Justin Song and Irfan Jamil
Section 315 (both)
jsong69@jhu.edu and ijamil1@jhu.edu
Project link:
https://www.ugrad.cs.jhu.edu/~jsong69/dbfinalproj.html

Our application domain is political and sociological data. We explore unemployment trends, crime trends, income inequality trends while also looking at approval ratings for recent presidents. We also make a brief foray into exploring data on election results in House, Senate, and Presidential races.

3. Our project is quite a bit different than the plan and design we had in mind at the time of our Phase I submission. In our Phase I submission, we planned on focusing more on exploring polling data. However, as we progressed on the project, we shifted more to wanting to explore data that better encapsulates and describes America. New datasets we integrated into our database include S&P 500 data from Yahoo Finance that dates back close to 70 years, annual unemployment data ranging back to the 1960s and data on the share of income going to certain groups from the Bureau of Labor Statistics, and crime data from the FBI.

Furthermore, the special advanced topic we intended to explore shifted from data mining to more specifically data visualization. Although the relational model is extremely powerful and can very easily capture important information, we realized that in the relational model it is hard to conceptualize what the data is actually conveying. So, we spent time researching and learning about various methods to visualize data from a mysql database dynamically.

4. We used https://codebeautify.org/csv-to-sql-converter to convert csv files into a sql script. We could just copy and paste the content of the csv files into the website and it would give us a sql script that was just a series of "Insert into" commands for each row of the csv file. This website made populating our database with the data much easier. However, we did have to spend quite a bit of time writing sql scripts to restructure the data and the tables to contain the attributes and information we wanted in a way that would be easier for us later on in the project.

From https://www.presidency.ucsb.edu/statistics/data/presidential-job-approval, we were able to extract approval rating data for many of the past U.S. presidents. We then used the codebeautify site to convert the csv files into sql scripts that would automate loading the data into mysql. However, the data was structured so that the approval ratings for a president would be in subsequent groupings of 5 or 6 days at a time (ie: one week of approval ratings likely averaged followed by the next week and so forth). We did not like the data being in this way, so we wrote a script that, for each table containing approval rating data for a president, declared a cursor, looped over the each row the table, used the sql STR_TO_DATE and DATE_ADD function to add the row into a new table. The key difference was that instead of having the approval ratings data correspond to a group of 5 or 6 days, we "broke" the data up and assigned each day of that 5-6 day interval, its own row in the new table. That is the reason we used the STR_TO_DATE and DATE_ADD functions. Now, in this new table, we could query the approval ratings for a president by a specific date. We acknowledge that attributing the average value over a short interval to each day of that interval isn't exact but for the purposes of our project we believed we lost little information in doing so.

We had to also write a sql script to process our unemployment data that we got from https://www.bls.gov/cps/tables.htm#empstat. We wanted a table in our database that was simple and directly gave us the unemployment rate for a year. However, the historical unemployment data that we pulled from the BLS gave us many unnecessary attributes like total US population, the gross number of employed people, the gross number of unemployed people, and so forth. We wrote a script that declared a cursor for this table, read in a row of the table at a time, extracted the year, the size of the labor force, and the number of unemployed individuals as integers, and inserted into a new table the year along with the unemployment rate that we were able to calculate from the data that we just pulled from the cursor.

In the removeCols sql script, we drop attributes from the House election history, Senate election history, and POTUS election history table that were not needed or redundant. Some of these dropped attributes include things like version, state fips code, and other bookkeeping measures.

In the winners.sql script, we extract the actual winners from the election data that we had. In the original data, we had a row in the table for each candidate in a given race and the number of votes he or she received. For example, for some Senate race in New Jersey in 2000, we could easily have had 5 or 6 rows in the Senate data corresponding to the general Senate election race in NJ in 2000. For all of the House, Senate, and POTUS election data, we wrote a sql query that would group a specific race by its unique characteristics (ie: year, state, district for a House race) and extract the candidate who received the most votes in that race. Thus, we were able to extract the candidates that won their races and so we aptly named this script winners.sql.

Sources of our data:

https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/42MVDX (historical POTUS election data)

https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/IG0UN2 (historical House election data)

https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/PEJ5QU (historical Senate election data)

https://www.presidency.ucsb.edu/statistics/data/presidential-job-approval (historical POTUS approval/disapproval ratings)

https://www.bls.gov/cps/tables.htm#empstat (unemployment data and data on income inequality by percentile)

https://finance.yahoo.com/quote/%5EGSPC/history/ (S&P 500 data)

https://crime-data-explorer.app.cloud.gov (FBI Crime Explorer which we used to get the crime data)

5 We used mysql on the dbase JHU server.

6 It is not needed to “run” our code. A user just needs to navigate to the correct browser to interact with our application. https://www.ugrad.cs.jhu.edu/~jsong69/dbfinalproj.html. It is user-friendly; **however, for the parts of the page that allow a user to input “Date (From)” and “Date (To)” parameters corresponding to an actual day and not just a year, the dates need to be formatted in the following way: year-month-day**

7 Our area of specialization was: (2) an interesting interface demonstrating significant accomplishment, expanded education and/or originality. First, we built upon what we learned in homework 3 by creating a PHP/HTML interface that can let a user interact with the mysql database. Then, we explored a variety of techniques to be able to visualize our data. After doing this research, we settled upon using Google Charts which lets us use a Google API to plot our data by embedding the calls to the API in a Javascript script which itself is embedded in an HTML document. Through this project, we increased our Javascript knowledge and became familiarized with a new technology (Google Charts and the associated API) used to visualize data. We enjoyed learning about this data visualization technology as it is extremely applicable and was a natural tool to add to our repertoire after doing hw3. Our visualizations are easy-on-the eyes and lets users easily detect what the data is conveying; these visualizations also make sense given our domain of interest. It makes sense to want to see the trends for unemployment, for violent crime, or for the S&P 500. It makes sense to prefer to see the approval ratings between presidents compared on a graph rather than paper. Thus, our visualizations not only were an area of expanded education but also a practical part of our project that made it far more informative and powerful than it otherwise would have been.

Additionally, we specialized (minor) in extraction and processing of our data. Processing was far from trivial as we needed to refresh our knowledge of cursors, sql conditionals, sql functions, and needed to write some complex queries to get tables representing what we desired.

8

Selling Points:

- Interesting domain (crime trends, unemployment trends, income inequality trends, comparing approval ratings between presidents and so forth)
- Interesting, dynamic visualizations
- Error-checking on user input/queries

9 Our limitations include a basic User Interface in the sense that the possible queries we allow are all listed on the html page. We don't allow for a user to enter a query (ie: no NLI). Incorporating a NLI would be a possible next step if we had more time. In addition, our visualizations are not as complex/sophisticated as we'd like. If we had more time, we'd have liked to work on making them more interesting. One such example is being able to plot the unemployment trend line over the past 50-60 years but having a color-coded segment of the line correspond to the president at the time with a legend somewhere on the graph. Another example could be clustering more specific election data (county election data) to provide data visualization that shows groups of people who voted for a certain party.

