# Task 1

## (1)

### (a)

Two new data quality problems that have emerged since the publication of the paper are:

1. Handling Unstructured and Semi-Structured Data: With the rise of big data, unstructured (e.g., text, images, videos) and semi-structured data (e.g., JSON, XML) have become more popular. Cleaning and standardizing these data types pose significant challenges, as traditional methods designed for structured data often fail to address issues like inconsistent formatting, embedded metadata, and contextual relevance.

2. Data Privacy and Compliance: As regulations like GDPR(General Data Protection Regulation) have been published, ensuring data quality now includes maintaining privacy and compliance. This introduces challenges in anonymizing personal data while preserving its utility. Inaccuracies or over-cleaning could lead to the loss of critical information, making it harder to ensure compliance without compromising data quality.

### (b)

Yes, many problems raised in the paper are still relevant today.

Those issues stems from the enduring nature of data quality challenges. Despite technological advancements, the **core problems** of data inconsistencies, duplicates, missing values, and integration across heterogeneous sources remain fundamental to ensuring data integrity. These issues are intrinsic to any data management process, whether dealing with traditional databases or modern big data environments.

The **ETL process**, central to the paper, is still crucial for transforming and loading data into warehouses or other storage systems. Even with the advent of real-time processing and advanced tools, ETL remains a vital step where data quality issues must be addressed.

Moreover, the **limitations of tools** and the need for **manual intervention**, as highlighted in the paper, continue to be a challenge. While automation has improved, complex data scenarios often require human expertise, especially when dealing with domain-specific data or ambiguous cases.

Additionally, the expansion of **data types** (e.g., unstructured and semi-structured data) and the growth in **data volume** have magnified these problems, making the paper’s insights even more pertinent today. Thus, the foundational issues and solutions discussed remain relevant as they underpin the challenges faced in modern data environments.

## (2)

When cleaning and integrating the latest Australian census data with previous years’ datasets for decision-making about Aboriginal and Torres Strait Islanders' access to health care, three key data wrangling aspects to consider are:

1. Data Quality and Consistency: Ensuring high data quality across all datasets is critical. This involves identifying and correcting errors like duplicate records, missing values, and inconsistencies that may arise due to variations in data collection methods or human error. For example, changes in the way health conditions or ethnic identities are recorded over different census years could lead to discrepancies. Ensuring consistency in data definitions, formats, and units of measurement across different years is essential for meaningful analysis and decision-making.

2. Data Integration and Schema Matching: Integrating datasets from multiple census years requires careful schema matching. This involves aligning different data structures, such as column names, data types, and categories, to create a unified dataset. For instance, if the way Aboriginal and Torres Strait Islander status is coded has changed over the years, these differences must be reconciled. Effective schema matching helps in combining data without losing context or meaning, ensuring that trends over time are accurately captured.

3. Ethical Considerations and Sensitivity: Handling sensitive data about Aboriginal and Torres Strait Islander communities requires strict attention to ethical considerations. It is vital to ensure that the data is anonymized where necessary and that privacy is protected throughout the data wrangling process. Additionally, care must be taken to avoid introducing bias or misrepresenting these communities. The data should be processed and presented in a way that supports equitable access to healthcare and respects the cultural significance of the information.

# Task 2

L= [25, 11, 40, 17, 17, 41, 21, 31, 46, 26, 86, 74, 100, 28, 15, 97];

L final=[25, 11, 40, 17, 17, 41, 21, 31, 46, 26, 86, 74, 100, 28, 15, 97, 75, 68, 82, 3]

## （1）

Mean=45.15;

Standard deviation =30.25;

## (2)

Median = 35.5;

median absolute deviation=19.5;

## (3)

Mode=17；

## （4）

The dataset is likely positively skewed (right-skewed) because the mode (17) is less than the median (35.5), which is generally less than the mean (which would likely be higher given the presence of large values like 86, 97, and 100). This pattern typically indicates a distribution with a long tail on the right side.

