University of Victoria

CSC 501

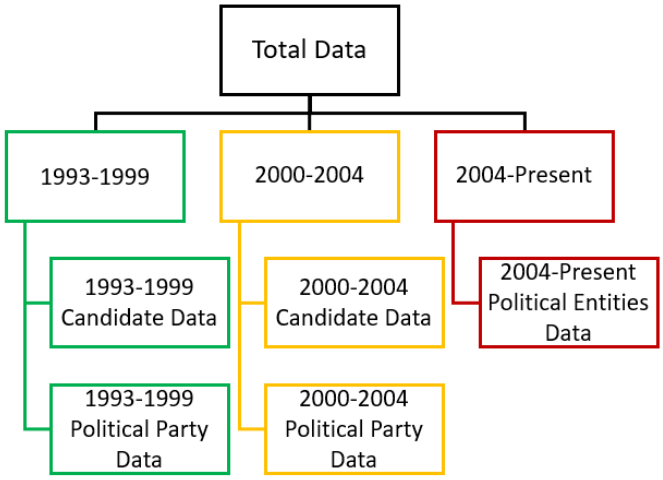
Report (Assignment-1)

Submitted by:

Name	Student ID
SIDDHARTH CHADDA	V00947906
DHRUVRAJ SINGH	V00970352
ABHISHEK KATHURIA	V00959831

Data Modelling Strategy

Inorder to retain the maximum amount of information and conduct an in-depth data analysis of the dataset we have chosen to model the datasets by grouping together the candidate, political party, political entities data according to specific time periods, namely: 1993-1999, 2000-2004, 2004-present. To summarize:



Conceptual Data Model

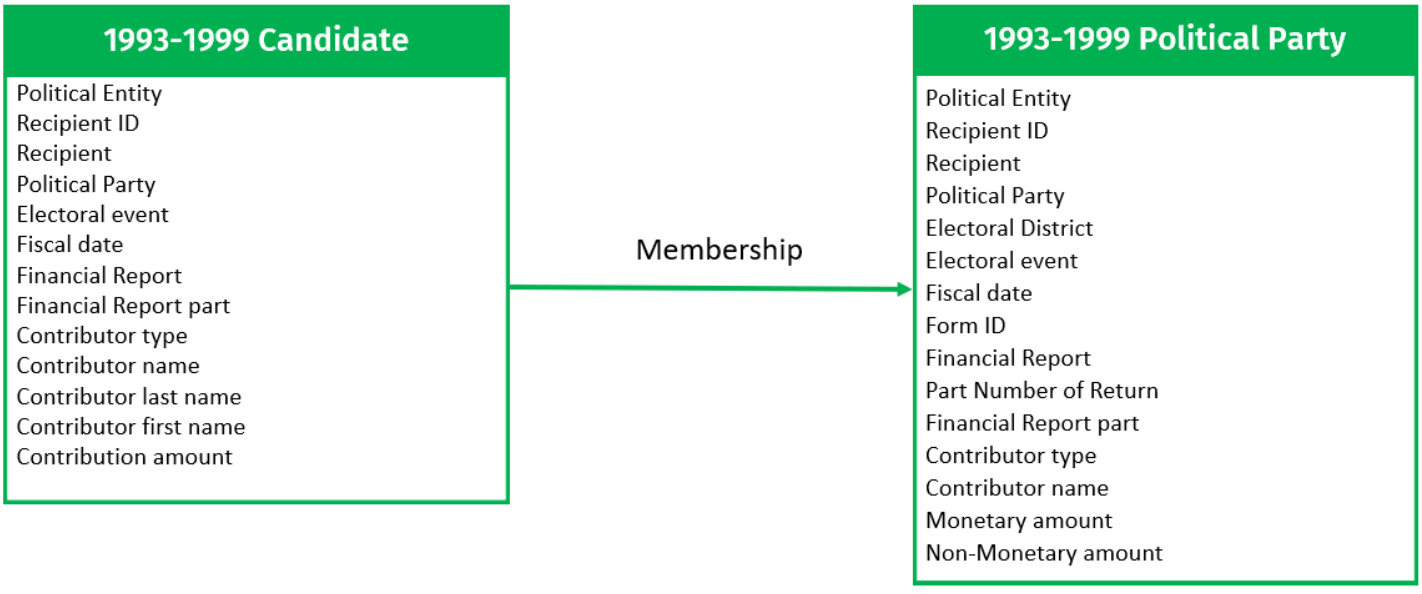
Conceptual Model Justification:

Inorder to better understand the realtionship between political party, candidates and contribution amount received, and to understand how their relationship has evolved over the time period from 1993-present, we have chosen to group the candidate, political party and contribution data present within the 5 datasets according to their respective time periods (1993-1999, 2000-2004, 2004-present).