10 All of the code is written by us except for many of the sql files.These files are just the initial "insert into" scripts on our raw, unprocessed data using the codebeautify site. We did write the following sql files: convertARRelations.sql, ConvertUEData.sql, removeCols.sql, and winners.sql.

https://codebeautify.org/csv-to-sql-converter

11

So, for the first half of our project, we displayed tables that represented our SQL output. These queries get more and more specific, tackling simple to more advanced topics. This method of visualization closely relates to the outputs we had for HW3, in that we show some specific SQL outputs for a variety of different questions pertaining to our database. For example, one of the most simple queries we displayed was finding out *all of the states* that had elected a Democrat to POTUS since 1976. With one click of a button, we are able to easily display this information:

STATE
District of Columbia
Minnesota

As we can see here, Washington D.C. and Minnesota are the only two states that have elected a Democrat to the President of United States *every year* since 1976. We are also able to view the opposing relationship for all the states that elected a *Republican* POTUS since 1976:

STATE
Alaska
Idaho
Kansas
Nebraska
North Dakota
Oklahoma
South Dakota
Utah
Wyoming

Now that you've seen the simplest query that our project can compute, let's move into some more advanced queries our SQL database can execute and display.

List how many Republican senators there were from a single State since a certain year

State: Kentucky Year: 2002
Submit

COUNT
6

So now we know that there have been 6 total Republican senators from Kentucky since 2002. However, if we look at a different state, such as Delaware:

List how many Republican senators there were from a single State since a certain year

State: Delaware Year: 2002
Submit

COUNT
0

We see that there have been 0 Republican senators since 2002.

Another interesting relationship that we are able to see from our SQL queries is the states that voted for Obama twice in a row (for his double presidential term) but then voted for Trump in 2016:

STATE
Florida
Iowa
Michigan
Ohio
Pennsylvania
Wisconsin

This shows a relationship essentially shows the states that elected a Democratic POTUS two presidential terms in a row, but then switched in 2016 to a Republican POTUS. It is intriguing to see these states listed, and brings up questions about the reasons they flipped to the opposing side.

Show Trump's Average Approval Rating between two dates during his Presidency (1/20/2017 - 11/18/2020)

Date (From): 01/20/2017 Date (To): 11/18/2020
Submit

Average Approval Ratings
47.964014

Show Obama's Average Approval Rating between two dates during his Presidency (1/19/2009 - 1/18/2017)

Date (From): 01/19/2009 Date (To): 01/18/2017
Submit

Average Approval Rating
41.006470

We are also able to show the average approval rating for candidates, such as Trump and Obama.

Our project is also able to explore areas such as comparing SP500 data with presidential approval ratings. For an example, we were able to calculate the dates that had the SP500 lows and highs corresponding to the dates that Donald Trump had his lowest/highest approval ratings:

Dates	SP500
2017-08-21	2430.58
2017-08-22	2454.77
2017-08-23	2448.91
2017-08-24	2450.39
2017-08-25	2453.96
2017-10-23	2578.29
2017-10-24	2572.18
2017-10-25	2567.40
2017-10-26	2567.07
2017-10-27	2582.98
2017-11-27	2606.41
2017-11-28	2627.69
2017-11-29	2634.89
2017-11-30	2657.74
2017-12-01	2650.62
2017-12-11	2660.33
2017-12-12	2669.72
2017-12-13	2671.88
2017-12-14	2668.09
2017-12-15	2679.63

These are SP500 highs for Donald Trump’s lowest approval ratings, and we are also able to view his SP500 lows for his *highest* approval ratings:

Dates	SP500
2020-01-16	3302.82
2020-01-17	3318.86
2020-01-21	3316.61
2020-01-22	3320.04
2020-01-23	3301.87
2020-01-24	3281.53
2020-01-27	3234.50
2020-01-28	3253.22
2020-02-03	3235.66
2020-02-04	3280.61
2020-02-05	3313.75
2020-02-06	3334.39
2020-02-07	3322.12
2020-02-10	3317.77
2020-02-11	3352.72
2020-02-12	3369.72
2020-02-13	3360.52
2020-02-14	3366.15
2020-03-13	2492.37
2020-03-16	2380.94
2020-03-17	2367.04
2020-03-18	2280.52
2020-03-19	2319.78
2020-03-20	2295.56
2020-04-14	2805.10
2020-04-15	2761.54
2020-04-16	2764.32
2020-04-17	2830.88
2020-04-20	2820.43
2020-04-21	2727.10
2020-04-22	2775.95
2020-04-23	2794.26
2020-04-24	2791.76
2020-04-27	2852.89
2020-05-01	2821.61
2020-05-04	2797.85
2020-05-05	2863.55
2020-05-06	2847.65
2020-05-07	2876.48
2020-05-08	2902.88
2020-05-11	2903.44
2020-05-12	2869.59

This extensive list shows the SP500 lows for the dates that DJT had his highest approval ratings.

Another area we are able to explore are income distributions, and we are able to explore this data by showing data that corresponds to the highest 10 years of income that were allocated to the top 5 percent:

Year
2017
2018
2019
2016
2001
2012
2011
2006
2013
2005

But, more importantly we are able to show data such as listing the tuples of size 2 in which one element is the share of income that goes to the 80th percentile and below and the other element is the share of income that goes to the top 20 percent:

Year	80%	20%
2000	50.30	49.80
2001	49.80	50.10
2002	50.40	49.70
2003	50.30	49.80
2004	50.00	50.10
2005	49.60	50.40
2006	49.40	50.50
2007	50.30	49.70
2008	50.00	50.00
2009	49.80	50.30
2010	49.80	50.30
2011	48.90	51.10
2012	48.90	51.00
2013	48.60	51.40
2014	48.80	51.20
2015	48.80	51.10
2016	48.50	51.50
2017	47.70	52.30
2018	48.10	52.00
2019	48.20	51.90

One last thing advanced topic this area can tackle is being able to show the years with the highest and lowest unemployment rates since 1949:

Year	Rate
1982	9.69

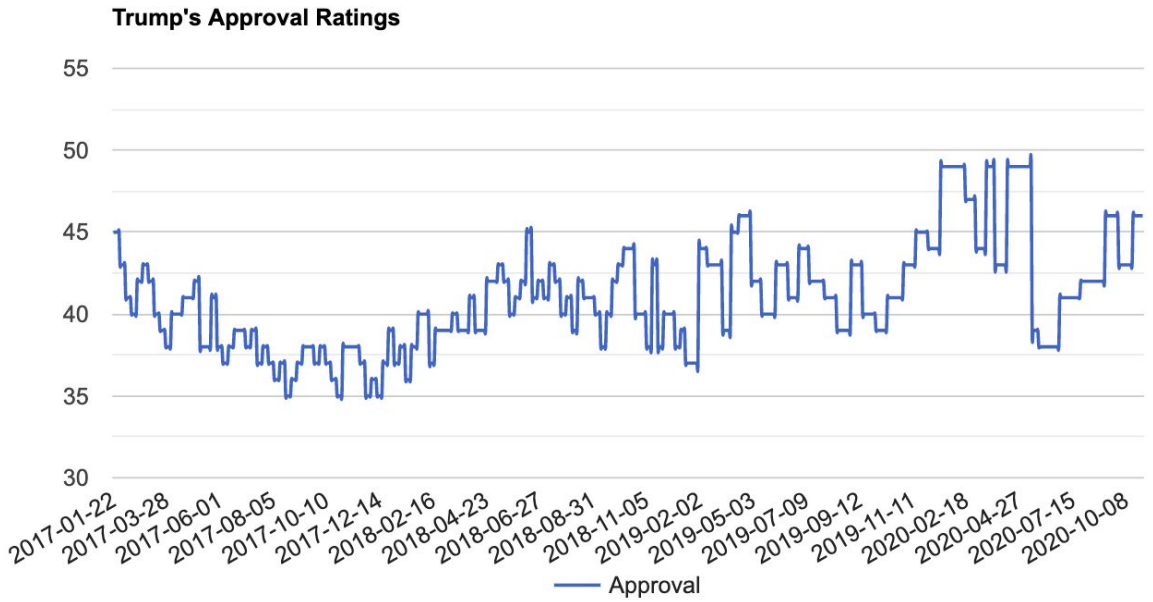
This is the highest unemployment rate the US has experienced since 1949, with an unemployment rate of 9.69%

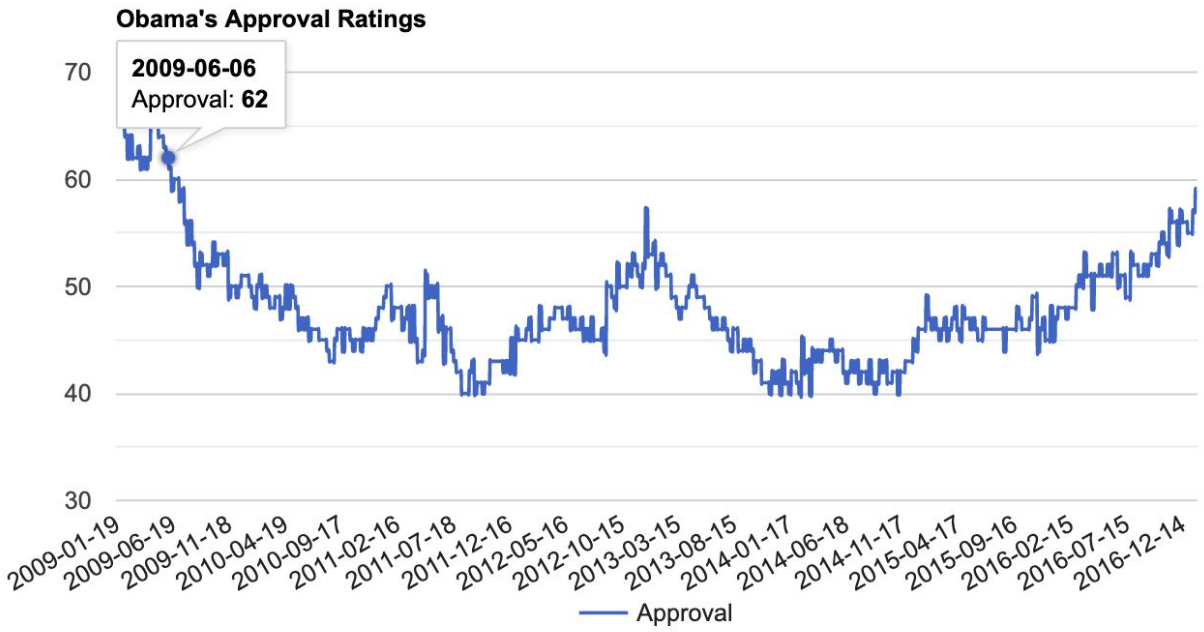
Year	Rate
1953	2.91

This is the lowest unemployment rate the US has experienced since 1949, with an unemployment rate of 2.91%.

Now, this is the perfect time to show some of the capabilities our application has when it comes to data *visualization.*

From our last few SQL queries, we were able to see the average approval ratings of candidates Obama and Trump, but in addition to the tables to show the actual averages, we also provide a graph showing the trends in each candidates average approval rating between a certain period of time:





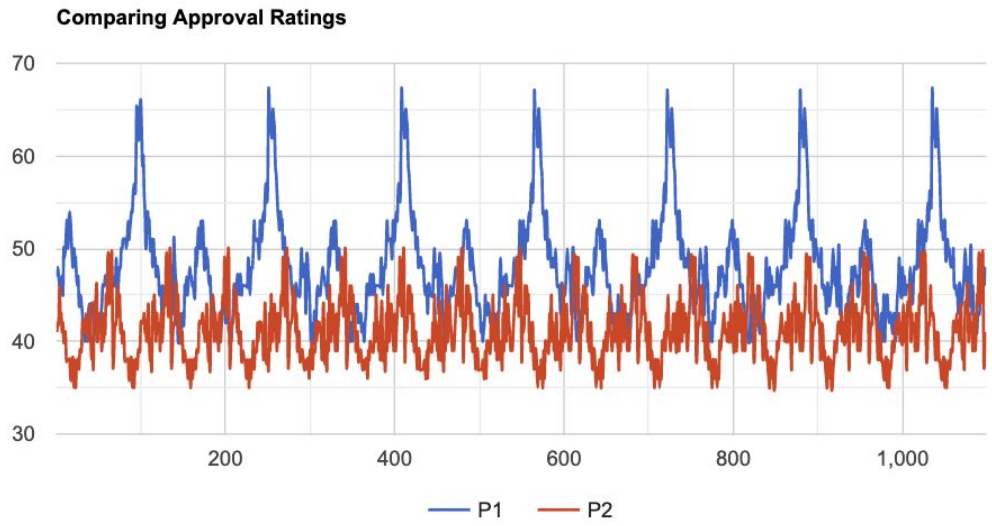
This shows the trends of approval ratings for Trump (1/22/2017 - 11/03/2020) and Obama (1/19/2009 - 1/18/2017)

However, our data visualization can do a lot more, such as being able to show how the share of total annual income going to the top 5 percent has changed over time (since 1967):

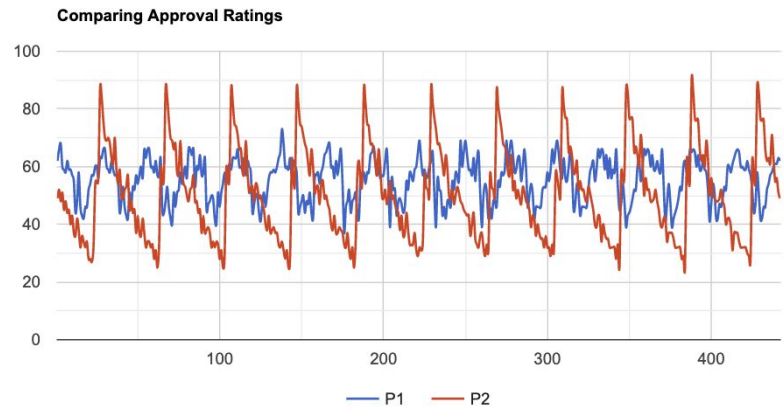
This data shows a mostly positive trend since the year 1967, which is obviously not a great sign, as it shows that the top 5% income share has only been increasing which means that the top 5% are slowly getting richer and richer, which leaves the poorer to get even more poorer.

