Report

Assignment 1: Relational Data Model

Political Contributions Analyst

CSC 501

University of Victoria

Submitted to:

Prof. Sean Chester

Submitted by:

Anushka Halder (V00961967)

Sanjana Arora (V00966221)

Tavanpreet S. Oberoi (V00963163)

1. Introduction

Canada holds regular elections for legislatures or governments in various jurisdictions. An important aspect of these elections is how candidates and parties control the sourcing of campaigns. A combination of public and private funds finances the activities of these entities during and outside of elections.
In this assignment, we are focusing on the contributions and the willingness of the different contributors over all the electoral events. We will be analyzing how the contributions have helped parties win elections, the contributor distribution over the provinces, the contribution trend in the past two decades and how these are related to the behavioral and the wealth of different provinces.

We used five comma-separated-values (CSV) files from political contributions data from Elections Canada covering different time periods back to 1993 and the degrees of auditing. From the seven files that were included in the Elections Canada dataset, the 'contributions as reviewed' files were used for analysis. We developed a conceptual model based on our area of interest of analysis and created four tables with the 'details of contributor', 'provincial details of contributors' ,'recipient and respective political parties' and the 'electoral event' information using SQLite3. We then joined the tables based on the primary and foreign keys and performed Exploratory Data Analysis.
As observed from the given dataset, the data size is humongous. It is important to design a model that supports the continuously increasing data keeping the scalability and time complexity of the model in mind.
In the below report, there is a discussion on conceptual and logical model of our data model, keeping the scalability factor in the process.

2. Data Modelling

2.1 Conceptual Data Model

Conceptual Data Model here covers two features:
- Entity
- Relationship (Many-One, Many-Many etc.)

Based on it, 4 major entities are created for our data analysis. Those entities are 'Contributor', 'Recipient', 'EventDetails', and 'Area'. It has been designed such that it can easily be updated with newer dataset and if required more entities can be created and modelled. The process involves creating the model from 'Pre 2000' dataset and then enhancing the model with '2000-2004' dataset. Finally, merging the dataset from 'Post 2000' into the model.
Initially, while modelling 'Pre 2000' dataset only 3 entities are present i.e. 'Contributor', 'Recipient' and 'EventDetails'. When the '2000-2004' dataset is seen, then the entity 'Area' is added and linked with contributor.

Entities description is explained below:
- *Recipient* -> Details of Person/Party receiving funds from contributor
- *Contributor* -> Amount of contribution and details of an individual who contributes during election to support the favorite party
- *Area* -> address of the contributor (multiple contributors could belong to same single postal code)
- *EventDetails* – When and what kind of election take place.

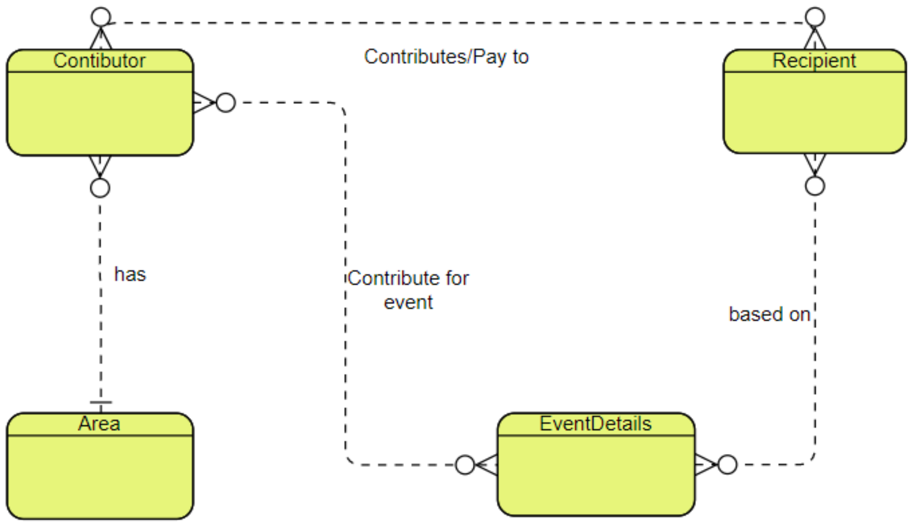


Fig.1. Conceptual Data Model for Elections Dataset

2.2 Logical Data Model

Logical Data Model adds ‘columns’ to the conceptual data model.
Initially while modelling ‘Pre 2000 Contributions to Candidates’ dataset, the entity ‘Recipient’ does not have column ‘Political Entity’, but when other ‘Pre 2000 Contributions to Political Parties’ is seen, then ‘Political Entity’ is added so that all different type of recipients can be merged together which is required for the data analysis. Similar actions are taken when designing the logical data model. It also helps in scaling the model while keeping the concepts of reducing redundancy in mind. Some of the columns like ‘contributor address is dropped because that does not contribute to the required data analysis. Some other actions like renaming the column, merging the columns, dropping the columns are performed on all the files to append the data into single database which can be used for data analysis.