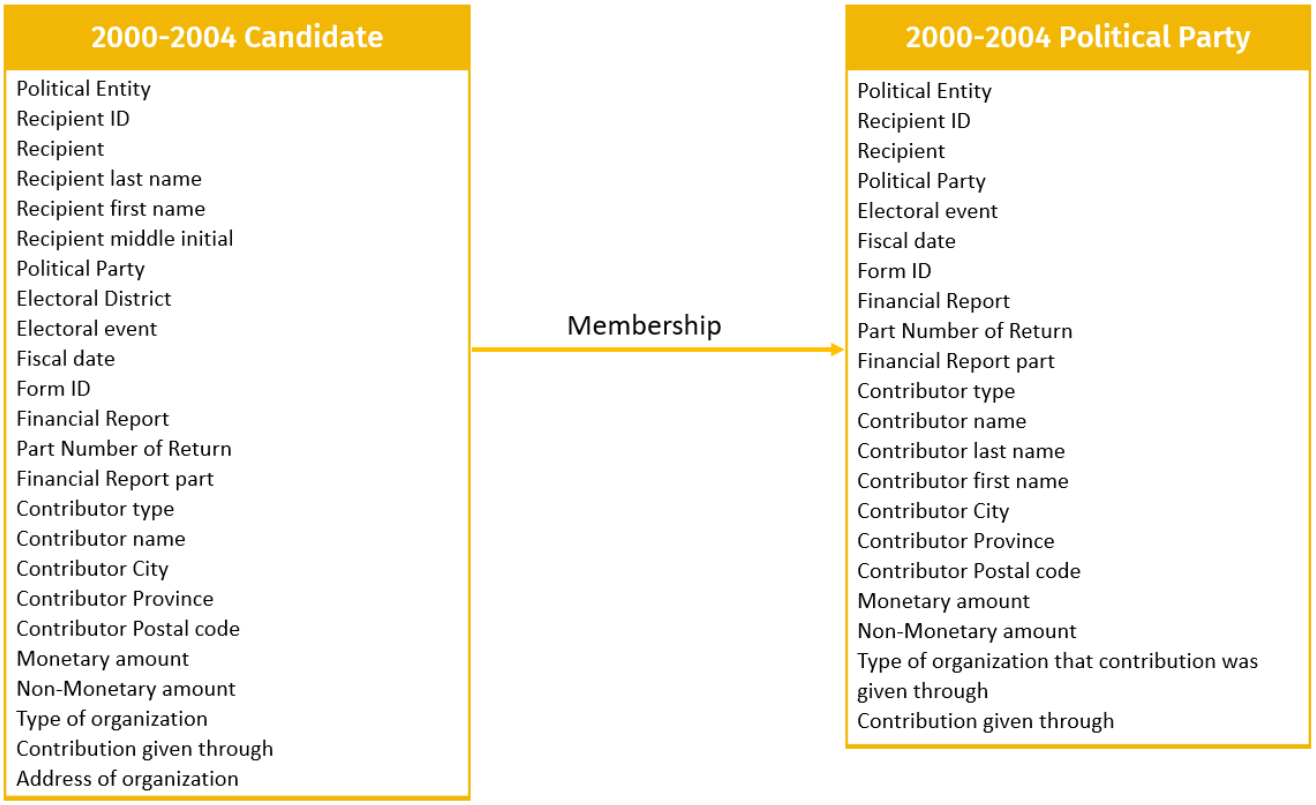
This modelling strategy has 2 advantages: (1) First it allows us to preserve the maximum amount of information from the original datasets and (2) It introduces modularity in our dataset and allows us to conduct an independent and more holistic investigation to uncover the underlying hidden relationships present in the 5 datasets.

Going by the above strategy in mind we have constructed the below conceptual data models:

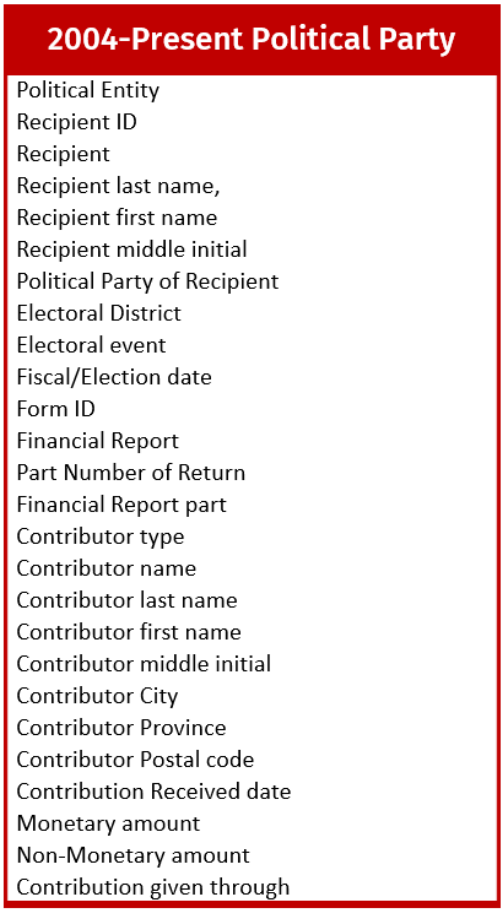
For the year 1993-1999



For the year 2000-2004



For the years 2004-Present



Logical Data Model

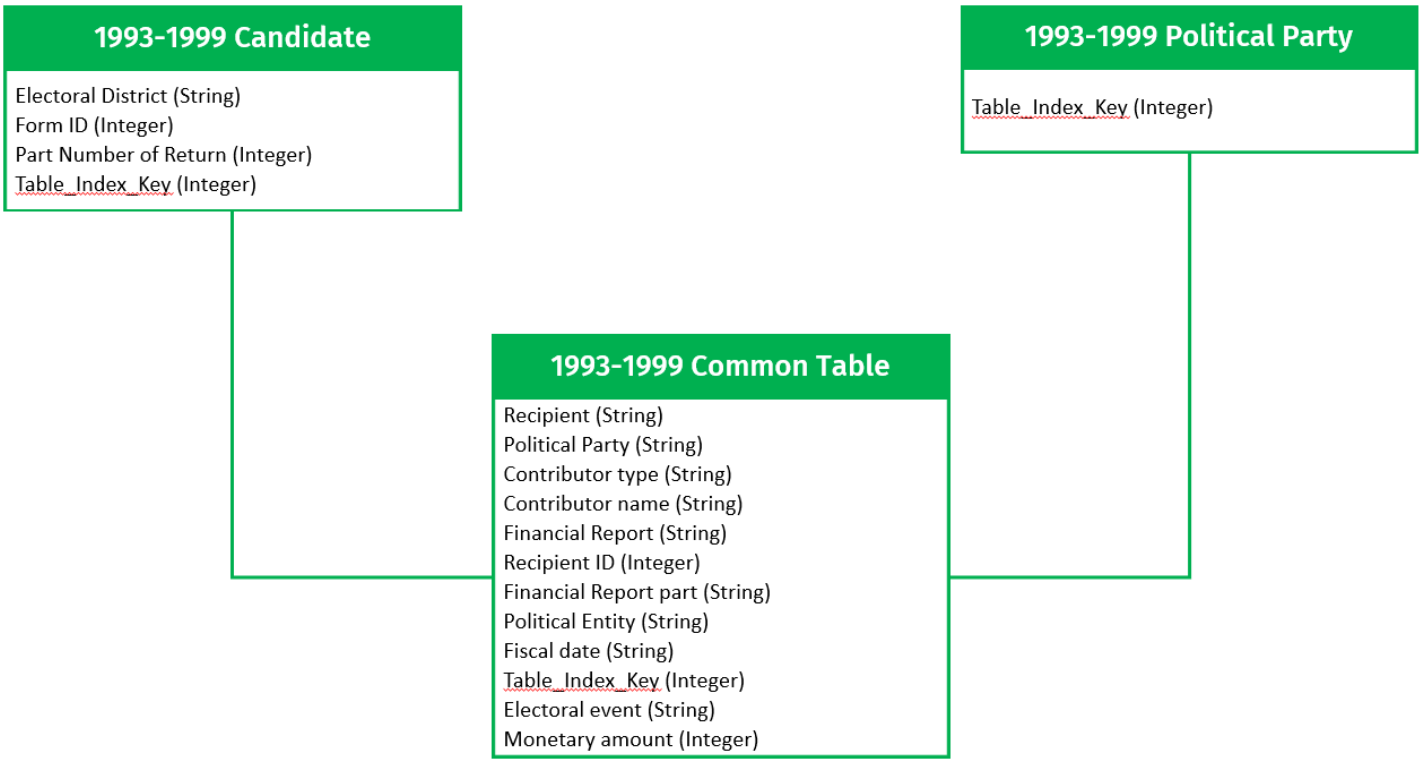
Logical Model Justification:

We studied the dataset in-depth and have implemented the below Normalization and Redundancy removal Techniques:

i. First Normal Form Implemented: For optimization the table now only contains atomic attributes
ii. Second Normal Form Implemented: No Partial Dependency exists in the tables
iii. Third Normal Form Implemented: No transitive dependency exists in the tables
iv. Superfluous attributes have been removed, For eg. First Name, Middle Name, Last Name columns concatenated into Name column
v. To build homogeneity and to further reduce the redundancy we have groped the similar elements from both the candidate and the political party tables into one common table (this is done according to the respective time periods)

Going by the above design strategies we have constructed the below logical data models:

For the year 1993-1999



For the year 2000-2004

2000-2004 Candidate

Electoral District (String)
Address of organization (String)
Table_Index_Key (Integer)

2000-2004 Political Party

Table_Index_Key (Integer)

2000-2004 Common Table

Recipient (String)
Contributor name (String)
Contribution given through (String)
Political Entity (String)
Non-Monetary amount
Political Party (String)
Financial Report (String)
Part Number of Return
Contributor City (String)
Contributor type (String)
Form ID (Integer)
Fiscal date (String)
Contributor Province (String)
Contributor Postal code (String)
Monetary amount (Integer)
Type of organization (String)
Recipient ID (Integer)
Financial Report part (String)
Table_Index_Key (Integer)
Electoral event (String)

