grocery_score

August 30, 2019

[1]:
```
import pymysql
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import warnings

warnings.filterwarnings("ignore")

def qry(sql, database=None):

    connection = pymysql.connect('localhost', 'root', 'root', database)
    df = pd.read_sql(sql, connection)

    connection.close()

    print("\nReturned {0} rows".format(len(df.index)))
    return df

database = 'property_value'
```

Selecting all grocery stores from the database.

[2]:
```
sql = "SELECT * FROM groceries;"
groceries = qry(sql, database)
groceries.head
```

```
Returned 506 rows
```

[2]:
```
<bound method NDFrame.head of      grocery_id
store_name  license_id  zip_code  \
0             1              BOB'S LIQUORS & GROCERIES         957     60644
1             2                   QUICK STOP FOOD MART       39791     60645
2             3                   DOMINICK'S #147/1147       41674     60607
3             4                        CERMAK PRODUCE     1515206     60632
4             5                     WHOLE FOODS MARKET       54059     60657
5             6                              PATEL BROS       25338     60659
6             7                        OHIO FOOD MART     1225367     60644
```

7	8	JOE'S FOOD & LIQUOR	822	60625
8	9	TWO BLUE FOODS INC	66710	60628
9	10	JEWEL FOOD STORE #3349	1353385	60639
10	11	200 LIQUORS	48663	60653
11	12	CARNICERIA LA GLORIA	1471030	60632
12	13	WHOLE FOODS MARKET	1678043	60607
13	14	PETERSON FOOD MART	1272746	60659
14	15	STANLEY'S	15042	60622
15	16	FOOD 4 LESS	1884255	60639
16	17	COMMERCIAL GROCERY	56894	60617
17	18	JEWEL FOOD STORE	1953613	60643
18	19	LA UNICA FOOD MART, INC.	491	60660
19	20	MID TOWN FOODS	39968	60651
20	21	7400 S HALSTED FOOD AND LIQUORS, INC.	980	60621
21	22	THREE BROTHER	6321	60622
22	23	CUENCA'S BAKERY & GROCERIES	1619464	60641
23	24	JEWEL FOOD STORES #3262	1409	60641
24	25	ROMAN BROS 1 INC	11900	60626
25	26	TAI NAM CORPORATION	34508	60640
26	27	LA JALISCIENCE	4749	60623
27	28	HOLLYWOOD TOWER MKT	1858	60660
28	29	A & R FOOD MART	1954411	60639
29	30	S. S. FOOD & LIQUOR	1476109	60614
..	...	...	...	...
476	477	ALDI	1959623	60660
477	478	FAIRPLAY FOODS INC	10947	60643
478	479	HILLTOP LIQUORS MART	788	60628
479	480	BIG APPLE FINER FOODS INC	4900	60614
480	481	AVENUE FOOD MAR INC	1246771	60634
481	482	FRESH FARMS INTERNATIONAL	30144	60659
482	483	LAKE VIEW FOODS INC	2055	60613
483	484	THE CELLAR	1574603	60634
484	485	MARIANO'S FRESH MARKET #8507	2163771	60630
485	486	RAINBOW GROCERY	61801	60636
486	487	CARNICERIA LA HACIENDA, INC.	1492826	60629
487	488	CICERO DELI INC.	20884	60641
488	489	K BUENA	1999075	60629
489	490	JEWEL FOOD STORE #3210	1333	60610
490	491	NANCY FOODS INC	79959	60628
491	492	WINDY CITY INC	24289	60644
492	493	ONE STOP FOOD & LIQUOR STORE	1094	60653
493	494	D & A FOOD & LIQUOR	46444	60636
494	495	ULTRA FOODS	1741082	60652
495	496	PETE'S FRESH SUPERMARKET	1771242	60608
496	497	CARNICERIA GUANAJUATO #3	2060960	60618
497	498	MINI MERCADO LA BETANIA	33473	60623
498	499	JERUSALEM LIQUORS	440	60625