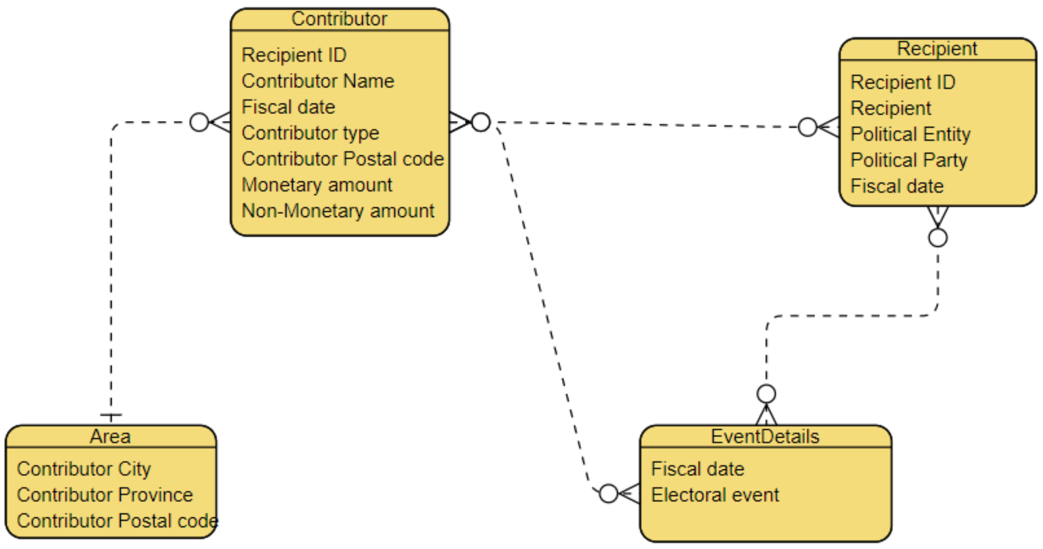


Fig.2. Logical Data Model for Elections Dataset

Pandas library was used for internal data modeling to do basic SQL operations such as querying, joining, discovering and rejecting null values, sorting tables, and indexing. All forms of SQL joins were made feasible thanks to the pandas merge function. Although, employing similar techniques with pandas was not a problem for smaller data sets, pandas had high query times for larger datasets. We later decided to use SQLite for these tasks. The most likely explanation was that pandas fetched data for the code, but SQLite performed coding operations on the data in place, eliminating the requirement for data shifting.

2.3 Physical Data Model

Physical Data models helps in building the database while creating links between how different entities are connected and related to each other. As it can be seen 'contributor postal code(contributorpost)' is primary key for table *'area'* which can be considered as foreign key for table 'contributor', however, it has not been declared foreign key as in dataset 'Pre 2000', there are no postal codes. These observations are included while designing the physical data model.

The representation used in below model is:

Table -> eventdetails	Primary key -> +FiscalDate
Table -> recipient	Foreign key -> *FiscalDate*