Another thing our data can be visualized into is being able to compare approval ratings *between* candidates, our examples include candidates such as Clinton, Bush, Trump, and Obama. Here are some visual examples:

Obama V. Trump:

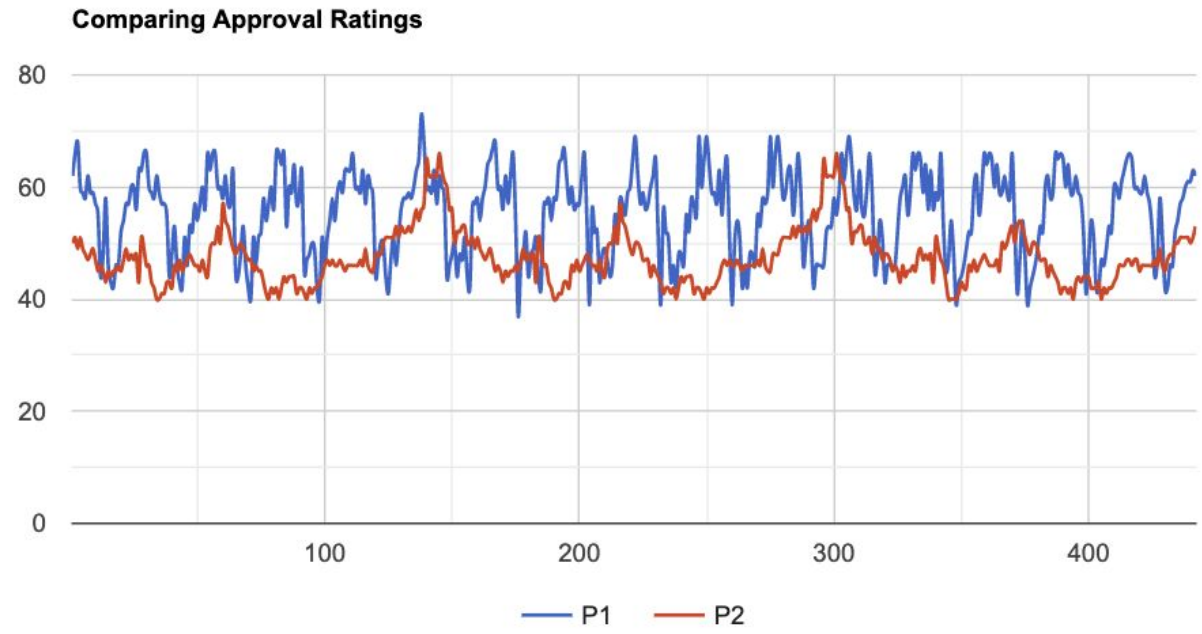


Where Obama is in blue and Trump is in red. Now, let's look at Clinton (blue) v. Bush (red):



From our data visualization, it is clear to see now that there was a drastic difference between Trump and Obama's approval ratings and Clinton and Bush's approval ratings. Obama and Trump's seems more volatile, while Clinton and Bush's seem to be more consistent.

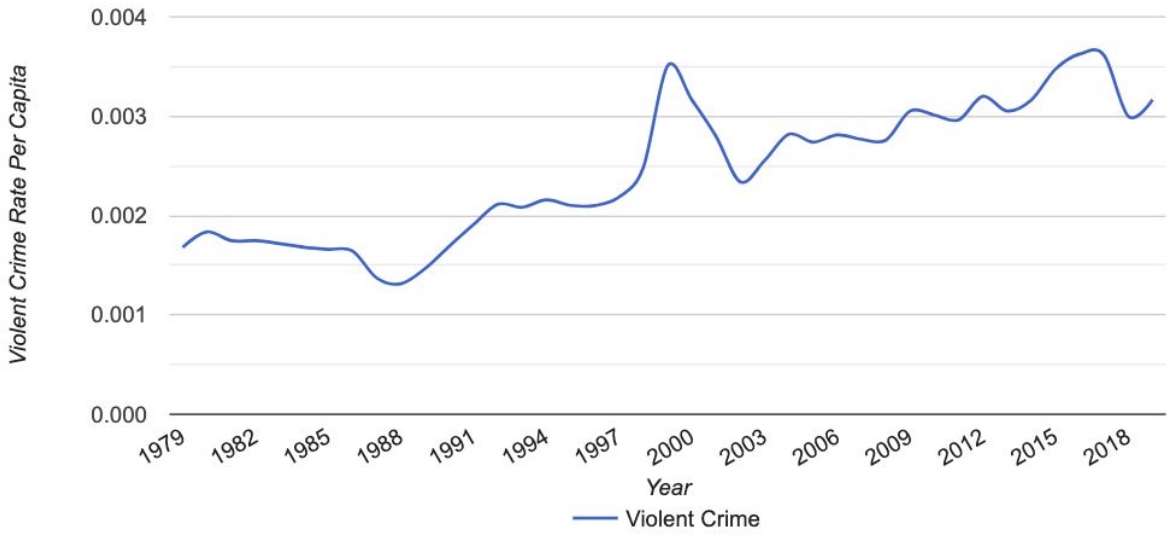
One more comparison… Obama V. Clinton:



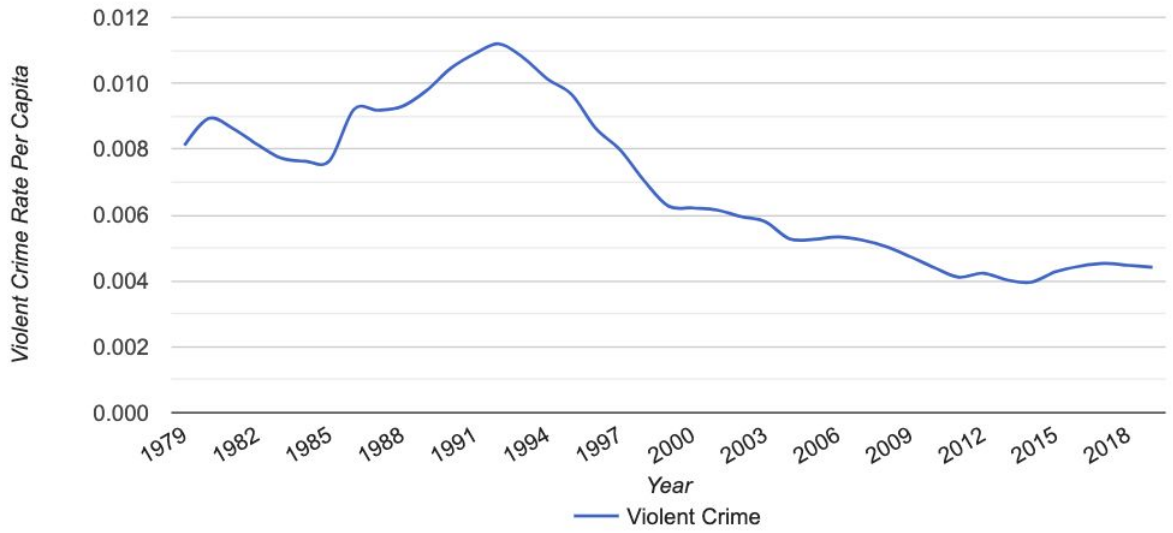
P1: Obama, P2: Clinton

Finally, one more advanced topic our data can visualize is visualizing crime data since 1979:

