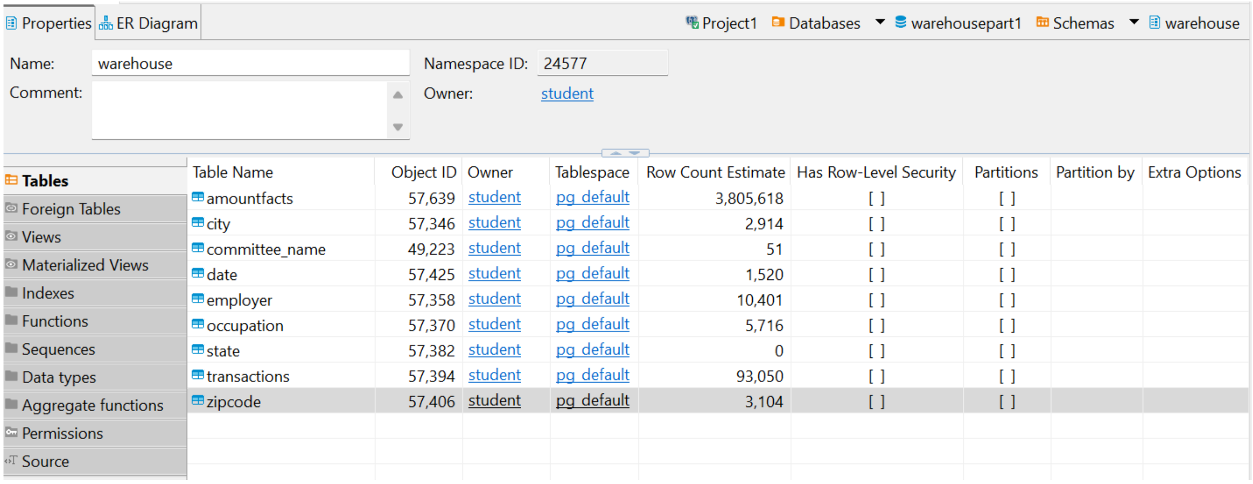


Table Name: state
Tablespace: pg_default
□ Has Row-Level Security □ Partitions
Partition by:
Comment:

Object ID: 57382
Owner: student
Extra Options:

| Column | Index Name | Table | Ascending | Nullable | Unique | Operator Class | Predicate | Rel Size | Access Method | Tablespace |
|---|---|---|---|---|---|---|---|---|---|---|
| name | state_name_idx_1 | state | – | – | [ ] | – | | 16K | btree | pg_default |
| name | name | – | [v] | [ ] | – | text_ops | – | – | – | – |
| id | state_pk_1 | state | – | – | [v] | – | | 16K | btree | pg_default |
| id | id | – | [v] | [ ] | – | int4_ops | – | – | – | – |

# Overall Schema of Datawarehouse:



Name: warehouse

Comment:

Namespace ID: 24577

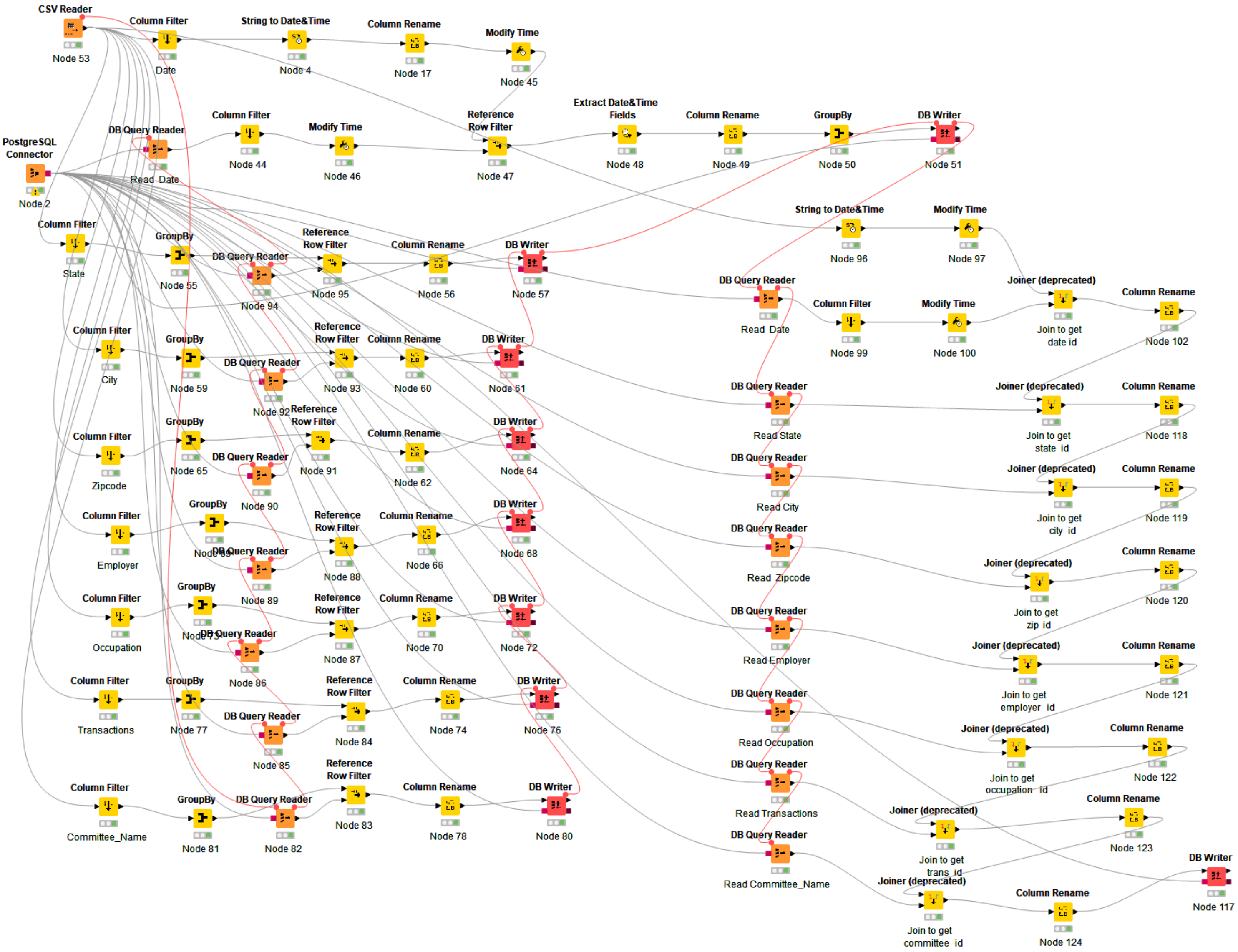
Owner: student

| Table Name | Object ID | Owner | Tablespace | Row Count Estimate | Has Row-Level Security | Partitions | Partition by | Extra Options |
|---|---|---|---|---|---|---|---|---|
| amountfacts | 57,639 | student | pg_default | 3,805,618 | [ ] | [ ] | | |
| city | 57,346 | student | pg_default | 2,914 | [ ] | [ ] | | |
| committee_name | 49,223 | student | pg_default | 51 | [ ] | [ ] | | |
| date | 57,425 | student | pg_default | 1,520 | [ ] | [ ] | | |
| employer | 57,358 | student | pg_default | 10,401 | [ ] | [ ] | | |
| occupation | 57,370 | student | pg_default | 5,716 | [ ] | [ ] | | |
| state | 57,382 | student | pg_default | 0 | [ ] | [ ] | | |
| transactions | 57,394 | student | pg_default | 93,050 | [ ] | [ ] | | |
| zipcode | 57,406 | student | pg_default | 3,104 | [ ] | [ ] | | |

# 3. Data Preparation

## 3.1 Relational Data Warehouse Implementation ETL considerations

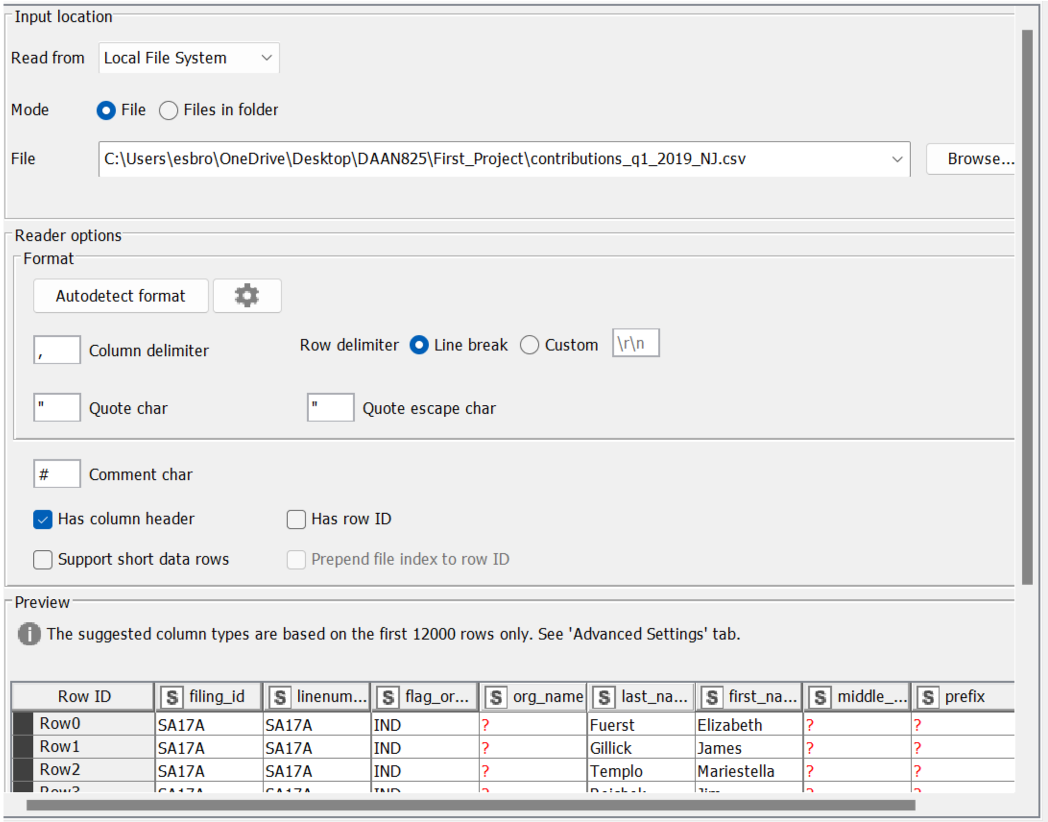
## ETL Process Flow with description



As one could see this ETL process included many steps to draw data from the initial source (4 election contribution data files). Then transfer the data into the data warehouse under warehousepart 1 project created from the student database under the schema warehouse in PostgreSQL.

## Step 1: Extract the data from the input files

In the first step I used a CSV reader to access the data from each of the four coma delimited files stored on my computer of the four state's election contributions data.



