**Project title:**   Covid-19 Cases Analysis

**Phase 3:** Development

**Part 1**

In this part you will begin building your project by loading and preprocessing the dataset.

Begin conducting the Covid-19 Cases analysis by collecting and preprocessing the data.

Collect and preprocess the Covid-19 Cases data for analysis.

**Data Preprocessing:**

* Data preprocessing is a crucial step within the statistics analysis and gadget gaining knowledge of pipeline.
* It includes a sequence of strategies and operations finished on uncooked statistics to clean, organize, and transform it right into a layout that is suitable for analysis or device mastering version schooling.
* Data preprocessing goals to enhance the first-class of the records, making it greater reliable and conducive to generating accurate consequences.

Here are some common tasks and techniques involved in data preprocessing:

**Data Cleaning:**

* Handling missing values: Deciding how to deal with missing data, whether by imputing values or removing incomplete records.
* Outlier detection and treatment: Identifying and handling data points that significantly deviate from the norm.

**Noise reduction:**

* Smoothing noisy data through techniques like filtering.

**Data Transformation:**

* **Data normalization:** Scaling numerical features to a standard range (e.g., between 0 and 1) to ensure that they have similar influence in the analysis.
* **Encoding categorical variables:** Converting categorical data into numerical format, such as one-hot encoding or label encoding.
* **Feature engineering:** Creating new features or modifying existing ones to capture more meaningful information from the data.
* **Dimensionality reduction:** Reducing the number of features while retaining essential information, using methods like Principal Component Analysis (PCA).

**Data Integration:**

* **Merging or joining datasets:** Combining data from multiple sources into a single dataset for analysis.

**Aggregation:** Summarizing data at a higher level of granularity, such as aggregating daily sales into monthly totals.

**Data Reduction:**

* **Sampling:** Reducing the size of a large dataset by randomly selecting a representative subset.
* **Binning:** Grouping continuous data into discrete bins to simplify analysis.
* **Filtering:** Selecting a subset of data based on specific criteria.

**Data Standardization:**

* Ensuring that data follows a consistent format and structure.
* Date and time format conversion: Converting date and time data into a uniform format.
* Currency conversion: Converting monetary values into a common currency.

**Data Scaling:**

* Scaling numerical data to a common range to prevent some features from dominating the analysis.

Data preprocessing is an iterative process that may involve several of these steps in various orders, depending on the specific dataset and the analysis goals. Proper data preprocessing is essential for improving the accuracy and effectiveness of machine learning models, as well as for making data more accessible for traditional statistical analysis.

Here is the data preprocessing codes along with the output of the given dataset:

**Importing the libraries:**

Import three basic libraries which are very common in machine learning and will be used every time you train a model

* **NumPy:** it is a library that allows us to work with arrays and as most machine learning models work on arrays NumPy makes it easier
* **matplotlib:** this library helps in plotting graphs and charts, which are very useful while showing the result of your model
* **Pandas:** pandas allows us to import our dataset and also creates a matrix of features containing the dependent and independent variable.

**Code:**

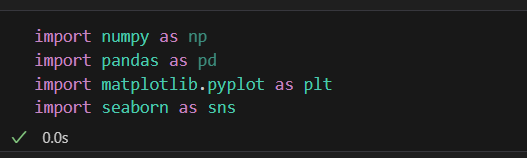
import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

**Output:**



**Load the dataset: (DATASET 1)**

* Data sets are available in .csv format. A CSV file stores tabular data in plain text.
* Each line of the file is a data record. We use the read\_csv method of the pandas library to read a local CSV file as a dataframe.
* Load our customer data from the CSV file

**Code:**

**import pandas as pd**

**# Try reading the file with different encodings**

**encodings = ['utf-8', 'latin1', 'ISO-8859-1']**

**for encoding in encodings:**

**try:**

**dataset = pd.read\_csv(r'/content/Covid\_19\_cases4.csv', encoding=encoding)**

**print(f"Successfully read with encoding: {encoding}")**

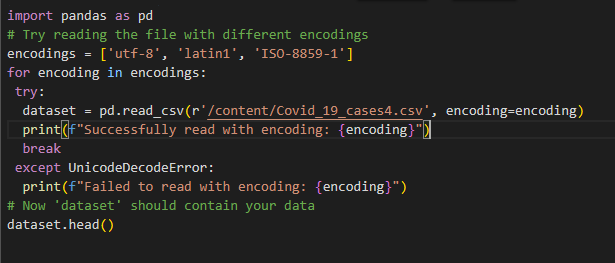
**break**

**except UnicodeDecodeError:**

**print(f"Failed to read with encoding: {encoding}")**

**# Now 'dataset' should contain your data**

**Output:**



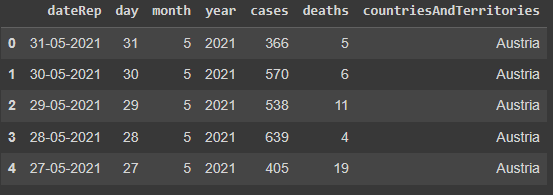
**Head() Function:**

* The head() function is used to get the first n rows.
* This function returns the first n rows for the object based on position.
* It is useful for quickly testing if your object has the right type of data in it.
* If the value of the n is not assigned it returns a default value of first 5 rows