499	500		WALGREENS # 05192	1144217	60617
500	501		JEWEL FOOD STORE #3114	1388	60655
501	502		WALGREENS #11410	1959845	60639
502	503		TIERRA CALIENTE MARKET	1769425	60622
503	504		QUALITY FOOD ON 71 ST INC	76481	60629
504	505		JEWEL FOOD STORE #3302	1402	60609
505	506		CERMAK PRODUCE	1770942	60623

	store_sqft	lat	lon
0	10001	41.880296	-87.752494
1	3000	42.004750	-87.699423
2	68000	41.864521	-87.639304
3	25000	41.797992	-87.723444
4	4501	41.941566	-87.668860
5	5000	41.997747	-87.695251
6	3500	41.891049	-87.759733
7	3500	41.968445	-87.719417
8	3000	41.712404	-87.605128
9	64285	41.926297	-87.785695
10	5000	41.809424	-87.620952
11	6500	41.800969	-87.688722
12	11310	41.868890	-87.639099
13	2500	41.990585	-87.693535
14	10001	41.910575	-87.662235
15	58000	41.909505	-87.746960
16	2500	41.713498	-87.550733
17	62603	41.680332	-87.662585
18	5000	41.997993	-87.668428
19	3520	41.902584	-87.723619
20	10001	41.759675	-87.644420
21	10001	41.898107	-87.699378
22	4500	41.960768	-87.733947
23	35000	41.953554	-87.745055
24	10001	42.008856	-87.674080
25	10001	41.972214	-87.659609
26	3000	41.844388	-87.706497
27	3000	41.985638	-87.655094
28	3500	41.920596	-87.775193
29	2703	41.926372	-87.640800
..	...	...	...
476	15680	41.995211	-87.660278
477	18500	41.691679	-87.679779
478	3200	41.707248	-87.620056
479	9000	41.924966	-87.639926
480	18000	41.938237	-87.797760
481	9500	41.997736	-87.695929
482	3100	41.957040	-87.654763

```
483        5000  41.936613 -87.806848
484       12198  41.978710 -87.757154
485        2500  41.768328 -87.673819
486        4000  41.786024 -87.710605
487        3600  41.939047 -87.747022
488        3500  41.771383 -87.730953
489       23845  41.904265 -87.631603
490        2500  41.670252 -87.622272
491        4225  41.875944 -87.745250
492       58000  41.816865 -87.598689
493        5500  41.793554 -87.664638
494       87600  41.735274 -87.703511
495       75000  41.852101 -87.689051
496        9000  41.938570 -87.697946
497        2640  41.840397 -87.730675
498        4500  41.968355 -87.707394
499       12198  41.728337 -87.552860
500       20860  41.706313 -87.699767
501       12198  41.910714 -87.726210
502        3125  41.907102 -87.667740
503        3000  41.764512 -87.687987
504       25431  41.827584 -87.682454
505       17000  41.844367 -87.707898

[506 rows x 7 columns]>
```

Finding all grocery stores with the "liqour" in the title because we do not want to count them as full grocery stores.

[3]:
```
liquorStores = groceries.loc[groceries['store_name'].str.contains(pat =␣
 ↪'liquor', case = False)]
liquorStores
```

[3]:

	grocery_id	store_name	license_id	zip_code	\
0	1	BOB'S LIQUORS & GROCERIES	957	60644	
7	8	JOE'S FOOD & LIQUOR	822	60625	
10	11	200 LIQUORS	48663	60653	
20	21	7400 S HALSTED FOOD AND LIQUORS, INC.	980	60621	
29	30	S. S. FOOD & LIQUOR	1476109	60614	
34	35	W K R B FOOD & LIQUOR	25165	60619	
57	58	DIA FOOD & LIQUOR	41854	60643	
60	61	HAREER FOOD & LIQUOR	52337	60609	
69	70	SUPER FOOD & LIQUORS, INC.	40429	60660	
88	89	J K & S FOOD & LIQUOR	1000978	60621	
89	90	K & K FOOD & LIQUOR	62092	60609	
93	94	CUPBOARD FOOD & LIQUOR	7110	60632	
100	101	LATIN GROCERY & LIQUOR, INC	20046	60647	
103	104	M & J LIQUOR & GROCERY	15168	60625	
121	122	Three Stars Liquors	1488855	60624	