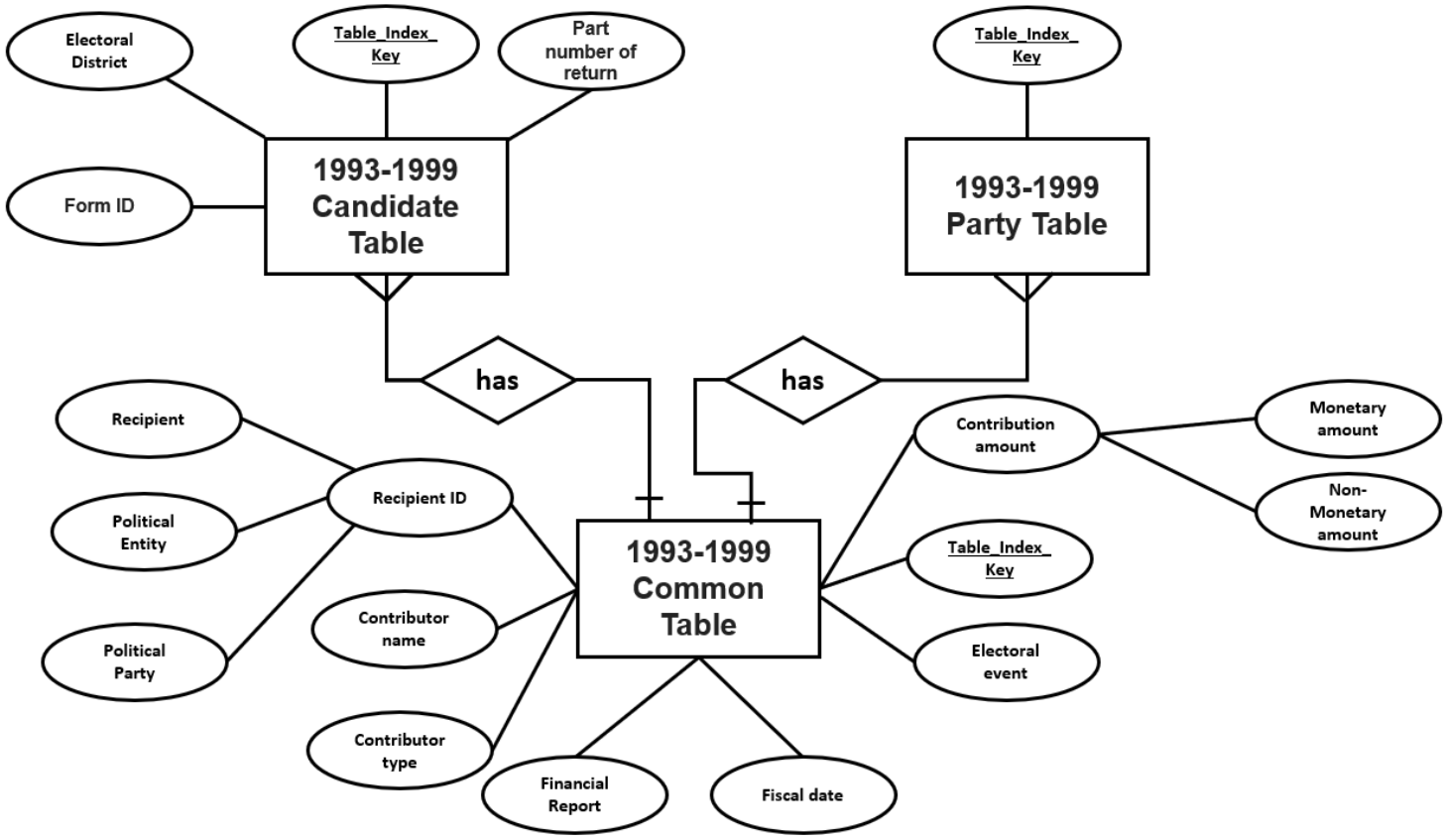
For the years 2004-Present

2004-Present Political Party

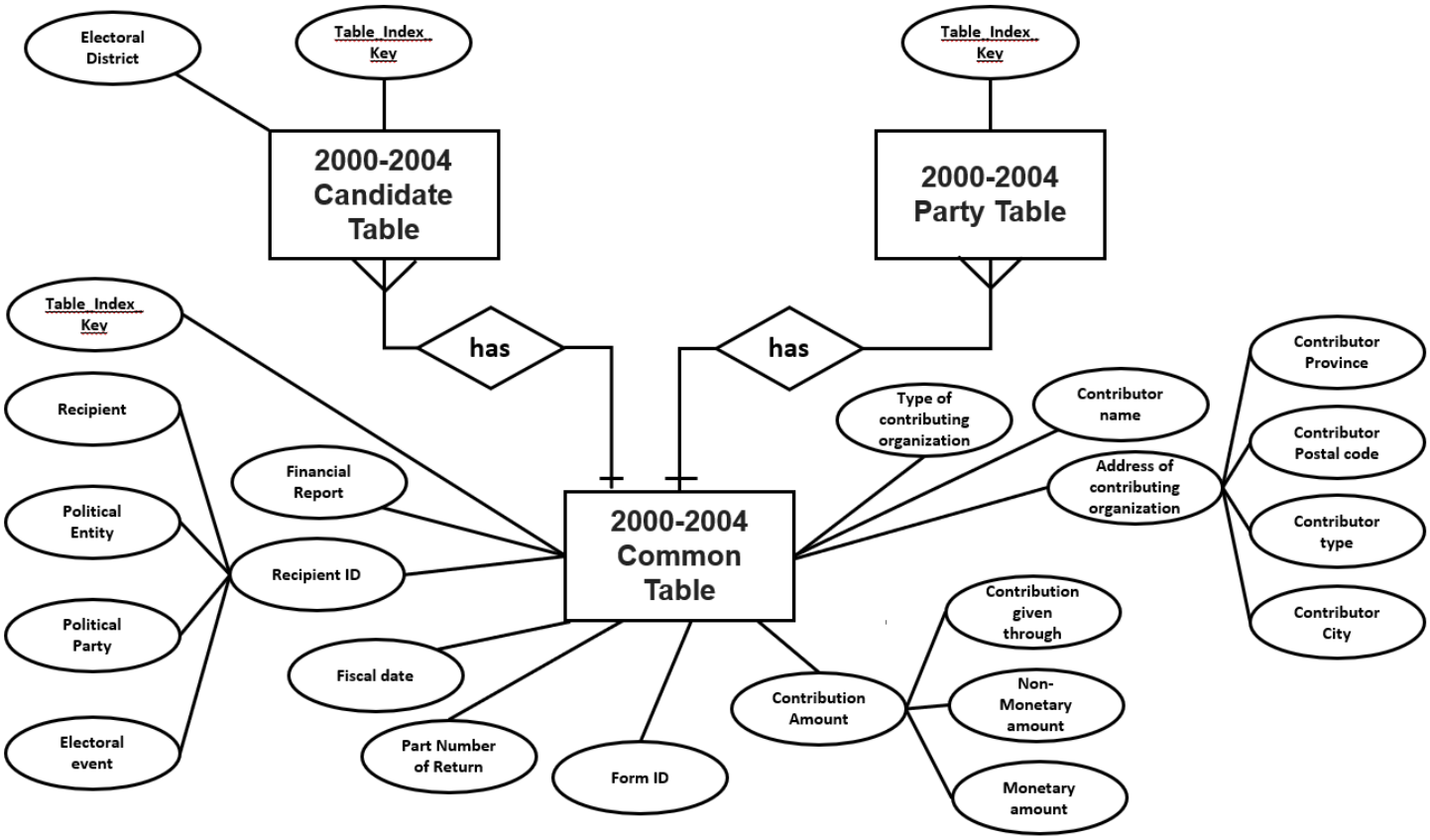
Political Entity (String)
Recipient ID (String)
Recipient (String)
Political Party of Recipient (String)
Electoral District (String)
Electoral event (String)
Fiscal/Election date (String)
Form ID (Integer)
Financial Report (String)
Part Number of Return (Integer)
Financial Report part (String)
Contributor type (String)
Contributor name (String)
Contributor City (String)
Contributor Province (String)
Contributor Postal code (String)
Contribution Received date (String)
Monetary amount (Integer)
Contribution given through (String)

