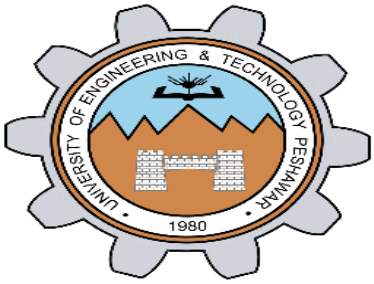
Lab report no 7,8



Fall 2022

Data Analytics Lab

Submitted By

Names	**Registration No**
Muhammad Ali	19pwcse1801

Section: A

Date: 21,12,22

Submitted to: Engr. Faiz Ullah

Department of Computer Systems Engineering

University of Engineering and Technology, Peshawar.

Data Analysis using python-pandas

Pandas is a popular Python library used for working in tabular data (similar to the data stored in a spreadsheet). Pandas provides helper functions to read data from various file formats like CSV, Excel spreadsheets, HTML tables, JSON, SQL, and more.

Considering an example of day-wise Covid-19 data for Italy in the tabular form as follows,
date,new_cases,new_deaths,new_tests
2020-04-21,2256.0,454.0,28095.0
2020-04-22,2729.0,534.0,44248.0
2020-04-23,3370.0,437.0,37083.0
2020-04-24,2646.0,464.0,95273.0
2020-04-25,3021.0,420.0,38676.0
2020-04-26,2357.0,415.0,24113.0
2020-04-27,2324.0,260.0,26678.0
2020-04-28,1739.0,333.0,37554.0
This format of storing data is known as comma-separated values or CSV.
We can now import the pandas module. As a convention, it is imported with the alias pd.

```
: import pandas as pd
```

```
: covid_df = pd.read_csv('italy-covid-daywise.csv')
```

```
: type(covid_df)
```

```
: pandas.core.frame.DataFrame
```

Data from the file is read and stored in a DataFrame object - one of the core data structures in Pandas for storing and working with tabular data. We typically use the _df suffix in the variable names for dataframes.

```
: covid_df
```

:

	date	new_cases	new_deaths	new_tests
0	2019-12-31	0	0	NaN
1	2020-01-01	0	0	NaN
2	2020-01-02	0	0	NaN
3	2020-01-03	0	0	NaN
4	2020-01-04	0	0	NaN
...	...	...	...	...
243	2020-08-30	1444	1	53541.0
244	2020-08-31	1365	4	42583.0
245	2020-09-01	996	6	54395.0
246	2020-09-02	975	8	NaN
247	2020-09-03	1326	6	NaN

248 rows × 4 columns

Here's what we can tell by looking at the dataframe:

- The file provides four day-wise counts for COVID-19 in Italy
- The metrics reported are new cases, deaths, and tests
- Data is provided for 248 days: from Dec 12, 2019, to Sep 3, 2020

Keep in mind that these are officially reported numbers. The actual number of cases & deaths may be higher, as not all cases are diagnosed.

We can view some basic information about the data frame using the .info method.

It appears that each column contains values of a specific data type. You can view statistical information for numerical columns (mean, standard deviation, minimum/maximum values, and the number of nonempty values) using the .describe method.

- pd.read_csv - Read data from a CSV file into a Pandas DataFrame object
- .info() - View basic infomation about rows, columns & data types
- .describe() - View statistical information about numeric columns
- .columns - Get the list of column names
- .shape - Get the number of rows & columns as a tuple

Tasks:

Find the total number of reported cases and deaths related to Covid-19 in Italy.
Find the overall death rate (ratio of reported deaths to reported cases).
Find the overall number of tests conducted? A total of 935310 tests were conducted before daily test numbers were reported.
Find the positive rate i.e. fraction of tests returned a positive result.

Tasks no 1: -

```
import pandas as pd
covid_df = pd.read_csv('italy-covid-daywise.csv')
newt_cases = covid_df['new_cases'].sum()
t_deaths = covid_df['new_deaths'].sum()
newt_test = covid_df['new_tests'].sum()
```

Tasks no 2: -

```
ratio = (t_deaths / newt_cases)*100
print('total no of death in itlay ',ratio)
```

Output: -

```
total no of death in itlay 13.073679170579894
```

Tasks no 3: -

```
initial_test = 935310
t_cases = (initial_test + newt_test)
print('total cases in itlay ',t_cases)
```

Output: -

```
total cases in itlay 5214766.0
```

Tasks no 4: -

```
t_newcases = covid_df['new_cases']
positivety = ( newt_cases /t_cases )*100
print('positivety in itlay ',positivety)
```