124	125	ASHLAND FOOD AND LIQUORS	11636	60643
137	138	KING FOOD & LIQUOR	40321	60621
161	162	Baltimore Food & Liquor	1446831	60633
164	165	NICKEL LIQUOR & MINI MART INC	22084	60651
172	173	EL YUNQUE FOOD & LIQUOR	1986802	60647
176	177	SAM FOOD AND LIQUOR	8745	60626
177	178	ALMUFLIHI FOOD & LIQUOR	62800	60609
185	186	W & R FOOD & LIQUOR INC	19348	60649
187	188	Z & S FOOD & LIQUOR	7382	60620
219	220	SUN LIQUORS	19461	60660
232	233	MANDY'S LIQUOR	17168	60640
244	245	LUX 2 FOOD & LIQUOR	1869021	60620
249	250	KHALIL FOOD & LIQUOR	46846	60628
251	252	SUNRISE FOOD AND LIQUORS	21927	60651
252	253	BOULEVARD FOOD & LIQUOR LLC	1931999	60634
256	257	S & K FOOD & LIQUOR, INC.	1544	60649
268	269	EXTRA VALUE FOOD & LIQUOR	1823321	60636
277	278	RJ Liquors	1597198	60639
279	280	PRESTIGE LIQUORS	1883995	60636
284	285	MR G'S FOOD & LIQUOR	1429	60637
288	289	MELISSA'S FOOD & LIQUORS INC	29494	60651
295	296	R & S DISCOUNT FOOD, WINE, & LIQUOR	8647	60647
299	300	JAMAICA FOOD & LIQUOR INC	22679	60653
338	339	P & I FOOD & LIQUOR INC.	16647	60651
348	349	Laxi Liquor	47780	60707
349	350	EDDIE'S SUNSHINE FOOD & LIQUOR, INC.	24217	60628
353	354	STEVE FOOD & LIQUOR	47965	60636
361	362	NORTHWEST FOOD & LIQUOR	1119636	60639
364	365	RICH'S DELI & LIQUOR, INC	16735	60622
377	378	SAFAH FOOD & LIQUOR INC	14103	60636
383	384	JOE'S FOOD AND LIQUOR DEPOT	1575140	60639
388	389	PUERTO RICO FOOD & LIQUORS	2033049	60622
395	396	RAINBOW FINER FOOD & LIQUOR	9138	60636
400	401	KEY FOOD LIQUOR	17551	60827
402	403	PERSONAL LIQUORS INC	8913	60624
404	405	GILMART'S FOOD & LIQUOR	22715	60632
412	413	A & G LIQUORS	44373	60637
427	428	JARDAN FOOD & LIQUOR	11226	60637
440	441	ALL STAR FOOD & LIQUOR	1922124	60608
478	479	HILLTOP LIQUORS MART	788	60628
492	493	ONE STOP FOOD & LIQUOR STORE	1094	60653
493	494	D & A FOOD & LIQUOR	46444	60636
498	499	JERUSALEM LIQUORS	440	60625