ER Diagrams

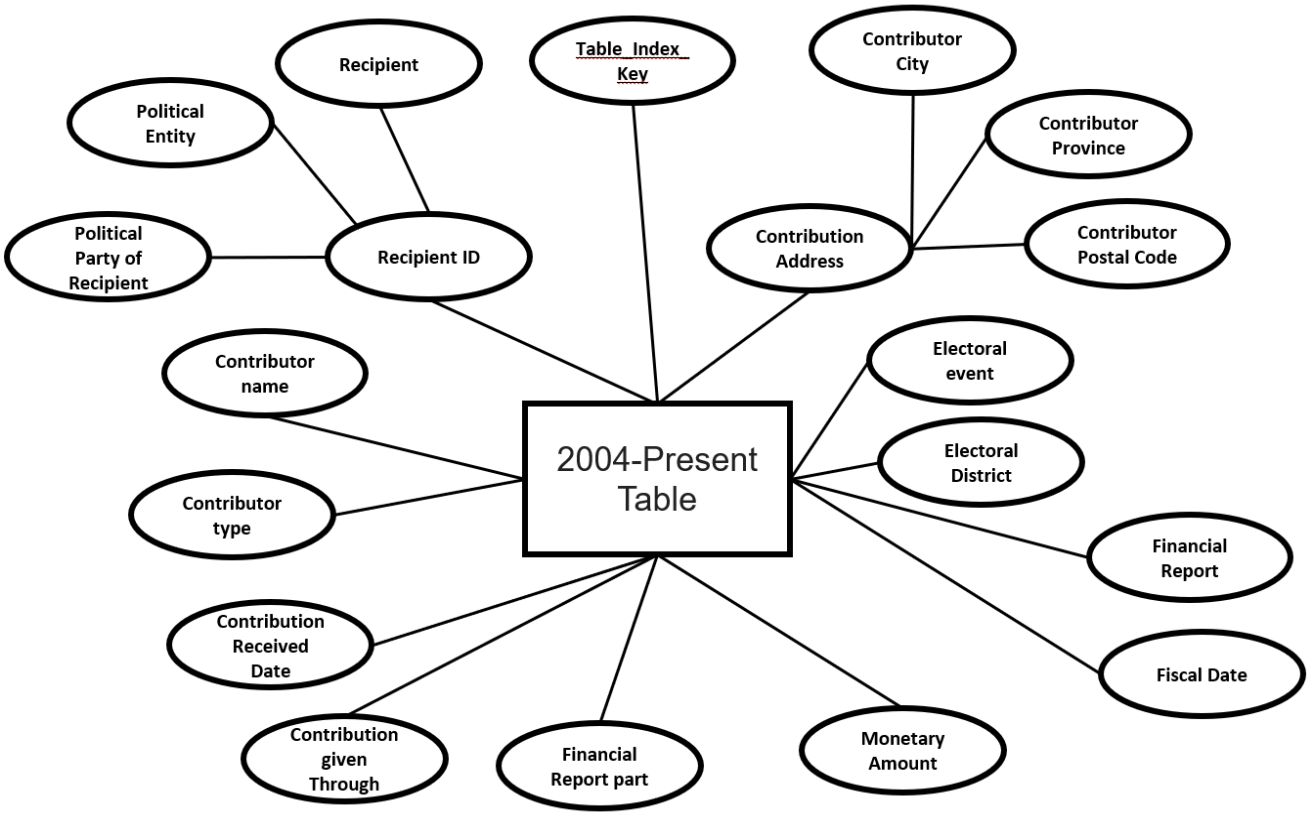
For the year 1993-1999



For the year 2000-2004



For the years 2004-Present



Exploratory Data Analysis

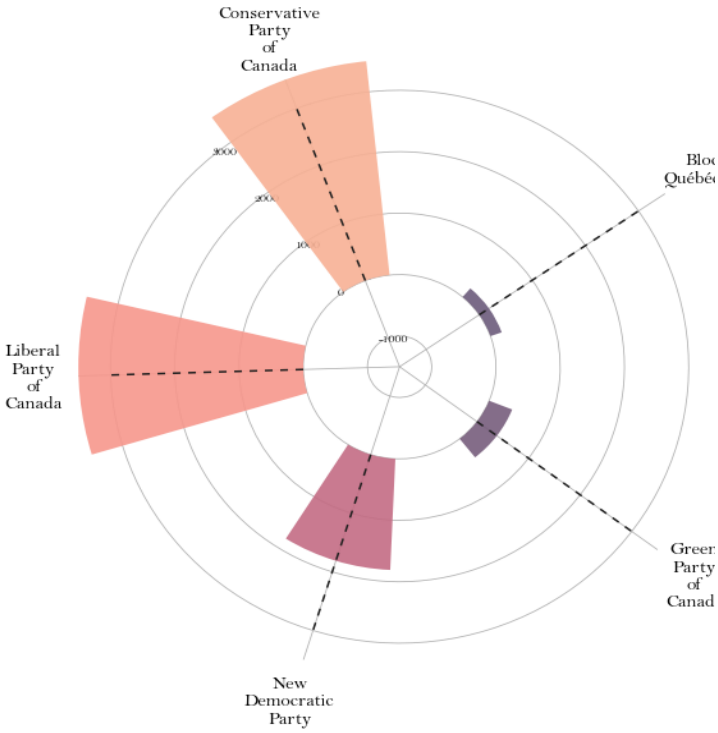
Plot 1: Political Party in terms of Monetary amount

SQL Tables used: party_annual_2000-2004_contributors_e, party_annual_1993-2000_contributors_e, candidate_pre_2000_contributors_e, candidate_2000_2004_contributors_audt_e

Question: Which Political Party occupied the largest portion of the Monetary amount in the 1993-1999, 2000-2004 and 2004-Present elections? Did the same party win the election for that particular year?

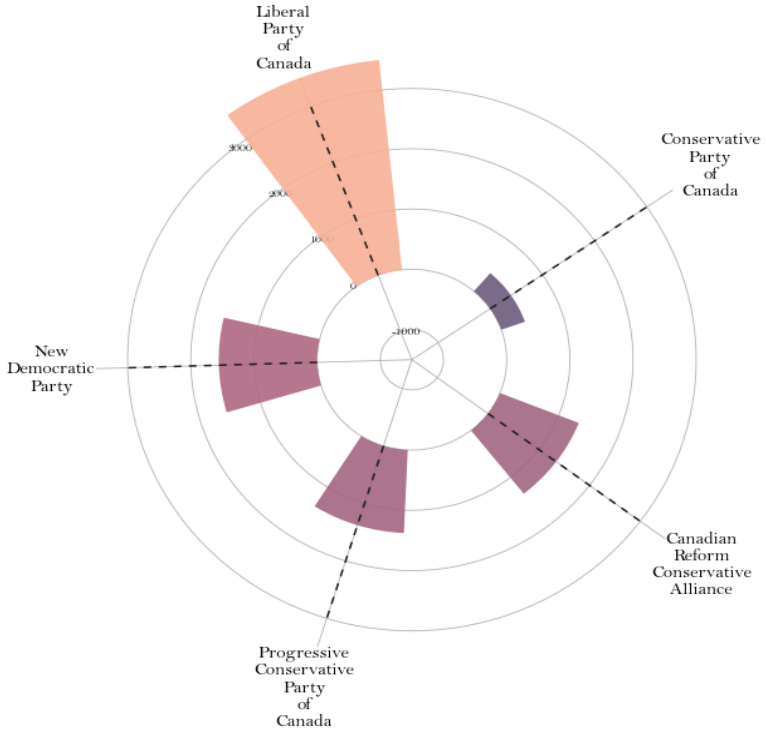
Insights: From Graph.1, Graph. 2, Graph. 3, the Liberal Party of Canada received the highest Monetary amount for all the three elections. Furthermore, according to our research from Wikipedia, the Liberal Party of Canada won in all the three elections which makes our assumption, the party with highest monetary amount received wins the election, true.