# Task 3

## （1）

Bin 1: [ 19.00 , 19.00, 19.00 , 19.00, 19.00 , 19.00, 19.00 , 19.00 , 19.00 , 19.00 ];

Bin 2: [74.50, 74.50, 74.50, 74.50, 74.50, 74.50, 74.50, 74.50, 74.50, 74.50];

## （2）

Bin 1: [19.40, 19.40, 19.40, 19.40, 19.40, 19.40, 19.40, 19.40, 19.40, 19.40]

Bin 2: [42.33, 42.33, 42.33]

Bin 3: [83.14, 83.14, 83.14, 83.14, 83.14, 83.14, 83.14]

## (3)

Bin 1: [3.00,3.00,3.00,26.00, 26.00, 26.00, 26.00 , 26.00]

Bin 2: [28.00 , 28.00, 46.00, 46.00, 46.00]

Bin 3: [75.00 , 75.00, 75.00]

Bin 4: [82.00, 82.00 ,100.00,100.00]

## (4)

Bin 1: [3.00 ,17.00 ,17.00,17.00,17.00]

Bin 2: [21.00 ,21.00 ,21.00 ,31.00 ,31.00 ]

Bin 3: [40.00 ,40.00 ,40.00 ,74.00 ,74.00 ]

Bin 4: [75.00 ,75.00 ,75.00 ,100.00 ,100.00 ]

# Task 4

## (1)

postcode phone email

6771 1 1 1 0

4573 1 0 1 1

2809 1 1 0 1

1874 1 0 0 2

1641 0 1 1 1

1145 0 0 1 2

692 0 1 0 2

495 0 0 0 3

3973 8087 5870 17930

## （2）

### （a）

To be check???

Spearman function

bmi age\_at\_consultation

bmi 1.00 0.27

age\_at\_consultation 0.27 1.00

Given that BMI and age at consultation are measurements that might not have a straightforward linear relationship (since BMI could increase with age up to a certain point and then decrease), Spearman correlation might be a better choice because it does not assume linearity and is more robust to outliers.

### （b）

Cramér's V

Correlation between state and valid marital status : 0.02

This function is ideal for determining the strength of association between two categorical variables (like state and marital status).

## (3)

### (a)

问问能否上传 代码

Completeness for **'middle name**': 90.18%

Steps:

# Calculate completeness for 'middle name'

total\_values\_middle\_name = df['middle\_name'].count() # Total non-missing values

total\_possible\_values\_middle\_name = len(df) # Total values

completeness\_middle\_name =

total\_values\_middle\_name / total\_possible\_values\_middle\_name

Completeness for **'email'**: 70.65%

Steps:

# Calculate completeness for 'email'

total\_values\_email = df['email'].count() # Total non-missing values

total\_possible\_values\_email = len(df) # Total values

completeness\_email = total\_values\_email / total\_possible\_values\_email

### (b)

Validity for **'weight'**: 90.09%

For **weight**: Values should be numeric and within a reasonable range (e.g., 0 to 500 kg).

# define a function to check the validity for 'weight'

def is\_valid\_weight(weight):

return pd.notnull(weight) and (0 <= weight <= 500)

valid\_weight = df['weight'].apply(is\_valid\_weight)

valid\_weight\_count = valid\_weight.sum() #the number of valid weight data

total\_weight\_count = df['weight'].notnull().sum() # Only consider non-empty weights

validity\_weight = valid\_weight\_count / total\_weight\_count

//----------

Validity for **'email'**: 84.86%

For **email**: Values should not be empty and must contain the '@' symbol.

# define a function to check the validity for 'email'

def is\_valid\_email(email):

return pd.notnull(email) and '@' in email

# get all valid email data by implementing the method above

valid\_email = df['email'].apply(is\_valid\_email)

valid\_email\_count = valid\_email.sum()

total\_email\_count = df['email'].notnull().sum() # Only consider non-empty emails

validity\_email = valid\_email\_count / total\_email\_count

### (c)

Uniqueness for **'first\_name'**: 18.09%

# Total number of values (including duplicates)

total\_values\_first\_name = len(df['first\_name'])

# Number of unique values

unique\_values\_first\_name = df['first\_name'].nunique()

#result

uniqueness\_first\_name = unique\_values\_first\_name / total\_values\_first\_name

### (d)

问问日期， 24点05 该如何处理

Consistency between 'age\_at\_consultation' and 'birth date': 46.31%

Steps:

import pandas as pd

from datetime import datetime

# Load the dataset

df = pd.read\_csv('data\_wrangling\_medical\_2024\_u7568823.csv')

