FR. CONCEICAO RODRIGUES COLLEGE OF ENGINEERING

Department of Computer Engineering

Course, Subject & Experiment Details

| | |
|---|---|
| Practical No: | 3 |
| Title: | Data Cleaning and Storage- Pre-process, filter and store social media data for business (Using Python, MongoDB, R, etc). |
| Name of the Student: | Warren Fernandes |
| Roll No: | 8940 |
| Date of Performance: | 07/02/2023 |
| Date of Submission: | 14/02/2023 |

Evaluation:

| Sr. No. | Rubric | Grade |
|---|---|---|
| 1 | On time submission/completion (2) | |
| 2 | Preparedness (2) | |
| 3 | Skill (4) | |
| 4 | Output (2) | |

Signature of the Teacher

## Data Cleaning and Storage

Preprocess, filter and store social media data for business (Using Python, MongoDB, R, etc).

```
import pandas as pd
df = pd.read_csv('data_youtube.csv')

df.info(verbose=True)

df.head()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2500 entries, 0 to 2499
Data columns (total 12 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   id              2500 non-null   int64
 1   author          2500 non-null   object
 2   description     2500 non-null   object
 3   guid            2500 non-null   object
 4   to              0 non-null      float64
 5   likecount       2500 non-null   int64
 6   link            2500 non-null   object
 7   pubdate         2500 non-null   object
 8   replycount      2500 non-null   int64
 9   title           2500 non-null   object
 10  authorChannelUrl  2500 non-null   object
 11  Unnamed: 11     0 non-null      float64
dtypes: float64(2), int64(3), object(7)
memory usage: 234.5+ KB
```

| | id | author | description | guid | to | likecount | l: |
|---|---|---|---|---|---|---|---|
| **0** | 1 | Jc ʓ | Engineering give attendance 🤣 | UgxO3eyUaBuL18DwJFp4AaABAg | NaN | 0 | https://www.youtube.com/wat v=Fw1Fc_y_2Ek& |
| **1** | 2 | Kaustubh Ramteke | *be an engineer first then decide what to do ... | UgzB9BfavPwJpKcznON4AaABAg | NaN | 1 | https://www.youtube.com/wat v=Fw1Fc_y_2Ek& |
| **2** | 3 | listen something different | Best talk ever Now. | UgyXaBkz3xCtE09mJlF4AaABAg | NaN | 0 | https://www.youtube.com/wat v=Fw1Fc_y_2Ek& |
| **3** | 4 | Lohith P gowda | India is affected by British education system | UgwRrJO7WpP_8fS3W3l4AaABAg | NaN | 0 | https://www.youtube.com/wat v=Fw1Fc_y_2Ek& |
| **4** | 5 | bit coin | He is real hero I watched first video which is... | Ugw8NKYM9XuMG0NT59t4AaABAg | NaN | 0 | https://www.youtube.com/wat v=Fw1Fc_y_2Ek& |

```
numeric_cols = df.select_dtypes(include=['number']).columns
print(numeric_cols)

non_numeric_cols = df.select_dtypes(exclude=['number']).columns
print(non_numeric_cols)
```

```
Index(['id', 'to', 'likecount', 'replycount', 'Unnamed: 11'], dtype='object')
Index(['author', 'description', 'guid', 'link', 'pubdate', 'title',
       'authorChannelUrl'],
      dtype='object')
```

## Method 1: missing data (by columns) count & percentage

This is the most basic method to detect missing data among columns.

The info method that we've used earlier includes this information. For example, we print out the summary of all the non-numeric columns below. Note that we are not printing for the entire DataFrame df since there are too many columns.

```
df[non_numeric_cols].info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2500 entries, 0 to 2499
Data columns (total 7 columns):
 #   Column            Non-Null Count  Dtype 
---  ------            --------------  ----- 
 0   author            2500 non-null   object
 1   description       2500 non-null   object
 2   guid              2500 non-null   object
 3   link              2500 non-null   object
 4   pubdate           2500 non-null   object
 5   title             2500 non-null   object
 6   authorChannelUrl  2500 non-null   object
dtypes: object(7)
memory usage: 136.8+ KB
```

```
num_missing = df.isna().sum()
num_missing[:10]
```

```
id                0
author            0
description       0
guid              0
to             2500
likecount         0
link              0
pubdate           0
replycount        0
title             0
dtype: int64
```

```
df.isna().mean()
```