**Code:**

dataset.head()

**Output:**



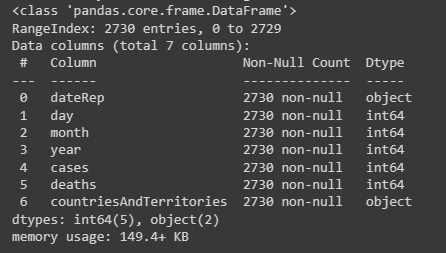
**Info() Function:**

* The info() method prints information about the DataFrame.
* The information contains the number of columns, column labels, column data types, memory usage, range index, and the number of cells in each column (non-nullvalues).

**Code:**

dataset.info()

**Output:**



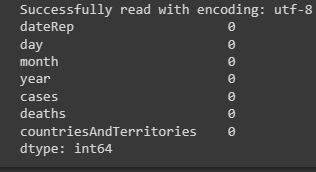
**Df.isnull().sum() Function:**

* This code is used to count the number of missing (null) values in each column of a DataFrame, denoted as df.
* It returns a summary of the missing data for each column, showing how many missing values are there in each column.
* This information is essential in data preprocessing and analysis to identify and handle missing data appropriately.Top of Form

**Code:**

dataset.isnull().sum()

**Output:**



**Describe Function:**

* The describe() function in pandas, a popular Python data analysis library, is used to generate summary statistics of a DataFrame or Series.

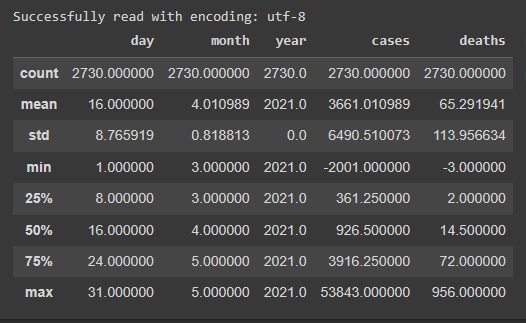
It provides a quick overview of the key statistics for numerical data in the dataset, including:

* **Count:** The number of non-null values.
* **Mean:** The average of the values.
* **Standard Deviation (std):** A measure of the spread or dispersion of the data.
* **Minimum:** The minimum value in the dataset.
* **25th Percentile (25%):** The value below which 25% of the data falls (the first quartile).
* **Median (50% or the 2nd quartile):** The middle value when the data is sorted.
* **75th Percentile (75%):** The value below which 75% of the data falls (the third quartile).
* **Maximum:** The maximum value in the dataset.

**Code:**

dataset.describe()

**Output:**



**Outliers:**

* Outliers are data points that significantly deviate from the rest of the data in a dataset.
* They can be exceptionally high or low values compared to the majority of the data.

**Code:**

**import matplotlib.pyplot as plt**

**# Ensure your dataset contains only numerical data for box plotting**

**dataset = pd.read\_csv(r'/content/Covid\_19\_cases4.csv', encoding=encoding)**

**numerical\_data = dataset.select\_dtypes(include='number')**

**# Transpose the data to prepare for box plotting**

**data\_to\_plot = numerical\_data.values.T**

**# Create subplots**

**fig, axs = plt.subplots(9, 1, dpi=95, figsize=(7, 17))**

**# Iterate through columns and create boxplots**

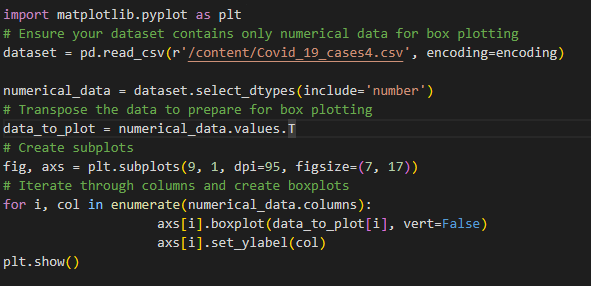
**for i, col in enumerate(numerical\_data.columns):**

