

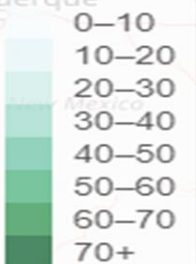
# Covid-19 Vaccination Status

## California Counties

Vaccination Status  
Hover over a county

Project – 3  
Data Visualization

Prepared by:  
TIKRAM SUBEDY



## Purpose of the Project:

- Create a dashboard with web based interactive charts, data, and maps using the database created in the previous project of ETL.

# Data Sources, Collection and Processing:

- Data Sources:

- California department of public health

- <https://data.chhs.ca.gov/dataset/vaccine-progress-dashboard>

- World Population Review

- <https://worldpopulationreview.com/us-counties/states/ca>

- Github:

- [https://github.com/codeforgermany/click\\_that\\_hood/blob/main/public/data/california-counties.geojson](https://github.com/codeforgermany/click_that_hood/blob/main/public/data/california-counties.geojson)



# Data Sources, Collection and Processing:

- Data Extraction:
  - Web scraping
  - Downloading datafiles

```
# California Population data by county (web scrapping)
# setting up splinter

executable_path = {'executable_path': ChromeDriverManager().install()}
browser = Browser('chrome', **executable_path, headless=False)

url = 'https://worldpopulationreview.com/us-counties/states/ca'
browser.visit(url)

table = pd.read_html(url)
county_population = table[0]

# saving the datafile as a csv file
county_population.to_csv('./data/county_population2021_data.csv', index = False)
county_population.head()
```

