**Contents of the project package:**

The project package has below main files:

***PopulatingTables.py***: Python script containing functions to make API calls to endpoint to gather the required data through REST API, create the connection to the PostgreSQL server, and populate the necessary tables on PostgreSQL server with information retrieved from website.

***ElectionAnalysis.py***: Python script containing functions for performing required analysis and computations to address and answer for parts of task 2, which are mainly analysis related to results of election.

***Explaining\_Results.docx***: docx file including final results and system’s outputs, as well as full description of how the designed system works and how to run it.

**Details of designed system:**

The developed system is mainly based on 2 different Python scripts.

***PopulatingTables.py***: is the Python script that is used to address the mentioned questions in task 1. It is responsible to create requests to endpoint website through REST API to retrieve necessary data, create connection to PostgreSQL, create 2 needed tables with asked schema and also populate tables with information retrieved from website.

This script is made of a class, named PopulatingTables which has some class functions to perform required computations in separate steps. Function *create\_connection* is responsible for creating a connection object through making connection to PostgreSQL by provided information about the host and server and database. This function handles exceptions and return the connection object. This class also has function *closing\_connection* which closes the connection to the server for that connection. Function *create\_tables* is responsible for creating and initiating 2 required tables in the database with asked schema. It also handles different exceptional situations. Function *populate\_candidate\_table* makes a request to endpoint using its API for each Riding, checks the response of the website and in case of correct response, continues to retrieve required information from the website. It handles the exceptions that might raise and in case of no exception, populate the candidate table in server with gathered and processed information. It also handles the possible raised exception while executing the sql queries. Function *populate\_voteshare\_table* does the same thing for table vote\_share in the PostgreSQL server.

***ElectionAnalysis.py***: is the main script to address the question of task 2, and has one distinct class function for each part of task 2 to perform the required analysis and processes to create the needed output. Function *part\_a* is to answer first question and in order to do so, it connects to tables on PostgreSQL database, for each Riding in vote\_share table, calculate the winner party and compute the total number of Riding’s win for each party and then produces the required output as a printed string. Function *part\_b* computes the win ratio for each Riding for the winner in vote\_share table and finds the max ratio and the winner party, and then finds the name of candidate for the winner party in candidate table. After that, it creates the output through a printed string. Function *part\_c* runs through each Riding of table vote\_share to perform processes to perform the winner party and then find the name of candidate of that winner party in that Riding in candidate table. Then, in case the name of winner is John it adds that Riding to the result list and finally produces the output string of that Ridings. Function *part\_d* gets the data from vote\_share table to find the winner for each Riding and does it for all Ridings to compute the total wins of each party and then, by mans of these information create the required output through a printed string.

**Results of the system:**

***SQL statements to create tables:***

*table\_1\_query = '''CREATE TABLE vote\_share*

*(RidingNumber INT PRIMARY KEY NOT NULL,*

*RidingNameInEnglish VARCHAR(50) NOT NULL,*

*RidingNameInFrench VARCHAR(50) NOT NULL,*

*TotalVotes INT NOT NULL,*

*Turnout REAL NOT NULL,*

*ConservativeVoteShare INT NOT NULL,*

*LiberalVoteShare INT NOT NULL,*

*NDPVoteShare INT NOT NULL,*

*GreenVoteShare INT NOT NULL,*

*BlocQuebecoisVoteShare INT NOT NULL,*

*PeoplesPartyVoteShare INT NOT NULL);’''*

*table\_2\_query = '''CREATE TABLE candidates*

*(RidingNumber INT PRIMARY KEY NOT NULL,*

*LiberalCandidate VARCHAR(100),*

*ConservativeCandidate VARCHAR(100),*

*NDPCandidate VARCHAR(100),*

*GreenCandidate VARCHAR(100),*

*BlocQuebecoisCandidate VARCHAR(100),*

*PeoplesPartyCandidate VARCHAR(100));’''*

***Part a):***

*Parties have earned following number of Ridings:*

*ConservativeVoteShare ---> 121*

*LiberalVoteShare ---> 158*

*NDPVoteShare ---> 24*

*GreenVoteShare ---> 3*

*BlocQuebecoisVoteShare ---> 32*

*PeoplesPartyVoteShare ---> 0*

*---------*

*---------*

*And the winner of election is ---> Liberal*

***Part b):***

*Candidate --Damien Kurek-- earned the biggest share(ratio) of votes in their riding with vote ration of --0.8549443033027164-- and this happened in riding number --261-- and this candidate is in --Conservative-- party.*

***Part c):***

*There are 5 Ridings with a winner candidate named John and those are as such:*

*Riding number--> 30*

*Riding number--> 114*

*Riding number--> 193*

*Riding number--> 205*

*Riding number--> 281*

***Part d):***

*The party with lowest number of won ridings is Peoples Party and the number of ridings it's won is 0*

*-----------*

*The parties that have won at least one riding are as such:*

*Conservative*

*Liberal*

*NDP*

*Green*

*Bloc Quebecois*

**How to run the package:**

In order to have the program running properly, first go to PopulatingTables.py and run the main. This python scripts populates the vote\_share and candidate table with retrieved information from website. Note that since the designed is to drop the previously built tables before creating them, in case this script is not run, or stopped running in the middle of been run, Tables won’t be fully populated. Hence, before going to next script, make sure that you run this file get the tables fully populated. Upon completion of running, related messages indicating tables are fully populated are printed.

After that, go to ElectionAnalysis.py file and run the main there. It initiates to run each function which is related to create results for each part of task 2. It starts with part a, produces string output and goes to part d.

**Notes:**

* To run the main program, please make sure all the necessary packages. In case of any error regarding not finding a specific library, please pip instal that library.
* Please make sure that first PopulatingTables.py is run to create and populate tables. It might take a couple of minutes to completely create and populate the tables. Since there might be some internal errors in endpoint website server and information might not be retrieved swiftly, and the program continues running to make sure it retrieves all the information necessary for all the Ridings, it might take a while ( between 2-5 minutes based on endpoint website internal server response) to fully populate tables.
* Note that if PopulatingTables.py is not finished running, vote\_share and candidate tables won’t be fully populated and ElectionAnalysis.py won’t create accurate results.