Input location

Read from: Local File System

Mode: ● File ○ Files in folder

File: C:\Users\esbro\OneDrive\Desktop\DAAN825\First_Project\contributions_q1_2019_NJ.csv

Reader options

Format

Column delimiter: ,

Row delimiter: ● Line break ○ Custom \r\n

Quote char: "

Quote escape char: "

Comment char: #

☑ Has column header

□ Has row ID

□ Support short data rows

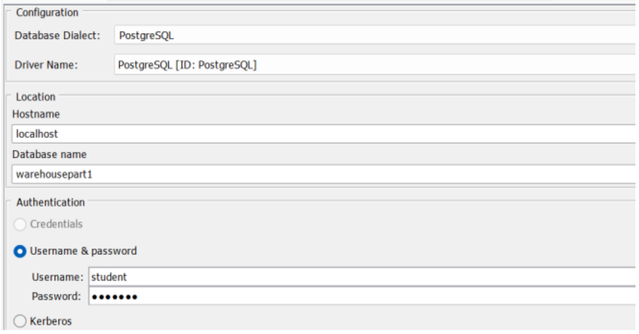
□ Prepend file index to row ID

Preview

The suggested column types are based on the first 12000 rows only. See 'Advanced Settings' tab.

| Row ID | S filing_id | S linenum... | S flag_or... | S org_name | S last_na... | S first_na... | S middle_... | S prefix |
|---|---|---|---|---|---|---|---|---|
| Row0 | SA17A | SA17A | IND | ? | Fuerst | Elizabeth | ? | ? |
| Row1 | SA17A | SA17A | IND | ? | Gillick | James | ? | ? |
| Row2 | SA17A | SA17A | IND | ? | Templo | Mariestella | ? | ? |

Then I used a PostgreSQL connector to access and connect to the data in the data warehouse in PostgreSQL using the following credentials:



Configuration

Database Dialect: PostgreSQL

Driver Name: PostgreSQL [ID: PostgreSQL]

Location

Hostname: localhost

Database name: warehousepart1

Authentication

○ Credentials

● Username & password

Username: student

Password: •••••••

○ Kerberos

## Step 2: Transform and Load data in the Dimension Tables

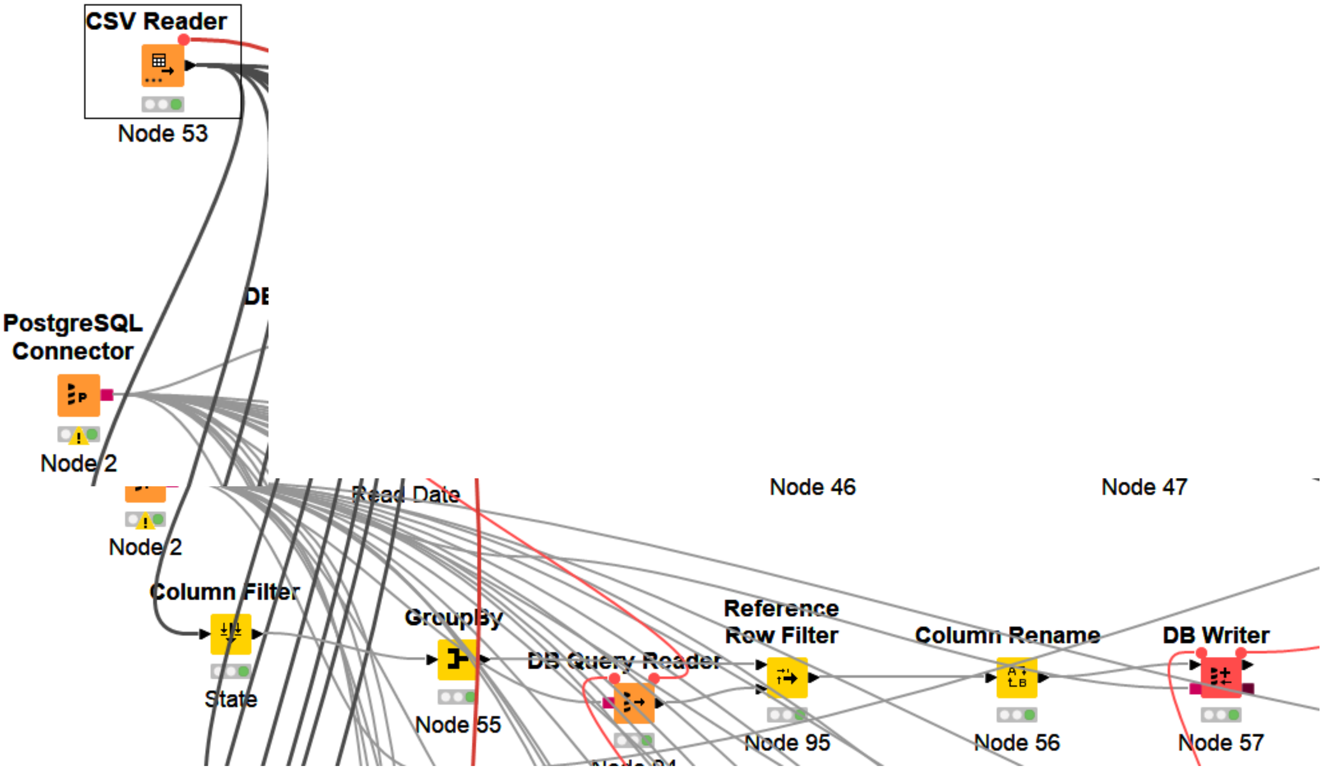
The I had to look at the dimensional data, for example State. We need to do the following for each dimension:

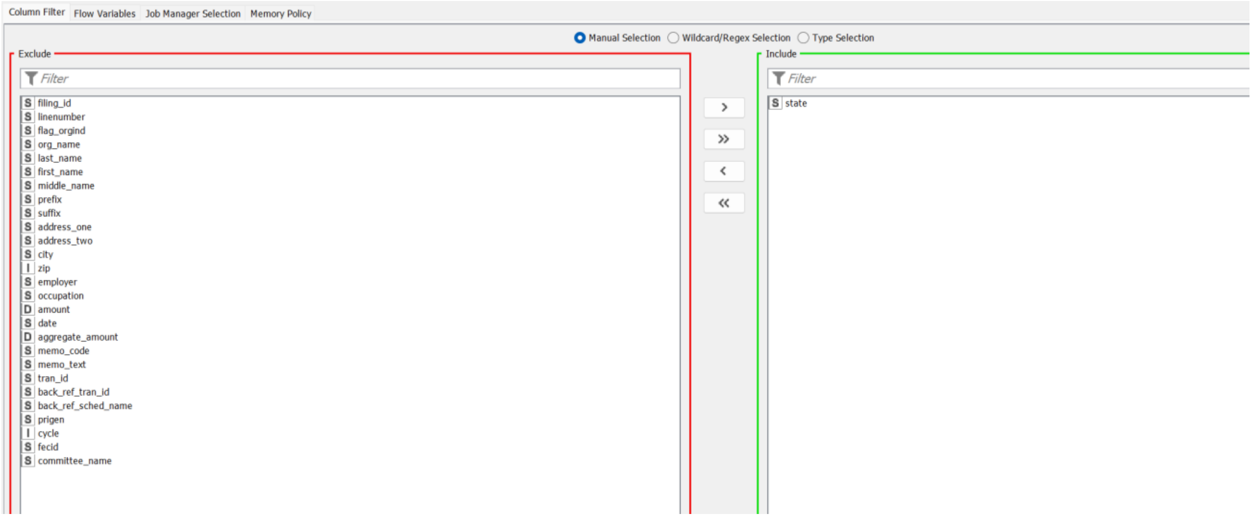
- Remove all other columns except for the dimension
- Keep only unique values for the dimension
- Save the dimensions in the PostgreSQl table

Note that you may get some errors if:

- You save the data file to a different path;
- Your Database runs under a different IP address

In Knime the corresponding operators are found in Manipulation. However you can search them by name. Below is the ETL flow for loading data in the “State” table.





● Manual Selection ○ Wildcard/Regex Selection ○ Type Selection

Exclude

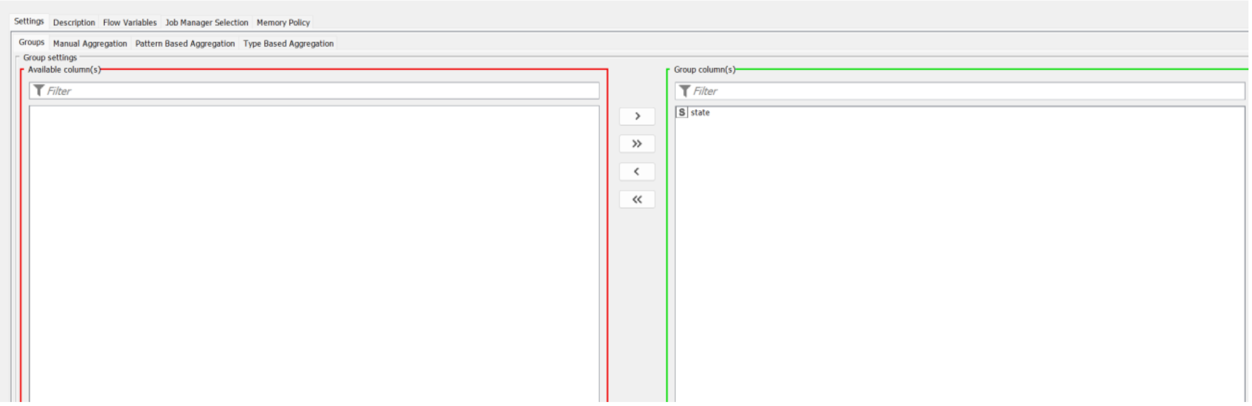
Filter

S filing_id
S linenumber
S flag_orgind
S org_name
S last_name
S first_name
S middle_name
S prefix
S suffix
S address_one
S address_two
S city
I zip
S employer
S occupation
D amount
S date
D aggregate_amount
S memo_code
S memo_text
S tran_id
S back_ref_tran_id
S back_ref_sched_name
S prigen
I cycle
S fecid
S committee_name

Include

Filter

S state



Group settings

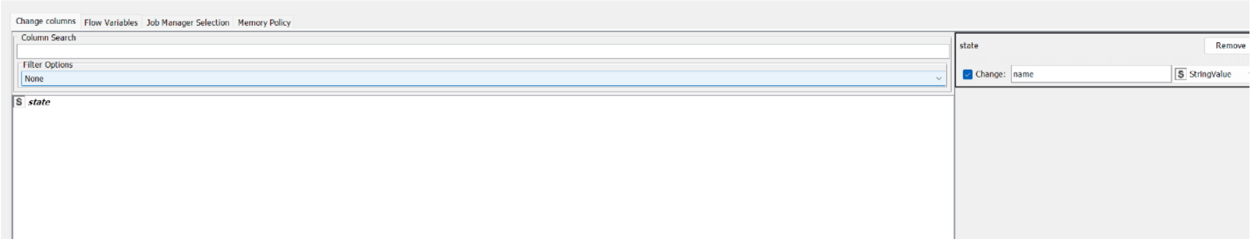
Available column(s)

Filter

Group column(s)

Filter

S state



Column Search

Filter Options: None

S *state*

state

☑ Change: name S StringValue

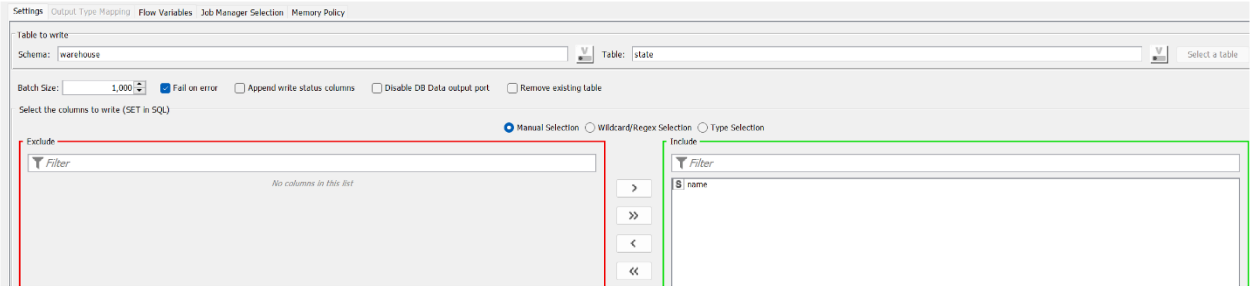


Table to write

Schema: warehouse Table: state

Batch Size: 1,000 ☑ Fail on error □ Append write status columns □ Disable DB Data output port □ Remove existing table

Select the columns to write (SET in SQL)

● Manual Selection ○ Wildcard/Regex Selection ○ Type Selection

Exclude

Filter

*No columns in this list*

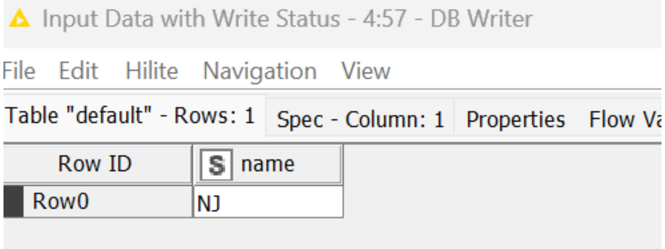
Include

Filter

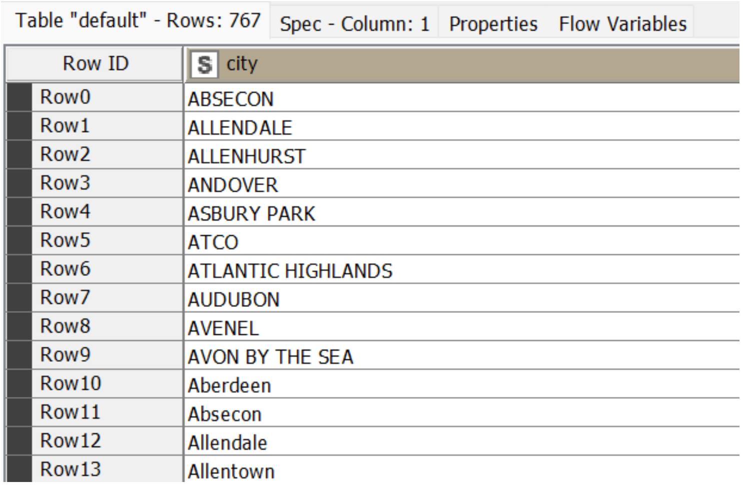
S name

I used a similar approach for the seven other dimensions that included city, zipcode, transactions, committee_name, employer, and occupation to load the data in the data warehouse. For date dimension I needed to follow a different approach due to the fact that dates are transient and every ETL will add new dates.

