DATA2001 Report

SID: 520102934, 520495526

Data Description

We are using 9 datasets provided in the canvas assignment specs description and 2 additional datasets. Businesses.csv, and SA2_2021_AUST_SHP_GDA2020 folder are from the Australian Bureau of Statistics. Stops.txt is from Transport for NSW. PollingPlaces2019.csv is from Australian Electoral Commision. The school catchments folder which includes primary, secondary, and future school datasets is from NSW Department of Education. As for income.csv and Population.csv, we couldn't find the origin of the dataset but it is uploaded on Canvas.

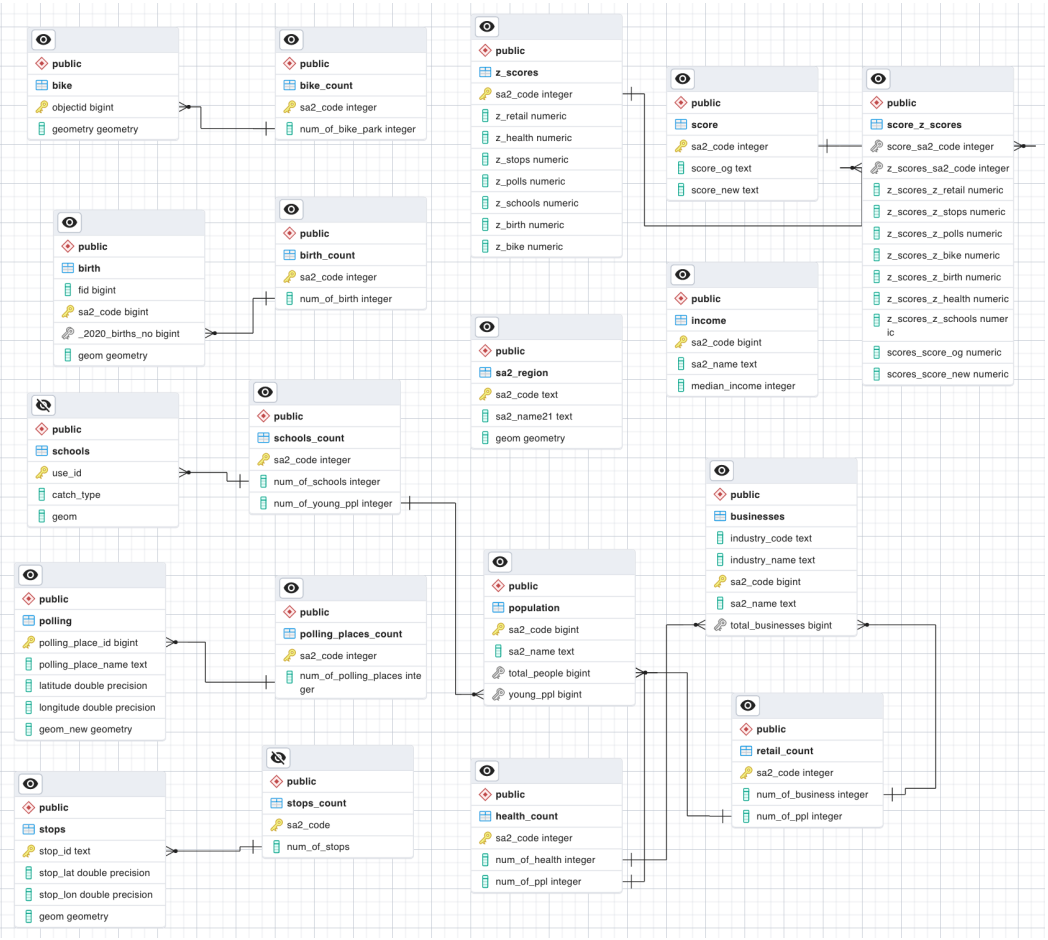
The first additional dataset is "birth.csv". We retrieved it from data.aurin.org but the origin is from the Australian Bureau of Statistics. It records the number of births and fertility rate in each sa2_code of each year from 2010 to 2020. The second dataset is "bicycle_parking.geojson". It lists all the available bicycle parking in NSW with its location as POINT in geometry. The dataset is from Transport for NSW. Both data contains spatial data but we decided to use the spatial data from bicycle_parking file when creating WKT elements and using ST functions because the birth dataset includes sa2_codes.