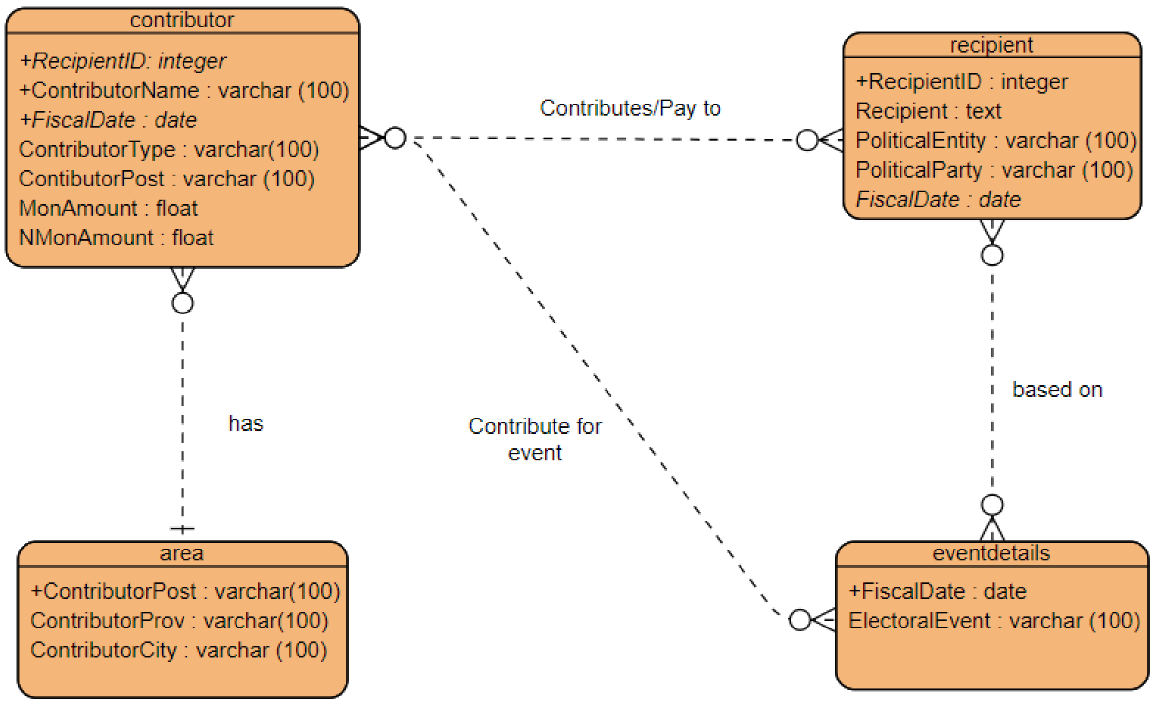

contributor
+RecipientID: integer
+ContributorName : varchar (100)
+FiscalDate : date
ContributorType : varchar(100)
ContibutorPost : varchar (100)
MonAmount : float
NMonAmount : float
recipient
+RecipientID : integer
Recipient : text
PoliticalEntity : varchar (100)
PoliticalParty : varchar (100)
FiscalDate : date
Contributes/Pay to
has
Contribute for event
based on
area
+ContributorPost : varchar(100)
ContributorProv : varchar(100)
ContributorCity : varchar (100)
eventdetails
+FiscalDate : date
ElectoralEvent : varchar (100)

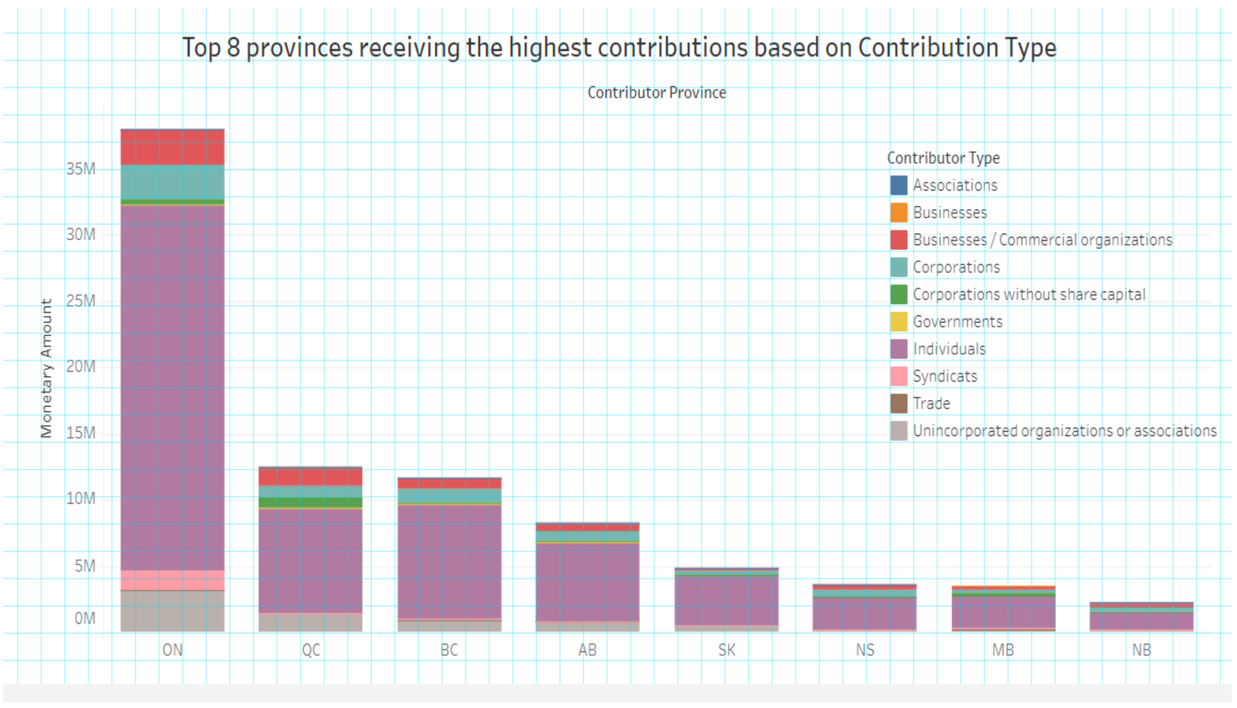
Fig.3. Physical Data Model for Elections Dataset