Political party in terms of Monetary Value from 1993-199



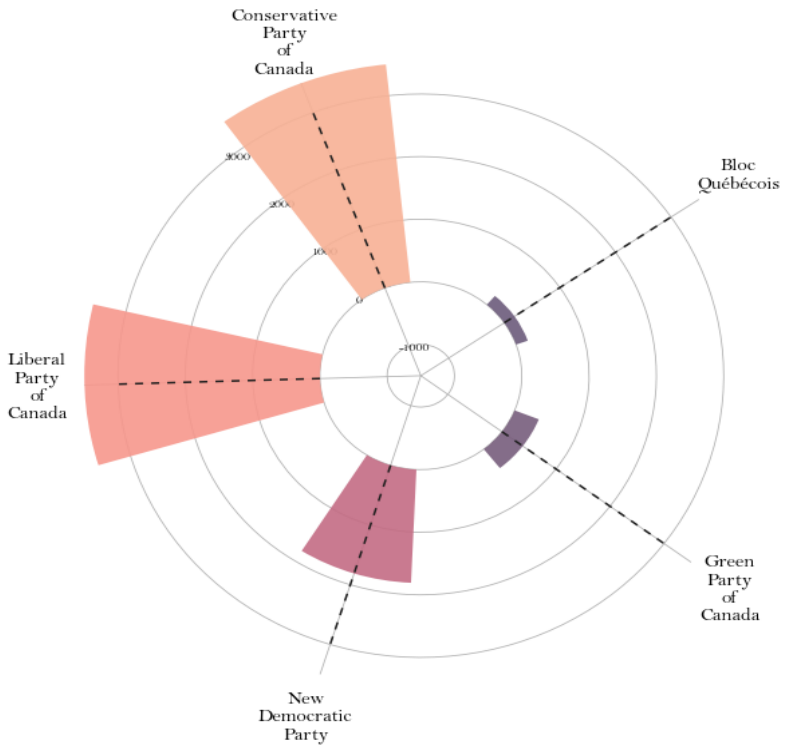
Graph. 1

Political party in terms of Monetary Value from 2000-2004



Graph. 1

Political party in terms of Monetary Value from 2004-Present



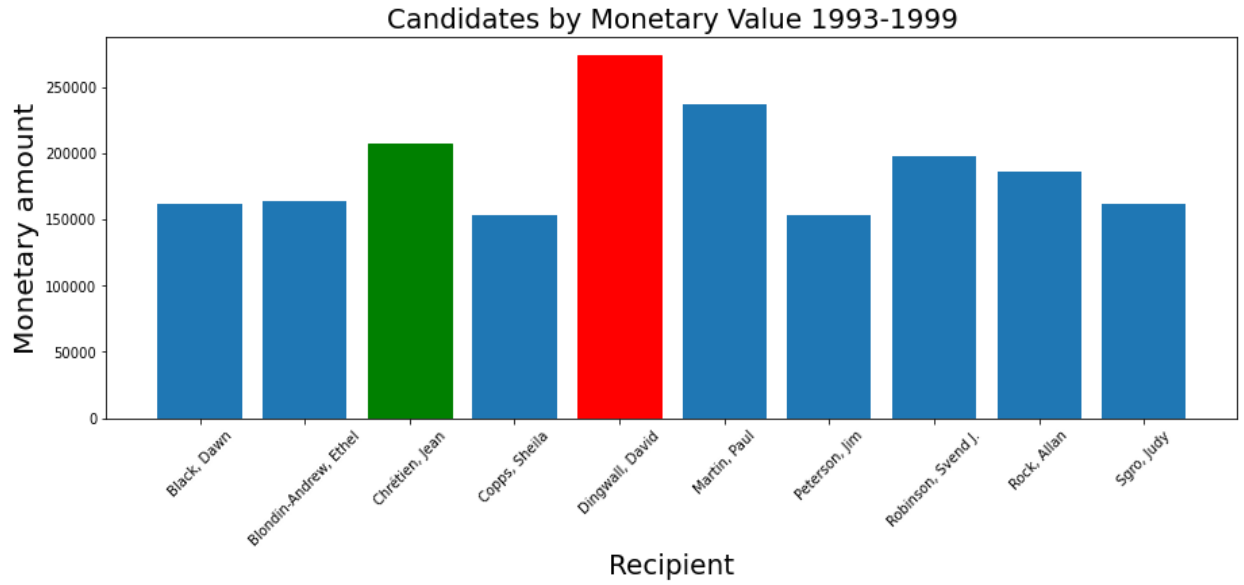
Graph. 3

Plot 2: Candidates in terms of Monetary amount

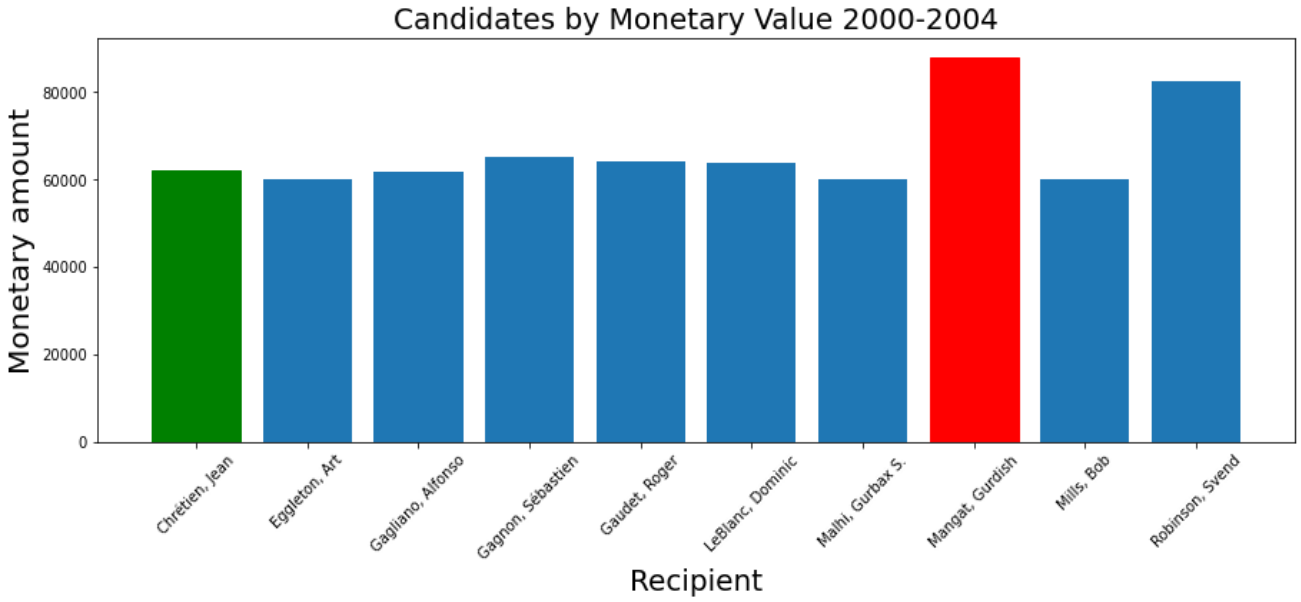
SQL Tables used: party_annual_2000-2004_contributors_e, party_annual_1993-2000_contributors_e, candidate_pre_2000_contributors_e, candidate_2000_2004_contributors_audt_e

Question: How was the Monetary amount distributed between the Candidates for 1993-1999 and 2000-2004 elections? Does the candidate with the highest monetary amount win the elections?