For preprocessing the data for sql queries, we check whether each dataset contains null values. We found that some datasets don't have null values. For the sa2_region dataset, it has null values in the geometry column. This is undesirable when using ST functions to find if the sa2_region geom intersects or contains the related dataset's geometry so we decided to remove rows with null values in geom. For a population dataset where we would have to divide the number of something with the number of people in the area to find the z-scores, we decided to remove rows with 0 as their total number of people or young people in each area. For the rest of the datasets with null values, it doesn't include in the columns we will be using to make queries.

We also decided to combine primary school, secondary school, and future school datasets together into one school dataset to make it easier for later integrations. We first put all rows from three dataset together then we used pandas drop_duplicate on school use_id to remove the same schools. We also decided to use srid = 4283 because to use ST functions, it says we have to have the same srid in the two input parameters so we decided to use the most common srid out of the datasets.

Database Description

Using the 11 datasets, we create columns in sql for each table corresponding to each dataset. Not all the columns are needed to calculate the z_score so we don't name a column for every column in the datasets. Most of the time, the primary key is the sa2_code of the dataset but there are also tables where it uses ids so that we can count how many of those are in the dataset according to sa2_code. The picture below shows the overview of our schema.



To calculate the z-score easier for us, we decided to create a table for every related z-score.

We decided to comment outside the picture instead that each sa2_code is registered either through the original dataset or using the ST functions using the spatial data, since it will be too messy to include them in the picture. After getting the z-score for each sa2_code, we update the value in the z_scores table, which is used in calculating the sigmoid function. The scores after doing sigmoid function are stored in the score table.

We decided to create two indexes to speed up our queries. One index called "stop_spatial_index" on the stops dataset's "geom" column. Another is called "fid_index" on the birth dataset's "fid" column. We decided to use indexes on these two columns because the stops dataset has around 100,000 rows and the birth dataset has around 2,000 rows. We used the "fid" column to make the index because birth's primary key is sa2_code so it would be useful. Similar logic is applied for stops in the dataset.

Score Analysis

To compute a score for how “well - resourced” each individual neighbourhood is according to the following formula, where S is the sigmoid function, z is the normalised z-score for each dataset and “young people” is defined as anyone aged 0-19.

$$\text{Score} = S(z_{\text{retail}} + z_{\text{health}} + z_{\text{stops}} + z_{\text{polls}} + z_{\text{schools}})$$

Z_retail & Z_health

```
# calculation for z_retail
# get Retail businesses per 1000 people

conn.execute("""
DROP TABLE IF EXISTS retail_count;
CREATE TABLE retail_count (
    sa2_code INTEGER PRIMARY KEY,
    num_of_business INTEGER,
    num_of_ppl INTEGER
);""")
)
```

```
<sqlalchemy.engine.cursor.LegacyCursorResult at 0x7feeae7c24f0>
```

```
sql = """
INSERT INTO retail_count (sa2_code, num_of_business, num_of_ppl)

SELECT businesses.sa2_code as sa2_code,
    total_businesses as num_of_business,
    total_people as num_of_ppl
FROM businesses
JOIN population ON businesses.sa2_code = population.sa2_code

WHERE industry_code = 'G'
returning sa2_code, num_of_business, num_of_ppl
"""

query(conn, sql)
```

	sa2_code	num_of_business	num_of_ppl
0	102011028	45	7530
1	102011029	50	11052
2	102011030	57	4748
3	102011031	152	14803
4	102011032	180	21346
...	...	...	...
363	128021537	0	45
364	128021538	124	23369
365	128021607	68	17379
366	128021608	29	7354
367	128021609	9	3551

368 rows × 3 columns

To calculate the z-score of the number of retail businesses per 1000 people in each area, as the code shown on the left, we first create a table called “retail_count” to store the data we are using (sa2 code, number of business, number of people). Then we add our data set into the table and merge the business’s sa2 code with the population one to find the common area they have.

```
JOIN population ON businesses.sa2_code = population.sa2_code
```

Because we only focus on the retail industry right now, we only select the data that has an industry code G. The output table is shown on the right.