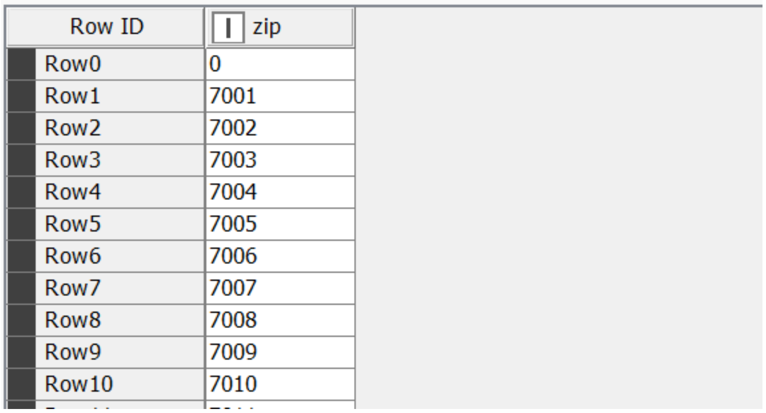
To execute the flow, right-click on the last node and click “Execute”. You can view the data in DBeaver by right-clicking on the table and then click “View Data” and then “View Top 100 Rows.


Input Data with Write Status - 4:57 - DB Writer
File Edit Hilite Navigation View
Table "default" - Rows: 1
Spec - Column: 1
Properties
Flow Va

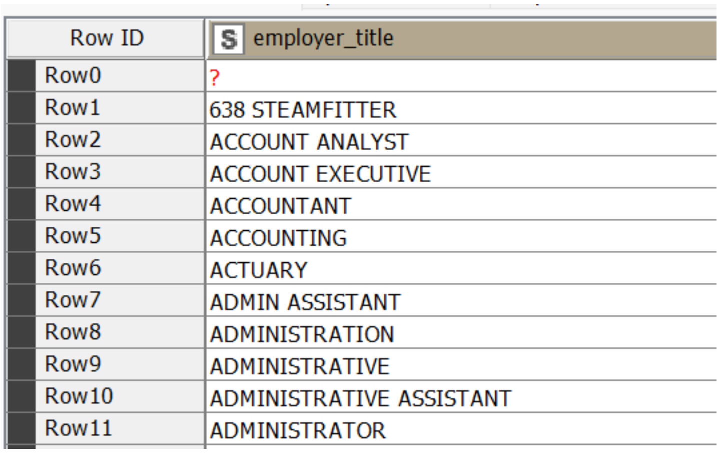
| Row ID | S name |
|---|---|
| Row0 | NJ |



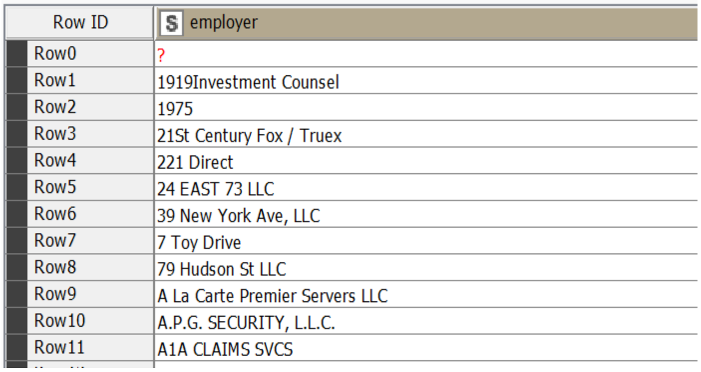
| Row ID | S city |
|---|---|
| Row0 | ABSECON |
| Row1 | ALLENDALE |
| Row2 | ALLENHURST |
| Row3 | ANDOVER |
| Row4 | ASBURY PARK |
| Row5 | ATCO |
| Row6 | ATLANTIC HIGHLANDS |
| Row7 | AUDUBON |
| Row8 | AVENEL |
| Row9 | AVON BY THE SEA |
| Row10 | Aberdeen |
| Row11 | Absecon |
| Row12 | Allendale |
| Row13 | Allentown |



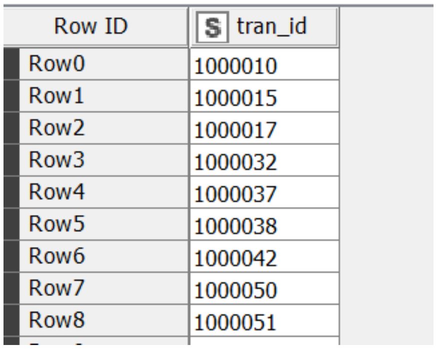
| Row ID | I zip |
|---|---|
| Row0 | 0 |
| Row1 | 7001 |
| Row2 | 7002 |
| Row3 | 7003 |
| Row4 | 7004 |
| Row5 | 7005 |
| Row6 | 7006 |
| Row7 | 7007 |
| Row8 | 7008 |
| Row9 | 7009 |
| Row10 | 7010 |



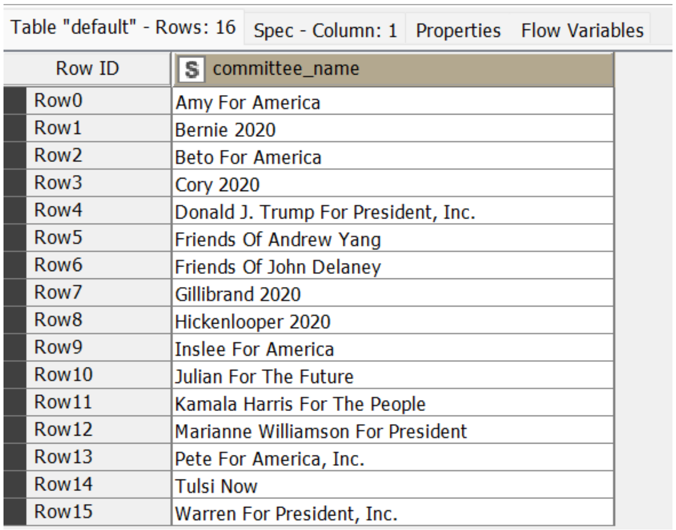
| Row ID | S employer_title |
|---|---|
| Row0 | ? |
| Row1 | 638 STEAMFITTER |
| Row2 | ACCOUNT ANALYST |
| Row3 | ACCOUNT EXECUTIVE |
| Row4 | ACCOUNTANT |
| Row5 | ACCOUNTING |
| Row6 | ACTUARY |
| Row7 | ADMIN ASSISTANT |
| Row8 | ADMINISTRATION |
| Row9 | ADMINISTRATIVE |
| Row10 | ADMINISTRATIVE ASSISTANT |
| Row11 | ADMINISTRATOR |



| Row ID | S employer |
|---|---|
| Row0 | ? |
| Row1 | 1919Investment Counsel |
| Row2 | 1975 |
| Row3 | 21St Century Fox / Truex |
| Row4 | 221 Direct |
| Row5 | 24 EAST 73 LLC |
| Row6 | 39 New York Ave, LLC |
| Row7 | 7 Toy Drive |
| Row8 | 79 Hudson St LLC |
| Row9 | A La Carte Premier Servers LLC |
| Row10 | A.P.G. SECURITY, L.L.C. |
| Row11 | A1A CLAIMS SVCS |



| Row ID | S tran_id |
|---|---|
| Row0 | 1000010 |
| Row1 | 1000015 |
| Row2 | 1000017 |
| Row3 | 1000032 |
| Row4 | 1000037 |
| Row5 | 1000038 |
| Row6 | 1000042 |
| Row7 | 1000050 |
| Row8 | 1000051 |



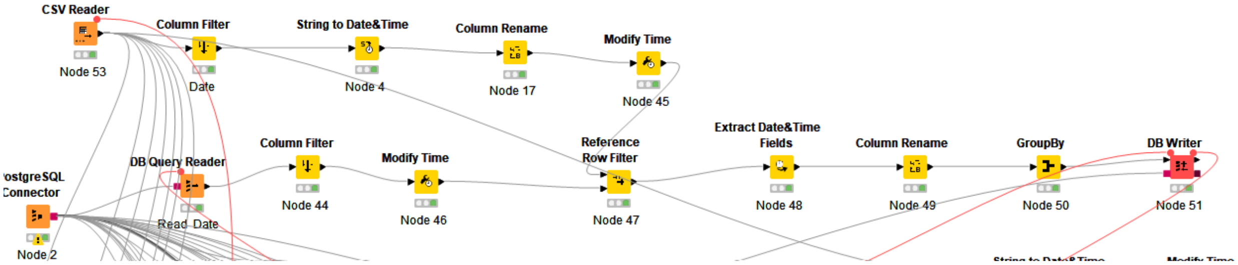
| Row ID | S committee_name |
|---|---|
| Row0 | Amy For America |
| Row1 | Bernie 2020 |
| Row2 | Beto For America |
| Row3 | Cory 2020 |
| Row4 | Donald J. Trump For President, Inc. |
| Row5 | Friends Of Andrew Yang |
| Row6 | Friends Of John Delaney |
| Row7 | Gillibrand 2020 |
| Row8 | Hickenlooper 2020 |
| Row9 | Inslee For America |
| Row10 | Julian For The Future |
| Row11 | Kamala Harris For The People |
| Row12 | Marianne Williamson For President |
| Row13 | Pete For America, Inc. |
| Row14 | Tulsi Now |
| Row15 | Warren For President, Inc. |

## Step 3: Load Data in the Date Dimension Table

To load data in the facts table, we need to:

- Load the date dimension into dimension table
- Change all the dimension names with dimensions ids.
- Remove data that is not needed

For this, we need to first read the data file and the dimension tables. The dimension tables need to be joined with the data from the data table so we can find the dimension id. The final workflow is shown below.



The most interesting operator here is the “Reference Row Filter” node that performs a “minus” operation on data, that is keeps only rows that exist in the first table but do not exist in the second table. That is necessary because we need to add only dimensions that do not already exist in the dimension table.

Another addition is the red line between the Excel Reader and the PostgreSQL Connector nodes on the left. This connection, noted in red line, will delay the execution of the PostgreSQL connection till the successful completion of the Excel Reader node. It can be useful to get a new connection every time we change the input file. You can enable this by right-clicking on each node and selecting "Show Flow Variable Ports" and then connecting the ports.

Note: although not shown in this example, it is important that you avoid duplications in the dimension tables by using the "Reference Row filter" method.

**Step 4: Load Data in the Facts Table**

Once we have added the dates, we can load the data in the facts table. The flow at the very beginning of the ETL section of this project shows the flow for loading the election data in the warehouse schema data warehouse.