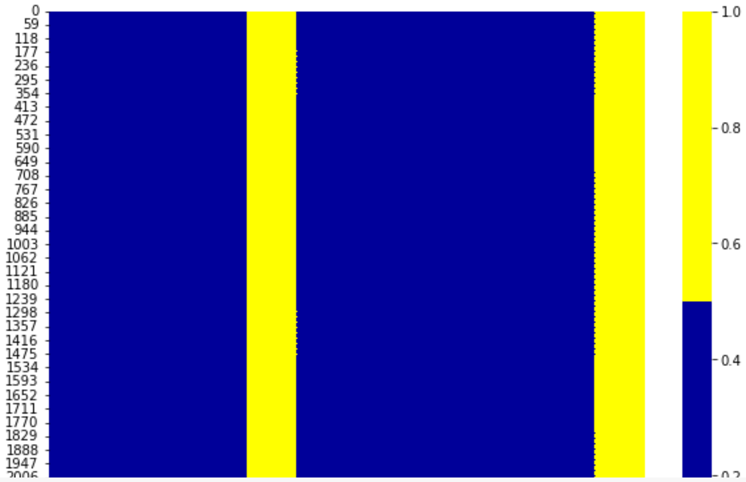
```
id                  0.0
author              0.0
description         0.0
guid                0.0
to                  1.0
likecount           0.0
link                0.0
pubdate             0.0
replycount          0.0
title               0.0
authorChannelUrl    0.0
Unnamed: 11         1.0
dtype: float64
```

## Method 2: missing data (by columns) heatmap

Sometimes a picture could be worth a thousand words. We can build a heatmap to visualize the missing data. This technique is proper when you have a smaller number of columns.

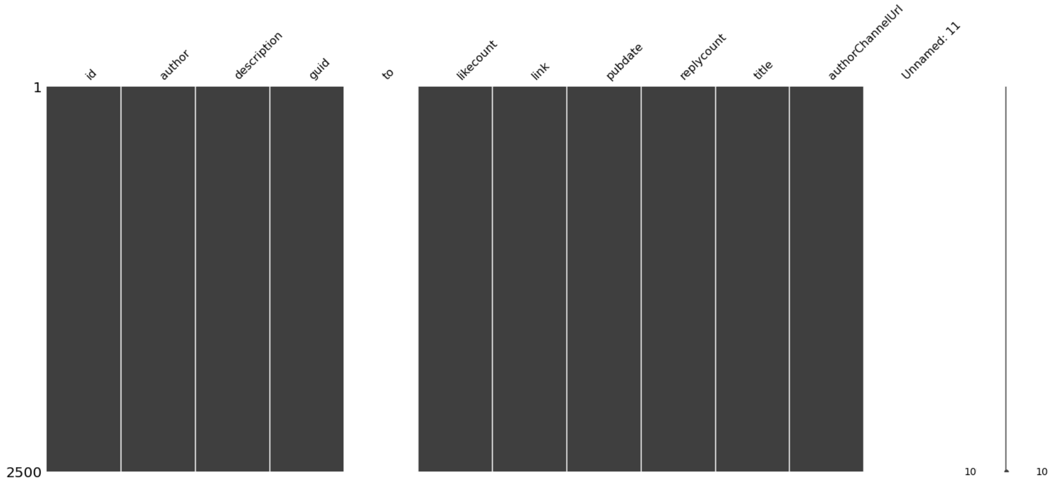
```
import seaborn as sns
import matplotlib.pyplot as plt

plt.figure(figsize=(10,8))

cols = df.columns[:30]
colours = ['#000099', '#ffff00'] # specify colours: yellow - missing. blue - not missing
sns.heatmap(df[cols].isna(), cmap=sns.color_palette(colours))
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fc7878a98b0>



```
import missingno as msno
msno.matrix(df.iloc[:, :30])
```

<matplotlib.axes._subplots.AxesSubplot at 0x7fc786f8c160>

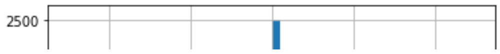



## Method 3: missing data (by rows) histogram

We've been looking at missing data by columns. But we can also summarize the missing data by rows. Missing data histogram is a technique for summarizing such information.