After forming a table, we can now calculate the number of retail industries per person in that area by dividing num_of_business by num_of_ppl. And use that value to calculate the z-score for retail using the z-score formula, $z = (x - \mu)/\sigma$, where x is the raw score, μ is the mean and σ is the standard deviation.

```
sql = """
SELECT
    sa2_code,
    num_of_business/num_of_ppl as num_of_retail_per_ppl,
    (num_of_business/num_of_ppl - mean) / standard_deviation AS z_score
FROM
    retail_count,
    (SELECT AVG(num_of_business/num_of_ppl) AS mean,
           STDDEV(num_of_business/num_of_ppl) AS standard_deviation
    FROM retail_count) AS stats

GROUP BY sa2_code, num_of_retail_per_ppl, mean, standard_deviation
"""

query(conn,sql)
```

	sa2_code	num_of_retail_per_ppl	z_score
0	126021498	0	-0.083771
1	127031732	0	-0.083771
2	121031408	0	-0.083771
3	115021298	0	-0.083771
4	125041717	0	-0.083771
...	...	...	...
363	117021328	0	-0.083771
364	125041589	0	-0.083771
365	116011304	0	-0.083771
366	115041625	0	-0.083771
367	127011595	0	-0.083771

```
sql = """
UPDATE z_scores
SET z_retail = (num_of_business/num_of_ppl - mean) / standard_deviation
FROM
    retail_count,
    (SELECT AVG(num_of_business/num_of_ppl) AS mean,
           STDDEV(num_of_business/num_of_ppl) AS standard_deviation
    FROM retail_count) AS stats

WHERE z_scores.sa2_code::INTEGER = retail_count.sa2_code
returning z_scores.sa2_code, z_retail

"""

query(conn,sql)
```

	sa2_code	z_retail
0	102011028	-0.08377127110529276092
1	102011029	-0.08377127110529276092
2	102011030	-0.08377127110529276092
3	102011031	-0.08377127110529276092
4	102011032	-0.08377127110529276092
...	...	...
363	128021537	-0.08377127110529276092
364	128021538	-0.08377127110529276092
365	128021607	-0.08377127110529276092
366	128021608	-0.08377127110529276092
367	128021609	-0.08377127110529276092

368 rows × 2 columns

By getting the z-score, we store the result into the table called “scores” to make it clear to see all the output at the end.

At the same time, we can get the z-score for the health industry with the same step as we did for the retail industry.

Z_stops

In order to calculate the z-score for the number of public transport stops in each region. We created a table called “stops_count” this time to store the dataset. And we use the COUNT() function to get the number of stops in each region. Below is the code we wrote and the table we created.

```
# calculation for z_stops
# get Public transport stops

conn.execute("""
DROP TABLE IF EXISTS stops_count;
CREATE TABLE stops_count (
    sa2_code INTEGER PRIMARY KEY,
    num_of_stops INTEGER
);"""
)
```

```
<sqlalchemy.engine.cursor.LegacyCursorResult at 0x7feee4f2be50>
```

```
sql = """
INSERT INTO stops_count (sa2_code, num_of_stops)

SELECT sa2_region.sa2_code::INTEGER AS sa2_code,
    COUNT(stops.stop_id) AS num_of_stops
FROM sa2_region
JOIN stops ON ST_Contains(sa2_region.geom, stops.geom)

GROUP BY sa2_region.sa2_code
returning sa2_code, num_of_stops
"""
query(conn, sql)
```

	sa2_code	num_of_stops
0	101021007	427
1	101021008	86
2	101021009	107
3	101021010	88
4	101021012	168
...	...	...
735	801091102	12
736	801091105	2
737	801091106	2
738	801091108	4
739	801091109	18