Again, the most interesting node is the “Joiner” node that we use to join the data with the dimension tables, on description/name to the id of the dimension. A second important operator is the String to date, that converts the string in the file to a date using a mask.

## 3.2 Hadoop Implementation

To be completed with project number 2

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

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

Using KNIME I was able to learn how to automate the ETL process through different nodes and functions to save time, reduce manual errors, and ensure that my data integration tasks were performed consistently and efficiently. I enjoyed that KNIME used a graphical interface to make the data extraction, transforming, and loading phases easier to understand, build, and test compared to other software applications I used to conduct this process.

Since the ETL process should be automated in most cases. And should be able to run at fixed times with limited human interaction. I believe I was able to accomplish this objective, since I only had to perform two steps to run the total ETL workflow in KNIME. These two steps included:

1: load each of the four respective new files into the csv node one at a time
2: execute the last DB Writer node on the right

# 4. Reporting System

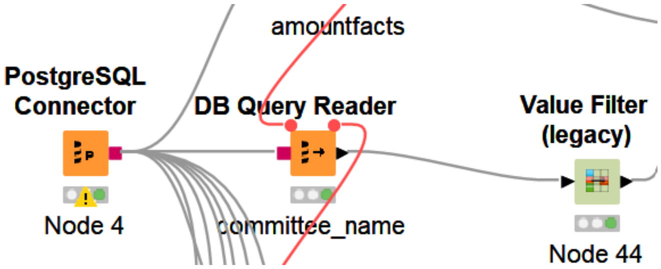
Provide screenshot and sample results with discussion on potential knowledge that was elicited.

## 4.1 Relational Data Warehouse Implementation

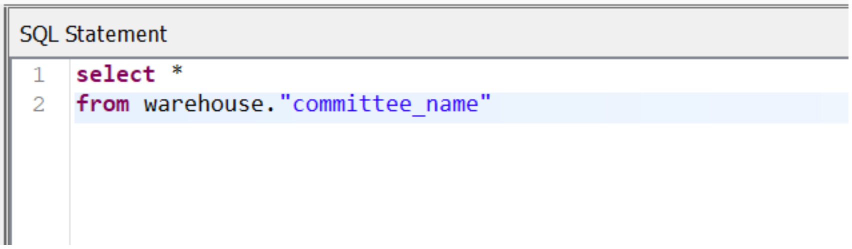
**Retrieving and aggregating the data from the data warehouse**

Our goal is to evaluate election contributions by state, political party, occupation, employer, month of the year, or day of year.

The first step is to create an interactive Knime workflow that will aggregate the data existing in the data warehouse. For this we will use the QuickForms "Value filter" that allows us to select the desired values from the database. For example, the flow with the three nodes below, establish a connection to the PostgreSQL server, read the data from the committee_name table, and then allows us to interactively select political party values.

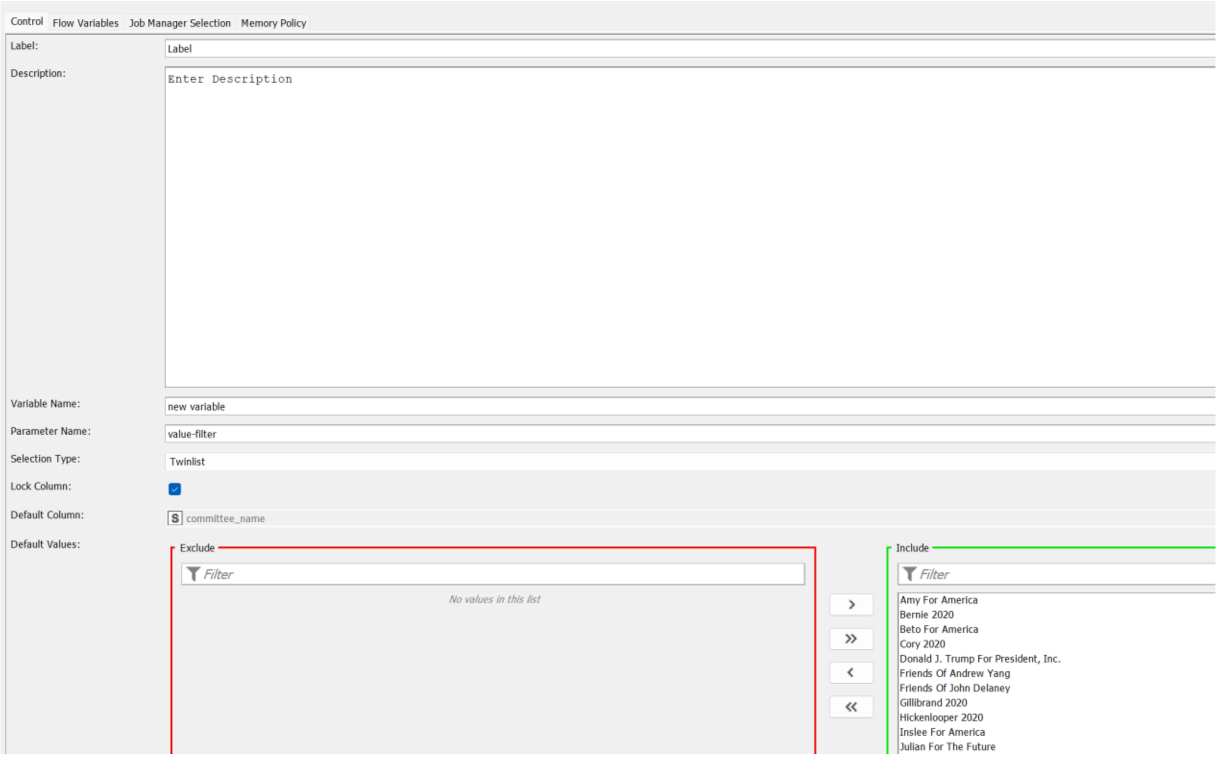


The first node returns the id and committee name of the political party in the database table.


SQL Statement

```
select *
from warehouse."committee_name"
```

The second node allows us to define the column that is used for selection. As you can see in the screenshot below, I have chosen the "priority" column and I have locked it. I have also chosen to use all the values by default, see the "Include" panel.



Label: Label

Description: Enter Description

Variable Name: new variable

Parameter Name: value-filter

Selection Type: Twinlist

Lock Column: ☑

Default Column: S committee_name

Default Values:

Exclude

Filter

No values in this list

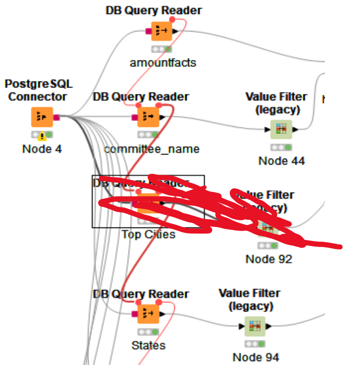
Include

Filter

Amy For America
Bernie 2020
Beto For America
Cory 2020
Donald J. Trump For President, Inc.
Friends Of Andrew Yang
Friends Of John Delaney
Gillibrand 2020
Hickenlooper 2020
Inslee For America
Julian For The Future

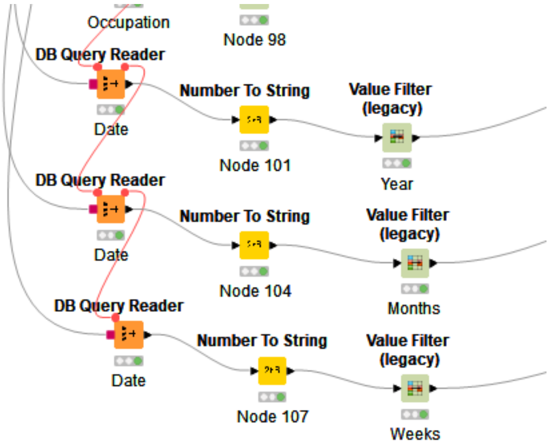
The selection of the priority values is done by right clicking on the "Value Filter" node and then clicking "Execute and Open View". Note that if the node is green you need to first reset the node to have the "Execute and Open View" available. The new window will give you the option to select the desired values. The new selected values will be applied when you click the button "Apply" and choose the "temporarily" option.

We repeat this process for two dimensions state and amountfacts tables, as shown in the figure below. The first node is reading from the facts table and it is not used yet. The following code is used to retrieve data from the database:

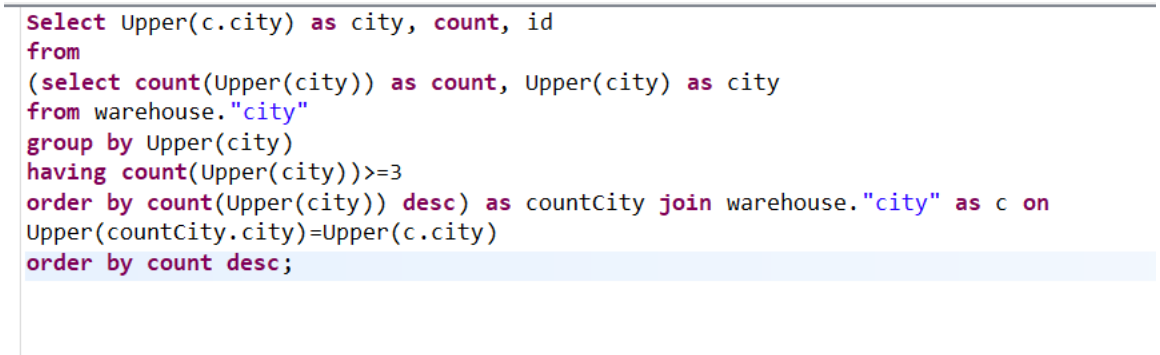


state Table: SELECT * FROM "state"
amountfacts Table: SELECT *  FROM "amountfacts"

The data in the “Date” table needs to be processed some more due to the fact that the “Value Filter” node accepts only string columns/values while all the values in this table are numeric. To accomplish this we convert the Day of Month, day of week, and year to string. And then we feed them into the “value Filter” nodes.



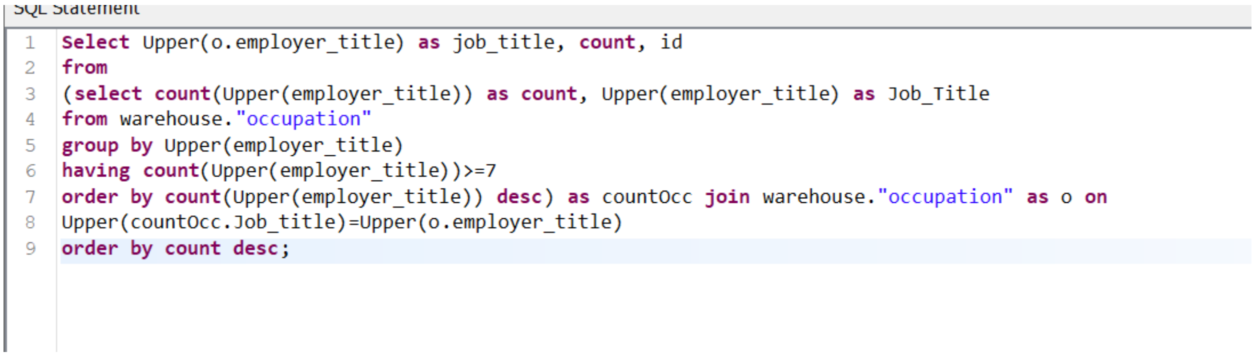
The other three dimensions city, employer, and occupation I had use a slightly different retrieval method to get the proper data from these dimensions in the data warehouse.



```
Select Upper(c.city) as city, count, id
from
(select count(Upper(city)) as count, Upper(city) as city
from warehouse."city"
group by Upper(city)
having count(Upper(city))>=3
order by count(Upper(city)) desc) as countCity join warehouse."city" as c on
Upper(countCity.city)=Upper(c.city)
order by count desc;
```

For example for the “city” dimension I really only wanted to select city values that appeared in the table the most often. This is why I used a count function to count the number of occurrences of each city value in the city table. Then I only outputted values that occurred three times or more in the table. Which is basically outputting cities that contributed the most frequent during the 2020 elections in the first quarter.