```
missing_by_row = df.isna().sum(axis='columns')
missing_by_row.hist(bins=50)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc786f07af0>
```




## Technique 1: drop columns / features

This technique is straightforward. We drop the entire column or feature with missing data, which will certainly cause a loss of information. So we should only perform this when we are sure that the missing data is not informative. Otherwise, we should consider other solutions.

```
pct_missing = df.isna().mean()
pct_missing[pct_missing > .3]
```

```
to             1.0
Unnamed: 11    1.0
dtype: float64
```

```
df_less_missing_cols = df.loc[:, pct_missing <= .3].copy() # equivalent to df.drop(columns=pct_missing[pct_missing > .3].index)
df_less_missing_cols.shape
```

```
(2500, 10)
```

## Technique 2: drop rows / observations

We can drop the entire row with missing data like the first technique. Again, please be aware of the loss of information when removing rows.

```
df_less_missing_rows = df[missing_by_row < 35].copy()
df_less_missing_rows.shape # equivalent to df.dropna(axis='index', thresh=292-35+1).shape
```

```
(2500, 12)
```

## Technique 3: impute the missing with constant values

Instead of dropping data, we can also replace the missing. An easy method is to impute the missing with constant values.

```
df_copy = df.copy()
df_copy[numeric_cols] = df_copy[numeric_cols].fillna(-999)
df_copy[non_numeric_cols] = df_copy[non_numeric_cols].fillna('_MISSING_')
df_copy.head()
```

| | id | author | description | guid | to | likecount | |
|---|---|---|---|---|---|---|---|
| 0 | 1 | Jc ʒ | Engineering give attendance 🤣 | UgxO3eyUaBuL18DwJFp4AaABAg | -999.0 | 0 | https://www.youtube.com/wa v=Fw1Fc_y_2Ek |
| 1 | 2 | Kaustubh Ramteke | *be an engineer first then decide what to do ... | UgzB9BfavPwJpKcznON4AaABAg | -999.0 | 1 | https://www.youtube.com/wa v=Fw1Fc_y_2Ek |
| 2 | 3 | listen something different | Best talk ever Now. | UgyXaBkz3xCtE09mJlF4AaABAg | -999.0 | 0 | https://www.youtube.com/wa v=Fw1Fc_y_2Ek |
| 3 | 4 | Lohith P gowda | India is affected by British education system | UgwRrJO7WpP_8fS3W3l4AaABAg | -999.0 | 0 | https://www.youtube.com/wa v=Fw1Fc_y_2Ek |
| 4 | 5 | bit coin | He is real hero I watched first video which is... | Ugw8NKYM9XuMG0NT59t4AaABAg | -999.0 | 0 | https://www.youtube.com/wa v=Fw1Fc_y_2Ek |

## Technique 4: impute the missing with statistics

Besides constants, we can also impute the missing values with statistics.

```
df_copy = df.copy()
med = df_copy[numeric_cols].median()
df_copy[numeric_cols] = df_copy[numeric_cols].fillna(med)
df_copy.head()
```

| | id | author | description | guid | to | likecount | l: |
|---|---|---|---|---|---|---|---|
| **0** | 1 | Ͻc ʒ | Engineering give attendance 🤣 | UgxO3eyUaBuL18DwJFp4AaABAg | NaN | 0 | https://www.youtube.com/wat v=Fw1Fc_y_2Ek& |
| **1** | 2 | Kaustubh Ramteke | *be an engineer first then decide what to do ... | UgzB9BfavPwJpKcznON4AaABAg | NaN | 1 | https://www.youtube.com/wat v=Fw1Fc_y_2Ek& |
| **2** | 3 | listen something different | Best talk ever Now. | UgyXaBkz3xCtE09mJlF4AaABAg | NaN | 0 | https://www.youtube.com/wat v=Fw1Fc_y_2Ek& |
| **3** | 4 | Lohith P gowda | India is affected by British education system | UgwRrJO7WpP_8fS3W3l4AaABAg | NaN | 0 | https://www.youtube.com/wat v=Fw1Fc_y_2Ek& |
| **4** | 5 | bit coin | He is real hero I watched first video which is... | Ugw8NKYM9XuMG0NT59t4AaABAg | NaN | 0 | https://www.youtube.com/wat v=Fw1Fc_y_2Ek& |