After we get the number of stops, we can now calculate the z-score using the formula. And update the score table with the new z_score_stops we found.

```
sql = """
SELECT
    sa2_code,
    num_of_stops,
    (num_of_stops - mean) / standard_deviation AS z_score
FROM
    stops_count,
    (SELECT AVG(num_of_stops) AS mean,
           STDDEV(num_of_stops) AS standard_deviation
    FROM stops_count) AS stats

GROUP BY sa2_code, num_of_stops, mean, standard_deviation
"""
query(conn,sql)
```

	sa2_code	num_of_stops	z_score
0	113031270	225	0.401159
1	115011553	104	-0.291284
2	123031447	222	0.383991
3	127011505	206	0.292428
4	119031666	113	-0.239780
...	...	...	...
735	105031105	309	0.881863
736	117031336	60	-0.543081
737	115011294	148	-0.039486
738	105021098	24	-0.749097
739	120031681	149	-0.033764

Z_polls

To calculate the z-score for the number of polling places in each region, we did exactly the same step as the stops. And update the z_score_polls into the score table to help us see the results clearly.

Z_schools

In order to calculate the z-score of how many schools there are per young people. We do the same steps as we did for the retail industry and health. However, there is a slight difference between the number of people, instead of using the number of people, we will need to use the number of young people this time.

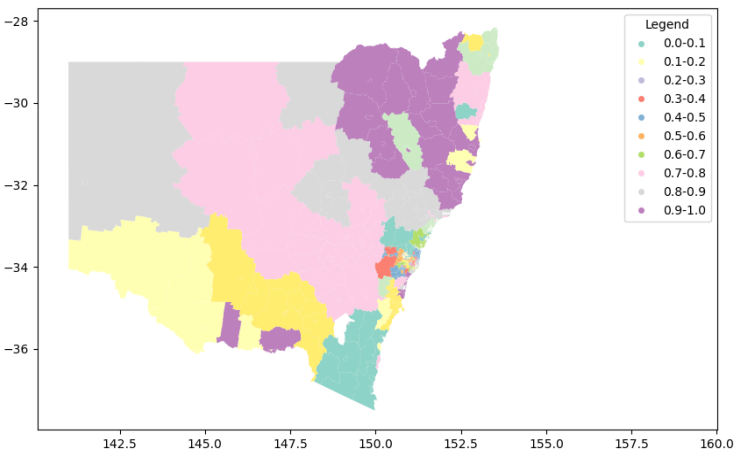
After getting each z-score of the given dataset, we will use the sigmoid function to find out the score we need by adding the z-scores together. The sigmoid function we use is the following, where x is the sum of z-scores.

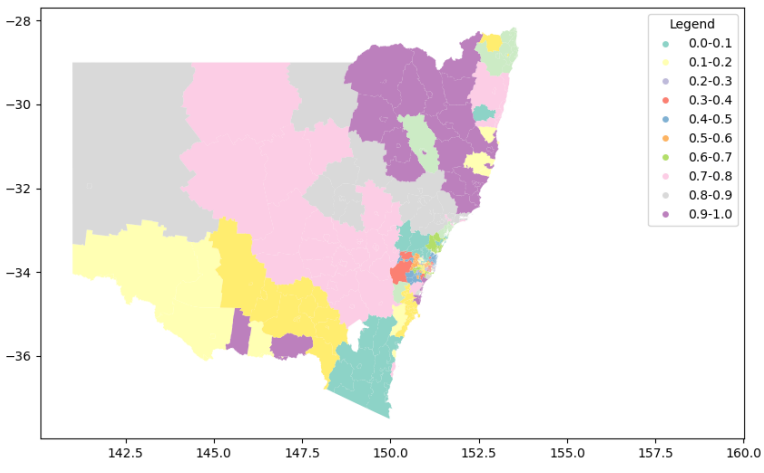
$$S(x) = \frac{1}{1+e^{-x}} = \frac{e^x}{e^x+1} = 1 - S(-x).$$

We then write the following code to calculate the score using the sigmoid function.

```
sql = """
UPDATE score
SET score_new =
    1 / (1 +
        EXP(-(COALESCE(CASE WHEN z_retail = 'none' THEN NULL ELSE z_retail::float END, 0) +
        COALESCE(CASE WHEN z_health = 'none' THEN NULL ELSE z_health::float END, 0) +
        COALESCE(CASE WHEN z_stops = 'none' THEN NULL ELSE z_stops::float END, 0) +
        COALESCE(CASE WHEN z_polls = 'none' THEN NULL ELSE z_polls::float END, 0) +
        COALESCE(CASE WHEN z_schools = 'none' THEN NULL ELSE z_schools::float END, 0) +
        COALESCE(CASE WHEN z_bike = 'none' THEN NULL ELSE z_bike::float END, 0) +
        COALESCE(CASE WHEN z_birth = 'none' THEN NULL ELSE z_birth::float END, 0)
        )))

    FROM z_scores

WHERE z_scores.sa2_code = score.sa2_code

returning score.sa2_code, score_og, score_new
"""
query(conn, sql)
```