For an example, in West Virginia:



But in California:



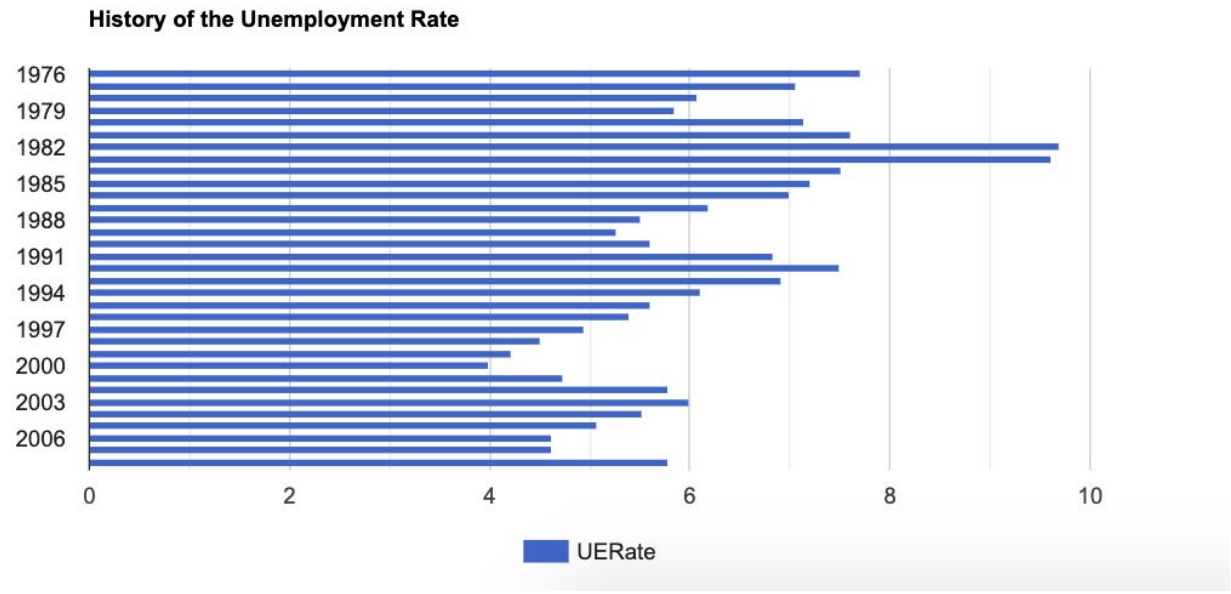
From these two graphs, we can make one clear speculation. Comparing West Virginia's violent crime rate and California's, it is definitely very clear that West Virginia's crime rates are on a positive trend, while California's is on the decline.

See Unemployment by Year(s)

Date (From): 01/20/1976 Date (To): 02/15/2008

Submit

Year	Rate
1976	7.70
1977	7.06
1978	6.07
1979	5.85
1980	7.14
1981	7.61
1982	9.69
1983	9.61
1984	7.52
1985	7.20
1986	6.99
1987	6.19
1988	5.51
1989	5.27
1990	5.60
1991	6.83
1992	7.50
1993	6.92
1994	6.10
1995	5.60
1996	5.40
1997	4.94
1998	4.51
1999	4.22
2000	3.99
2001	4.73
2002	5.78
2003	5.99
2004	5.53
2005	5.08
2006	4.62
2007	4.62
2008	5.78



This is a great example of our data being able to show the unemployment rate between two different dates (which can be inputted into the HTML seen above). You can also see our data being translated from our SQL outputs to a graph visualization.

12

```
CREATE TABLE IF NOT EXISTS PresHistory(
  year INT(11),
  state VARCHAR(100),
  state_po VARCHAR(100),
  state_fips DECIMAL(10, 2),
  state_cen DECIMAL(10, 2),
  state_ic DECIMAL(10, 2),
  office VARCHAR(100),
  candidate VARCHAR(100),
  party VARCHAR(100),
  writein VARCHAR(100),
  candidatevotes DECIMAL(10, 2),
  totalvotes DECIMAL(10, 2),
  version DECIMAL(10, 2),
  notes VARCHAR(100)
);


CREATE VIEW PresWinner AS
        SELECT year as Year, state as State, candidate as Winner, party as Party, candidatevotes as Votes
        FROM PresHistory as p
        WHERE p.candidatevotes IN
                (SELECT max(candidatevotes) FROM PresHistory GROUP BY year, state HAVING year=p.year
                AND state = p.state);
```

PRIMARY KEY in PresWinner=> (Year, State)

SAMPLE ENTRY IN PresWinner:
//Year, State, Winner, Party, Votes
(1976, Alabama, Carter, Jimmy, democrat, 659170.00)

```
CREATE TABLE IF NOT EXISTS SenateHistory(
  year INT(11),
  state VARCHAR(100),
  state_po VARCHAR(100),
  state_fips DECIMAL(10, 2),
  state_cen DECIMAL(10, 2),
  state_ic DECIMAL(10, 2),
  office VARCHAR(100),
  district VARCHAR(100),
  stage VARCHAR(100),
  special VARCHAR(100),
  candidate VARCHAR(100),
  party VARCHAR(100),
  writein VARCHAR(100),
  mode VARCHAR(100),
  candidatevotes DECIMAL(10, 2),
  totalvotes DECIMAL(10, 2),
  unofficial VARCHAR(100),
  version DECIMAL(10, 2)
);

CREATE VIEW SenateWinner AS
        SELECT year as Year, state as State, candidate as Winner, candidatevotes as Votes
        FROM SenateHistory as s
        WHERE s.candidatevotes IN
                (SELECT max(candidatevotes) FROM SenateHistory GROUP BY year, state HAVING year=s.year
                AND state=s.state);
```

PRIMARY KEY in SenateWinner=> (Year, State)