```
most_freq = df_copy[non_numeric_cols].describe().loc['top']
most_freq
```

```
author                                           Kunal krishann
description                                                True
guid                                 UgxO3eyUaBuL18DwJFp4AaABAg
link              https://www.youtube.com/watch?v=Fw1Fc_y_2Ek&lc...
pubdate                                     2023-01-21 05:49:45
title                                                      True
authorChannelUrl  http://www.youtube.com/channel/UCu4CMjBTOwBpf6...
Name: top, dtype: object
```

```
df_copy[non_numeric_cols] = df_copy[non_numeric_cols].fillna(most_freq)
```

## Irregular data (outliers)

Outliers are data that is distinct from other observations. They could bias our data analysis results, providing a misleading representation of the data. Outliers could be real outliers or mistakes.

## Method 1: descriptive statistics

First, let's look at kurtosis. Kurtosis is a statistical measure of 'tailedness'. The higher kurtosis is often linked to the greater extremity of deviations (or outliers) in the data. So this is a single statistic to detect potential outliers.

```
df = pd.read_csv('data_youtube.csv')
df.kurt(numeric_only=True)[:10]
```

```
id              -1.200000
to                    NaN
likecount      393.934705
replycount     173.721757
Unnamed: 11           NaN
dtype: float64
```

```
df['likecount'].describe()
```

```
count     2500.000000
mean        46.872800
std        443.505308
min          0.000000
25%          0.000000
```

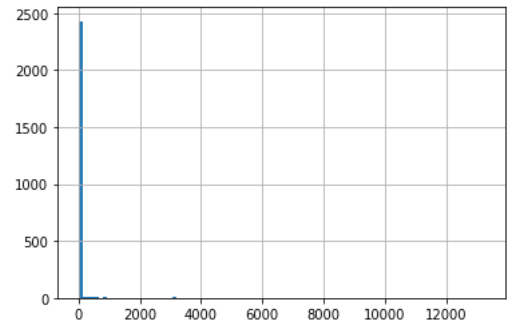
```
50%          0.000000
75%          1.000000
max      13274.000000
Name: likecount, dtype: float64
```

## Method 2: histogram & box plot

Let’s use the data visualization method to detect outliers. We’ll plot a histogram and a box plot of the column likecount.

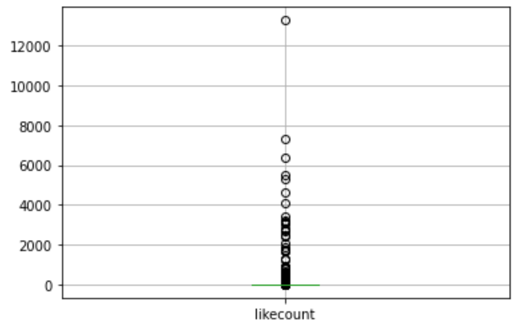
```
df['likecount'].hist(bins=100)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc786e3fac0>
```



```
df.boxplot(column=['likecount'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc786abf970>
```

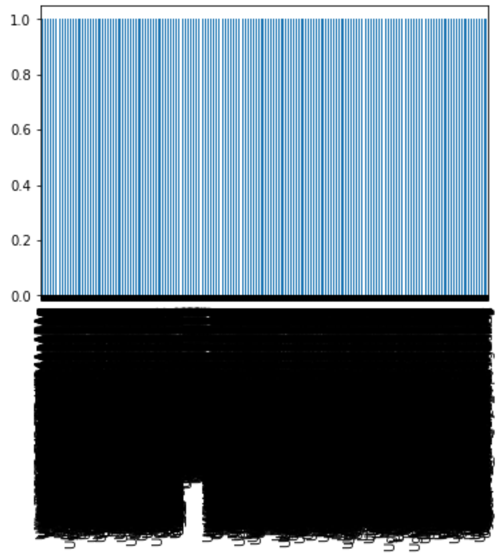


## Method 3: bar chart

As mentioned, outliers are mainly defined for numeric data. But for non-numeric data, there could be irregular values too. We can use a bar chart to learn about the categories and their distributions.