# Data Sources, Collection and Processing:

- Data Processing:
  - Python Pandas

```
# California COVID-19 cases and tests by county downloaded from
# https://data.chhs.ca.gov/dataset/covid-19-time-series-metrics-by-county-and-state

cases_data = pd.read_csv("../data/statewide_covid_19_cases_by_county.csv", encoding = 'utf8')

cases_data = cases_data.query('area != "All CA Counties"')
cases_data = cases_data.query('area != "All CA and Non-CA Counties"')
cases_data = cases_data.query('area != "Outside California"')
cases_data = cases_data.query('area != "Unknown"')
cases_data = cases_data.query('area != "California"')
cases_data = cases_data.query('area_type == "County"')
cases_data = cases_data.dropna()
cases_data
```

```
# Selecting only the residence county as reported in the data (elimination method is used)
ca_vaccin_data = csv_data.query('county != "All CA Counties"')
ca_vaccin_data = ca_vaccin_data.query('county != "All CA and Non-CA Counties"')
ca_vaccin_data = ca_vaccin_data.query('county != "Outside California"')
ca_vaccin_data = ca_vaccin_data.query('county != "Unknown"')
ca_vaccin_data = ca_vaccin_data.query('county != "California"')
# getting new dataframe for total vaccination (fully, partially vaccinated) by counties

county_vac_data = pd.DataFrame(ca_vaccin_data, columns = ['county', 'administered_date', \
                                                         'partially_vaccinated', 'fully_vaccinated'])

county_vac_data['month'] = pd.DatetimeIndex(county_vac_data['administered_date']).month_name()

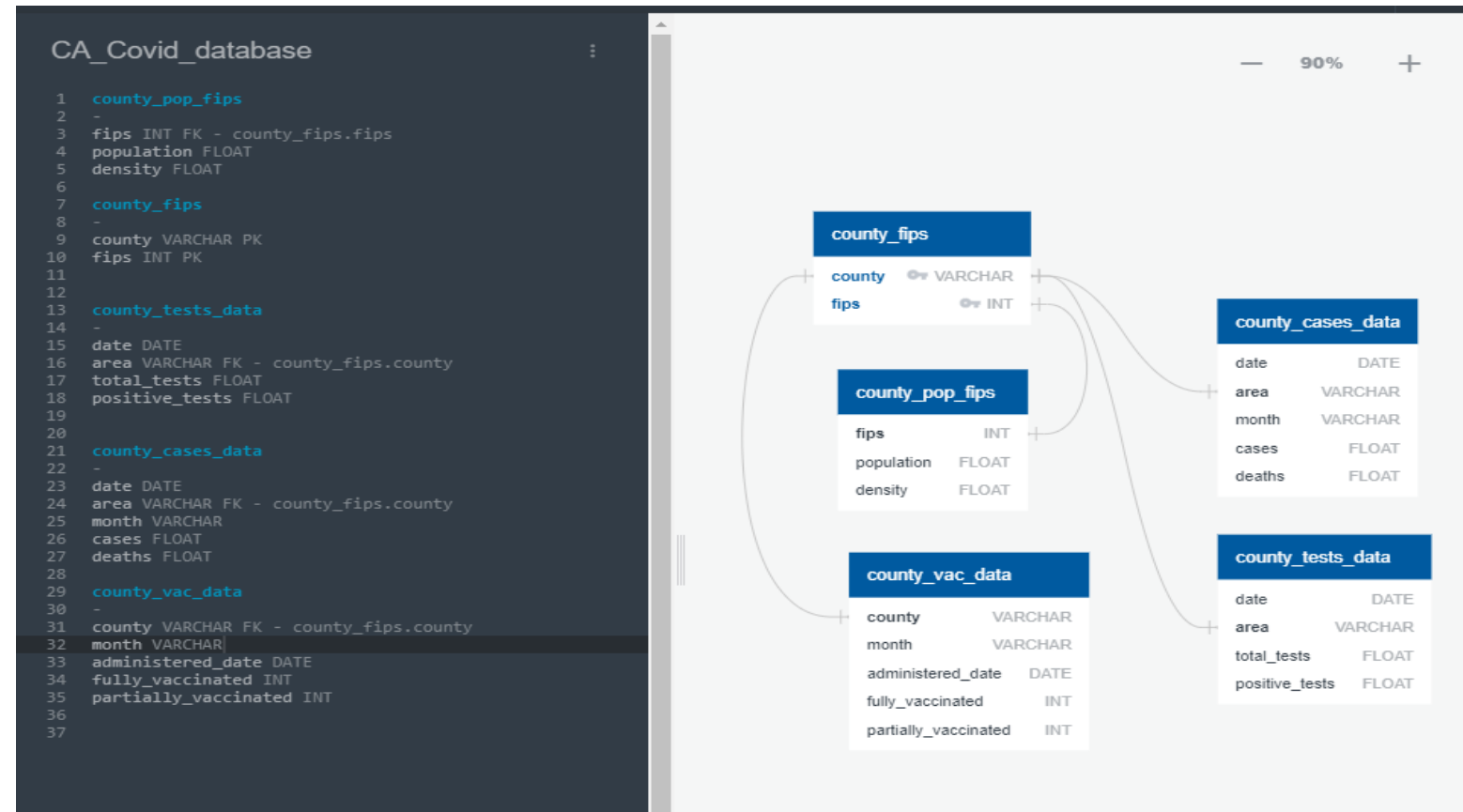
# Creating a new dataframe with months as variable to be utilized for plotting by months
county_vac_data_cleaned = pd.DataFrame(county_vac_data, columns=['county', 'month', 'administered_date', 'fully_vaccinated', 'partially_vaccinated'])

county_vac_data_cleaned.to_csv('../data/county_vac_data_cleaned.csv', index = False)
county_vac_data_cleaned
```

	total_tests	cumulative_total_tests	positive_tests	cumulative
0		30087	65.0	
0		31392	78.0	
0		32731	38.0	

# Data Sources, Collection and Processing:

- Creating Database (Relational):
  - ERD, SQL (Postgres, PgAdmin)





# Data Sources, Collection and Processing:

- Creating Database (Relational):
  - ERD, SQL (Postgres, PgAdmin)

The screenshot displays the PgAdmin 4 interface. The left sidebar shows a tree view of the database structure, including 'cases', 'population', 'tests', 'vac', and 'employees'. The main pane shows a SQL query editor with the following queries:

```
-- cases data query
create view cases as
SELECT area as county, sum(cases) as cases, sum(deaths) as deaths
from county_cases_data
group by area;

-- vaccine data query
create view vac as
SELECT county, sum(fully_vaccinated) as full, sum(partially_vaccinated) as partial
FROM county_vac_data
GROUP BY county;

-- combined data by county
create view county_wise_combined_data as
select p.county, p.population, t.tests, t.positive, c.cases, c.deaths, v.total_full, v.total_partial
from population p
join tests t
on p.county = t.county
join cases c
on p.county = c.county
join vac v
on p.county = v.county;

-- select counties with high population density
create view high_pop_density as
SELECT f.county, p.population
FROM county_fips f
join county_pop p
on f.fips = p.fips
where p.density >= 1000;

-- high density counties combined data
select a.*
from county_wise_combined_data a
join high_pop_density h
on a.county = h.county;
```

