THE UNIVERSITY OF SYDNEY

DATA1002 Group Assignment Stage 1

Group

470000946 Xiaobing Wang

470021332 Shiyu Hu

Student Disclaimer

The work comprising this report is substantially our own, and to the extent that any part of this work is not my own I have indicated that it is not my own by acknowledging the source of that part or those parts of the work. I have read and understood the University of Sydney Student Plagiarism: Coursework Policy and Procedure. I understand that failure to comply with the University of Sydney Student Plagiarism: Coursework Policy and Procedure can lead to the University commencing proceedings against me for potential student misconduct under chapter 8 of the University of Sydney By-Law 1999 (as amended).

Section 1 Description of data sets

The aim of this report is to investigate the number of Covid-19 confirmed cases in different local government areas. Other attributes that can be used to group by are the population density, the proportion of the elderly, and month of the date.

We selected data from three different sources. The dataset is obtained from official and trustworthy source such as ABS and NSW data.

1.1 data sources and the rights associated with the data

Raw Dataset 1， Population estimates by Local Government Area, 2018 to 2019, accessed from https://www.abs.gov.au/statistics/people/population/regional-population/latest-release#data-downloads-geopackages

Raw Dataset 2， Population estimates by age and sex - summary statistics, by LGA, 2019, accessed from https://www.abs.gov.au/statistics/people/population/regional-population-age-and-sex/latest-release#data-download

Raw Dataset 3, NSW COVID-19 tests by location and result, accessed from https://data.nsw.gov.au/data/dataset/covid-19-cases-by-location

Both dataset 1 and dataset 2 authorized by The Australian Bureau of Statistics, a famous official Statistics organization of Australia. Dataset 3 is authorized by Data.NSW, a famous official Statistics organization supported by NSW government. These data are open to free use and gives us trusted official statistics for comprehensive research and use of these data follows the NSW Government Open Data Policy defined in the Government Information (Public Access) Act 2009 (NSW) (GIPA Act).

1.2 the format/contents of the data

- D1_population_by_lga.csv

This dataset contains estimates of the resident population of Local Government Areas of Australia for 30 June 2018 and 30 June 2019, according to the 2019 edition of the Australian Statistical Geography Standard (ASGS). Estimates are revised for 2018 and preliminary for 2019.

This dataset contains 8565 values and provides population density of various Local Government Areas in Australia. The formats of the data are shown as below, together with the first 5 rows displayed.

	LGA code	Local Government Area	2018 no.	2019 no.	...	Unnamed: 11	Area in KM2	Population Density 2019	State
0	20110.0	Alpine (S)	12730.0	12814.0	...	NaN	4788.2	2.7	VIC
1	20260.0	Ararat (RC)	11795.0	11845.0	...	NaN	4211.1	2.8	VIC
2	20570.0	Ballarat (C)	107324.0	109505.0	...	NaN	739.0	148.2	VIC
3	20660.0	Banyule (C)	130250.0	131631.0	...	NaN	62.5	2104.7	VIC
4	20740.0	Bass Coast (S)	35326.0	36320.0	...	NaN	865.8	41.9	VIC

Attribute	Description	Type and form
LGA code	Local Government Area code	String – digits
LGA Government Area	Local Government Area Name	String
2018 no.	2018 population number	String – digits
2019 no.	2019 population number	String – digits
Population Change	Population Change between 2018&2019	String – digits
Change in %	Population Change in %	String – digits
Natural Increase	Population Natural Increase	String – digits
Net Internal Migration	Population Net Internal Migration	String – digits
Net Overseas migration	Population Net Overseas migration	String – digits
Area in KM2	Local Government Area in km^ 2	String – digits
Population Density 2019	Population Density 2019	String – digits
State	State name in short form	String

- D2_Age_Summary_by_lga.csv

This dataset contains preliminary estimates of the resident population by age and sex as at 30 June 2019. Data are provided for Local Government Areas (LGAs) of Australia, according to the 2019 edition of the Australian Statistical Geography Standard (ASGS). This dataset contains 8190 values and provides age summary of various Local Government Areas in Australia.

The formats of the data are shown as below, together with the first 5 rows displayed.