```
df['guid'].value_counts().plot(kind='bar')
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc786e51100>
```



## Unnecessary data

Those are a lot of hard work for missing data and outliers! Let's clean something more straightforward in this section: the unnecessary data.

## Unnecessary type 1: repetitive & uninformative

One column can have many observations being the same value. When an extremely high percentage of the column has a repetitive value, we should investigate whether such a column provides valuable information.

```
num_rows = len(df)

for col in df.columns:
    cnts = df[col].value_counts(dropna=False)
    top_pct = (cnts/num_rows).iloc[0]

    if top_pct > 0.999:
        print('{0}: {1:.2f}%'.format(col, top_pct*100))
        print(cnts)
        print()
```

```
to: 100.00%
NaN    2500
Name: to, dtype: int64

Unnamed: 11: 100.00%
NaN    2500
Name: Unnamed: 11, dtype: int64
```

## Unnecessary type 2: irrelevant

Again, the data needs to provide valuable information for the project. If the features are not related to the question we are trying to solve, they are irrelevant.

### How to find out?

We need to skim through the features to identify irrelevant ones. For example, a feature recording the temperature in the US wouldn't provide direct insights into housing prices in Russia.

### What to do?

When the features are not serving the project's goal, we can remove them. You could use the drop method in pandas.

## Unnecessary type 3: duplicates

The duplicate data is when copies of the same observation exist. Let's look at 2 main types of duplicate data and clean them in Python.

## Duplicates type 1: all columns based

How to find out? This is easy to understand. Such duplicate occurs when all the columns' values within the observations are the same.

```
df[df.duplicated()]
```

| id | author | description | guid | to | likecount | link | pubdate | replycount | title | authorChannelUrl | Unna |
|---|---|---|---|---|---|---|---|---|---|---|---|

```
df.drop_duplicates()
```

| | id | author | description | guid | to | likecount | |
|---|---|---|---|---|---|---|---|
| **0** | 1 | Jc ʒ | Engineering give attendance 🤣 | UgxO3eyUaBuL18DwJFp4AaABAg | NaN | 0 | https://www.youtube v=Fw1F |
| **1** | 2 | Kaustubh Ramteke | *be an engineer first then decide what to do ... | UgzB9BfavPwJpKcznON4AaABAg | NaN | 1 | https://www.youtube v=Fw1F |
| **2** | 3 | listen something different | Best talk ever Now. | UgyXaBkz3xCtE09mJlF4AaABAg | NaN | 0 | https://www.youtube v=Fw1F |
| **3** | 4 | Lohith P gowda | India is affected by British education system | UgwRrJO7WpP_8fS3W3l4AaABAg | NaN | 0 | https://www.youtube v=Fw1F |
| **4** | 5 | bit coin | He is real hero I watched first video which is... | Ugw8NKYM9XuMG0NT59t4AaABAg | NaN | 0 | https://www.youtube v=Fw1F |
| **...** | ... | ... | ... | ... | ... | ... | |
| **2495** | 2496 | Mihir J. | sad ! | UgjXZi33yEQlj3gCoAEC | NaN | 0 | https://www.youtube v=Fw1F |
| **2496** | 2497 | Naman Sharma | Pause at 2:43 with the auto subtitles on. He s... | UghttxK2CaxHQHgCoAEC | NaN | 0 | https://www.youtube v=Fw1F |
| **2497** | 2498 | TimeWalker | this person was throwing out the truth like a ... | Ugia1fJgy5BPqHgCoAEC | NaN | 1 | https://www.youtube v=Fw1F |

## Duplicates type 2: key columns based

Instead of looking at all columns, sometimes we want to detect duplicates based on a set of identifiers (columns).

system...