3. Exploratory Data Analysis

PLOT-1: Top 8 provinces with the highest contribution (In each contributor type)

FILES OR TABLES USED – area.csv and contri.csv
We have used inner join on area and contri tables on Contributor Postal Code column to join the two tables and find the topmost provinces that receive the highest funds for each type of contributor.

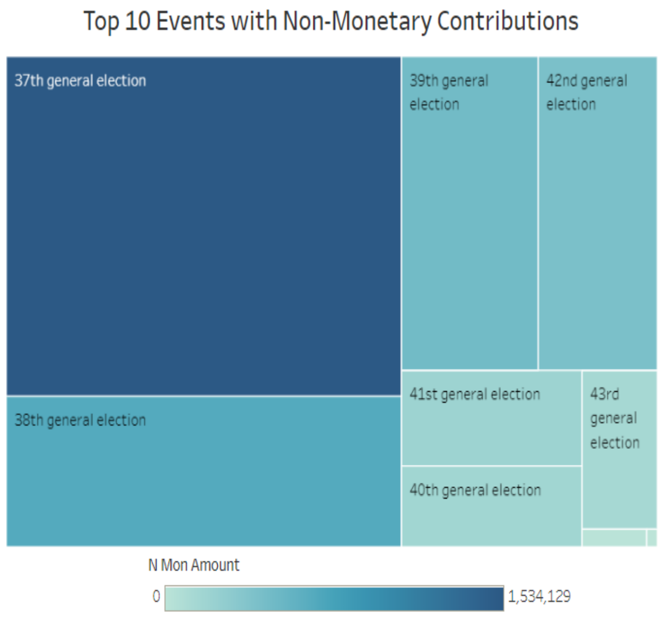
INSIGHTS – We comprehend that the contribution amount is directly proportional to the population. It can also be clearly observed that individuals tend to contribute most in all the top provinces followed by commercial organizations. Governments and smaller associations do not contribute much towards elections. Also, we observe that the bigger provinces have wealthier and more politically educated people who contribute and influence the elections.

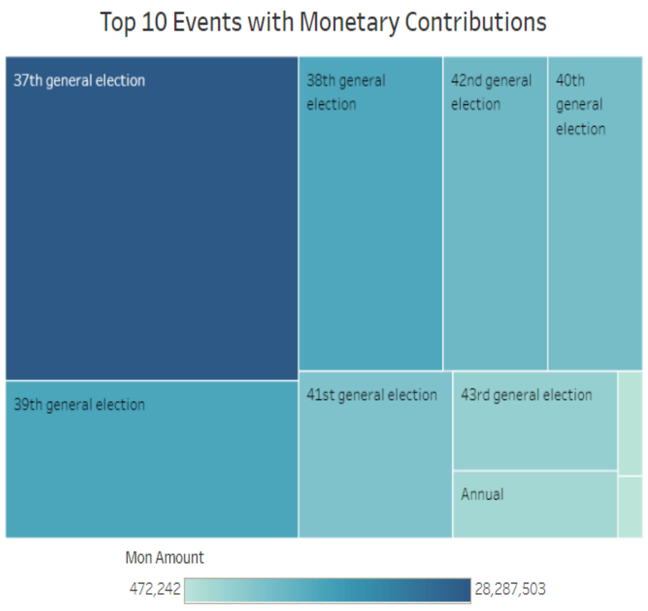


PLOT-2&3: Top 10 events with the highest monetary & non-monetary contributions

FILES OR TABLES USED – event.csv and contri.csv
We have used inner join on event and contri tables on Fiscal Date column to join the two tables and find which elections got the highest contributions.