Insights: Even though the candidates David Dingwall and Gurdish Mangat got the highest Monetary amount for the 1993-1999 and 2000-2004 elections respectively, the winner for both the elections was Jean Chretien, denoted by the green bar in Graph. 4 and Graph. 5.



Graph. 4



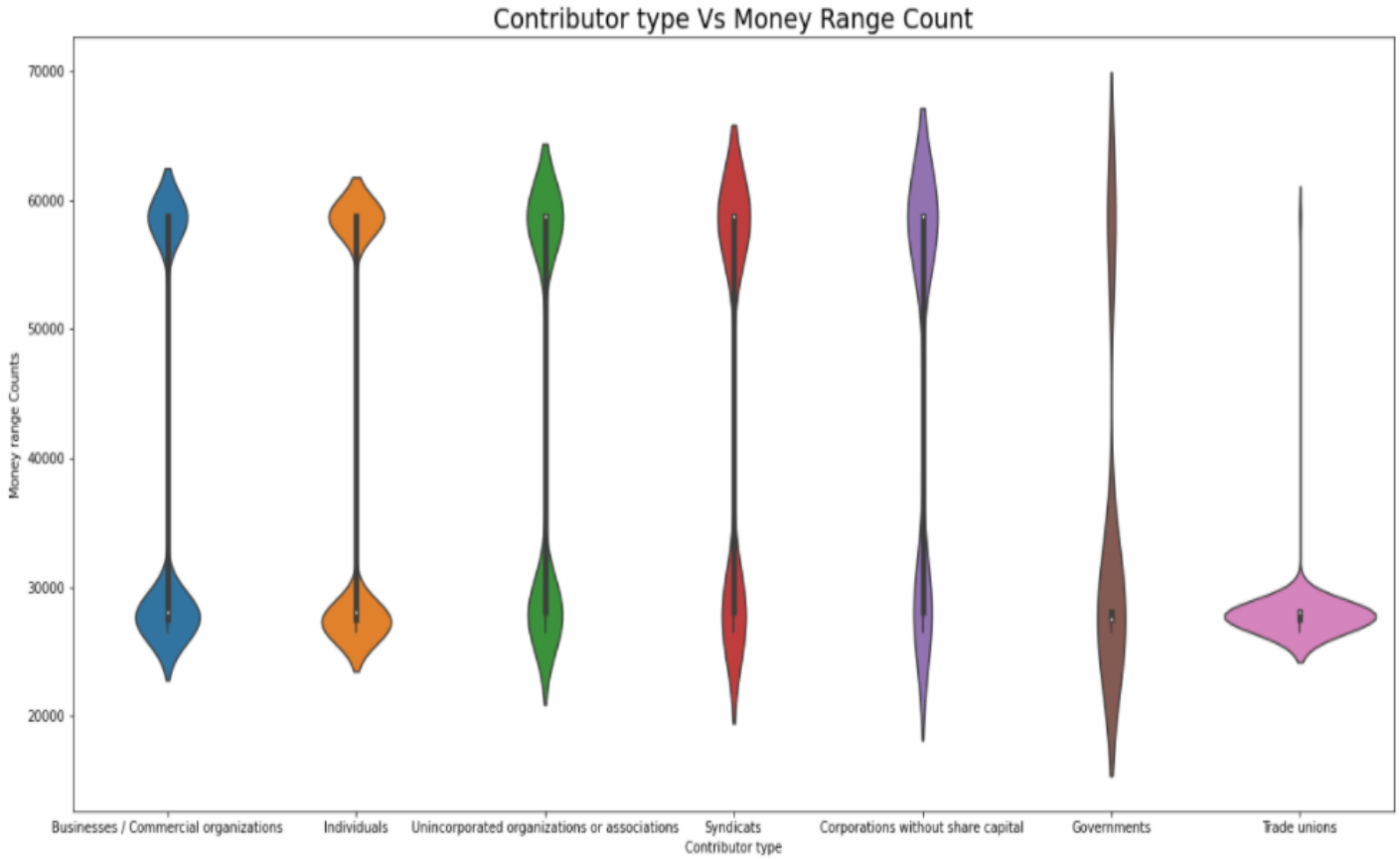
Graph. 5

SQL Tables used: od_cntrbtn_audt_e

Plot: Money count range by different Contributor types

Question: What is the distribution between the Money Count and the Contributor type for the election dataset from 2004-present?

Insights: As we can observe from the violin plot in Graph. 6, the maximum amount count ranges from 25K to 30K and 57K to 60K. The extremes show the amount range for the different types of contributors. Both Business/Commercial organizations and Individuals had the highest contribution for the 2004-present election.