	S/T code	S/T name	...	People aged 15–64 years in %	People aged 65 years and over in %
0	1.0	New South Wales	...	62.5	18.1
1	1.0	New South Wales	...	65.4	17.0
2	1.0	New South Wales	...	57.5	25.6
3	1.0	New South Wales	...	61.6	17.9
4	1.0	New South Wales	...	64.2	16.3

Attribute	Description	Type and form
S/T code	Suburb & Town	String – digits
S/T name	Suburb & Town Name	String
LGA code	Local Government Area code	String – digits
LGA Name	Local Government Area Name	String
Males no.	Males number	String – digits
Females no.	Females number	String – digits
Persons no.	Total number	String – digits
Sex ratio males per 100 females	Sex ratio males per 100 females	String – digits
Median age in years	Median age in years	String – digits
People aged 0-14 years in %	People age between 0-14 number in total	String – digits
People aged 15-64 years in %	People aged between 15-64 number in total	String – digits
People aged 65 years and over in %	People aged 65 years and over in total	String - digits

- D3_confirmed_cases_by_lga_postcode.csv

This dataset is about COVID-19 confirmed cases in New South Wales – COVID-19 data including notification date and postcode, local health district, and local government area. The dataset is updated daily, except on weekends.

This dataset contains 25182 values and provides age summary of various Local Government Areas in Australia. The formats of the data are shown as below, together with the first 5 rows displayed.

	notification_date	postcode	lhd_2010_code	lhd_2010_name	lga_code19	lga_name19
0	2020-01-25	2134.0	X700	Sydney	11300.0	Burwood (A)
1	2020-01-25	2121.0	X760	Northern Sydney	16260.0	Parramatta (C)
2	2020-01-25	2071.0	X760	Northern Sydney	14500.0	Ku-ring-gai (A)
3	2020-01-27	2033.0	X720	South Eastern Sydney	16550.0	Randwick (C)
4	2020-03-01	2163.0	X710	South Western Sydney	12850.0	Fairfield (C)

Attribute	Description	Type and form
notification_date	The date of notification	String – year/month/day
postcode	Suburb & Town postcode	String - dights
lhd_2010_code	Local health district code	String - digits
lhd_2010_name	Local health district name	String
lga_code19	Local Government Area code	String - digits
lga_name19	Local Government Area name	String

1.3 some comments on any strengths or limitations of the dataset

The official datasets are accurate and useful for research, adding credibility to the conclusion of the study. However, come of the test cases are having unknown sources, limiting the researchers to fully investigate every confirm cases and group by local government areas. In some research, these unknown cases are ignored and rows of these unknown cases are excluded to make the number of cases grouped by area more reliable and sensible.

Due to time constraint, the datasets does not include more factors that may be important in influencing number of cases in certain area, such as access to health service, number of commuters and medical care awareness. With more database to further improve the depth of this project, more meaningful insights can be drawn to help the general public to know more about the pandemic.

Section 2 Data transformation

2.1 CHECK AND REMOVE NULL VALUES, FILL IN MISSING VALUES

Code are used to check and remove null values for datasets and ensure there is no null values for dataset 1 and 2

```
# Check and remove null values for dataset 1, State value cannot be null and density cannot be 0
density_df = density_df[density_df['State'] == 'NSW']
density_df = density_df[density_df['Population Density'] != 0]
density_df.dropna()
print("Check if dataset 1 have nulls", density_df.isnull().any())
```

```
# Check and remove null values for dataset 1, State value cannot be null and density cannot be 0
aged_df = aged_df[aged_df['State'] == 'NSW']
aged_df.dropna()
print("Check if dataset 2 have nulls", aged_df.isnull().any())
```

Code are used to remove null values for dataset 3. NA values are filled as ‘Unknown’ to represent confirm cases with unknown sources and code is used to ensure there is no null values for dataset 3.

```
cases_df.dropna(how = 'any')
# fill in missing values and transform data formats
cases_df["Post Code"].replace({NA: "Unknown"}, inplace=True)
cases_df["LGA code"].replace({NA: "Unknown"}, inplace=True)
cases_df['Local Government Area'].replace({NA: "Unknown"}, inplace=True)
print("Check if dataset 3 have nulls",cases_df.isnull().any())
```