INSIGHTS – From the visualization we see that contributions are mostly made for the federal elections. The contributions are low in other local elections which entails that people are more interested and involved in the national elections rather than the lower level elections.

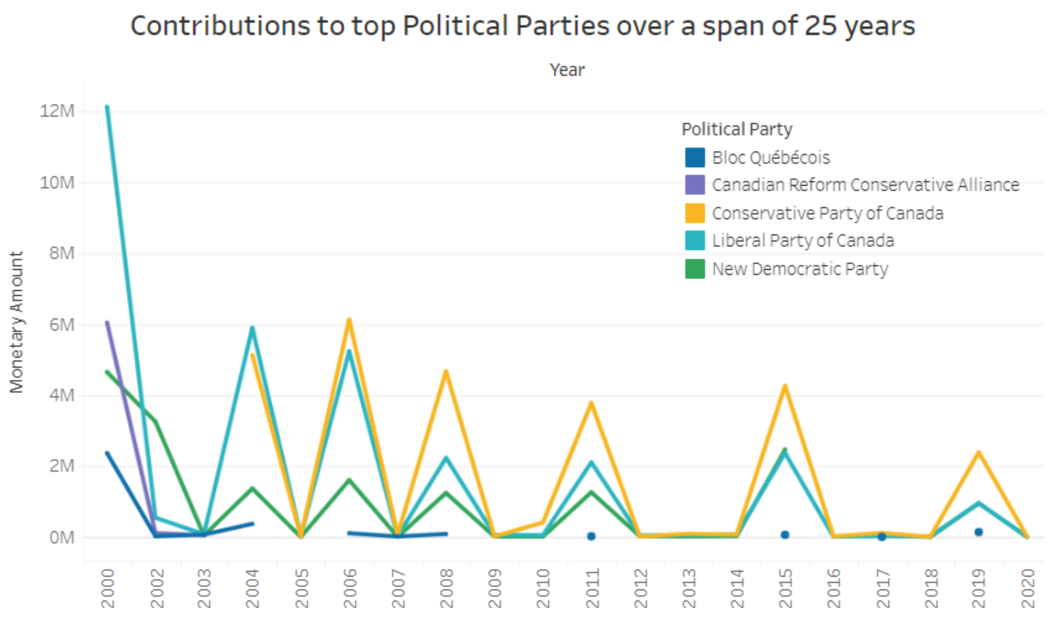




PLOT-4: **Average contribution to top political parties over 25 years**

FILES OR TABLES USED – recep.csv and event.csv
We have used inner join on recep and event tables on Fiscal Date column to join the two tables and find which parties got the highest contributions.

