

# City-Level Unemployment Map

*Interactive Map of Recent (August 2023) City-Level Unemployment Data for Top 100 US Cities*

*Ryan McReynolds*

*September 26, 2023*

*GitHub repo: <https://github.com/ryanmcr17/city-level-employment>*

# Executive Summary

# Project Plan/Proposal

The aim of this project is to clearly represent recent employment statistics for individual large US cities. I plan to provide a dashboard/page through which users can visualize city-level data from the federal Bureau of Labor Statistics (BLS) around key unemployment metrics. I will display city-specific data on an interactive map of the US with multiple visualization options for users.


# Key Files/Folders

- all data processing / ETL work contained within cityDataETL/ folder
  - cityDataETL.ipynb contains the bulk of the ETL work
  - .sqlite file contains the cleaned-up database produced by the Python ETL code
  - CityUnemploymentAPI.py is separate Python code which generates a Flask API allowing users to call the city data from the SQLite DB in JSON format
- index.html is the primary file to open which loads the interactive map
- the static/ folder contains the JS file that processes the city data (pulled from the Flask API) into circle markers and builds the Leaflet map

# Project

# Data Processing / ETL

CSV file of city lat/long coordinates and populations downloaded from <https://simplemaps.com/data/us-cities>



US Cities

Zips

Counties

Neighborhoods




World Cities

Pricing

All

## United States Cities Database

We're proud to offer a simple, accurate and up-to-date database of United States cities and towns. We've built it from the ground up using authoritative sources such as the U.S. Geological Survey and U.S. Census Bureau.

- ☒ **Up-to-date:** Data updated as of January 31, 2023.
- ☒ **Comprehensive:** Over 108,000 cities and towns from all 50 states, DC, Puerto Rico and the US Virgin Islands.
- ☒ **Useful fields:** From latitude and longitude to household income.
- ☒ **Accurate:** Aggregated and cleaned from official sources.
- ☒ **Simple:** A single CSV file, concise field names, only one entry per city/town.

Date:

Out[2]:

City	Latitude	Longitude	Population
Albany, NY	42.6525	-73.7568	97,886
Albany, OR	42.7284	-122.3718	18,338
Albany, TN	34.9011	-85.1887	19,375
Albany, VT	43.1734	-73.4706	14,657
Albany, WA	47.8763	-122.2822	18,338

	City	Latitude	Longitude	Population
0	New York	40.6943	-73.9249	18972871
1	Los Angeles	34.1141	-118.4068	12121244
2	Chicago	41.8375	-87.6866	8595181
3	Miami	25.7840	-80.2101	5711945
4	Dallas	32.7935	-96.7667	5668165

# Data Wrangling

*'Area codes' for US cities/locations scraped from BLS page and parsed as .tsv string*  
*BLS 'area codes' merged with city DF containing lat/long coordinates and population data*  
*'Area codes' used to build out BLS API URLs for unemployment data calls*

	area_type_code	area_code	area_text	display_level	selectable	sort_sequence
0	A	ST0100000000000	Alabama	0	T	1
1	A	ST0200000000000	Alaska	0	T	149
2	A	ST0400000000000	Arizona	0	T	193
3	A	ST0500000000000	Arkansas	0	T	257
4	A	ST0600000000000	California	0	T	383



	City Area Code	city_state_key
0	CT0100820000000	Alabaster, AL
1	CT0101852000000	Anniston, AL
2	CT0102956000000	Athens, AL
3	CT0103076000000	Auburn, AL
4	CT0105980000000	Bessemer, AL