SAMPLE ENTRY IN SenateWinner:
//Year, State, Winner, Votes
(1976, Arizona, Dennis DeConcini, 400334.00)

```
CREATE TABLE IF NOT EXISTS HouseHistory(
year INT(11),
state VARCHAR(100),
state_po VARCHAR(100),
state_fips DECIMAL(10,2),
state_cen DECIMAL(10,2),
state_ic DECIMAL(10,2),
office VARCHAR(100),
```

```
district DECIMAL(10,2),
stage VARCHAR(100),
runoff VARCHAR(100),
special VARCHAR(100),
candidate VARCHAR(100),
party VARCHAR(100),
writein VARCHAR(100),
mode VARCHAR(100),
candidatevotes DECIMAL(10,2),
totalvotes DECIMAL(10,2),
unofficial VARCHAR(100),
version DECIMAL(10,2));


CREATE VIEW HouseWinners AS
        SELECT year as Year, state as State, district as District, candidate as Winner, candidatevotes
         FROM HouseHistory as h
        WHERE h.candidatevotes IN
                (SELECT max(candidatevotes) FROM HouseHistory GROUP BY year, state, district HAVING
                year=h.year AND state = h.state AND district = h.district);
```

PRIMARY KEY in House Winners => (Year, State, District)

SAMPLE ENTRY IN HouseWinners:

//Year, State, District, Winner, Votes

(1976, Alabama, 1.00 Jack Edwards, 98257.00)

```
CREATE TABLE IF NOT EXISTS 38ApprovalRatings (
  Day DATE NOT NULL,
  Approve DECIMAL(10, 2),
  Disapprove DECIMAL(10,2),
  UnsureOrNoData DECIMAL(10,2),
  PRIMARY KEY (Day, Approve, Disapprove, UnsureOrNoData)
);

CREATE TABLE IF NOT EXISTS 39ApprovalRatings (
  Day DATE NOT NULL,
  Approve DECIMAL(10, 2),
  Disapprove DECIMAL(10,2),
  UnsureOrNoData DECIMAL(10,2),
  PRIMARY KEY (Day, Approve, Disapprove, UnsureOrNoData)
);

CREATE TABLE IF NOT EXISTS 40ApprovalRatings (
  Day DATE NOT NULL,
  Approve DECIMAL(10, 2),
  Disapprove DECIMAL(10,2),
  UnsureOrNoData DECIMAL(10,2),
```

```
  PRIMARY KEY (Day, Approve, Disapprove, UnsureOrNoData)
);

CREATE TABLE IF NOT EXISTS 41ApprovalRatings (
  Day DATE NOT NULL,
  Approve DECIMAL(10, 2),
  Disapprove DECIMAL(10,2),
  UnsureOrNoData DECIMAL(10,2),
  PRIMARY KEY (Day, Approve, Disapprove, UnsureOrNoData)
);

CREATE TABLE IF NOT EXISTS 42ApprovalRatings (
  Day DATE NOT NULL,
  Approve DECIMAL(10, 2),
  Disapprove DECIMAL(10,2),
  UnsureOrNoData DECIMAL(10,2),
  PRIMARY KEY (Day, Approve, Disapprove, UnsureOrNoData)
);

CREATE TABLE IF NOT EXISTS 43ApprovalRatings (
  Day DATE NOT NULL,
  Approve DECIMAL(10, 2),
  Disapprove DECIMAL(10,2),
  UnsureOrNoData DECIMAL(10,2),
  PRIMARY KEY (Day, Approve, Disapprove, UnsureOrNoData)
);

CREATE TABLE IF NOT EXISTS 44ApprovalRatings (
  Day DATE NOT NULL,
  Approve DECIMAL(10, 2),
  Disapprove DECIMAL(10,2),
  UnsureOrNoData DECIMAL(10,2),
  PRIMARY KEY (Day, Approve, Disapprove, UnsureOrNoData)
);

CREATE TABLE IF NOT EXISTS 45ApprovalRatings (
  Day DATE NOT NULL,
  Approve DECIMAL(10, 2),
  Disapprove DECIMAL(10,2),
  UnsureOrNoData DECIMAL(10,2),
  PRIMARY KEY (Day, Approve, Disapprove, UnsureOrNoData)
);
```

SAMPLE INSERT INTO Statement:

INSERT INTO 44ApprovalRatings VALUES (2014-12-15, 50.00, 46.00, 4.00);

NOTE***

For the above relations, our primary key is the entire tuple for the following reason: our original data had approval ratings given for a batch of days at a time; however, some of these batch of days overlapped, so when we converted the original relation into the above relation to give an approval rating on a day by day basis, we had a few days with multiple entries. Thus, we made the entire tuple the primary key. This wasn't guaranteed to work, but, for our data it did. We acknowledge it's an ad-hoc type of fix.

```
CREATE TABLE IF NOT EXISTS Crime(
  year INT(11) NOT NULL,
  state_abbr VARCHAR(100) NOT NULL,
  state_name VARCHAR(100)  NOT NULL,
  population INT NOT NULL,
  violent_crime DECIMAL(10, 2) NOT NULL,
  homicide DECIMAL(10, 2) NOT NULL,
  rape_legacy DECIMAL(10, 2) NOT NULL,
  robbery DECIMAL(10, 2) NOT NULL,
  aggravated_assault DECIMAL(10, 2) NOT NULL,
  property_crime DECIMAL(10, 2) NOT NULL,
  burglary DECIMAL(10, 2) NOT NULL,
  larceny DECIMAL(10, 2) NOT NULL,
  motor_vehicle_theft DECIMAL(10, 2) NOT NULL,
  PRIMARY KEY (year, state_abbr)
);
```

SAMPLE INSERT INTO Statement:

```
INSERT INTO Crime(
  year, state_abbr, state_name, population,
  violent_crime, homicide, rape_legacy,
  robbery, aggravated_assault, property_crime,
  burglary, larceny, motor_vehicle_theft
)
VALUES
  (
    1979, '', '', 220099000, 1208030, 21460,
    76390, 480700, 629480, 11041500, 3327700,
    6601000, 1112800
  );
```

```
CREATE TABLE IF NOT EXISTS IncomeDist(
  Year INT(11),
  Number VARCHAR(100),
  Lowest_fifth DECIMAL(10, 2),
  Second_Fifth DECIMAL(10, 2),
  Third_Fifth DECIMAL(10, 2),
  Fourth_Fifth DECIMAL(10, 2),
  Highest_Fifth DECIMAL(10, 2),
  Top_5Percent DECIMAL(10, 2),
  PRIMARY KEY (Year)
);
```

SAMPLE INSERT INTO Statement:

```
INSERT INTO IncomeDist(
  Year,
  Number,
  Lowest_fifth,
  Second_Fifth,
```

```
  Third_Fifth,
  Fourth_Fifth,
  Highest_Fifth,
  Top_5Percent
)
VALUES
  (
    2019, '128,451', 3.1, 8.3, 14.1, 22.7,
    51.9, 23.0
  );
```

```
CREATE TABLE IF NOT EXISTS sp500(
Date VARCHAR(100),
Open DECIMAL(10,2),
High DECIMAL(10,2),
Low DECIMAL(10,2),
Close DECIMAL(10,2),
PRIMARY KEY (Date)
);
```

SAMPLE INSERT INTO Statement:

```
INSERT INTO sp500(Date,Open,High,Low,Close) VALUES( '1940-01-03',12.770000,12.770000,12.770000,12.770000);
```

```
CREATE TABLE IF NOT EXISTS unemployment(
  Year INT NOT NULL,
  UERate DECIMAL(10,2) NOT NULL,
  PRIMARY KEY (Year)
);
```

SAMPLE INSERT INTO Statement:

```
INSERT INTO unemployment VALUES (2009, 9.18);
```

13

SQL code/scripts contained in ZIP file on Gradescope. Comments in the files explain what it is each script does.