```
df[df.drop(columns=['author']).duplicated()]
```

| id | author | description | guid | to | likecount | link | pubdate | replycount | title | authorChannelUrl | Unna |
|---|---|---|---|---|---|---|---|---|---|---|---|

kar lo bont...

```
df_dedupped = df.drop(columns=['author']).drop_duplicates()

print(df.shape)
print(df_dedupped.shape)
```

```
(2500, 12)
(2500, 11)
```

## Inconsistent data

It is crucial to have the dataset follow specific standards. There could be different inconsistent data that needs to be cleaned and we'll cover 4 common ones. Please note that the actual data may be even messier, be creative when cleaning it!

## Inconsistent type 1: capitalization

Inconsistent use of upper and lower cases in categorical values is typical. We need to clean it since Python is case-sensitive.

```
df['description'].value_counts(dropna=False)
```

```
True
4
Arjun Kapoor
3
👍
3
```

```
Dead audience
3
Amazing
3

..
He lived in saudi arabia
1
Look man i have heard these talks a lot of time but there has to be a replaced model
1
Rather than blaming the system, we should look into our mistakes first.
1
india has more unemployment becoz they follow Mark\'s but not there curiosity
1
Parents be like.. beta engineering kar lo boht scope hai. And when their kids are not able to get employment in their relevant
fields then "Banking hai na"    1
Name: description, Length: 2480, dtype: int64
```

```
df['sub_area_lower'] = df['description'].str.lower()
df['sub_area_lower'].value_counts(dropna=False)
```

```
amazing
4
true
4
dead audience
4
good
3
arjun kapoor
3

..
who is here after the new education policy ?modi2024
1
let\'s be honest here, we didn\'t get this in our recommended, we hate our system so much we searched this up.
1
he lived in saudi arabia
1
look man i have heard these talks a lot of time but there has to be a replaced model
1
parents be like.. beta engineering kar lo boht scope hai. and when their kids are not able to get employment in their relevant
fields then "banking hai na"    1
Name: sub_area_lower, Length: 2471, dtype: int64
```

## Inconsistent type 2: data types

Another standardization we often need to look at is the data types.

```
df['pubdate']
```

```
0       2023-01-21 05:49:45
1       2023-01-20 07:58:08
2       2023-01-09 07:06:41
3       2023-01-04 13:26:01
4       2023-01-03 21:40:20
                ...
2495    2017-04-08 16:41:58
2496    2017-04-08 16:05:03
2497    2017-04-08 14:43:17
2498    2017-04-08 11:20:28
2499    2017-04-07 14:11:27
Name: pubdate, Length: 2500, dtype: object
```

```
df['pubdate_dt'] = pd.to_datetime(df['pubdate'], format='%Y-%m-%d')
df['year'] = df['pubdate_dt'].dt.year
df['month'] = df['pubdate_dt'].dt.month
df['weekday'] = df['pubdate_dt'].dt.weekday

df[['pubdate_dt', 'year', 'month', 'weekday']].head()
```

| | pubdate_dt | year | month | weekday |
|---|---|---|---|---|
| **0** | 2023-01-21 05:49:45 | 2023 | 1 | 5 |
| **1** | 2023-01-20 07:58:08 | 2023 | 1 | 4 |
| **2** | 2023-01-09 07:06:41 | 2023 | 1 | 0 |
| **3** | 2023-01-04 13:26:01 | 2023 | 1 | 2 |
| **4** | 2023-01-03 21:40:20 | 2023 | 1 | 1 |

## Inconsistent type 3: typos of categorical values

A categorical column takes on a limited and usually fixed number of possible values. Sometimes it shows other values due to reasons like typos.

