# Project Report

## Summary

This project aims to demonstrate data extraction, transformation and loading (ETL) skills. As part of the scope, an analysis of election data from India (quinquennial in frequency) has been used. Data was scraped from Wikipedia, and loaded in pandas dataframes. Transformations on dataframes were made to standardize the column names and data structures. Eventually, this data was exported into CSV files and imported into a postgres database using pgAdmin tool.

**Extract**

The data was extracted from the multiple websites as listed below:

|  |  |
| --- | --- |
| **Year** | **Source** |
| 2009 | <https://eci.gov.in/> |
| 2014 | https://en.wikipedia.org/wiki/2014\_Indian\_general\_election |
| 2019 | https://en.wikipedia.org/wiki/2019\_Indian\_general\_election |

Extraction was done using Pandas function read\_html, which located all tables in the given Wikipedia page and put them in a list. One endeavor was to find the right table by iterating through the list. Once found this table would then needed to be cleaned up for further consumption

**Transform**

The following transformations were done on the table data both on the pandas and the postgres side

* Extra columns were removed, multi-index columns were eliminated
* Column names were standardized
* Constituency names were standardized by using 2019 names as dictionary and updating the respective yearly constituency names
* Non-abbreviated party names from 2019 were abbreviated using 2014 abbreviations as dictionary

Some examples of transformation would include:

* *Standardizing column names*

df\_2014 = df\_2014.rename(columns = {'PC Name': 'Name', 'Winner Candidate': 'Candidate', 'Winner Party': 'Party'})

* *Standardizing constituency names*

UPDATE winner\_2009 SET constituency\_name = 'Ahmednagar' WHERE constituency\_name = 'Ahmadnagar';

* *Standardizing party abbreviations*

UPDATE winner\_2019 SET winning\_party = 'SAD' WHERE winning\_party = 'Shiromani Akali Dal';

**Load**

Cleaned up CSV files were exported from pandas dataframes and then using Import utility in pgAdmin were imported into postgres DB. Once loaded, checks were run to ensure that names were standardized in all three yearly tables. That ensured, tables were joined to come up with the final table which comprises the elections data from 2009 – 2019.

|  |
| --- |
| CREATE TABLE all\_years AS ( |
|  | SELECT a.constituency\_name, a.state\_name, |
|  | a.winning\_party as winning\_party\_2009, a.winning\_candidate\_name as winning\_candidate\_2009, |
|  | b.winning\_party as winning\_party\_2014, b.winning\_candidate\_name as winning\_candidate\_2014, |
|  | c.winning\_party as winning\_party\_2019, c.winning\_candidate\_name as winning\_candidate\_2019 |
|  | FROM winner\_2009 a, winner\_2014 b, winner\_2019 c |
|  | WHERE a.constituency\_name = b.constituency\_name |
|  | AND a.constituency\_name = c.constituency\_name); |
|  |  |

**Key Learnings**

* Using webscraping as a tool to extract rich content from the web, not available otherwise through APIs or other methods
* Exploring ETL, via toggling multiple technologies from webscraping to postgres DB with Python as an intermediary

**Next Steps**

* This exercise can be continued for years further back to understand the political landscape
* This data can then be subsequently combined with other datasets to understand how different political issues and promises have impacted elections