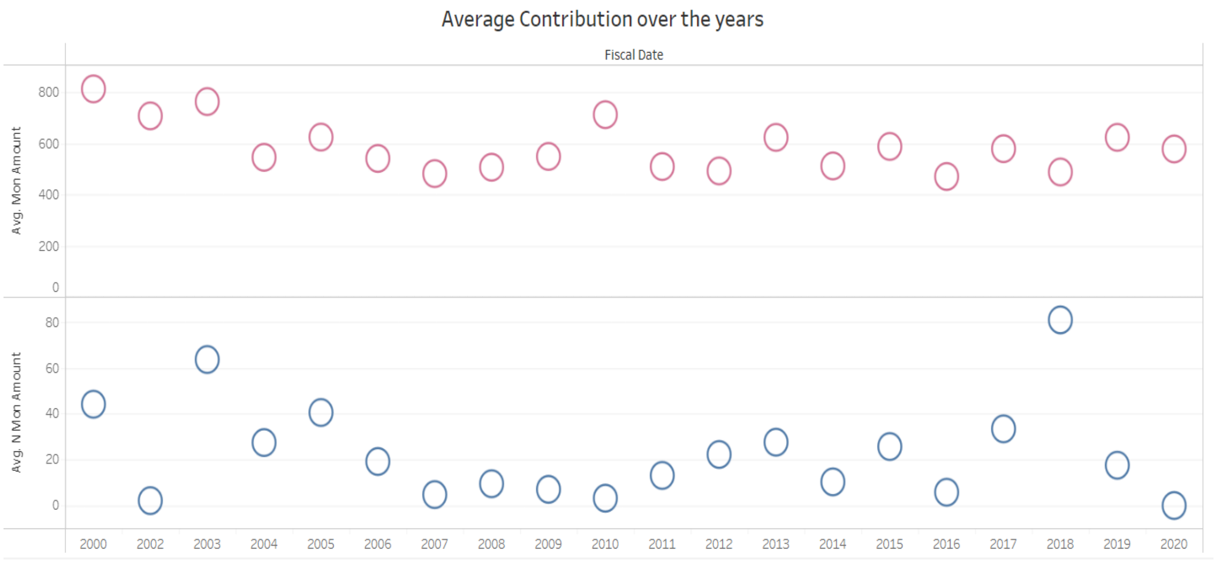
INSIGHTS – Based on the visualization below we have observed that the parties with the highest contributions won the federal elections. In years 2000 and 2004, Liberal Party took away majority seats and also the maximum contributions. However, for the next few elections Conservative party won the elections and the maximum contributions. It shows that contributions can help the parties to conduct influential campaigns and win more supporters.



PLOT-5: Average contribution over 20 years

FILES OR TABLES USED – recep.csv and event.csv
We have used inner join on recep and event tables on Fiscal Date column to join the two tables and find the average contribution over the years.

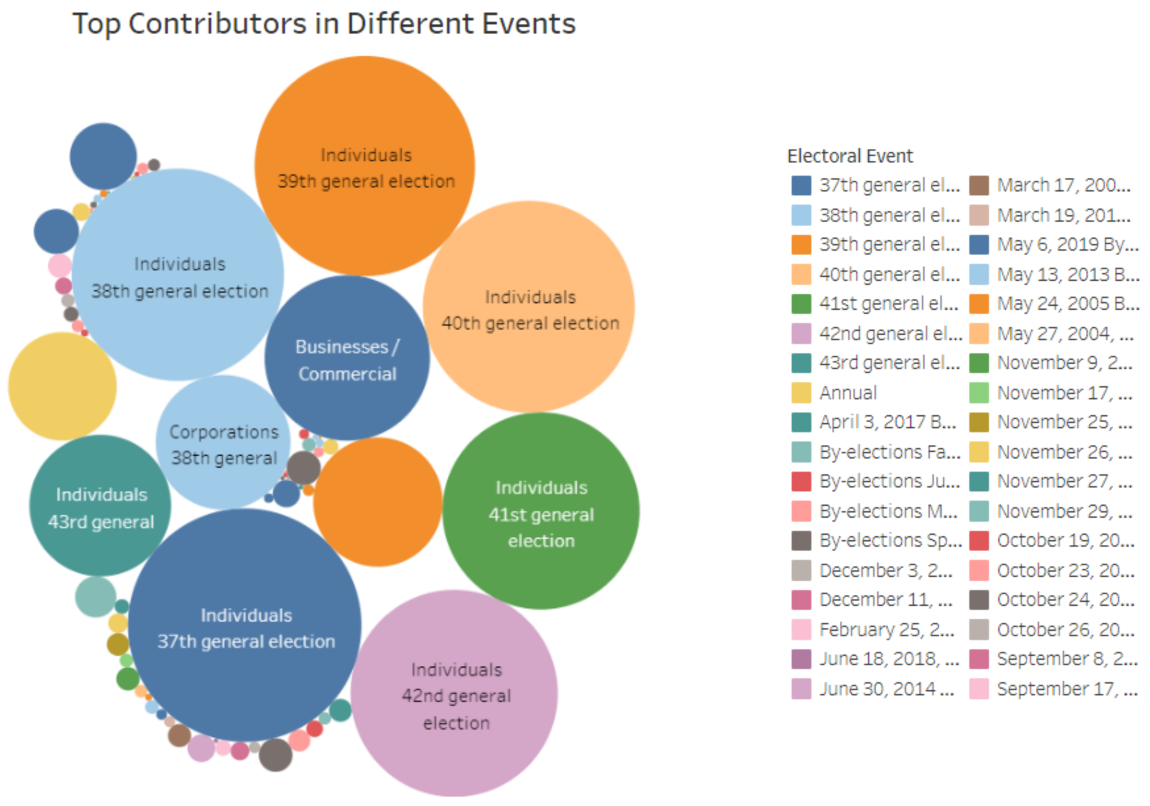
INSIGHTS – From the visualization we have observed that higher amount of contributions are made just before the federal elections are conducted. For example, before 2015, the elections were held in the first quarter so the majority contributions were made a year before the elections were conducted as the campaigns had to be started in the beginning of the year. However, from 2015, the elections were held in the fall session so the contributions are higher in the same year the elections take place.