	store_sqft	lat	lon
0	10001	41.880296	-87.752494
7	3500	41.968445	-87.719417

10	5000	41.809424	-87.620952
20	10001	41.759675	-87.644420
29	2703	41.926372	-87.640800
34	4340	41.723860	-87.604365
57	4500	41.684956	-87.659726
60	2500	41.794559	-87.645313
69	4000	41.983564	-87.657146
88	3024	41.787039	-87.637813
89	4200	41.801577	-87.653788
93	3000	41.793589	-87.705946
100	10001	41.910049	-87.721458
103	2900	41.968435	-87.720481
121	4600	41.885823	-87.735788
124	10001	41.721327	-87.662534
137	3750	41.768926	-87.644656
161	3404	41.652965	-87.547299
164	10001	41.902634	-87.717931
172	2500	41.915761	-87.697296
176	3000	42.005945	-87.661175
177	4500	41.795871	-87.645344
185	2700	41.751419	-87.582206
187	6000	41.735757	-87.657828
219	4350	41.994466	-87.658393
232	3000	41.968656	-87.678438
244	2500	41.756163	-87.652820
249	3000	41.707218	-87.627752
251	7000	41.895315	-87.720200
252	3350	41.953183	-87.775259
256	10001	41.758993	-87.560378
268	5000	41.779615	-87.664077
277	3550	41.924064	-87.767660
279	2900	41.764496	-87.683384
284	3000	41.789430	-87.617840
288	3500	41.905059	-87.723951
295	2800	41.932047	-87.688979
299	2500	41.817030	-87.606858
338	3000	41.902707	-87.731018
348	3200	41.923681	-87.798676
349	3000	41.697818	-87.621011
353	3500	41.783334	-87.654743
361	3180	41.909615	-87.755478
364	3400	41.897461	-87.686739
377	3000	41.764896	-87.653965
383	4000	41.923992	-87.772289
388	2500	41.899222	-87.691738
395	3500	41.778922	-87.664353
400	3800	41.651973	-87.617084

```
402        6000  41.880539 -87.732216
404       10001  41.801400 -87.726044
412        4800  41.767831 -87.624766
427        9500  41.794325 -87.618521
440        4200  41.841484 -87.658022
478        3200  41.707248 -87.620056
492       58000  41.816865 -87.598689
493        5500  41.793554 -87.664638
498        4500  41.968355 -87.707394
```

Selecting all zip codes in the database.

[4]:
```
sql = "SELECT * FROM property_value.zipcode;"
zipcodes = qry(sql, database)
zipcodes.head()
```

```
Returned 33120 rows
```

[4]:

	zip_id	zip_code	population	aggpop
0	1	601	17800	127967
1	2	602	39716	287227
2	3	603	51565	374522
3	4	606	6320	45251
4	5	610	27976	201340

Creating a list of all Chicago Zip codes.

[5]:
```
chicagozips = [60601,
60602,
60603,
60604,
60605,
60606,
60607,
60608,
60609,
60610,
60611,
60612,
60613,
60614,
60615,
60616,
60617,
60618,
60619,
60620,
60621,
60622,
```

```
60623,
60624,
60625,
60626,
60628,
60629,
60630,
60631,
60632,
60633,
60634,
60636,
60637,
60638,
60639,
60640,
60641,
60642,
60643,
60644,
60645,
60646,
60647,
60649,
60651,
60652,
60653,
60654,
60655,
60656,
60657,
60659,
60660,
60661,
60706,
60707,
60712,
60714,
60803,
60804,
60805,
60827,
]
```

Creating a dataframe of the Chicago zip codes.

[6]:
```
chicagozipcodes = zipcodes.loc[zipcodes['zip_code'].isin(chicagozips)]
chicagozipcodes
```

[6]:

	zip_id	zip_code	population	aggpop
20694	20695	60601	13695	75402
20695	20696	60602	1252	9115
20696	20697	60603	1029	5585
20697	20698	60604	619	3389
20698	20699	60605	26623	172848
20699	20700	60606	3011	18568
20700	20701	60607	28377	178133
20701	20702	60608	78072	560234
20702	20703	60609	62250	440223
20703	20704	60610	38438	268191
20704	20705	60611	31563	208913
20705	20706	60612	35559	247365
20706	20707	60613	49519	344588
20707	20708	60614	69817	476348
20708	20709	60615	40257	286810
20709	20710	60616	52580	350400
20710	20711	60617	80002	586171
20711	20712	60618	95632	670358
20712	20713	60619	62822	453180
20713	20714	60620	69299	508281
20714	20715	60621	31383	235658
20715	20716	60622	54467	378982
20716	20717	60623	88137	612841
20717	20718	60624	38134	272236
20718	20719	60625	79157	552560
20719	20720	60626	50090	352681
20720	20721	60628	68077	500754
20721	20722	60629	115104	797117
20722	20723	60630	57627	395717
20723	20724	60631	28238	200708
...	...	...	...	...
20728	20729	60637	49158	340854
20729	20730	60638	57746	394118
20730	20731	60639	90211	637891
20731	20732	60640	67088	458042
20732	20733	60641	70642	500251
20733	20734	60642	19508	133097
20734	20735	60643	50507	353727
20735	20736	60644	49645	338879
20736	20737	60645	47131	322969
20737	20738	60646	27454	195089
20738	20739	60647	88866	612493
20739	20740	60649	45218	317218
20740	20741	60651	61759	437228
20741	20742	60652	43228	296716
20742	20743	60653	31045	215775

```
20743   20744     60654       17328  108932
20744   20745     60655       28741  198193
20745   20746     60656       27926  194102
20746   20747     60657       70105  479869
20747   20748     60659       38995  273559
20748   20749     60660       41490  292015
20749   20750     60661        9343   56170
20750   20751     60706       23604  162839
20751   20752     60707       43451  299610
20752   20753     60712       12637   88178
20753   20754     60714       29730  209540
20754   20755     60803       22762  159541
20755   20756     60804       83972  589175
20756   20757     60805       19849  139146
20757   20758     60827       28864  199135

[64 rows x 4 columns]
```

Calculating the total square footage of grocery stores that are not liquor stores in each Chicago zip code.

[7]:
```
grocerySqft = pd.pivot_table(
    chicagozipcodes.merge(groceries.loc[~groceries['store_name'].str.
 ↪contains(pat = 'liquor', case = False)], on = 'zip_code', how = 'left'),
    index = 'zip_code',
    values = 'store_sqft',
    aggfunc = 'sum',
    fill_value = 0
)

grocerySqft
```

[7]:
```
          store_sqft
zip_code
60601          14260
60602              0
60603              0
60604              0
60605          63221
60606              0
60607          97310
60608         261801
60609         222814
60610         200940
60611          77836
60612          20984
60613         146680
60614         136787
60615          42047
```

60616	121039
60617	206545
60618	257688
60619	90587
60620	241698
60621	27910
60622	152134
60623	149118
60624	40365
60625	121486
60626	133089
60628	166628
60629	108700
60630	109791
60631	14500
...	...
60637	83703
60638	143642
60639	373286
60640	193854
60641	107400
60642	74883
60643	118071
60644	56726
60645	74488
60646	49836
60647	213457
60649	65618
60651	45121
60652	103900
60653	46500
60654	0
60655	43361
60656	47097
60657	67230
60659	66771
60660	58881
60661	44965
60706	0
60707	15179
60712	0
60714	0
60803	0
60804	0
60805	0
60827	10001

```
[64 rows x 1 columns]
```

Counting the number of grocery stores per Chicago zip code.

[8]:
```
groceryCount = pd.pivot_table(
    chicagozipcodes.merge(groceries.loc[~groceries['store_name'].str.
 ↪contains(pat = 'liquor', case = False)], on = 'zip_code', how = 'left'),
    index = 'zip_code',
    values = 'grocery_id',
    aggfunc = 'count',
    fill_value = 0
)

groceryCount
```