Code are used to replace NA values with meaningful strings. Rows with numeric attributes as NaN are removed. Code is used to explicitly show that no null value is present.

```
# fill in missing values for areas in NSW but without state value
result_df['State'].replace({NA: "NSW"}, inplace=True)
result_df["Post Code"].replace({NA: "Unknown"}, inplace=True)
result_df["LGA code"].replace({NA: "Unknown"}, inplace=True)
result_df["Date"].replace({NA: "Unknown"}, inplace=True)
# Remove null values for computation
result_df = result_df[result_df['Population Density'] != NA]
result_df = result_df[result_df['Percentage of The Aged'] != NA]
```

```
# Checking for quality verification
print("Check if result dataset have nulls after filling missing values", result_df.isnull().any())
```

2.2 TRANSFORM DATA FORMATS

Datasets with different column names for the same attribute are being modified to have consistent column names. While dataset 2 use for form for state 'New South Wales' while dataset 1 uses 'NSW', dataset 2 is modified to be consistent with dataset 1.

```
# Unify names as different datasets may have different column names for the same attribute
# Unify representations while dataset 2 use for form for state 'New South Wales' while dataset 1 uses 'NSW'
aged_df.columns = ['State', 'LGA code', 'Local Government Area', 'Percentage of The Aged']
# Transform data formats
aged_df["State"].replace({"New South Wales": "NSW"}, inplace=True)
```

Numeric attributes are converted to floats from string of digits before being saved into the result dataset.

```
# Transform data formats
result_df['Population Density'] = result_df['Population Density'].astype(float)
result_df['Percentage of The Aged'] = result_df['Percentage of The Aged'].astype(float)
```

2.3 MERGE DATASET AND REMOVE DUPLICATES

Datasets are merged using the attribute the Local Government Area. This attribute is checked to be consistent in all datasets. Duplicates are removed.

```
# Merge dataset by Local Government Area
cases_density_df = pd.merge(cases_df, density_df, on = 'Local Government Area', how = 'outer')
result_df = pd.merge(cases_density_df, aged_df, on = 'Local Government Area', how = 'outer')
```

```
# Remove duplicates if there is any
result_df = result_df.iloc[:,[0,1,8,3,4,6,9]]
result_df.columns = ['Date', 'Post Code', 'LGA code', 'Local Government Area','State', 'Population Density', 'Percentage
print("Check if result dataset have nulls in LGA", result_df['Local Government Area'].isnull().any())
```

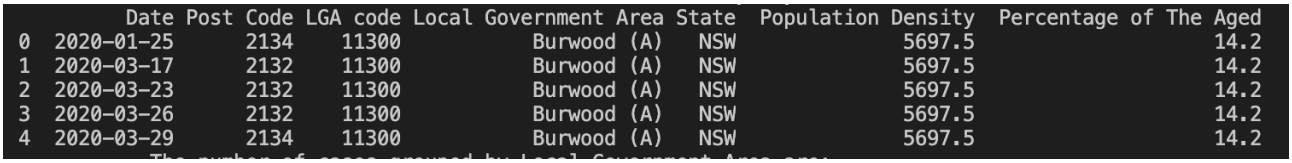
2.5 Automated checking for quality verification

Beside checking done after each dataset is cleaned and transformed. The final clean dataset is checked again to ensure there is no null values and all values are meaningful for further processing.

```
# Remove duplicates if there is any
result_df = result_df.iloc[:,[0,1,8,3,4,6,9]]
result_df.columns = ['Date', 'Post Code', 'LGA code', 'Local Government Area','State', 'Population Density', 'Percentage
print("Check if result dataset have nulls in LGA", result_df['Local Government Area'].isnull().any())
# fill in missing values for areas in NSW but without state value
result_df['State'].replace({NA: "NSW"}, inplace=True)
result_df["Post Code"].replace({NA: "Unknown"}, inplace=True)
result_df["LGA code"].replace({NA: "Unknown"}, inplace=True)
result_df["Date"].replace({NA: "Unknown"}, inplace=True)
# Remove null values for computation
result_df = result_df[result_df['Population Density'] != NA]
result_df = result_df[result_df['Percentage of The Aged'] != NA]
# Transform data formats
result_df['Population Density'] = result_df['Population Density'].astype(float)
result_df['Percentage of The Aged'] = result_df['Percentage of The Aged'].astype(float)
# Checking for quality verification
print("Check if result dataset have nulls after filling missing values", result_df.isnull().any())
```