	City	Latitude	Longitude	Population	Unemployment Rate Series ID	Unemployment Series ID	Labor Force Participation Rate Series ID	City Area Code
0	New York, NY	40.6943	-73.9249	18972871	LAUCT3651000000000003	LAUCT3651000000000004	LAUCT3651000000000008	CT3651000000000000
1	Los Angeles, CA	34.1141	-118.4068	12121244	LAUCT0644000000000003	LAUCT0644000000000004	LAUCT0644000000000008	CT0644000000000000
2	Chicago, IL	41.8375	-87.6866	8595181	LAUCT1714000000000003	LAUCT1714000000000004	LAUCT1714000000000008	CT1714000000000000
3	Miami, FL	25.7840	-80.2101	5711945	LAUCT1245000000000003	LAUCT1245000000000004	LAUCT1245000000000008	CT1245000000000000
4	Dallas, TX	32.7935	-96.7667	5668165	LAUCT4819000000000003	LAUCT4819000000000004	LAUCT4819000000000008	CT4819000000000000
5	Houston, TX	29.7860	-95.3885	5650910	LAUCT4835000000000003	LAUCT4835000000000004	LAUCT4835000000000008	CT4835000000000000
6	Atlanta, GA	33.7628	-84.4220	5046555	LAUCT1304000000000003	LAUCT1304000000000004	LAUCT1304000000000008	CT1304000000000000
7	Washington, DC	38.9047	-77.0163	4810669	LAUCT1150000000000003	LAUCT1150000000000004	LAUCT1150000000000008	CT1150000000000000
8	Boston, MA	42.3188	-71.0852	4208580	LAUCT2507000000000003	LAUCT2507000000000004	LAUCT2507000000000008	CT2507000000000000
9	Phoenix, AZ	33.5722	-112.0892	4047095	LAUCT0455000000000003	LAUCT0455000000000004	LAUCT0455000000000008	CT0455000000000000



# Final ETL Output

*Unemployment data pulled via BLS API and merged back with city location/population data*  
*Final cleaned-up dataframe saved to SQLite DB file*

	Unemployment Rate Series ID	August 2023 Unemployment Rate
0	LAUCT3651000000000003	5.6
1	LAUCT0644000000000003	5.7
2	LAUCT1714000000000003	4.4
3	LAUCT1245000000000003	1.8
4	LAUCT4819000000000003	4.3

	Unemployment Series ID	August 2023 Unemployment
0	LAUCT3651000000000004	234715
1	LAUCT0644000000000004	117804
2	LAUCT1714000000000004	61024
3	LAUCT1245000000000004	4350
4	LAUCT4819000000000004	32665



	city	latitude	longitude	population	unemploymentRate	unemploymentCount
0	New York, NY	40.6943	-73.9249	18972871	5.6	234715
1	Los Angeles, CA	34.1141	-118.4068	12121244	5.7	117804
2	Chicago, IL	41.8375	-87.6866	8595181	4.4	61024
3	Miami, FL	25.7840	-80.2101	5711945	1.8	4350
4	Dallas, TX	32.7935	-96.7667	5668165	4.3	32665

# Flask API

# Flask API

Separate Python file creates the Flask API which responds to API calls with the city dataset from the SQLite database in JSON format

```
@app.route("/USCityUnemploymentDATA", methods=['GET'])
def unemploymentDATA():
    """Return the city-level data (location, population, unemployment) as JSON"""

    # Create our session (link) from Python to the DB
    session = Session(engine)

    # Query all city unemployment data from the database + transfer to DF then to dictionary then return as JSON

    city_data = session.query(CityData).all()

    city_df = pd.DataFrame([(cd.city, cd.latitude, cd.longitude, cd.population, cd.unemploymentRate, cd.unemploymentCount) for
                             cd in city_data])

    print(city_df)

    df_to_json = city_df.to_json(orient="records")
    parsed_json = loads(df_to_json)
    json_city_data = dumps(parsed_json, indent=4)

    return json_city_data

    session.close()
```