[8]:

zip_code	grocery_id
60601	3
60602	0
60603	0
60604	0
60605	2
60606	0
60607	4
60608	17
60609	17
60610	11
60611	5
60612	4
60613	9
60614	10
60615	4
60616	8
60617	14
60618	17
60619	9
60620	15
60621	4
60622	12
60623	22
60624	6
60625	9
60626	10
60628	11
60629	15
60630	8
60631	2
...	...
60637	8

```
60638              5
60639             29
60640             15
60641             10
60642              1
60643              7
60644              6
60645              4
60646              5
60647             15
60649              6
60651              7
60652              2
60653              3
60654              0
60655              3
60656              2
60657              5
60659             14
60660              7
60661              3
60706              0
60707              2
60712              0
60714              0
60803              0
60804              0
60805              0
60827              1

[64 rows x 1 columns]
```

Merging the square footage and counts with the Chicago data frame.

[9]:
```
groceryScore = grocerySqft.merge(chicagozipcodes[['zip_code', 'population']],␣
↪on = 'zip_code')
groceryScore = groceryCount.merge(groceryScore, on = 'zip_code')
groceryScore.set_index('zip_code', inplace = True)
groceryScore.columns = ['count', 'sqft', 'population']
groceryScore
```

[9]:
```
           count    sqft  population
zip_code
60601          3   14260       13695
60602          0       0        1252
60603          0       0        1029
60604          0       0         619
60605          2   63221       26623
60606          0       0        3011
```

60607	4	97310	28377
60608	17	261801	78072
60609	17	222814	62250
60610	11	200940	38438
60611	5	77836	31563
60612	4	20984	35559
60613	9	146680	49519
60614	10	136787	69817
60615	4	42047	40257
60616	8	121039	52580
60617	14	206545	80002
60618	17	257688	95632
60619	9	90587	62822
60620	15	241698	69299
60621	4	27910	31383
60622	12	152134	54467
60623	22	149118	88137
60624	6	40365	38134
60625	9	121486	79157
60626	10	133089	50090
60628	11	166628	68077
60629	15	108700	115104
60630	8	109791	57627
60631	2	14500	28238
...	...	...	...
60637	8	83703	49158
60638	5	143642	57746
60639	29	373286	90211
60640	15	193854	67088
60641	10	107400	70642
60642	1	74883	19508
60643	7	118071	50507
60644	6	56726	49645
60645	4	74488	47131
60646	5	49836	27454
60647	15	213457	88866
60649	6	65618	45218
60651	7	45121	61759
60652	2	103900	43228
60653	3	46500	31045
60654	0	0	17328
60655	3	43361	28741
60656	2	47097	27926
60657	5	67230	70105
60659	14	66771	38995
60660	7	58881	41490
60661	3	44965	9343

```
60706          0       0       23604
60707          2   15179       43451
60712          0       0       12637
60714          0       0       29730
60803          0       0       22762
60804          0       0       83972
60805          0       0       19849
60827          1   10001       28864

[64 rows x 3 columns]
```

Calculating the count of grocery stores per capita and the total square footage per capita in each zip code

[10]:
```
groceryScore['count_score'] = groceryScore['count'] /␣
 ↪groceryScore['population'] * 10000
groceryScore['sqft_score'] = groceryScore['sqft'] / groceryScore['population']
groceryScore
```

[10]:

zip_code	count	sqft	population	count_score	sqft_score
60601	3	14260	13695	2.190581	1.041256
60602	0	0	1252	0.000000	0.000000
60603	0	0	1029	0.000000	0.000000
60604	0	0	619	0.000000	0.000000
60605	2	63221	26623	0.751230	2.374676
60606	0	0	3011	0.000000	0.000000
60607	4	97310	28377	1.409592	3.429186
60608	17	261801	78072	2.177477	3.353328
60609	17	222814	62250	2.730924	3.579341
60610	11	200940	38438	2.861751	5.227639
60611	5	77836	31563	1.584133	2.466052
60612	4	20984	35559	1.124891	0.590118
60613	9	146680	49519	1.817484	2.962095
60614	10	136787	69817	1.432316	1.959222
60615	4	42047	40257	0.993616	1.044464
60616	8	121039	52580	1.521491	2.301997
60617	14	206545	80002	1.749956	2.581748
60618	17	257688	95632	1.777648	2.694579
60619	9	90587	62822	1.432619	1.441963
60620	15	241698	69299	2.164533	3.487756
60621	4	27910	31383	1.274575	0.889335
60622	12	152134	54467	2.203169	2.793141
60623	22	149118	88137	2.496114	1.691889
60624	6	40365	38134	1.573399	1.058504
60625	9	121486	79157	1.136981	1.534747
60626	10	133089	50090	1.996406	2.656997
60628	11	166628	68077	1.615817	2.447640
60629	15	108700	115104	1.303169	0.944363

```
60630         8  109791       57627     1.388238    1.905201
60631         2   14500       28238     0.708265    0.513492
...         ...     ...         ...          ...         ...
60637         8   83703       49158     1.627406    1.702734
60638         5  143642       57746     0.865861    2.487480
60639        29  373286       90211     3.214686    4.137921
60640        15  193854       67088     2.235869    2.889548
60641        10  107400       70642     1.415588    1.520342
60642         1   74883       19508     0.512610    3.838579
60643         7  118071       50507     1.385947    2.337716
60644         6   56726       49645     1.208581    1.142633
60645         4   74488       47131     0.848698    1.580446
60646         5   49836       27454     1.821228    1.815255
60647        15  213457       88866     1.687935    2.402010
60649         6   65618       45218     1.326905    1.451148
60651         7   45121       61759     1.133438    0.730598
60652         2  103900       43228     0.462663    2.403535
60653         3   46500       31045     0.966339    1.497826
60654         0       0       17328     0.000000    0.000000
60655         3   43361       28741     1.043805    1.508681
60656         2   47097       27926     0.716178    1.686493
60657         5   67230       70105     0.713216    0.958990
60659        14   66771       38995     3.590204    1.712296
60660         7   58881       41490     1.687154    1.419161
60661         3   44965        9343     3.210960    4.812694
60706         0       0       23604     0.000000    0.000000
60707         2   15179       43451     0.460289    0.349336
60712         0       0       12637     0.000000    0.000000
60714         0       0       29730     0.000000    0.000000
60803         0       0       22762     0.000000    0.000000
60804         0       0       83972     0.000000    0.000000
60805         0       0       19849     0.000000    0.000000
60827         1   10001       28864     0.346452    0.346487

[64 rows x 5 columns]
```