2.6 Metadata of the merged dataset (这个应该和 clean data 在一起)

This clean dataset provides confirmed cases of various Local Government Areas in Australia, together with population density, percentage of the elderly statistics. The formats of the data are shown as below, together with the first 5 rows displayed.



	Date	Post Code	LGA code	Local Government Area	State	Population Density	Percentage of The Aged
0	2020-01-25	2134	11300	Burwood (A)	NSW	5697.5	14.2
1	2020-03-17	2132	11300	Burwood (A)	NSW	5697.5	14.2
2	2020-03-23	2132	11300	Burwood (A)	NSW	5697.5	14.2
3	2020-03-26	2132	11300	Burwood (A)	NSW	5697.5	14.2
4	2020-03-29	2134	11300	Burwood (A)	NSW	5697.5	14.2

Attribute	Description	Type and form
Date	The date of notification	String – year/month/day
postcode	Suburb & Town postcode	String of digits
LGA code	Local Government Area code	String of digits
LGA Government Area	Local Government Area Name	String
State	State in short form	String
Population Density	Population divided by area	float
Percentage of The Aged	Percentage of population over 65	float

Section 3 Analysis of the data

3.1 number of cases, grouped by Local Government Area

The result is analysed and the number of cases are grouped by Local Government Area.

```
81   # number of cases, Group by Local Government Area
82   counts_by_LGA = result_df.groupby(by = 'Local Government Area').size().to_dict()
83   print("========== The number of cases grouped by Local Government Area are: ==========")
84   for k in counts_by_LGA:
85       print(k, counts_by_LGA[k])
86   print()
```

```
========== The number of cases grouped by Local Government Area are: ==========
Albury (C) 11
Armidale Regional (A) 4
Ballina (A) 8
Balranald (A) 1
Bathurst Regional (A) 9
Bayside (A) 78
Bega Valley (A) 9
Bellingen (A) 1
Berrigan (A) 5
Blacktown (C) 186
Bland (A) 1
Blayney (A) 3
Blue Mountains (C) 38
Bogan (A) 1
Bourke (A) 1
Brewarrina (A) 1
Broken Hill (C) 2
Burwood (A) 14
Byron (A) 18
Cabonne (A) 5
Camden (A) 69
Campbelltown (C) (NSW) 84
Canada Bay (A) 40
Canterbury-Bankstown (A) 169
Carrathool (A) 1
```

The highest number of cases is shown after grouping counts by Local Government Area, excluding unknown cases.

```
The Local Government Area with highest number of cases is:
Waverley (A) 205
```

3.2 Average number of cases, Grouped by population density and Local Government Area

Population Density is categorized into 3 groups, excluding unknown cases. Cases are calcaluated and grouped by population density. The averages number of cases of each group are shown.

```
101  # Average number of cases, which are Grouped by density and Local Government Area
102  # denstity is splitted into 3 groups, excluding unknown cases
103  highestD = df['Population Density'].max()
104  lowestD = df['Population Density'].min()
105  crowed_density = lowestD + (highestD - lowestD)*0.6667
106  safe_density = lowestD + (highestD - lowestD)*0.3333
107  cases_in_most_crowed = df[df['Population Density'] > crowed_density].groupby(by = 'Local Government Area').size().mean()
108  cases_in_crowed = df[df['Population Density'] <= crowed_density]
109  cases_in_moderately_crowed = cases_in_crowed[cases_in_crowed['Population Density'] >= safe_density].groupby(by = 'Local
110  cases_in_less_crowed_area = cases_in_crowed[cases_in_crowed['Population Density'] < safe_density].groupby(by = 'Local Go
111  print('========== Average number of cases, grouped by density and Local Government Area are: ==========')
112  print('The most crowed area have an average cases of',cases_in_most_crowed)
113  print('The moderately crowed area have an average cases of',cases_in_moderately_crowed)
114  print('The less crowed area have an average cases of',cases_in_less_crowed_area)
115  print()
```

```
========== Average number of cases, grouped by density and Local Government Area are: ==========
The most crowed area have an average cases of 149.33333333333334
The moderately crowed area have an average cases of 74.92857142857143
The less crowed area have an average cases of 21.836363636363636
```