127.0.0.1:5000/USCityUnemploymentDATA

```
[{"city": "New York, NY", "latitude": "40.6943", "longitude": "-73.9249", "population": "18972871", "unemploymentRate": "5.6", "unemploymentCount": "234715"}, {"city": "Los Angeles, CA", "latitude": "34.1141", "longitude": "-118.4068", "population": "12121244", "unemploymentRate": "5.7", "unemploymentCount": "117804"}, {"city": "Chicago, IL", "latitude": "41.8375", "longitude": "-87.6866", "population": "8595181", "unemploymentRate": "4.4", "unemploymentCount": "161024"}, {"city": "Miami, FL", "latitude": "25.7840", "longitude": "-80.2101", "population": "5711945", "unemploymentRate": "1.8", "unemploymentCount": "4350"}, {"city": "Dallas, TX", "latitude": "32.7935", "longitude": "-96.7667", "population": "5668165", "unemploymentRate": "4.3", "unemploymentCount": "32665"}, {"city": "Houston, TX", "latitude": "29.7860", "longitude": "-95.3855", "population": "5650910", "unemploymentRate": "5.2", "unemploymentCount": "62158"}, {"city": "Atlanta, GA", "latitude": "33.7628", "longitude": "-84.4220", "population": "5046555", "unemploymentRate": "3.9", "unemploymentCount": "10846"}, {"city": "Washington, DC", "latitude": "38.9047", "longitude": "-77.0163", "population": "4810669", "unemploymentRate": "5.5", "unemploymentCount": "21877"}, {"city": "Boston, MA", "latitude": "42.3188", "longitude": "-71.0852", "population": "4208580", "unemploymentRate": "3.0", "unemploymentCount": "11719"}, {"city": "Phoenix, AZ", "latitude": "33.5722", "longitude": "-112.0892", "population": "4047095", "unemploymentRate": "4.2", "unemploymentCount": "38801"}, {"city": "Detroit, MI", "latitude": "42.3834", "longitude": "-83.1024", "population": "3522856", "unemploymentRate": "8.3", "unemploymentCount": "21211"}, {"city": "Seattle, WA", "latitude": "47.6211", "longitude": "-122.3244", "population": "3438221", "unemploymentRate": "3.4", "unemploymentCount": "17375"}, {"city": "San Diego, CA", "latitude": "32.8313", "longitude": "-117.1222", "population": "3084174", "unemploymentRate": "3.8", "unemploymentCount": "27621"}, {"city": "Minneapolis, MN", "latitude": "44.9635", "longitude": "-93.2678", "population": "2856952", "unemploymentRate": "3.1", "unemploymentCount": "7864"}, {"city": "Tampa, FL", "latitude": "27.9945", "longitude": "-82.4447", "population": "2683956", "unemploymentRate": "3.2", "unemploymentCount": "7337"}, {"city": "Baltimore, MD", "latitude": "39.3051", "longitude": "-76.6144", "population": "2205092", "unemploymentRate": "2.3", "unemploymentCount": "6439"}, {"city": "Las Vegas, NV", "latitude": "36.2333", "longitude": "-115.2654", "population": "2150373", "unemploymentRate": "6.3", "unemploymentCount": "455371"}, {"city": "St. Louis, MO", "latitude": "38.6359", "longitude": "-90.2451", "population": "2092481", "unemploymentRate": "4.3", "unemploymentCount": "6679"}, {"city": "Portland, OR", "latitude": "45.5371", "longitude": "-122.6500", "population": "2036875", "unemploymentRate": "3.7", "unemploymentCount": "14126"}, {"city": "Riverside, CA", "latitude": "33.9381", "longitude": "-117.3949", "population": "2022285", "unemploymentRate": "4.5", "unemploymentCount": "7153"}, {"city": "Orlando, FL", "latitude": "28.4773", "longitude": "-81.3370", "population": "1927699", "unemploymentRate": "3.0", "unemploymentCount": "5500"}, {"city": "Sacramento, CA", "latitude": "38.5677", "longitude": "-121.4685", "population": "1924167", "unemploymentRate": "4.6", "unemploymentCount": "11353"}, {"city": "San