Furthermore, while we add our new dataset into the sigmoid function, the only difference is the sum of z-scores. We add these two new z-scores, z_bike and z_birth, and we got to calculate the new sum of z-scores and input it into the same sigmoid function as the code shown above. The output will be the new score that includes the extension dataset.





We got this graph by filtering for New South Wales state. From the graphs, we can see that northern parts have higher sigmoid scores while southern parts have very low sigmoid scores. One reason could be that there isn't enough data from southern parts. These two graphs are actually generated by using 2 different columns. The left map uses the score of sigmoid function over z-scores of the given datasets. The right map uses the score of sigmoid function of z-scores of given+additional datasets. We can see that there is no striking difference between the two. This could be because the bike data doesn't have data from many area codes in NSW. However, since birth data have sufficient data, we can see that adding birth data z-score in doesn't impact the sigmoid score.

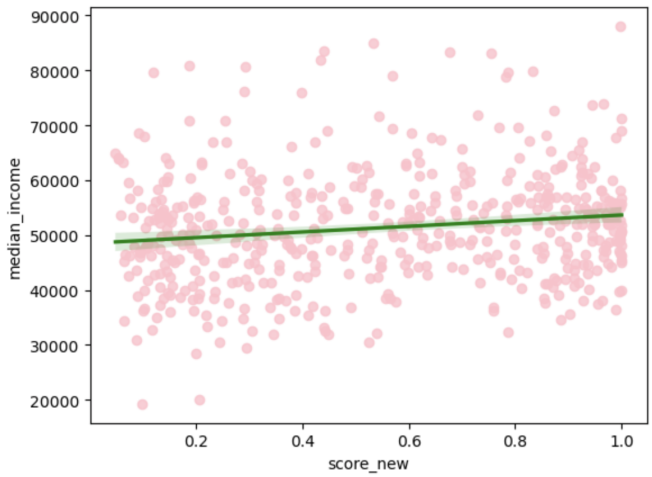
Correlation Analysis

The correlation coefficient is a statistical measure of the strength of a linear relationship between two variables, this time score and the median income of each region. Its value can range from -1 to 1, where -1 presents a perfect negative relationship between the values and 1 presents a strong positive relationship. In this case, we are interested in exploring the correlation between score and the median income of each region.

To determine the correlation between these two values, we first created a new dataframe that merged from scores table and Income.csv to get the most important part in together. Then we use Pandas Corr() function to get the correlation coefficient between median income and score. The output is 0.08514 suggests a very weak positive correlation. This means that there is no relationship between the median income and score because the correlation is too weak. Below is the code that visualises the regression line using regplot.

```
# use regplot
sb.regplot(x = 'score_new', y = 'median_income',data = merged_score,
           scatter_kws={'color': 'pink'}, line_kws={'color': 'green'})
```

<Axes: xlabel='score_new', ylabel='median_income'>



As we can see, the plot in the graph is too dispersed to the regression line. This also shows out the weak relationship between our values. This means how "well-resourced" each individual neighbourhood is does not impact the median income of each region.

The usefulness of our scores may be limited in predicting or explaining median income variations across SA2 regions and there are many other socioeconomic factors such as employment rates and regional characteristics that may play more substantial roles in determining the income levels.

Citations:

Australian Urban Research Infrastructure Network (AURIN) and The University of Melbourne. Created April 27, 2023, from https://data.aurin.org.au/dataset/au-govt-abs-abs-births-sa2-2010-2020-sa2-2016

Transport of NSW (2020, November 9). *City of Sydney Bicycle Parking*. Open Data. Retrieved May 20, 2023, from https://opendata.transport.nsw.gov.au/dataset/city-sydney-bicycle-parking