PHASE 1 DOCUMENT COPY/PASTED BELOW

Databases Project Phase 1

1 Our team consists of Justin Song (jsong69) and Irfan Jamil (ijamil1). We are both in 315.

2 Our domain of interest is political data. Namely, we will be working with historical US federal election data, presidential approval data, and polling data. With respect to the US federal election data, we are interested in a variety of questions such as touching up midterm results for the party in the White House, historical battleground state results, and states that had a “split party vote” (ie: Presidential contest went in favor of the Republican candidate, yet Senate races and House races were favorable for Dems). Additionally, we are interested in exploring approval ratings between presidents over the course of their presidency and after major events in their respective presidencies (ie: Hurricane Katrina for Bush 43 and Covid-19 for Trump). Finally, and perhaps the most interesting, we intend to explore how final polling forecasts compare to actual results in House, Senate, and Presidential races in the past few election cycles. Note, for polling forecasts, we plan to use FiveThirtyEight’s forecasts (if data available or their compilation of polls by various pollsters).

3

I) List all of the Senatorial candidates who were favored to win, but lost between the years of 2000 – 2020
II) List the states which have elected a Democrat in each of the presidential contests since 1976.
III) List the states which have elected a Republican in each of the presidential contests since 1976.
IV) List the approval ratings of Trump and Obama in their first month in Office.
V) List the approval ratings of Trump and Obama in their last month in Office.
VI) Compare the approval ratings of Bush 43 and Trump during the first 2 months after each of the following events in their respective presidencies: Lehman Brothers collapse, COVID-19 declared as a national emergency
VII) For each president since 1976, how did the opposing party perform in midterm elections?
VIII) List all of the pollsters whose final polling had the eventual winner (Trump) ahead in the PA 2016 race.
IX) List all of the pollsters whose final polling had the eventual winner (Trump) ahead in WI, PA, and MI.
X) What states voted for Obama in 2012, Trump in 2016, and Biden in 2020?

XI) Were there any states whose voter turnout dropped from 1992 to 1996?
XII) Has California sent any Republicans to the Senate in the 21st century? List the names of the Senators?
XIII) What states, in 2016, elected more representatives of one of the major political parties to the House but elected the presidential candidate of the other party?
XIV) In which states is the average polling lead for the eventual winner of the state in 2016 less than the actual margin of victory?
XV) Of the Senatorial candidates who were favored to win but lost in 2016, what percentage were democrats? In 2020?

4

```
CREATE TABLE PresidentHistoricalResults (
Year      INT                NOT NULL
State     VARCHAR(30)    NOT NULL
Winner  VARCHAR(40)    NOT NULL
Party     VARCHAR(30)    NOT NULL
Candidatevotes   INT       NOT NULL
Totalvotes         INT       NOT NULL
PRIMARY KEY (Year, State, Winner));
```

```
CREATE TABLE SenateHistoricalResults (
Year      INT                NOT NULL
State     VARCHAR(30)    NOT NULL
Winner  VARCHAR(40)    NOT NULL
Party     VARCHAR(30)    NOT NULL
Candidatevotes   INT       NOT NULL
Totalvotes         INT       NOT NULL
PRIMARY KEY (Year, State, Winner));
```

```
CREATE TABLE HouseHistoricalResults (
Year      INT                NOT NULL
State     VARCHAR(30)    NOT NULL
Winner  VARCHAR(40)    NOT NULL
Party     VARCHAR(30)    NOT NULL
Candidatevotes   INT       NOT NULL
Totalvotes         INT       NOT NULL
PRIMARY KEY (Year, State, Winner));
```

```
CREATE TABLE PresidentialApproval (
Name              VARCHAR(30)    NOT NULL
Inauguration      DATE              NOT NULL
StartDate         DATE              NOT NULL
EndDate           DATE              NOT NULL
Approval          DECIMAL          NOT NULL
Disapproval       DECIMAL          NOT NULL
PRIMARY KEY (Name, StartDate, EndDate));
```

```
CREATE TABLE 2016PresidentialPolls (
Date      DATE              NOT NULL
State     VARCHAR(30)    NOT NULL
```

```
Pollster  VARCHAR(50)    NOT NULL
Clinton   DECIMAL          NOT NULL
Trump   DECIMAL          NOT NULL
PRIMARY KEY (Date, State, Pollster));

CREATE TABLE 2020PresidentialPolls (
Date      DATE              NOT NULL
State     VARCHAR(30)    NOT NULL
Pollster  VARCHAR(50)    NOT NULL
Biden    DECIMAL          NOT NULL
Trump   DECIMAL          NOT NULL
PRIMARY KEY (Date, State, Pollster));

CREATE TABLE 2020SenatePolls (
Date              DATE              NOT NULL
State             VARCHAR(30)    NOT NULL
Pollster          VARCHAR(40)    NOT NULL
Candidate         VARCHAR(40)    NOT NULL
Party             VARCHAR(20)    NOT NULL
Percentage        DECIMAL          NOT NULL
PRIMARY KEY (Date, State, Pollster, Candidate));


CREATE TABLE 2018SenatePolls (
Date              DATE              NOT NULL
State             VARCHAR(30)    NOT NULL
Pollster          VARCHAR(40)    NOT NULL
Candidate         VARCHAR(40)    NOT NULL
Party             VARCHAR(20)    NOT NULL
Percentage        DECIMAL          NOT NULL
PRIMARY KEY (Date, State, Pollster, Candidate));
```

Omitting other polling data relations for brevity since the polling data in other cycles will have identical attributes/schema.

5) Some SQL Queries (the number the query refers to is listed before the actual SQL query itself)

- (ii) (SELECT DISTINCT ph.State FROM PresidentHistoricalResults as ph) EXCEPT (SELECT DISTINCT ph.State FROM PresidentHistoricalResults as ph WHERE ph.Party = ‘democratic’;
- (iii) (SELECT DISTINCT ph.State FROM PresidentHistoricalResults as ph) EXCEPT (SELECT DISTINCT ph.State FROM PresidentHistoricalResults as ph WHERE ph.Party = ‘republican’;
- (iv) SELECT 100End.PresidentName, AVG(pa2.Approval) as AverageApproval FROM (SELECT pa.Name as PresidentName, DATEADD(month, 1, pa.Inauguration) as 1Month FROM PresidentialApproval as pa WHERE pa.Name = ‘Barack Obama’ OR pa.Name = ‘Donald Trump’) as 1Temp, PresidentialApproval as pa2 WHERE pa2.Name = 1Temp.PresidentName AND pa2.StartDate >= pa2.Inauguration AND pa2.EndDate <= 1Temp.1Month GROUP BY pa.Name;
- (v) SELECT finaltemp.PresidentName, AVG(pa3.Approval) as AverageApproval FROM (SELECT pa2.Name as PresidentName, DATEADD(month, -1, EndOfTerm.TempEndDate) as TempStart, EndOfTerm.TempEndDate FROM (pa.Name,SELECT MAX(pa.EndDate) as TempEndDate FROM PresidentialApproval as pa WHERE pa.Name = ‘Barack Obama’ OR pa.Name = ‘Donald Trump’) as EndOfTerm, PresidentialApproval as pa2 WHERE pa2.Name = EndOfTerm.Name) as finaltemp, PresidentialApproval as pa3 WHERE pa3.Name = finaltemp.PresidentName AND pa3.StartDate >= finaltemp.TempStart AND pa3.EndDate <= finaltemp.TempEndDate;
- (vi) (SELECT 100End.PresidentName, AVG(pa2.Approval) as AverageApproval FROM (SELECT pa.Name as PresidentName, DATEADD(month, 2, ‘9/15/2008’) as 1Month FROM PresidentialApproval as