# Convert 'birth\_date' to datetime format using the correct format

df['birth\_date'] = pd.to\_datetime(df['birth\_date'], format='%d/%m/%Y')

# Convert 'consultation\_timestamp' to datetime format

df['consultation\_timestamp'] = pd.to\_datetime(df['consultation\_timestamp'], format='%Y-%m-%dT%H:%M%z', errors='coerce')

# Drop rows where 'consultation\_timestamp' is NaT due to parsing errors

df = df.dropna(subset=['consultation\_timestamp'])

# Extract the consultation date (ignoring the time part)

df['consultation\_date'] = df['consultation\_timestamp'].dt.date

# Calculate the expected age at consultation

df['calculated\_age'] = df['consultation\_date'].apply(lambda x: x.year) - df['birth\_date'].dt.year - (

((df['consultation\_date'].apply(lambda x: x.month) < df['birth\_date'].dt.month) |

((df['consultation\_date'].apply(lambda x: x.month) == df['birth\_date'].dt.month) &

(df['consultation\_date'].apply(lambda x: x.day) < df['birth\_date'].dt.day)))

)

# Validate both the calculated age and the recorded age at consultation

df['valid\_calculated\_age'] = df['calculated\_age'] < 150

df['valid\_recorded\_age'] = df['age\_at\_consultation'] < 150

# Only consider rows with valid ages in both fields

valid\_rows = df[df['valid\_calculated\_age'] & df['valid\_recorded\_age'] & df['age\_at\_consultation'].notnull()]

# Define a function to check consistency

def is\_consistent\_age(row):

return row['calculated\_age'] == row['age\_at\_consultation']

# Apply the consistency check function to each row with valid 'age at consultation'

consistent\_age = valid\_rows.apply(is\_consistent\_age, axis=1)

# Calculate the consistency ratio

consistency\_ratio = consistent\_age.mean()

## (4)

### (a)

First Digit Distribution for Cholesterol Level (%):

cholesterol\_level

1 63.8

2 28.0

3 0.8

4 0.4

5 0.5

6 0.9

7 1.3

8 1.8

9 2.6

The cholesterol level data does not follow Benford's law. The distribution shows a strong skew towards the first digits "1" and "2", which are much more frequent than Benford's law would suggest.

### (b)

First Digit Distribution for Blood Pressure (%):

blood\_pressure

5 0.0

6 8.8

7 78.9

8 12.3

9 0.0

The blood pressure data does not follow Benford's law. Benford's law applies to naturally occurring, unrestricted datasets across many orders of magnitude, which is not the case for blood pressure data that falls within a medically relevant range.

### (c)

First Digit Distribution for Medicare Number (%):

medicare\_number

1 11.2

2 11.0

3 11.0

4 11.0

5 11.2

6 11.3

7 11.2

8 11.2

9 11.0

The medicare number data does not follow Benford's law. The observed distribution is nearly uniform across all digits from 1 to 9, indicating that each digit has roughly the same probability of appearing as the first digit.

### (d)

Two common data warehousing operations that can be applied are drill-down and slice.

1. Drill-Down Operation

Description: The drill-down operation allows for a more detailed view of the data by navigating from less detailed data to more detailed data. It reduces the level of aggregation.

Dimension Applied: Consultation Time (Year)

Example: I can perform a drill-down operation on the consultation time dimension from the year level to the month level. Initially, I may have aggregated data showing the number of consultations for each disease type per state per year. After applying the drill-down, I can see the data broken down into each month within the year.

- Result Example: If the original data cube shows 500 cases of hereditary diseases in New South Wales for the year 2020, the drill-down operation could reveal that 200 cases occurred in January, 150 in February, and 150 in March.

2. Slice Operation

Description: The slice operation selects a single dimension from the cube, producing a sub-cube by fixing a specific value of one dimension.

Dimension Applied: Disease Type

Example: I can apply a slice operation on the disease type dimension to isolate data related to infectious diseases only. This operation would create a sub-cube that contains consultation counts across different states and years, but only for infectious diseases.

- Result Example: The resulting sub-cube might show that in Victoria, there were 300 consultations for infectious diseases in 2019, 400 in 2020, and 350 in 2021.