PLOT-6: Top contributors in different elections

FILES OR TABLES USED – contri.csv and event.csv
We have used inner join on contri and event tables on Fiscal Date column to join the two tables and find top contributors in different elections.

INSIGHTS – This chart shows the highest contributor type in different events and the proportion of contributors among the various events. The federal elections have the highest individual contributors. 39th general election has the highest contributors. This can be related to the fact that Conservative Party came to power in 2006 after defeating Liberal Party who reigned from 1993-2006. In 2006 elections, many Conservative supporters contributed in order to impact the elections.



4. Scalability

Processor – Intel(R) Core(TM) i7-4510U CPU @ 2.00GHz 2.60 GHz

RAM – 8GB

Tool/software – Jupyter Notebook and Google Colab

Language – Python 3.7 and SQLite3

Resource Consumption

Google colab provides each user with 61.54 GB memory. For comparing the resource consumption by the original data files and our data model created on the SQL database, we downloaded the tables into .csv format. The Figures 4 and 6 below show the RAM & Disk consumption with the original five files were being used in Google colab. While, Fig.5 and Fig.7 show the amount of RAM and disk consumption when the 4 files of the tables that we created. As can be seen from comparison of these images, the original data files are consuming much more RAM than the data model we have created. Further, as seen from Fig.5 and Fig.7, the amount of memory available after the original .csv files is 60.16 GB, while it is 61.48 GB with the .csv files of the tables modeled. Therefore, the fetching from tables modeled is much more efficient in terms of memory and resource consumption as compared to the original dataset.