Antonio, TX", "latitude": "29.4632", "longitude": "-98.5238", "population": "1910785", "unemploymentRate": "4.2", "unemploymentCount": "33127"}, {"city": "San Jose, CA", "latitude": "37.3012", "longitude": "-121.8480", "population": "1729879", "unemploymentRate": "3.7", "unemploymentCount": "20468"}, {"city": "Pittsburgh, PA", "latitude": "40.4397", "longitude": "-79.9763", "population": "1720279", "unemploymentRate": "3.5", "unemploymentCount": "5472"}, {"city": "Cincinnati, OH", "latitude": "39.1413", "longitude": "-84.5060", "population": "1712287", "unemploymentRate": "3.7", "unemploymentCount": "5721"}, {"city": "Cleveland, OH", "latitude": "41.4764", "longitude": "-81.6805", "population": "1683059", "unemploymentRate": "4.5", "unemploymentCount": "7261"}, {"city": "Austin, TX", "latitude": "30.3005", "longitude": "-97.7522", "population": "1659251", "unemploymentRate": "3.7", "unemploymentCount": "25096"}, {"city": "Kansas City, MO", "latitude": "39.1238", "longitude": "-94.5541", "population": "1644497", "unemploymentRate": "3.6", "unemploymentCount": "9934"}, {"city": "Columbus, OH", "latitude": "39.9862", "longitude": "-82.9855", "population": "1556848", "unemploymentRate": "3.3", "unemploymentCount": "16045"}, {"city": "Charlotte, NC", "latitude": "35.2083", "longitude": "-80.8303", "population": "1516107", "unemploymentRate": "3.5", "unemploymentCount": "18784"}, {"city": "Virginia Beach, VA", "latitude": "36.7335", "longitude": "-76.0435", "population": "1500764", "unemploymentRate": "2.5", "unemploymentCount": "6173"}, {"city": "Milwaukee, WI", "latitude": "43.0642", "longitude": "-87.9675", "population": "1340981", "unemploymentRate": "4.6", "unemploymentCount": "12760"}, {"city": "Providence, RI", "latitude": "41.8230", "longitude": "-71.4187", "population": "1270149", "unemploymentRate": "3.1", "unemploymentCount": "2749"}, {"city": "Jacksonville, FL", "latitude": "30.3322", "longitude": "-81.6749", "population": "1220191", "unemploymentRate": "3.4", "unemploymentCount": "17502"}, {"city": "Salt Lake City, UT", "latitude": "40.7766", "longitude": "-111.9311", "population": "1135344", "unemploymentRate": "2.9", "unemploymentCount": "3875"}, {"city": "Raleigh, NC", "latitude": "35.8324", "longitude": "-78.6429", "population": "1062018", "unemploymentRate": "3.4", "unemploymentCount": "9451"}, {"city": "Memphis, TN", "latitude": "35.1087", "longitude": "-89.9663", "population": "1034498", "unemploymentRate": "5.4", "unemploymentCount": "15901"}, {"city": "Richmond, VA", "latitude": "37.5295", "longitude": "-77.4756", "population": "1008069", "unemploymentRate": "5.4", "unemploymentCount": "15901"}]
```