The 'Data Output' pane shows the results of the queries, displaying a table with columns: county, population, tests, positive, cases, deaths, total\_full, and total\_partial. The table contains 13 rows of data for various counties in California.

county	population	tests	positive	cases	deaths	total_full	total_partial
1 Alameda	1680480	3590879	130133	107950	1230	1151524	1128245
2 Contra Costa	1159540	2141866	102018	89960	882	804166	805267
3 Fresno	1013400	1605085	144932	121531	1857	477528	517126
4 Kern	913090	1396151	128279	111923	1421	366141	395875
5 Los Angeles	9969510	29303282	1702841	1324628	24128	6010757	6285936
6 Orange	3175130	4728003	346437	284020	5235	1947863	2015165
7 Riverside	2520060	3680001	394759	331043	4531	1189318	1282784
8 Sacramento	1578680	2585300	162204	141319	1910	850700	895469
9 San Bernardino	2206750	3803020	389404	329165	5265	997026	1067484
10 San Diego	3347270	6427084	403751	336712	3798	2117134	2175592
11 San Francisco	883255	2580665	59299	45668	562	646138	644212
12 San Joaquin	781462	1335718	103721	89882	1574	365491	402790
13 San Mateo	762357	2150850	61476	48308	487	550648	556058

# Data Sources, Collection and Processing:

- Loading Database:
  - SQLAlchemy

```
from sqlalchemy import create_engine
engine = create_engine(f'postgresql://{username}:{password}@localhost:5432/county_covid_database')
connection = engine.connect()
```

```
# Loading combined dataset (using sql view 'county_wise_combined_data')
```

```
combined_data = pd.read_sql("SELECT * from county_wise_combined_data", connection)
combined_data.to_csv('./data/combined_data.csv', index = False)
combined_data.head()
combined_data["percentFullVax"] = round(combined_data['total_full']/combined_data['population'] *100, 2)

combined_data["percentCases"] = round(combined_data['cases']/combined_data['population'] *100, 2)

combined_data["percentDeaths"] = round(combined_data['deaths']/combined_data['population'] *100, 2)

combined_data
```

county	population	tests	positive	cases	deaths	total_full	total_partial	percentFullVax	percentCases	percentDeaths
Alameda	1680480.0	3590879.0	130133.0	107950.0	1230.0	1151524	1128245	68.52	6.42	0.07
Alpine	1209.0	2151.0	46.0	98.0	0.0	705	790	58.31	8.11	0.00
Amador	40446.0	149585.0	5485.0	4745.0	58.0	17826	19639	44.07	11.73	0.14
Butte	196880.0	289588.0	18739.0	17542.0	228.0	96451	100901	48.99	8.91	0.12
Calaveras	46319.0	55933.0	3768.0	3270.0	65.0	20647	21932	44.58	7.06	0.14
Colusa	21805.0	23090.0	2199.0	2343.0	16.0	10478	11059	48.05	10.75	0.07
Contra Costa	1159540.0	2141866.0	102018.0	89960.0	882.0	804166	805267	69.35	7.76	0.08
Del Norte	27956.0	127754.0	3646.0	3210.0	29.0	10578	11220	37.84	11.48	0.10
El Dorado	197037.0	245215.0	14841.0	14052.0	124.0	100822	104156	51.17	7.13	0.06
Fresno	1013400.0	1605085.0	144932.0	121531.0	1857.0	477528	517126	47.12	11.99	0.18
Glenn	29245.0	31724.0	3152.0	3038.0	24.0	12613	12727	43.13	10.39	0.08
Humboldt	134186.0	103705.0	8638.0	7500.0	50.0	75301	76411	56.18	5.50	0.04