I used this same concept for extracting data from the “occupation” table in the data warehouse, as well. I used count function to count the number of occurrences of each job title listed in the occupation table. Then I only outputted values that occurred seven times or more in the table. Which is basically outputting job titles that contributed the most frequent during the 2020 elections in the first quarter.



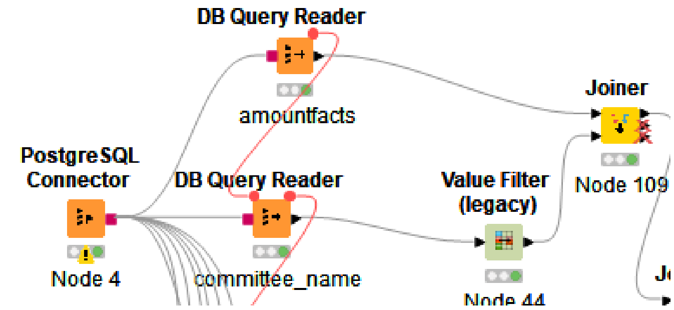
SQL Statement

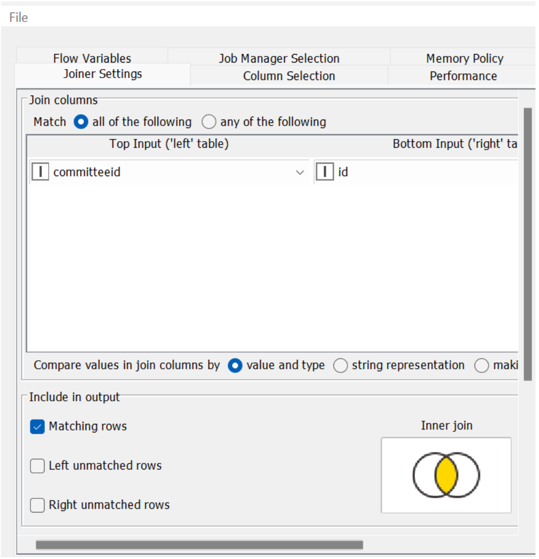
```
Select Upper(o.employer_title) as job_title, count, id
from
(select count(Upper(employer_title)) as count, Upper(employer_title) as Job_Title
from warehouse."occupation"
group by Upper(employer_title)
having count(Upper(employer_title))>=7
order by count(Upper(employer_title)) desc) as countOcc join warehouse."occupation" as o on
Upper(countOcc.Job_title)=Upper(o.employer_title)
order by count desc;
```

Lastly for the employer table I did not need to select certain values from this table. Since I wanted to get a full analysis of which employers contributed the most in 2020 elections. This did not mean the employers who contributed the most frequently.

employer Table: SELECT * FROM "employer"

Next, we will join the values in each dimension table with the fact table. For example, the flow below, joins the data in the facts table with the selected political parties. Joining is done on the pair “committeeid” from the facts table and the “id” field from the committee_name dimension table. We also have chosen to discard the id fields related to committee_name from the data set.





File

| Flow Variables | Job Manager Selection | Memory Policy |
|---|---|---|
| Joiner Settings | Column Selection | Performance |

Join columns

Match ● all of the following ○ any of the following

| Top Input ('left' table) | Bottom Input ('right' ta |
|---|---|
| committeeid | id |

Compare values in join columns by ● value and type ○ string representation ○ maki

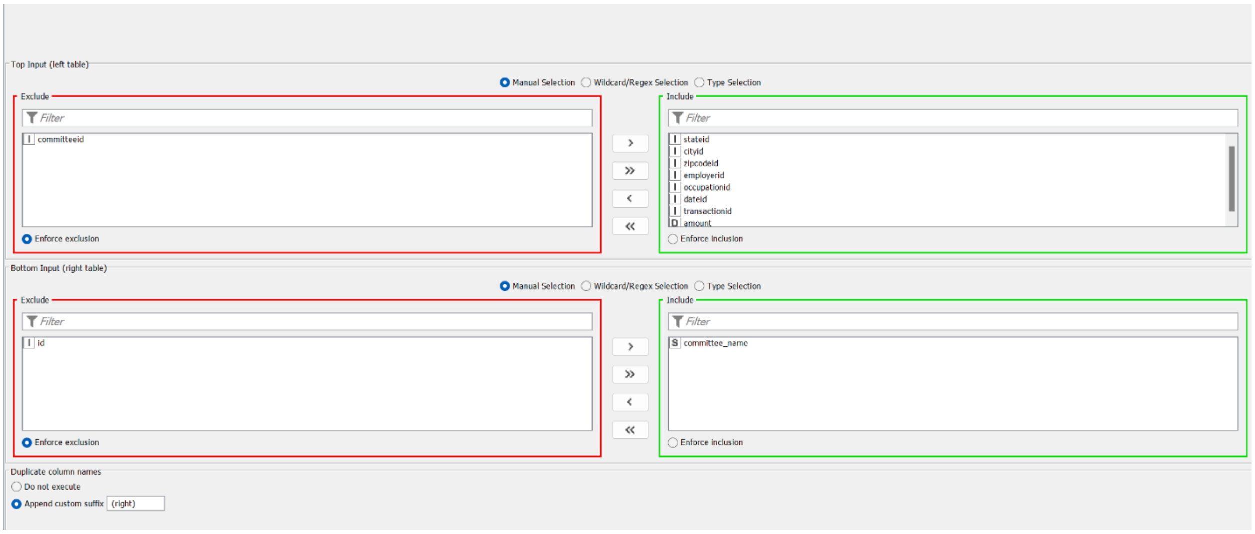
Include in output

☑ Matching rows

☐ Left unmatched rows

☐ Right unmatched rows

Inner join



Top Input (left table)

● Manual Selection ○ Wildcard/Regex Selection ○ Type Selection

Exclude

Filter

I committeeid

● Enforce exclusion

Include

Filter

I stateid
I cityid
I zipcodeid
I employerid
I occupationid
I dateid
I transactionid
D amount

○ Enforce inclusion

Bottom Input (right table)

● Manual Selection ○ Wildcard/Regex Selection ○ Type Selection

Exclude

Filter

I id

● Enforce exclusion

Include

Filter

S committee_name

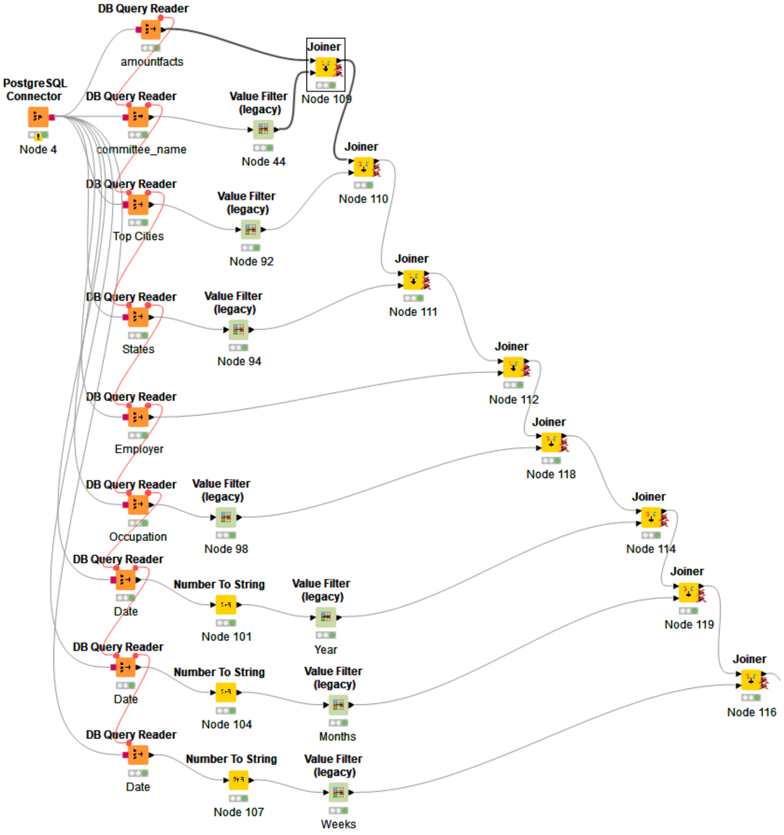
○ Enforce inclusion

Duplicate column names

○ Do not execute

● Append custom suffix (right)

We join the fact table with all the dimensions we have in a similar fashion. The result is shown in the figure below, together with a partial view of the resulting data.



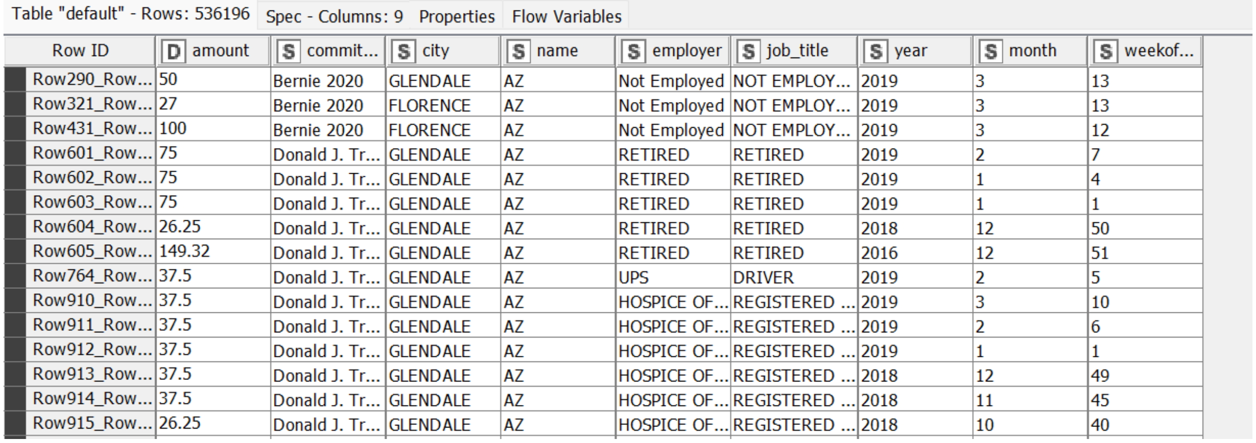
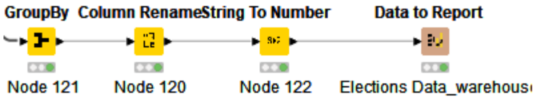
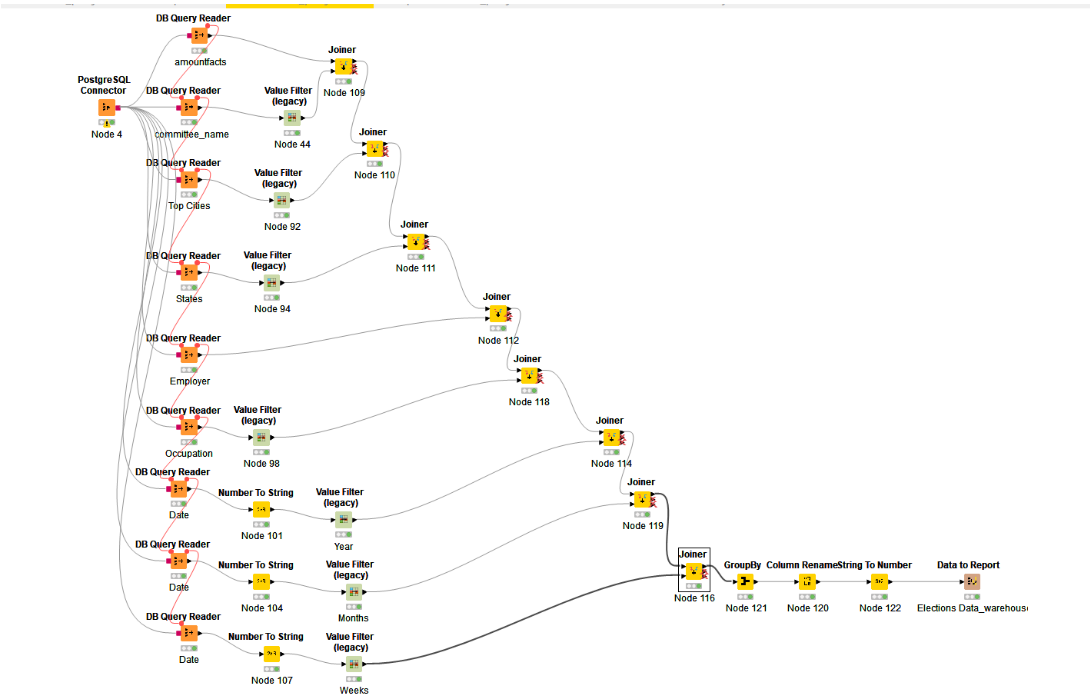