# JavaScript+Leaflet

# JavaScript+Leaflet

Upon loading the HTML, JavaScript calls the Flask API, processes the JSON data on cities, creates 2 sets of Leaflet circle markers the visualize the data in different ways, and builds the Leaflet map with layers

```
1 // save API response
2 const baseurl = "http://127.0.0.1:5000/"
3 const apiUrl = baseurl + "api/getemploymentdata/"
4 console.log("APIurl: " + apiUrl);
5
6
7 // JS request to the API URL
8 $(ajax(apiurl)).then(APIresponse => create employmentMap(APIresponse)).catch(console.log("No response from the API - make sure the flask API is active/serving"));
9 console.log("APIresponse: " + APIresponse);
10
11
12
13 // function to define/create the features (specific earthquakes) and to create the map
14 function create employmentMap(city_data) {
15
16     console.log("API response / city_data: " + city_data);
17
18     let cityMarkers = [];
19     let cityMarkers2 = [];
20
21     // loop through the earthquakes/features array creating a new circle marker with detailed popup for each quake and pushing them into the quakeMarkers array
22     for (let i = 0; i < Object.keys(city_data).length; i++) {
23
24         let city = city_data[i];
25         console.log("city object: " + city);
26
27         let name = city.city;
28         console.log("city name: " + name);
29
30         let coordinates = [city.latitude, city.longitude];
31         console.log("coordinates: " + coordinates);
32
33         let population = Number(city.population);
34         let population_string = population.toLocaleString();
35         console.log("population: " + population);
36         console.log("population string: " + population_string);
37     }
38 }
```



```
74 // create circle marker based on coordinates + attributes dict, with detailed popup showing additional info
75 cityMarkers.push(
76     L.circle(coordinates, {featureAttributes})
77     .bindPopup('<h3>{name}</h3><br>
78     Unemployment Rate (August 2023): {unemploymentRate}<br>
79     Unemployment Count (August 2023): {unemploymentCount_string}')
80 );
81
82 // save attributes dict/JSON for circle marker
83 let featureAttributes2 = {
84     fillOpacity: 0.5,
85     fillColor: colorBasedOnRate,
86     radius: Math.pow(unemploymentRate, 2.5)*1000,
87     stroke: true,
88     color: "black",
89     weight: 0.5
90 };
91
92 // create circle marker based on coordinates + attributes dict, with detailed popup showing additional info
93 cityMarkers2.push(
94     L.circle(coordinates, {featureAttributes2})
95     .bindPopup('<h3>{name}</h3><br>
96     Unemployment Rate (August 2023): {unemploymentRate}<br>
97     Unemployment Count (August 2023): {unemploymentCount_string}')
98 );
99
100
101
102 // create overlay layer from quakeMarkers array
103 let cityLayer = L.layerGroup(cityMarkers);
104 let cityLayer2 = L.layerGroup(cityMarkers2);
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

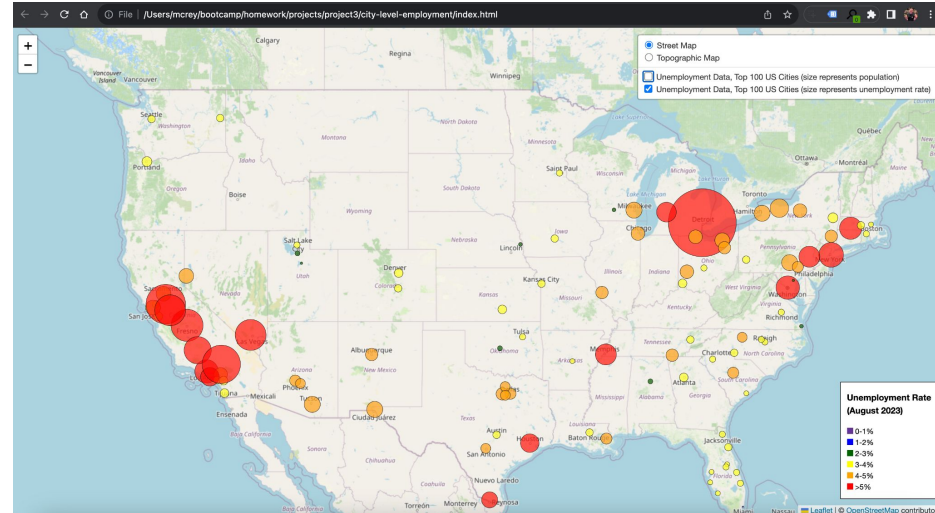
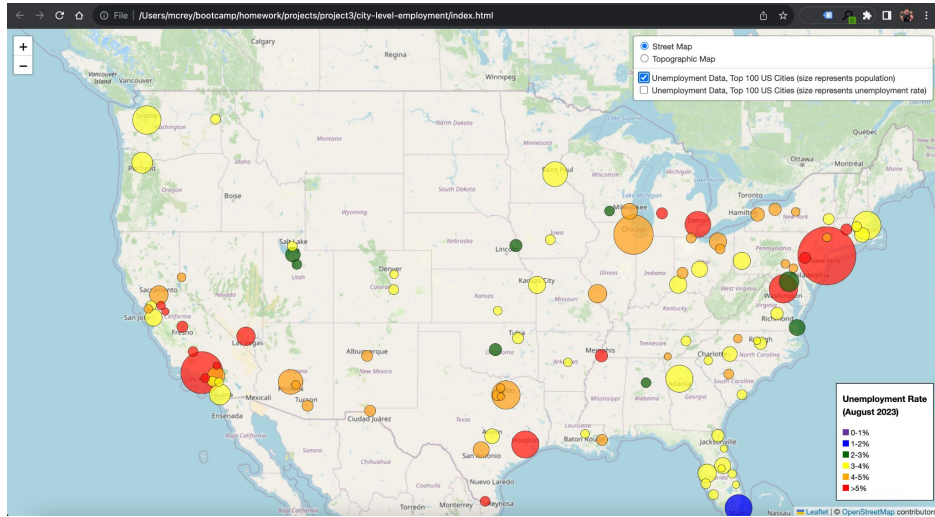