pa WHERE pa.Name = 'George W. Bush') as 1Temp, PresidentialApproval as pa2 WHERE pa2.Name = 1Temp.PresidentName AND pa2.StartDate >= pa2.Inauguration AND pa2.EndDate <= 1Temp.1Month GROUP BY pa.Name) UNION ALL (SELECT 100End.PresidentName, AVG(pa2.Approval) as AverageApproval FROM (SELECT pa.Name as PresidentName, DATEADD(month, 2, '01/31/2020') as 1Month FROM PresidentialApproval as pa WHERE pa.Name = 'Donald Trump') as 1Temp, PresidentialApproval as pa2 WHERE pa2.Name = 1Temp.PresidentName AND pa2.StartDate >= pa2.Inauguration AND pa2.EndDate <= 1Temp.1Month GROUP BY pa.Name);

- (vii) SELECT phr.Winner FROM PresidentHistoricalResults as phr GROUP BY phr.Year
- (viii) SELECT p.Pollster, MAX(p.Date) FROM 2016PresidentialPolls WHERE p.Date = (SELECT MAX(p.Date) FROM 2016PresidentialPolls as temp GROUP BY temp.Pollsters) WHERE p.Trump > p.Clinton AND p.State = 'PA' GROUP BY p.Pollster;
- (ix) (SELECT p.Pollster, MAX(p.Date) FROM 2016PresidentialPolls WHERE p.Date = (SELECT MAX(p.Date) FROM 2016PresidentialPolls as temp GROUP BY temp.Pollsters) WHERE p.Trump > p.Clinton AND p.State = 'PA' GROUP BY p.Pollster) INTERSECT (SELECT p.Pollster, MAX(p.Date) FROM 2016PresidentialPolls WHERE p.Date = (SELECT MAX(p.Date) FROM 2016PresidentialPolls as temp GROUP BY temp.Pollsters) WHERE p.Trump > p.Clinton AND p.State = 'WI' GROUP BY p.Pollster) INTERSECT (SELECT p.Pollster, MAX(p.Date) FROM 2016PresidentialPolls WHERE p.Date = (SELECT MAX(p.Date) FROM 2016PresidentialPolls as temp GROUP BY temp.Pollsters) WHERE p.Trump > p.Clinton AND p.State = 'MD' GROUP BY p.Pollster);
- (x) (SELECT phr.State FROM PresidentHistoricalResults as phr WHERE phr.Year = 2012 AND phr.Winner = 'Obama') INTERSECT (SELECT phr.State FROM PresidentHistoricalResults as phr WHERE phr.Year = 2016 AND phr.Winner = 'Trump') INTERSECT (SELECT phr.State FROM PresidentHistoricalResults as phr WHERE phr.Year = 2020 AND phr.Winner = 'Biden');
- (xii) SELECT s.Winner FROM SenateHistoricalResults as s WHERE s.Year >= 2000 AND s.State = 'CA' AND s.Party = 'republican';
- (xiii) SELECT phr.State (SELECT hhr.State, hhr.Party FROM (SELECT temp.State, COUNT(temp.Party) as PartyCount FROM (SELECT h.State, h.Party FROM HouseHistoricalResults as h WHERE h.Party = 'republican' OR h.Party = 'democratic' AND h.Year = 2016 GROUP BY h.State ORDER BY PartyCount DESC LIMIT 1) as temp2, HouseHistoricalResults as hhr WHERE hhr.Party = 'republican' OR hhr.Party = 'democratic' AND hhr.Year = 2016 GROUP BY HAVING COUNT(hhr.Party) = temp2.PartyCount) as finaltemp, PresidentHistoricalResults as phr WHERE phr.State = finaltemp.State AND phr.Party != finaltemp.Party);

6 Our datasets are formatted in csv files. We plan on using a PHP/MySQL interface similar to what was implemented in HW 3. We have found that there are a variety of methods to import a csv file into a MySQL table, so we don't expect much trouble importing the data into a SQL environment. **********TODO********

https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/42MVDX (historical POTUS election data)

https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/IG0UN2 (historical House election data)

https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/PEJ5QU (historical Senate election data)

https://www.presidency.ucsb.edu/statistics/data/presidential-job-approval (historical POTUS approval/disapproval ratings)

https://projects.fivethirtyeight.com/polls/senate/ (2020 Senate polls FiveThirtyEight)

https://projects.fivethirtyeight.com/2020-election-forecast/ (2020 Presidential Polls FiveThirtyEight)

https://projects.fivethirtyeight.com/2016-election-forecast/national-polls/ (2016 Presidential Polls FiveThirtyEight)

https://elections.huffingtonpost.com/pollster/api/v2 (We will be using this API to extract polling data for past elections which FiveThirtyEight hasn't released data on).

https://drive.google.com/file/d/1PB9e1-gPTFcVuI-MLpxnEhrJTVK9Ebn0/view?usp=sharing (2000-2010 US State Populations)

https://drive.google.com/file/d/10B5V1CmpI6hENNm4jMOx09BgkPIwe-_K/view?usp=sharing (2010-2019 US State Populations)

7 I think that we would like to visualize the data by using some kind of visual of the USA and allowing users to click specifically on locations to see graphs and specific data corresponding to the location the user is looking for. In general, I think that some views can be created such as a view with some MAJOR events that happened in the US the past few decades alongside dates and the President's name. This would make it easier to find interesting information corresponding to special historical events in the US.

8 A special, advanced topic we plan to focus on is data mining/visualization. Political data lends itself well to charts, graphs, and the like. Some simple examples/scenarios of how and when we would actually use data mining include: plotting presidential approval ratings between various presidents in a time-series line-graph, plotting FiveThirtyEight's final polling forecasts against the actual results for a race in a bar graph for multiple races at a time (ie: could be a user input to determine which races/forecasts to compare), and plotting a pie chart of the political parties and the percentage of races that they have won since some baseline year for a given state. These are just a subset of the myriad of data mining and visualization routes that we can explore. Another topic we will likely explore is the advanced SQL topics of cursors, stored procedures, and views derived from the raw data.