## Data Sources, Collection and Processing:

- Loading Data:
  - GeoJson

```

1 import json
2 with open("../data/countyGeoJson.js") as f:
3     geodata = json.load(f)
4 with open("../data/combined_data.js") as f:
5     data = json.load(f)

1 for x in data:
2     for y in geodata["features"]:
3         if y["properties"]["name"] == x["county"]:
4             y["properties"]["data"] = x
5 countyGeoData = geodata
6 countyGeoData

: {'type': 'FeatureCollection',
  'features': [{'type': 'Feature',
    'properties': {'name': 'Alameda',
      'cartodb_id': 1,
      'created_at': '2015-07-04T21:04:58Z',
      'updated_at': '2015-07-04T21:04:58Z',
      'data': {'county': 'Alameda',
        'population': 1680480.0,
        'tests': 3590879.0,
        'positive': 130133.0,
        'cases': 107950.0,
        'deaths': 1230.0,
        'total_full': 1151524,
        'total_partial': 1128245,
        'percentFullVax': 68.52,
        'percentCases': 6.42,
        'percentDeaths': 0.07}},
      'geometry': {'type': 'MultiPolygon',
        'coordinates': [[[[[-122.312934, 37.897333],
          [122.312934, 37.897333],
          [122.312934, 37.897333],
          [-122.312934, 37.897333],
          [-122.312934, 37.897333]]]]]]]}]}]}

1 with open('countyGeoData.js', 'w') as fp:
2     json.dump(countyGeoData, fp, sort_keys=False, indent=4)
3

```

# Data Sources, Collection and Processing:

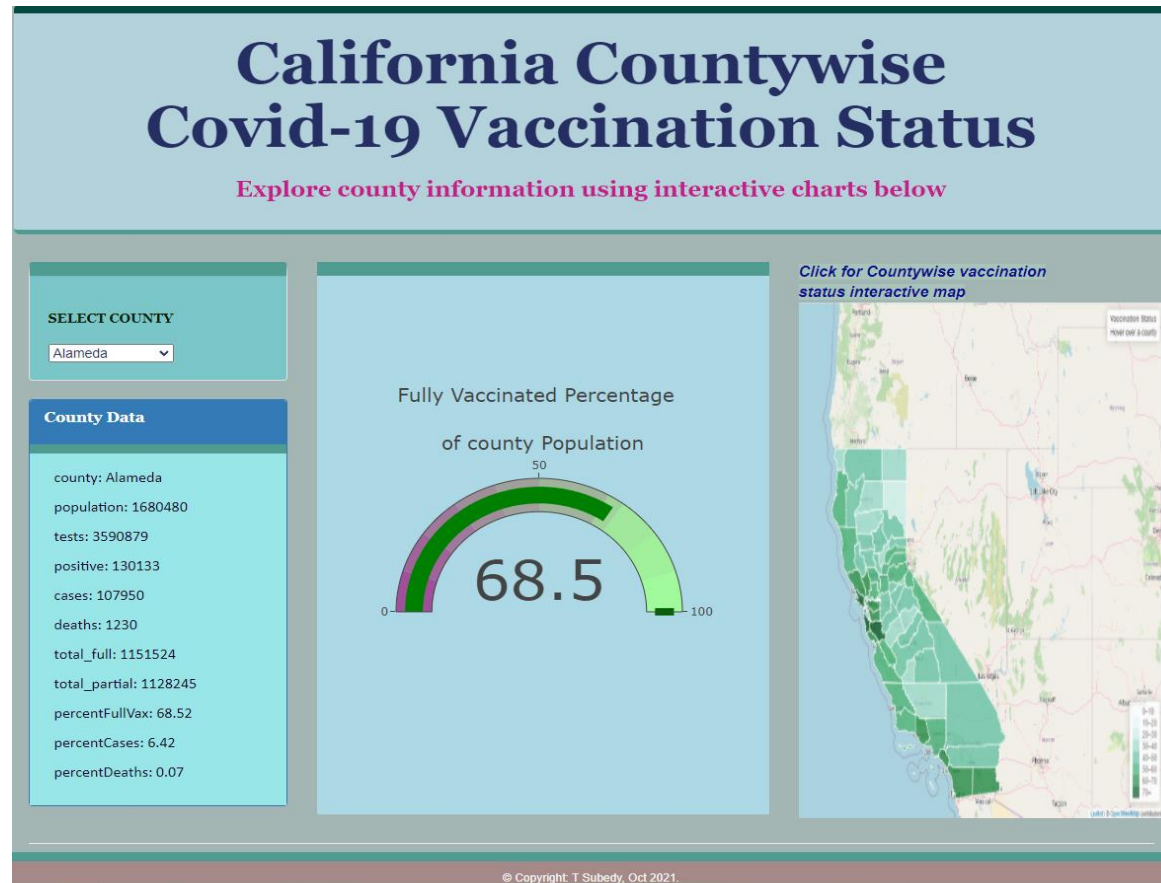
- Data Visualization:

- Javascript,
- Plotly,
- Leaflets
- HTML, CSS

# Dashboard and Data Visualization:

- Dashboard:
  - Javascript
  - Plotly
  - Html/ CSS
  - Mapping
  - Leaflets

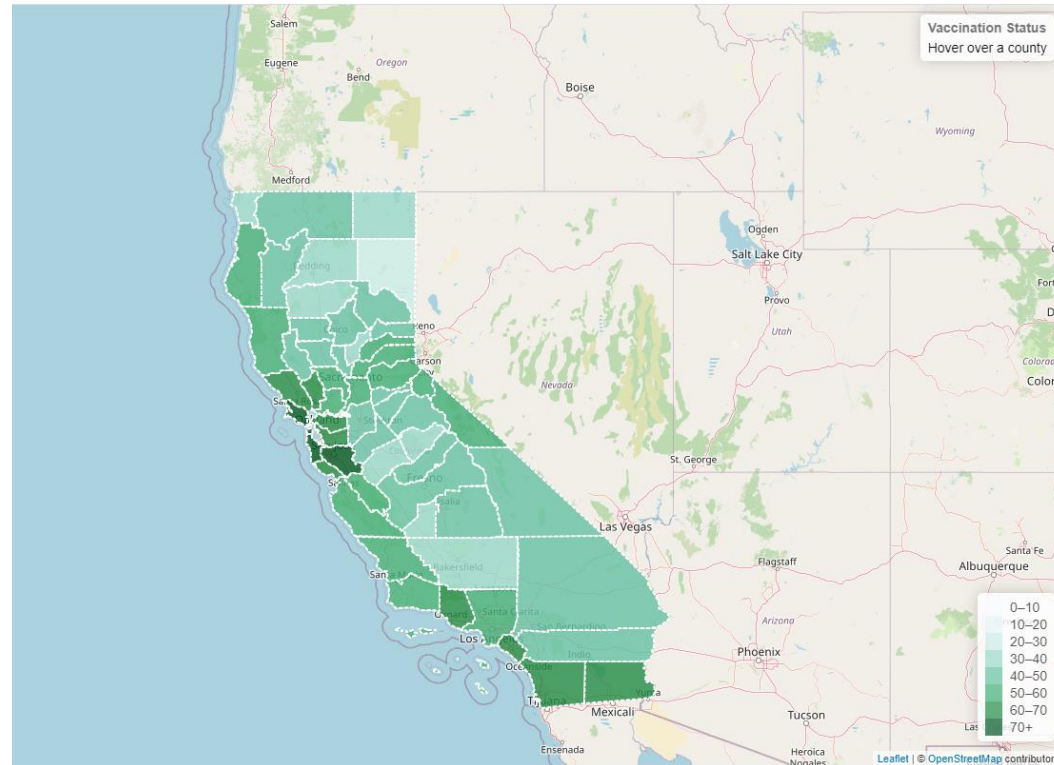
[https://tsubedy.github.io/Project\\_3/](https://tsubedy.github.io/Project_3/)





# Dashboard and Data Visualization:

- Dashboard:
  - Javascript
  - Plotly
  - Html/ CSS
  - Mapping
  - Leaflets



# Dashboard and Data Visualization:

- Dashboard:

- Javascript
- Plotly
- Html/ CSS
- Mapping
- Leaflets



## Limitations:

- The project is a part of the assignments from the class of Data Analytics Bootcamp and is limited to demonstrate the technical skills learned so far in the class.
- Some of data used for this project are not up to date as they were downloaded as csv files from the source sites.
- Data analyses are not included as it is beyond the scope of this project.



Questions:

???

THANK YOU !!!