Table "default" - Rows: 536196 Spec - Columns: 9 Properties Flow Variables

| Row ID | D amount | S commit... | S city | S name | S employer | S job_title | S year | S month | S weekof... |
|---|---|---|---|---|---|---|---|---|---|
| Row290_Row... | 50 | Bernie 2020 | GLENDALE | AZ | Not Employed | NOT EMPLOY... | 2019 | 3 | 13 |
| Row321_Row... | 27 | Bernie 2020 | FLORENCE | AZ | Not Employed | NOT EMPLOY... | 2019 | 3 | 13 |
| Row431_Row... | 100 | Bernie 2020 | FLORENCE | AZ | Not Employed | NOT EMPLOY... | 2019 | 3 | 12 |
| Row601_Row... | 75 | Donald J. Tr... | GLENDALE | AZ | RETIRED | RETIRED | 2019 | 2 | 7 |
| Row602_Row... | 75 | Donald J. Tr... | GLENDALE | AZ | RETIRED | RETIRED | 2019 | 1 | 4 |
| Row603_Row... | 75 | Donald J. Tr... | GLENDALE | AZ | RETIRED | RETIRED | 2019 | 1 | 1 |
| Row604_Row... | 26.25 | Donald J. Tr... | GLENDALE | AZ | RETIRED | RETIRED | 2018 | 12 | 50 |
| Row605_Row... | 149.32 | Donald J. Tr... | GLENDALE | AZ | RETIRED | RETIRED | 2016 | 12 | 51 |
| Row764_Row... | 37.5 | Donald J. Tr... | GLENDALE | AZ | UPS | DRIVER | 2019 | 2 | 5 |
| Row910_Row... | 37.5 | Donald J. Tr... | GLENDALE | AZ | HOSPICE OF... | REGISTERED ... | 2019 | 3 | 10 |
| Row911_Row... | 37.5 | Donald J. Tr... | GLENDALE | AZ | HOSPICE OF... | REGISTERED ... | 2019 | 2 | 6 |
| Row912_Row... | 37.5 | Donald J. Tr... | GLENDALE | AZ | HOSPICE OF... | REGISTERED ... | 2019 | 1 | 1 |
| Row913_Row... | 37.5 | Donald J. Tr... | GLENDALE | AZ | HOSPICE OF... | REGISTERED ... | 2018 | 12 | 49 |
| Row914_Row... | 37.5 | Donald J. Tr... | GLENDALE | AZ | HOSPICE OF... | REGISTERED ... | 2018 | 11 | 45 |
| Row915_Row... | 26.25 | Donald J. Tr... | GLENDALE | AZ | HOSPICE OF... | REGISTERED ... | 2018 | 10 | 40 |

The last three nodes aggregate the data by year, month of year, week of year, state, city, occupation, employer, and committee_name, and change the name of columns to more meaningful values.



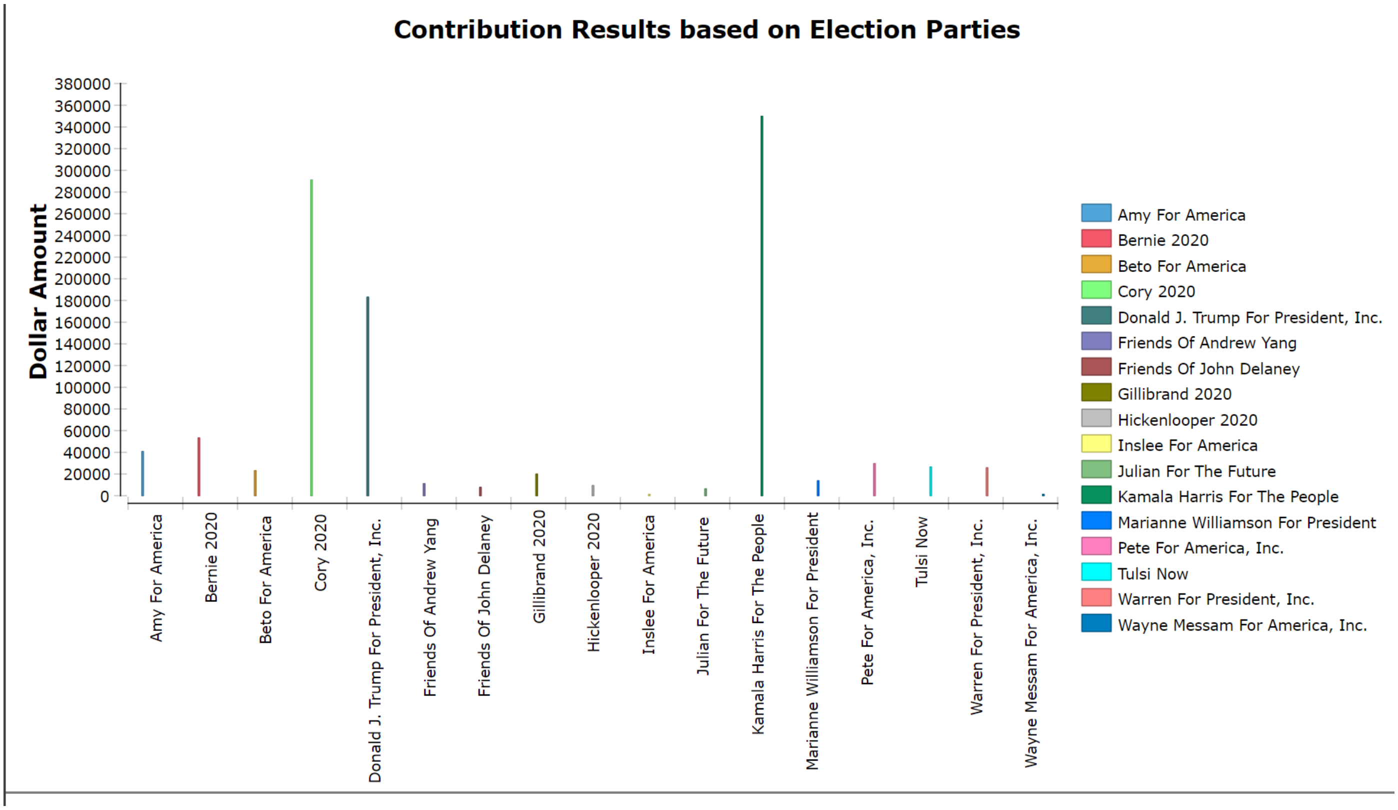


## Creating a Report

Once we have added the report node and connected it to the data, we can customize it by clicking on the “Open Report” button on the menu.

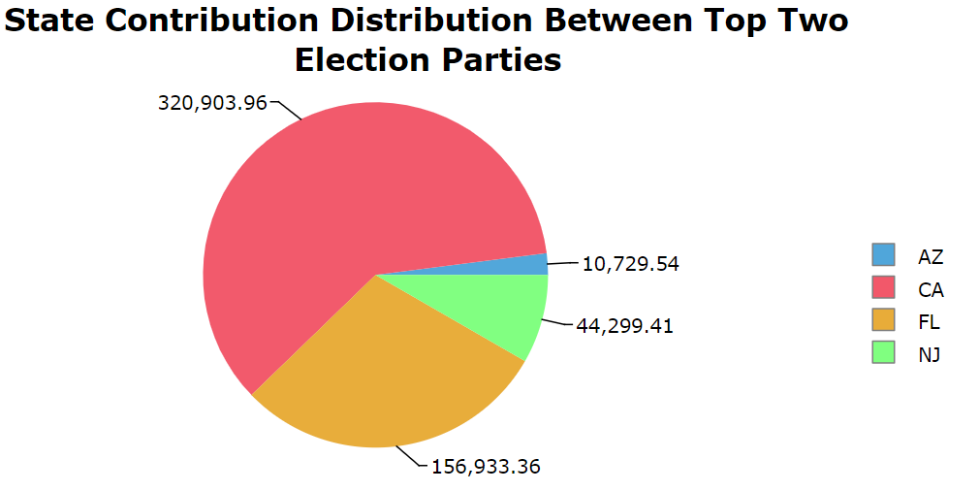
Once the data was established to conduct the report building process, I began to create the necessary charts and tables needed to answer each of the six business questions. The first chart I began to create was used to answer the first business question that I established. Which was *“Which campaigning party had the highest contributions in terms of dollars between the four states of California, New Jersey, Florida, and Arizona?”*

This chart, I felt needed to be a bar chart since we wanted to get a visual understanding of what the top campaigning/political parties were based on the four states. And bar charts should be used when you are showing segments of information. Vertical bar charts are useful to compare different categorical or discrete variables (*Statistics of Canada, 2023)*.



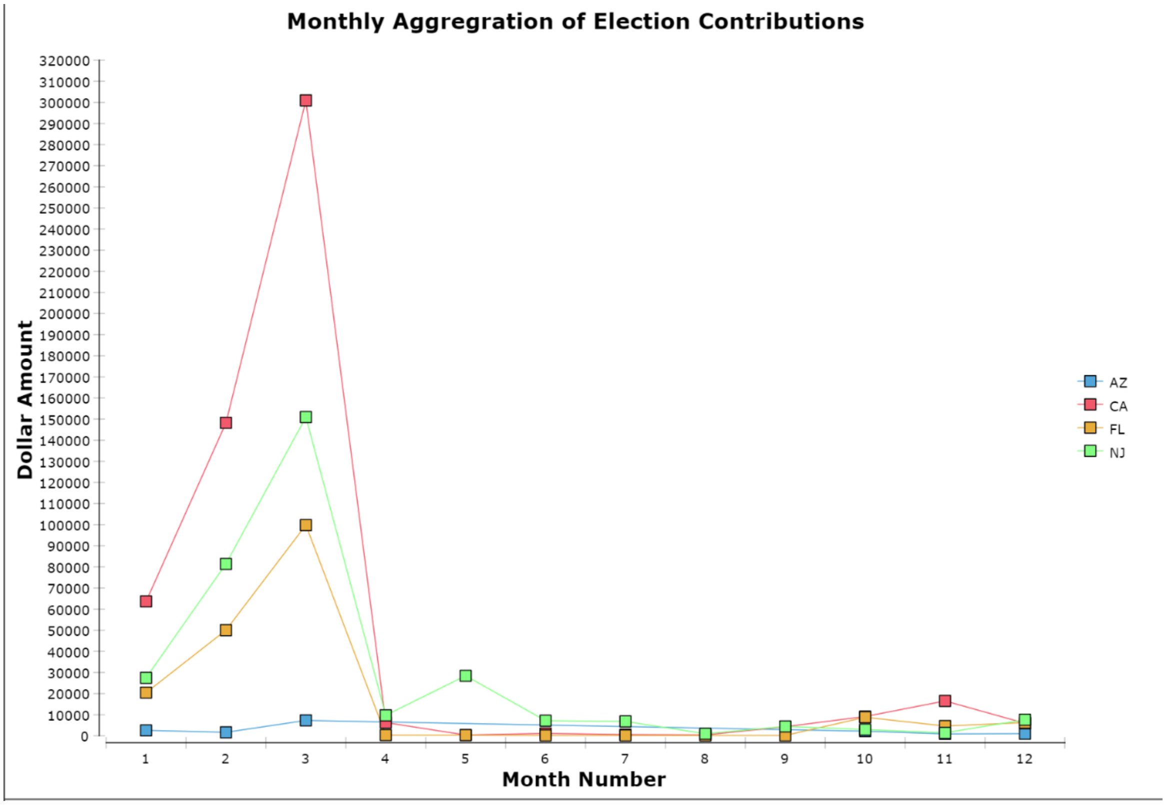
The second chart I began to create was used to answer the third business question that I established. Which was *"Which state had the highest contributions to the top two campaigning parties?"*

This chart, I felt, needed to be a pie chart since we wanted to get a visual understanding of which state contributed the most to the top political parties. And since the number of categorical variables that needed to be analyzed were less than five. I felt a pie chart could show this data point as a percentage of a whole. Instead of individual categorical aggregational values, which a bar chart would show.