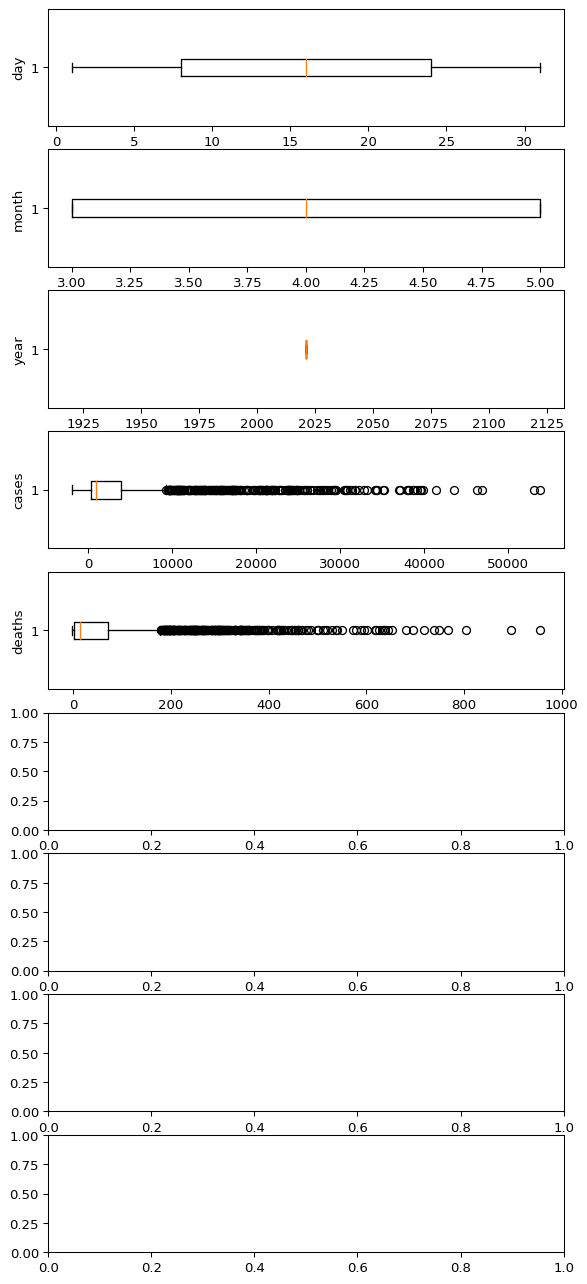
**axs[i].boxplot(data\_to\_plot[i], vert=False)**

**axs[i].set\_ylabel(col)**

**plt.show()**

**Output:**





**Corelation:**

* Correlation is a statistical measure that indicates the extent to which two or more variables fluctuate in relation to each other.
* Correlation describes the relationship between variables. It can be described as either strong or weak, and as either positive or negative.

**Code:**

dataset = pd.read\_csv(r'/content/Covid\_19\_cases4.csv', encoding=encoding)

numeric\_dataset = dataset.select\_dtypes(include=['number'])

corr = numeric\_dataset.corr()

import matplotlib.pyplot as plt

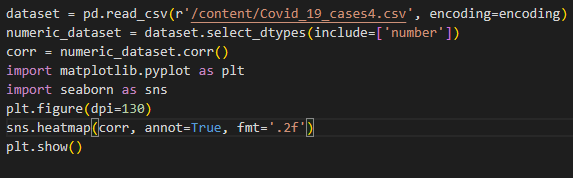
import seaborn as sns

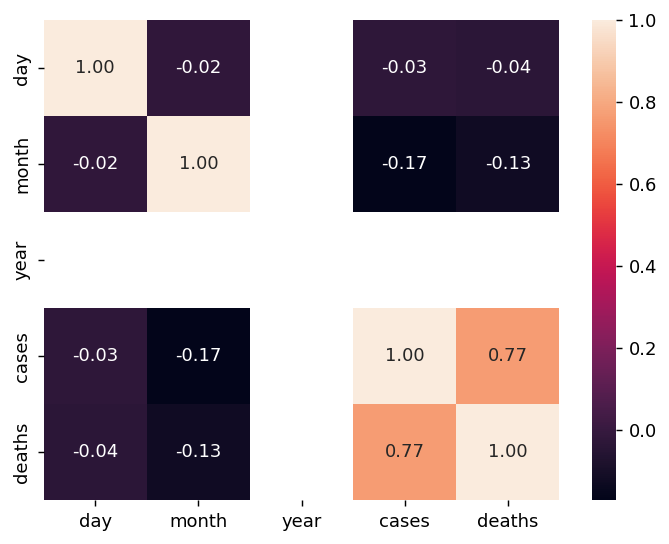
plt.figure(dpi=130)

sns.heatmap(corr, annot=True, fmt='.2f')

plt.show()

**Output:**





**Normalization**

* MinMaxScaler scales the data so that each feature is in the range [0, 1].
* It works well when the features have different scales and the algorithm being used is sensitive to the scale of the features, such as k-nearest neighbors or neural networks.
* Rescale your data using scikit-learn using the MinMaxScalar.