```
df_city_ex = pd.DataFrame(data={'city': ['torontoo', 'toronto', 'tronto', 'vancouver', 'vancover', 'vancouvr', 'montreal', 'calgary']})

cities = ['toronto', 'vancouver', 'montreal', 'calgary']
from nltk.metrics import edit_distance
for city in cities:
    df_city_ex[f'city_distance_{city}'] = df_city_ex['city'].map(lambda x: edit_distance(x, city))

df_city_ex
```

| | city | city_distance_toronto | city_distance_vancouver | city_distance_montreal | city_distance_cal |
|---|---|---|---|---|---|
| **0** | torontoo | 1 | 8 | 7 | |
| **1** | toronto | 0 | 8 | 7 | |
| **2** | tronto | 1 | 8 | 6 | |
| **3** | vancouver | 8 | 0 | 8 | |
| **4** | vancover | 7 | 1 | 7 | |
| **5** | vancouvr | 7 | 1 | 7 | |
| **6** | montreal | 7 | 8 | 0 | |
| **7** | calgary | 7 | 8 | 8 | |

```
msk = df_city_ex['city_distance_toronto'] <= 2
df_city_ex.loc[msk, 'city'] = 'toronto'

msk = df_city_ex['city_distance_vancouver'] <= 2
df_city_ex.loc[msk, 'city'] = 'vancouver'

df_city_ex
```

| | city | city_distance_toronto | city_distance_vancouver | city_distance_montreal | city_distance_cal |
|---|---|---|---|---|---|
| **0** | toronto | 1 | 8 | 7 | |
| **1** | toronto | 0 | 8 | 7 | |
| **2** | toronto | 1 | 8 | 6 | |
| **3** | vancouver | 8 | 0 | 8 | |
| **4** | vancouver | 7 | 1 | 7 | |
| **5** | vancouver | 7 | 1 | 7 | |
| **6** | montreal | 7 | 8 | 0 | |
| **7** | calgary | 7 | 8 | 8 | |

## Inconsistent type 4: addresses

This is the last data cleaning in Python problem we'll cover. If you've worked with addresses, you know how messy they can be. Just imagine how people can write addresses in all different ways!

```
df_add_ex = pd.DataFrame(['123 MAIN St Apartment 15', '123 Main Street Apt 12   ', '543 FirSt Av', '  876 FIRst Ave.'], columns=['address
df_add_ex
```

| | address |
|---|---|
| **0** | 123 MAIN St Apartment 15 |
| **1** | 123 Main Street Apt 12 |
| **2** | 543 FirSt Av |
| **3** | 876 FIRst Ave. |

```
df_add_ex['address_std'] = df_add_ex['address'].str.lower()
df_add_ex['address_std'] = df_add_ex['address_std'].str.strip() # remove leading and trailing whitespaces.
df_add_ex['address_std'] = df_add_ex['address_std'].str.replace('\\.', '', regex=True) # remove period.
```

```
df_add_ex['address_std'] = df_add_ex['address_std'].str.replace('\\bstreet\\b', 'st', regex=True) # replace street with st.
df_add_ex['address_std'] = df_add_ex['address_std'].str.replace('\\bapartment\\b', 'apt', regex=True) # replace apartment with apt.
df_add_ex['address_std'] = df_add_ex['address_std'].str.replace('\\bav\\b', 'ave', regex=True) # replace av with ave.

df_add_ex
```

| | address | address_std |
|---|---|---|
| **0** | 123 MAIN St Apartment 15 | 123 main st apt 15 |
| **1** | 123 Main Street Apt 12 | 123 main st apt 12 |
| **2** | 543 FirSt Av | 543 first ave |
| **3** | 876 FIRst Ave. | 876 first ave |

## Conclusion

In conclusion, data cleaning and storage are crucial steps in the process of preprocessing, filtering and storing social media data for business purposes. It involves removing irrelevant, duplicated, or inconsistent information and transforming data into a structured format that can be easily analyzed and utilized. Effective data cleaning and storage strategies ensure that the data collected from social media platforms is reliable and accurate, allowing businesses to make informed decisions based on the insights they gain.

It is important to invest in tools and techniques that automate the process of data cleaning and storage, as this can greatly reduce the time and effort required to prepare the data for analysis. Additionally, regular updates to data storage systems and the implementation of backup and recovery plans can help ensure that businesses have access to the data they need even in the event of a system failure or data loss.

In conclusion, data cleaning and storage play a critical role in ensuring that social media data is useful and valuable for businesses. By following best practices and utilizing the right tools, businesses can unlock the full potential of social media data to drive growth and success.