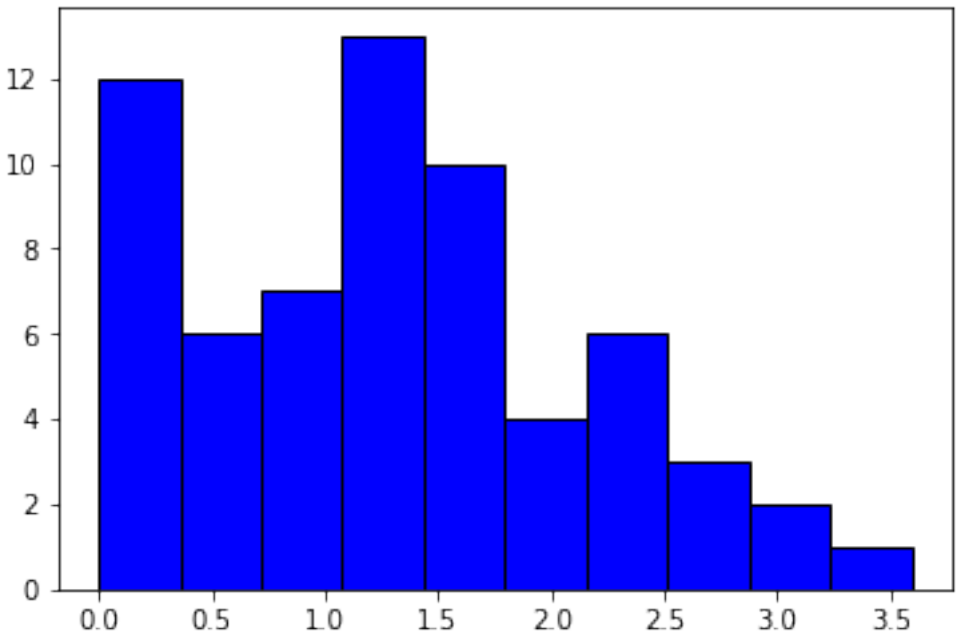
Creating the grocery scores csv

[11]:
```
groceryScore.to_csv("grocery_score.csv", sep=",", encoding='utf-8')
```

Checking for zero inflation in count of grocery stores per zip code.

[12]:
```
plt.hist(groceryScore['count_score'], color = 'blue', edgecolor = 'black')
```

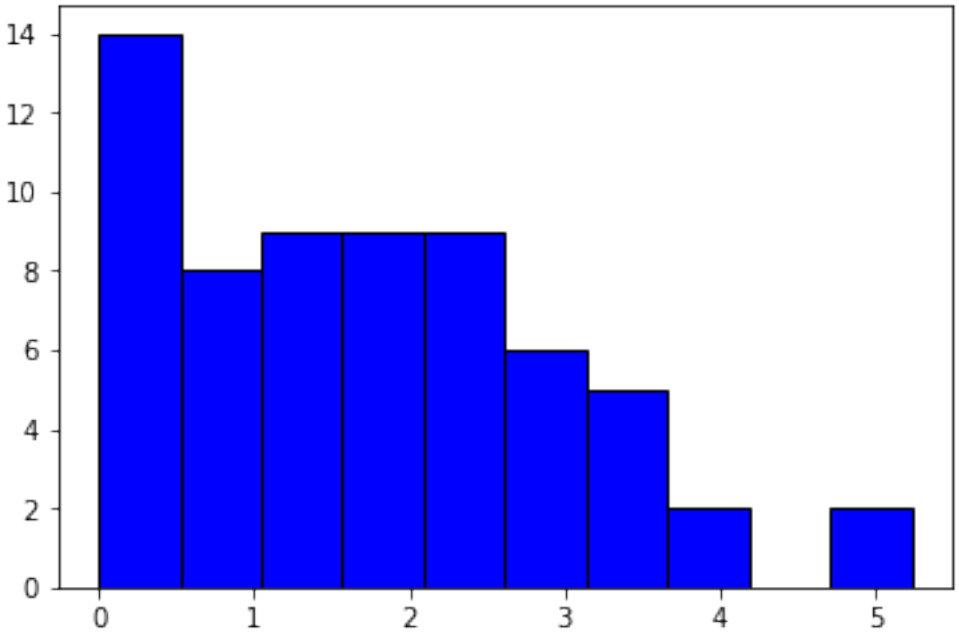
[12]:
```
(array([12.,  6.,  7., 13., 10.,  4.,  6.,  3.,  2.,  1.]),
 array([0.        , 0.35902039, 0.71804077, 1.07706116, 1.43608155,
        1.79510194, 2.15412232, 2.51314271, 2.8721631 , 3.23118349,
        3.59020387]),
 <a list of 10 Patch objects>)
```



Checking for zero inflation in total square footage of grocery stores per zip code.

[13]:
```
plt.hist(groceryScore['sqft_score'], color = 'blue', edgecolor = 'black')
```

[13]:
```
(array([14.,  8.,  9.,  9.,  9.,  6.,  5.,  2.,  0.,  2.]),
 array([0.        , 0.52276393, 1.04552786, 1.56829179, 2.09105573,
        2.61381966, 3.13658359, 3.65934752, 4.18211145, 4.70487538,
        5.22763932]),
 <a list of 10 Patch objects>)
```


14
12
10
8
6
4
2
0
0
1
2
3
4
5