```
126 // create overlayMaps object to hold our quakeLayer of earthquake markers with popups
127
128 let overlayMaps = {
129     "Unemployment Data, Top 100 US Cities (size represents population)": cityLayer,
130     "Unemployment Data, Top 100 US Cities (size represents unemployment rate)": cityLayer2,
131 };
132
133 // create map, passing streetmap and earthquakes layers to display on load, showing the entire world centered on [0,0]
134 let myMap = L.map("map", {
135     center: [39.8355, -99.0909],
136     zoom: 4,
137     layers: [street, cityLayer]
138 });
139
140
141 // add a legend showing how the marker colors represent earthquake depth
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

**Final Product**

# Interactive Map

*Upon loading the HTML, JavaScript calls the Flask API, processes the JSON data on cities, creates 2 sets of Leaflet circle markers the visualize the data in different ways, and builds the Leaflet map with layers*



# Conclusions and Next Steps



# Analysis/Takeaways/Value

- This map enables users to easily go beyond average national unemployment data and view how that data varies at the city level for the 100 largest US cities (by population), including potential correlation in unemployment statistics within sub-regions of the country
  - The first map overlay/view seems to indicate some correlation between city size/population and higher unemployment rates, as well as generally lower unemployment rates in the Southwest, especially Florida, and in the Northwest / Mountain West
  - The second map overlay/view makes it much more clear to see that the highest unemployment rates are quite concentrated within a few smaller regions of the country, specifically Detroit, MI then California / Las Vegas, then the Northeast

# Next Steps / Future Analysis

- Given additional time it would be valuable to add additional pages/options to help users explore more unemployment rate data
  - Analyzing unemployment data over longer periods of time for individual cities
  - Pulling similar data for larger areas/regions to allow users to view the data at different levels of aggregation
  - Running statistical analysis on various datasets to better describe and understand how current rates and/or rates for individual cities compare
- It would also be great to pull in additional datasets to allow users to view unemployment rate statistics within the context of other geographical data
  - GDP per capita, credit card debt per capita, median home prices, demographic data, job type data, etc
- Chart.JS more work needed
  - I added a Chart.JS chart as well but was not able to get the formatting to work without it getting in the way of the map or Plotly chart.. More HTML/CSS learning needed to make that work so I just removed it

# Appendix

# Acknowledgements and Data Sources

- SimpleMaps used for database of US cities with populations and lat+long coordinates: <https://simplemaps.com/data/us-cities>
- help with SQLite database creation via Python/Pandas pulled from <https://www.fullstackpython.com/blog/export-pandas-dataframes-sqlite-sqlalchemy.html>
- help with Flask+Python+HTML+JS for creating web applications pulled from <https://www.geeksforgeeks.org/pass-javascript-variables-to-python-in-flask/>
- reverse geocoding for looking up state names from lat/long coordinates via <https://geocode.maps.co/> API