1. Initial data load

Load the initial data “as-is”, and review it:

- Understand the structure (types)
- MAke sure you understand all the columns

```python
pd.read_csv()
df.info()
df.describe()
```

2. Organize data

Understand the changes you want to make to the data structure to achieve convenience of use + memory reduction (if needed):

- Rename columns
- Convert types
  - reduce numeric types size
  - set categories instead of strings
  - Dates
- Delete columns
- Convert values if needed
- Sort the data if needed
- Set index if needed
- Change numerical columns into bins

```python
# remember to use endless amount of params to load the data as you want
pd.read_csv()

# rename
df.rename(columns={'CITY_NAME':'city'})

# type conversion (or use params of pd.read_csv)
df['Age'].astype(np.int8)
df['season'].astype('category')
pd.to_datetime(df['date'])
```

```python
# delete columns
df.drop(columns=["unused_column_name"])

# sort
df.sort_index()
df.sort_values('name')

# indexing
df.set_index('id')
df.drop_index()

# Numeric to bins (categories)
pd.cut(df['Age'], bins=3)
```

3. Merge / Concat

If your data consists of multiple files, there might be a need to concatenate and/or merge the data

Python

```python
# concat
pd.concat([fruits, vegetables], axis='index')
pd.concat([fruits, vegetables], axis='columns')

# merge
inner_join = pd.merge(movies, actors, on='title', how='inner')
left_join = pd.merge(movies, actors, on='title', how='left')
right_join = pd.merge(movies, actors, on='title', how='right')
outer_join = pd.merge(movies, actors, on='title', how='outer')
```

4. Handle NaNs

Check whether there are missing values in your dataset, and decide what to do with them. The options are:

- Remove the rows with missing data
- Impute missing values.
- When filling NaNs in numeric values, make sure not to change initial distribution!!!

Python

```python
df.isna().any()
df.isna().sum()

df.dropna()
df.dropna(subset=['Weight', 'Height', 'Age'])

df.fillna({'Medal': 'No medal', 'Weight': df['Weight'].mean()})

df.groupby(['event', 'male', 'age_bin'])['height'].transform(lambda x:
x.fillna(x.median())).plot(kind='hist', bins=50)
```

5. Handle Duplicates

Check whether there are duplicates in the data and decide what to do with them. Make sure to understand which combination of columns is ok to be duplicated, and which is not.

Python

```python
# check duplicates
df.duplicated().sum()
df.duplicated(keep=False)

# drop duplicates
df.drop_duplicates(inplace=True)
```

6. Save cleaned data for future use

It is a good practice to store the cleaned data so next time you won’t need to clean it again

```python
df.to_csv()
```

7. Add external data that might be relevant

Find APIs or other data sources with information that might provide more interesting insights to your original data