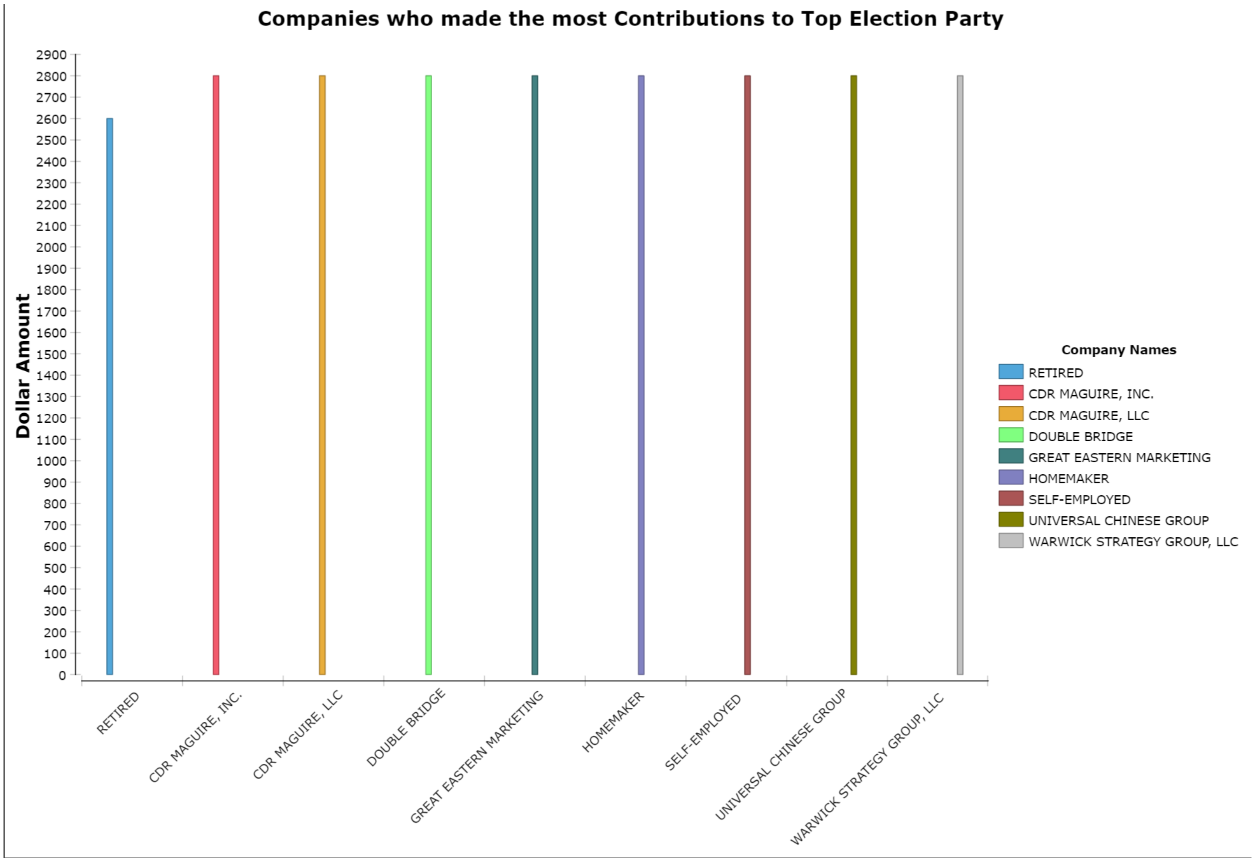
The third chart I began to create was used to answer the fourth business question that I established. Which was "*Display contributions made in California, New Jersey, Florida, and Arizona aggregated monthly in Q1 during the year of 2019.*"

This chart, I felt, needed to be a line graph since we wanted to get a visual understanding of how contributions by individuals changed over time in the first quarter of elections in 2019. And line graphs are best used when showing how something changes over time (KidsZone, 2023).



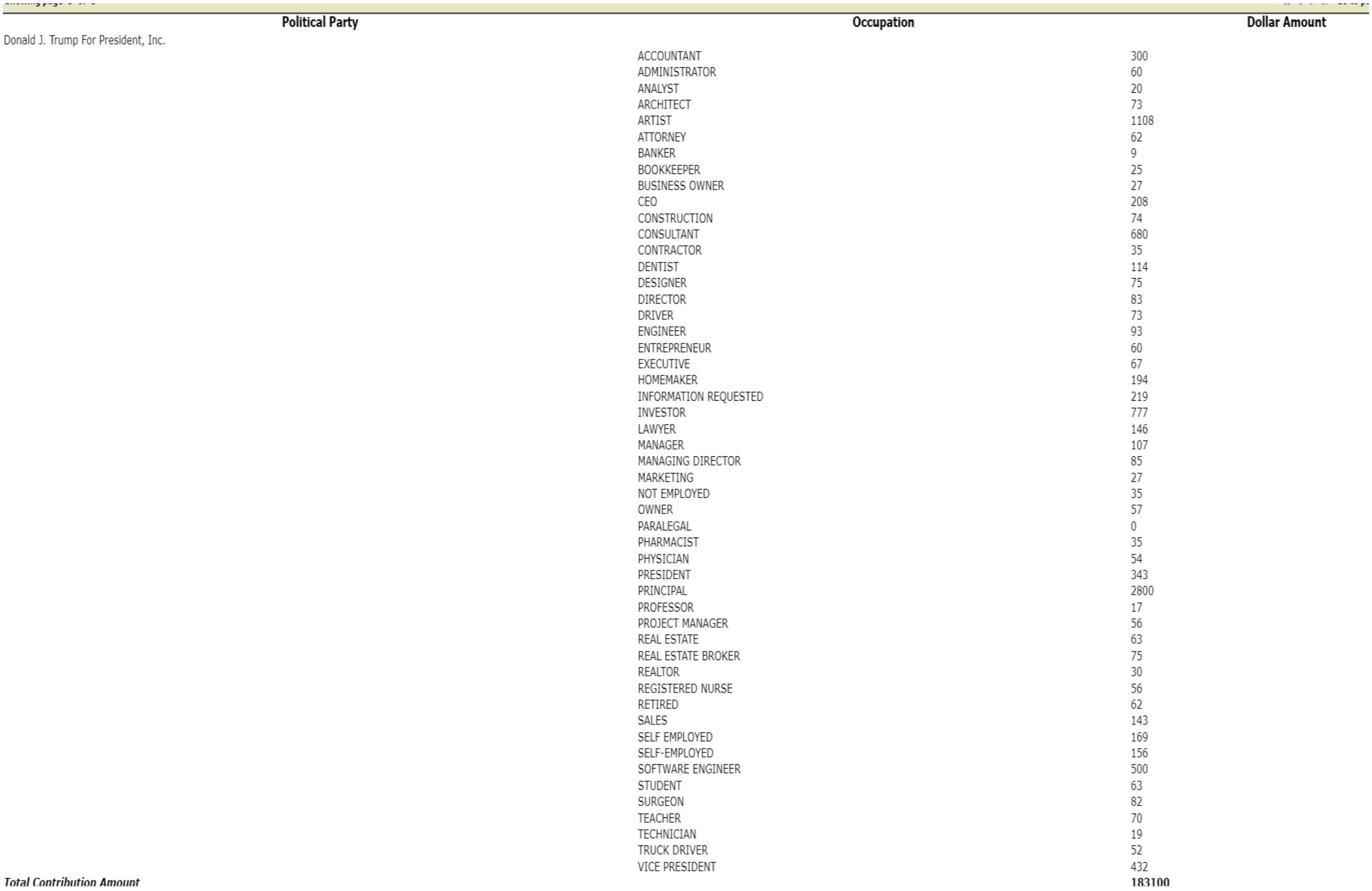
The fourth chart I began to create was used to answer the fifth business question that I established. Which was *"Which company made the most contributions in terms of dollars to the top campaigning party?"*

To answer this question, I felt I could choose from numerous different chart types. Though from further analysis I felt a bar chart would best depict the visual needed to answer this question. Since there was a great number of companies used to create the dataset for this report. And I felt a pie chart would not show the percentage of contributions from a given company compared to other companies in the dataset best. Because there would be too many companies to analyze and compare against with the use of slices in a pie chart. Also, a line graph would not suffice because we were not analyzing the change in values over a particular time period.

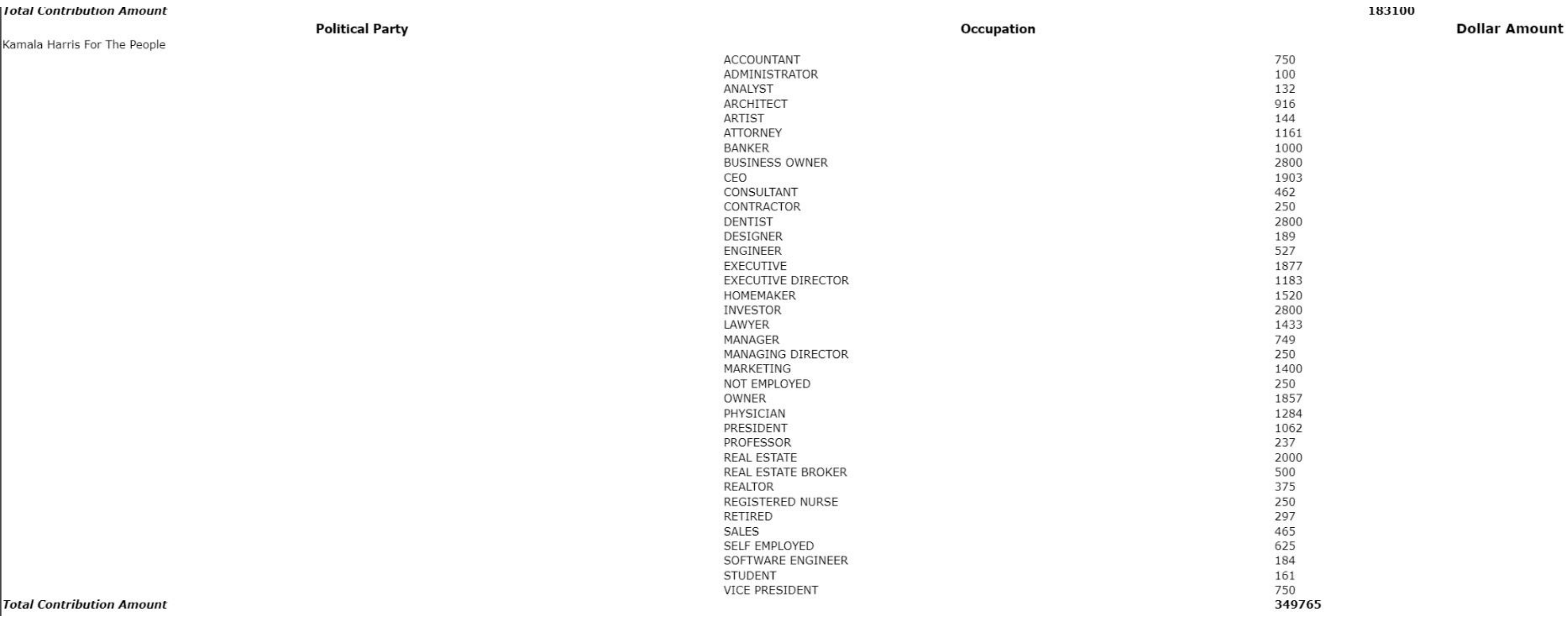


The fifth and final chart I began to create was used to answer the sixth business question that I established. Which was *“What was the overall demographic makeup of the contributors of each of the top campaigning parties?”*

For this question I had to think twice about what I actually wanted to show to the audience of the report. And I came to the conclusion that not a graph or plot would best describe the demograhic makeup of the contributors of the top campaigning parties. There would be too much information needed to be displayed to describe a given supporter of a given political party. So, I decided to use a table to show all the different type political party supporters for each of top campaigning parties. The information I decided to show the table would include the different job professions each political party had supporting their campaign in the 2020 elections.