Graph. 6

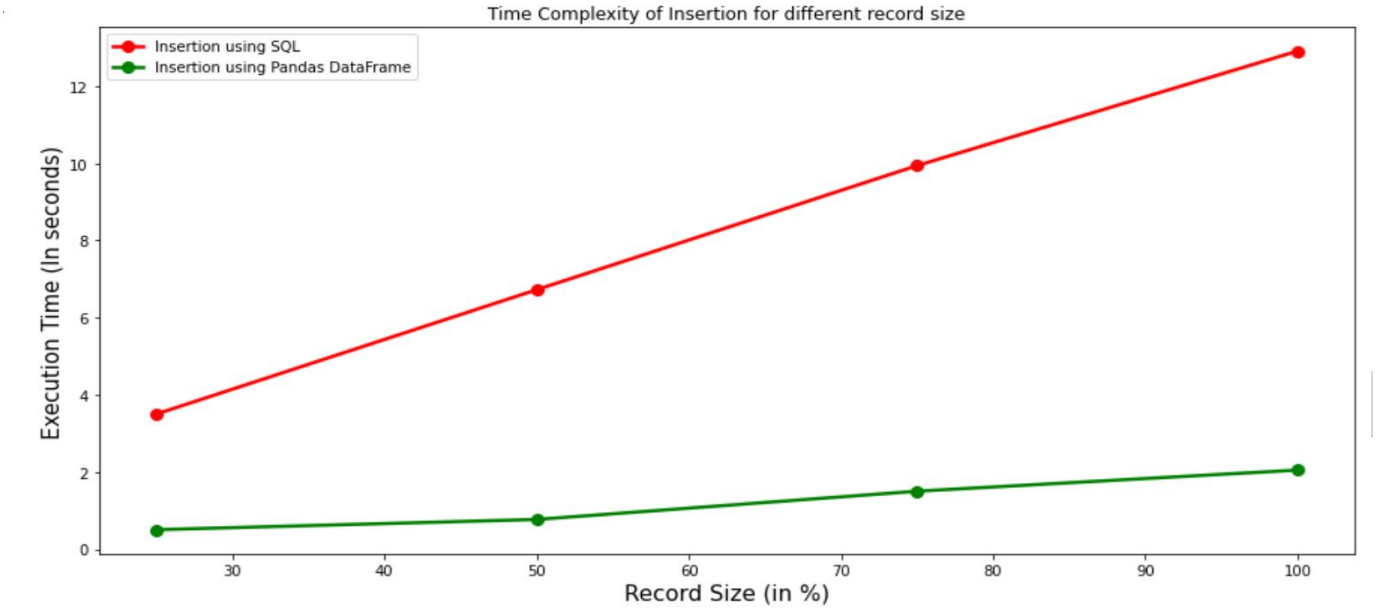
Experiments on Scalability

Experiment 1: Comparing the Execution Time for Insertion of records of different sizes using simple SQL Queries vs Pandas based DataFrame Technique.

SQL Tables used: od_cntrbtn_audt_e

Insights: It can be clearly observed from the below Graph that as the record size of the dataset increases, the execution time for insertion of records in the table using SQL queries (indicated in red) increases steeply as compared to the execution time for insertion of record in the table using Pandas DataFrame (indicated in green). Considering just the 25% of the total records, it is evident that the time for insertion using Pandas DataFrame is 0.502 seconds whereas for SQL queries, it is 3.498 seconds which is way higher for insertion of small chunk of records in the table. It is also noticeable from the Graph that when all the records were inserted in the table, the time taken by Pandas DataFrame was 2.047 seconds but for the SQL queries, it is 12.90 seconds which is 3.7 times more than the time taken to insert 25% of the records.

This shows that for insertion of records in the table, Pandas DataFrame is more optimized as it uses multithreading.



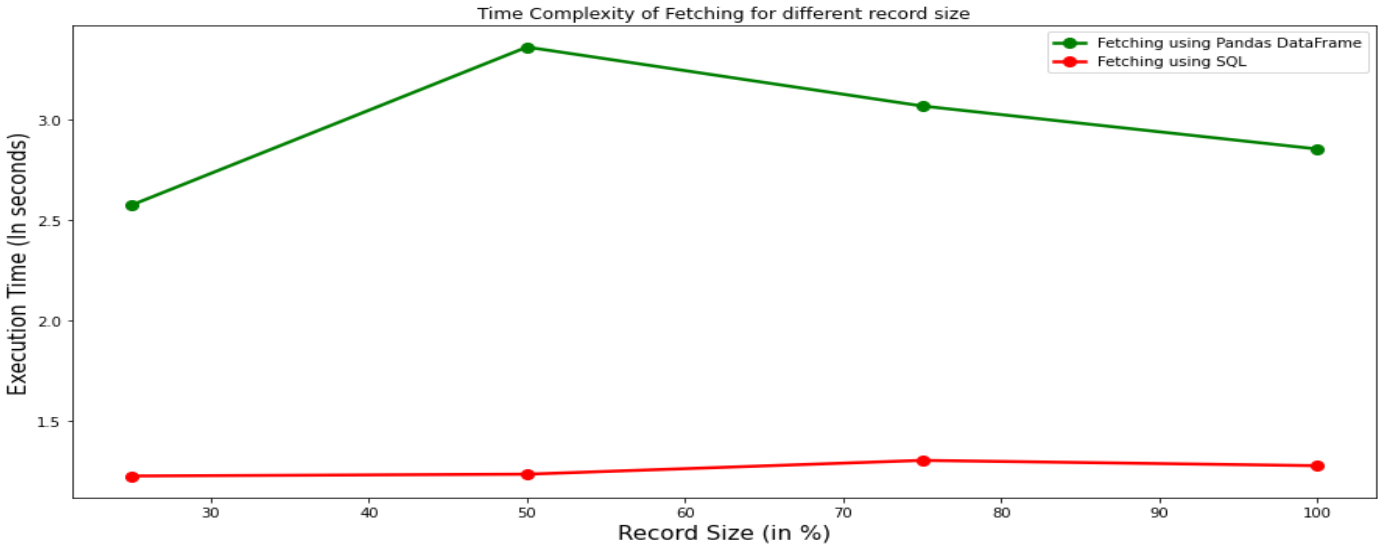
Graph. 7

Experiment 2: Comparing the Execution Time for sequential Fetching of records of different sizes using simple SQL Queries vs Pandas based DataFrame Technique.

SQL Tables used: od_cntrbtn_audt_e

Insights: The below Graph shows the time taken to fetch the data sequentially from local database using SQL queries as well as Pandas DataFrame. It can be noted that Pandas DataFrame has the highest execution time for fetching 50% of records with the value of 3.35 seconds. As the dataset size increases, the time taken fetching records using Pandas DataFrame first drastically increases and then slowly decreases with the increase in volumetric data. But, on the other hand, as the percentage of the records fetched increases, the execution time for fetching using SQL queries remains within the range of 1.22 seconds to 1.26 seconds.

This shows that for fetching the records sequentially, we should prefer using SQL queries as they are able to handle large volumetric data.



Graph. 8

Execution Time for Insertion

% of Records inserted	Execution time using Pandas Dataframe	Execution time using SQL queries
25%	0.50 seconds	3.49 seconds
50%	0.76 seconds	6.72 seconds
75%	1.50 seconds	9.93 seconds
100%	2.04 seconds	12.90 seconds

Execution Time for Fetching

% of Records fetched	Execution time using Pandas Dataframe	Execution time using SQL queries
25%	2.57 seconds	1.22 seconds
50%	3.35 seconds	1.23 seconds
75%	3.06 seconds	1.30 seconds
100%	2.85 seconds	1.27 seconds

Experiment 3: Calculating the space saved by data modelling

Type of Database	Memory size
Uncleaned Database	5.29 GB
Cleaned Database	4.09 GB

In the above table, we show the comparison between the memory size of the uncleaned data (raw unmodelled data) and cleaned data (modelled data). The memory size for raw Database was 5.29 Gigabytes.
Various operations were performed on the original database such as dropping repeated columns, combining related columns, handling Nan values, taking out common columns, etc. This resulted in a new cleaned database with a memory size of 4.09 GB.

Total Space saved = 1.2 GB
The percentage of space saved = [(5.29 - 4.09) / 5.29] * 100= 22.68%