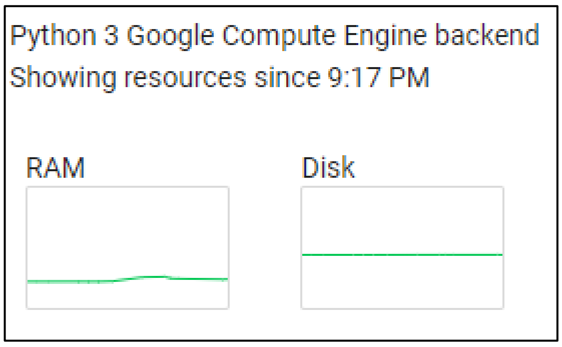

Python 3 Google Compute Engine backend
Showing resources since 9:17 PM
RAM
Disk

Fig.4. RAM and Disk for original data

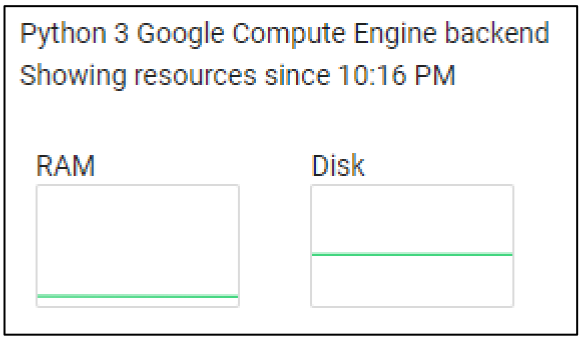

Python 3 Google Compute Engine backend
Showing resources since 10:16 PM
RAM
Disk

Fig.5. RAM and Disk for our tables

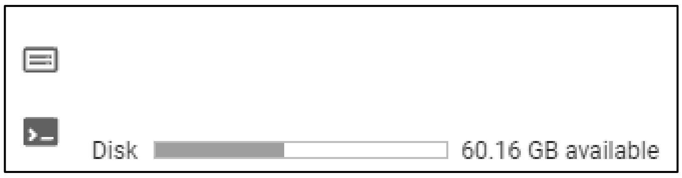


Fig.6. Memory available after original data

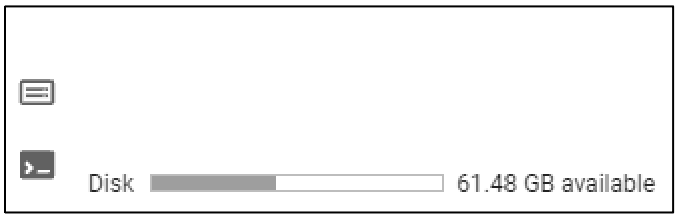


Fig.7. Memory available after our tables

Removing Data redundancy

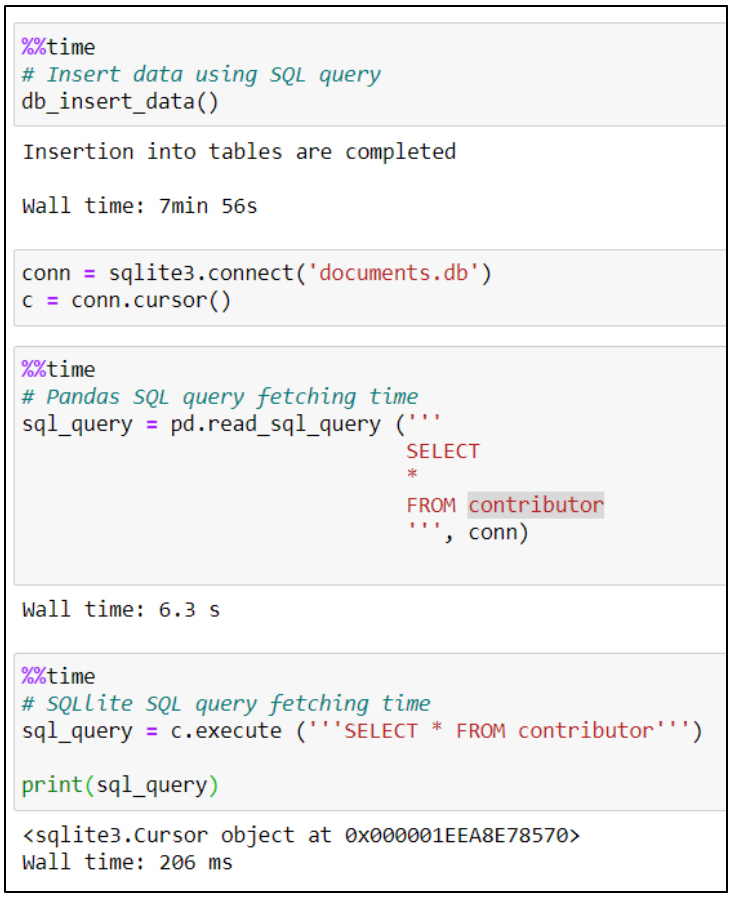
Further, if we would have stored the area related details of each data file in separate tables, the number of rows being used only by the area related details is **228141.** Our data model allows storing the area information of all the five data files in one table. The number of rows fetched from the area table is **195251** and hence, the data model has reduced data redundancy by storing only the distinct area details because of the presence of postal code attribute.

Data Scalability wrt. Data Analysis

As also captured in the Exploratory Data Analysis section of this report, we wanted to capture the analysis of contributions made for all the available data from 1993 to present. Hence, the data model is built such that it does not require separate tables to be created for a specific year range. Whenever, some new data needs to be added to this data model, we could easily do that by adding the data to these tables with respect to the attributes. Further, incase new data comes with some new attributes; those attributes could be easily included by creating more tables and relating those with the current tables.

<u>Performance</u>

As shown in the Fig.8, the amount of time taken to access the tables using pandas is 6.3 secs, while it is 206ms by using the SQL query. Hence, we preferred using the SQL queries for further calculations of our data analysis.



```
%%time
# Insert data using SQL query
db_insert_data()

Insertion into tables are completed

Wall time: 7min 56s

conn = sqlite3.connect('documents.db')
c = conn.cursor()

%%time
# Pandas SQL query fetching time
sql_query = pd.read_sql_query ('''
                               SELECT
                               *
                               FROM contributor
                               ''', conn)

Wall time: 6.3 s

%%time
# SQLlite SQL query fetching time
sql_query = c.execute ('''SELECT * FROM contributor''')

print(sql_query)

<sqlite3.Cursor object at 0x000001EEA8E78570>
Wall time: 206 ms
```

Fig.8. Python pandas vs. Python SQL query