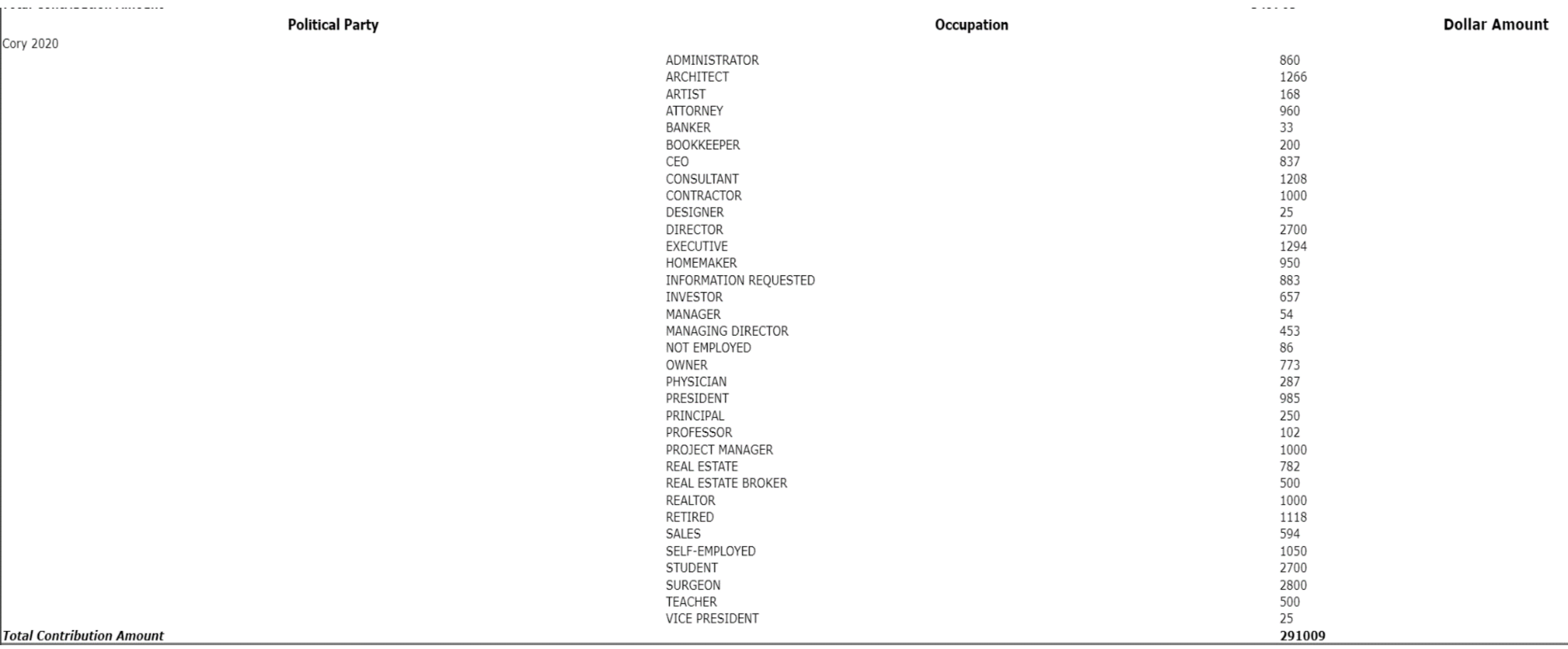


| Political Party | Occupation | Dollar Amount |
|---|---|---|
| Donald J. Trump For President, Inc. | | |
| | ACCOUNTANT | 300 |
| | ADMINISTRATOR | 60 |
| | ANALYST | 20 |
| | ARCHITECT | 73 |
| | ARTIST | 1108 |
| | ATTORNEY | 62 |
| | BANKER | 9 |
| | BOOKKEEPER | 25 |
| | BUSINESS OWNER | 27 |
| | CEO | 208 |
| | CONSTRUCTION | 74 |
| | CONSULTANT | 680 |
| | CONTRACTOR | 35 |
| | DENTIST | 114 |
| | DESIGNER | 75 |
| | DIRECTOR | 83 |
| | DRIVER | 73 |
| | ENGINEER | 93 |
| | ENTREPRENEUR | 60 |
| | EXECUTIVE | 67 |
| | HOMEMAKER | 194 |
| | INFORMATION REQUESTED | 219 |
| | INVESTOR | 777 |
| | LAWYER | 146 |
| | MANAGER | 107 |
| | MANAGING DIRECTOR | 85 |
| | MARKETING | 27 |
| | NOT EMPLOYED | 35 |
| | OWNER | 57 |
| | PARALEGAL | 0 |
| | PHARMACIST | 35 |
| | PHYSICIAN | 54 |
| | PRESIDENT | 343 |
| | PRINCIPAL | 2800 |
| | PROFESSOR | 17 |
| | PROJECT MANAGER | 56 |
| | REAL ESTATE | 63 |
| | REAL ESTATE BROKER | 75 |
| | REALTOR | 30 |
| | REGISTERED NURSE | 56 |
| | RETIRED | 62 |
| | SALES | 143 |
| | SELF EMPLOYED | 169 |
| | SELF-EMPLOYED | 156 |
| | SOFTWARE ENGINEER | 500 |
| | STUDENT | 63 |
| | SURGEON | 82 |
| | TEACHER | 70 |
| | TECHNICIAN | 19 |
| | TRUCK DRIVER | 52 |
| | VICE PRESIDENT | 432 |
| ***Total Contribution Amount*** | | **183100** |



*Total Contribution Amount* **183100**

| Political Party | Occupation | Dollar Amount |
|---|---|---|
| Kamala Harris For The People | | |
| | ACCOUNTANT | 750 |
| | ADMINISTRATOR | 100 |
| | ANALYST | 132 |
| | ARCHITECT | 916 |
| | ARTIST | 144 |
| | ATTORNEY | 1161 |
| | BANKER | 1000 |
| | BUSINESS OWNER | 2800 |
| | CEO | 1903 |
| | CONSULTANT | 462 |
| | CONTRACTOR | 250 |
| | DENTIST | 2800 |
| | DESIGNER | 189 |
| | ENGINEER | 527 |
| | EXECUTIVE | 1877 |
| | EXECUTIVE DIRECTOR | 1183 |
| | HOMEMAKER | 1520 |
| | INVESTOR | 2800 |
| | LAWYER | 1433 |
| | MANAGER | 749 |
| | MANAGING DIRECTOR | 250 |
| | MARKETING | 1400 |
| | NOT EMPLOYED | 250 |
| | OWNER | 1857 |
| | PHYSICIAN | 1284 |
| | PRESIDENT | 1062 |
| | PROFESSOR | 237 |
| | REAL ESTATE | 2000 |
| | REAL ESTATE BROKER | 500 |
| | REALTOR | 375 |
| | REGISTERED NURSE | 250 |
| | RETIRED | 297 |
| | SALES | 465 |
| | SELF EMPLOYED | 625 |
| | SOFTWARE ENGINEER | 184 |
| | STUDENT | 161 |
| | VICE PRESIDENT | 750 |
| *Total Contribution Amount* | | **349765** |



| Political Party | Occupation | Dollar Amount |
|---|---|---|
| Cory 2020 | | |
| | ADMINISTRATOR | 860 |
| | ARCHITECT | 1266 |
| | ARTIST | 168 |
| | ATTORNEY | 960 |
| | BANKER | 33 |
| | BOOKKEEPER | 200 |
| | CEO | 837 |
| | CONSULTANT | 1208 |
| | CONTRACTOR | 1000 |
| | DESIGNER | 25 |
| | DIRECTOR | 2700 |
| | EXECUTIVE | 1294 |
| | HOMEMAKER | 950 |
| | INFORMATION REQUESTED | 883 |
| | INVESTOR | 657 |
| | MANAGER | 54 |
| | MANAGING DIRECTOR | 453 |
| | NOT EMPLOYED | 86 |
| | OWNER | 773 |
| | PHYSICIAN | 287 |
| | PRESIDENT | 985 |
| | PRINCIPAL | 250 |
| | PROFESSOR | 102 |
| | PROJECT MANAGER | 1000 |
| | REAL ESTATE | 782 |
| | REAL ESTATE BROKER | 500 |
| | REALTOR | 1000 |
| | RETIRED | 1118 |
| | SALES | 594 |
| | SELF-EMPLOYED | 1050 |
| | STUDENT | 2700 |
| | SURGEON | 2800 |
| | TEACHER | 500 |
| | VICE PRESIDENT | 25 |
| *Total Contribution Amount* | | **291009** |

Lastly, to answer the second business question of *"Which campaigning party had the lowest contributions in terms of dollars amongst the four states of California, New Jersey, Florida, and Arizona?"*, I decided that I did not need to create a sixth and final chart. Mainly because my first bar chart of campaign contributions showed this finding.

## 4.2 Hadoop Implementation

To be completed with project number 2

## 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.

# Conclusions

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

When analyzing and answering the business question stated earlier in this project assignment, I was able to discover some interesting facts. One point I was able to discover was the campaigning party who had the highest contributions in terms of dollars between the four states was Kamala Harris. Second was Cory Booker and third was Donald J. Trump. Kamala Harris had raised around $350,000, Cory Booker raised around $291,000, and Donald J. Trump raised around $183,000 from the four states. The lowest campaigning party amongst the four states was a tie between Jay Inslee and Wayne Messam. Both raised only $1,250 dollars amongst the four states.

Then the state who had the highest contribution value amongst the top two political candidates of Donald J. Trump and Kamala Harris was California. With a total of $320,903.96 raised in the state. Florida came in second, New Jersey third, and Arizona last.

Next, in terms of monthly aggregations of contributions from each of the four states, I was able to see there was a spike of contributions from January to March. Contributions went from around $64,000 to $301,000 during this time period. However, after March contributions amongst the four states came crashing down to around $10,000 on average for the remainder of the year.

Then in terms of companies who contributed the most to Kamala Harris campaign were people who were retired, or worked for the following companies: CDR Maguire, Double Bridge, Great Eastern Marking, Homemaker, people self-employed, Universal Chinese Group, or Warwick Strategy Group. All these companies and/or people on average contributed around $2,800 to her campaign.

Lastly in terms of demographic makeup of the top campaigning candidates. Of which people who contributed the most were attorneys, accountants, artists, principals, real estate agents, executive directors, dentists, CEOs, business owners, bankers, people who worked in marketing, physicians, architects, surgeons, or students.

# Works Cited

Government of Canada, Statistics Canada. “5 Data Visualization 5.2 Bar Chart.” *5.2 Bar Chart*, 2 Sept. 2021, www150.statcan.gc.ca/n1/edu/power-pouvoir/ch9/bargraph-diagrammeabarres/5214818-eng.htm.

“Line Graph.” *Kids Graphing Page - Line Graph - NCES Kids’ Zone*, 14 Oct. 2023, nces.ed.gov/nceskids/graphing/classic/line.asp#:~:text=Line%20graphs%20can%20be%20used,for%20what%20is%20being%20measured.

*DBeaver documentation* (2023) *DBeaver*. Available at: https://dbeaver.com/docs/dbeaver/ (Accessed: 10 October 2023).

*KNIME documentation* (2023) *KNIME*. Available at: https://docs.knime.com/ (Accessed: 2 October 2023).

“The Collaborative Data Science Platform | Mode.” *Mode.com*, 2 Oct. 2023, mode.com/. Accessed 2 Oct. 2023.