3.3 Average number of cases, are Grouped by percentage of the olderly

Local Government Area that needs special care using Number of cases and aging population percentage

Aging percentage is spitted into 3 groups, excluding unknown cases. The averages number of cases of each group are shown.

```
# Local Government Area that needs special care using Number of cases and aging population percentage
# Average number of cases, are Grouped by percentage of the olderly and Local Government Area
# Aging percentage is splitted into 3 groups, excluding unknown cases
highestP = df['Percentage of The Aged'].max()
lowestP = df['Percentage of The Aged'].min()
dangerous = lowestP + (highestP - lowestP)*0.75
relatively_dangerous = lowestP + (highestD - lowestD)*0.5
relatively_safe = lowestP + (highestD - lowestD)*0.25

cases_in_danger_by_LGA = df[df['Percentage of The Aged'] > dangerous].groupby(by = 'Local Government Area').size().to_di
print('The areas that needs special are to the elderly are:')
for k in cases_in_danger_by_LGA:
    print(k, cases_in_danger_by_LGA[k])
print()
cases_in_q1 = df[df['Percentage of The Aged'] > dangerous].groupby(by = 'Local Government Area').size().mean()
cases_in_q234 = df[df['Percentage of The Aged'] <= crowed_density]
cases_in_q2 = cases_in_q234[cases_in_q234['Population Density'] > relatively_dangerous].groupby(by = 'Local Government A
cases_in_q34 = result_df[result_df['Percentage of The Aged'] <= relatively_dangerous]
cases_in_q3 = cases_in_q34[cases_in_q34['Population Density'] > relatively_safe].groupby(by = 'Local Government Area').s
cases_in_q4 = df[df['Percentage of The Aged'] < relatively_safe].groupby(by = 'Local Government Area').size().mean()
print('========== Average number of cases, grouped by percentage of the elderly and Local Government Area are: =========
print('The areas with most aging population have an average cases of', cases_in_q1)
print('The areas with moderately aging population have an average cases of', cases_in_q2)
print('The areas with moderately less aging population have an average cases of', cases_in_q3)
print('The areas with less aging population have an average cases of', cases_in_q4)
print()
```

```
========== Average number of cases, grouped by percentage of the elderly and Local Government Area are: ==========
The areas with most aging population have an average cases of 10.1875
The areas with moderately aging population have an average cases of 99.0
The areas with moderately less aging population have an average cases of 85.10526315789474
The areas with less aging population have an average cases of 30.24031007751938
```

3.4 Number of cases, Grouped by month

```python
# Month with highest Number of cases, using Group by month, including unknown cases
# number of cases, Group by month
cases_df = result_df[result_df['Date'] != 'Unknown']
month_df = cases_df.iloc[:,[0,3]]
month_df['Date'] = cases_df['Date'].apply(lambda x : x.split('-')[1])
counts_by_month = month_df.groupby('Date').size().to_dict()
print("========== The number of cases grouped by month are: ==========")
for k in counts_by_month:
    print(k, counts_by_month[k])

largest = None
month_with_most_cases = None
for month in counts_by_month:
    if largest is None or largest < counts_by_month[month]:
        month_with_most_cases = month
        largest = counts_by_month[month]
print("========== The month with highest number of cases is: ===========")
print('Month', month_with_most_cases, 'with number of cases as',largest)
print()
```

```
========== The number of cases grouped by month are: ==========
01 4
03 2148
04 867
05 66
06 112
07 388
08 291
09 163
10 158
```

Highest number is also shown.

```
========== The month with highest number of cases is: ===========
Month 03 with number of cases as 2148
```