Output: -

```
positivety in itlay 5.206657403227681
```

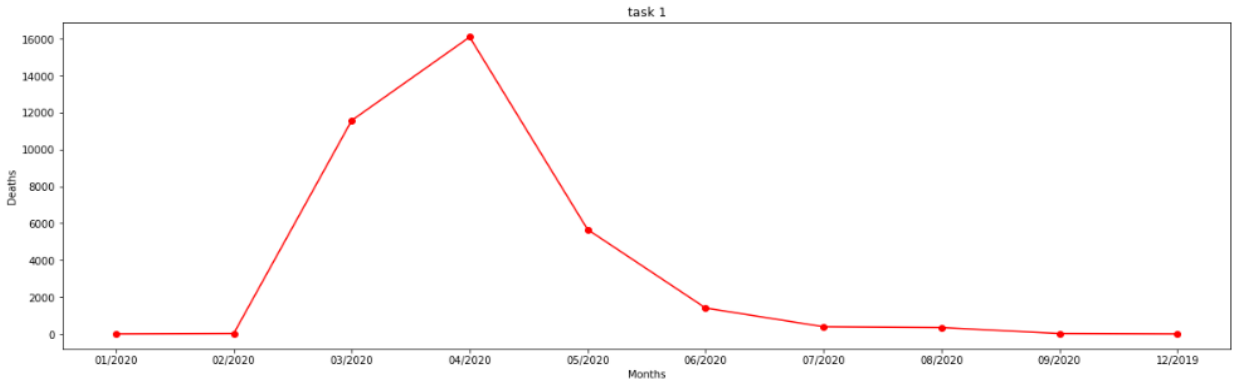
Lab report no 8

TASK 1: -

Display the graph of death cases verses months.

```
In [86]: #task 1
plt.figure(figsize=(20, 6))
plt.plot(data.new_deaths,marker="o",c="r")
plt.xlabel('Months')
plt.ylabel('Deaths')
plt.title("task 1")
```

```
Out[86]: Text(0.5, 1.0, 'task 1')
```

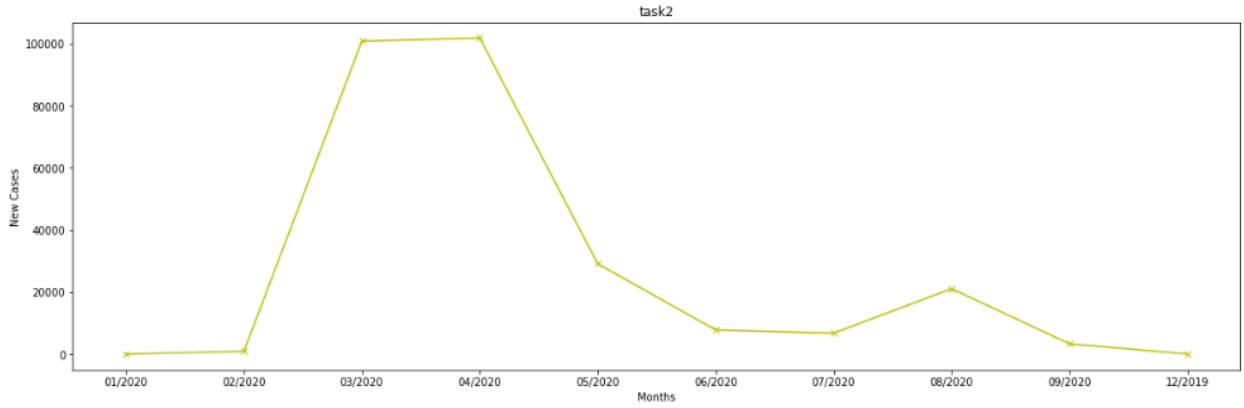


TASK 2:

Display the graph of new cases verses months.

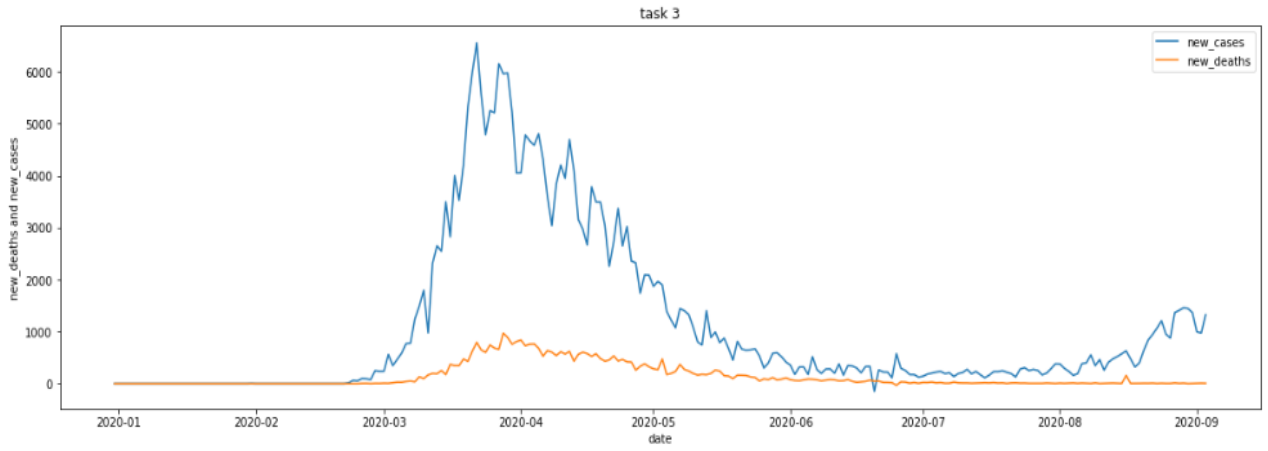
```
In [85]: #task 2
plt.figure(figsize=(20, 6))
plt.plot(data.new_cases,marker="x",c="y")
plt.xlabel('Months')
plt.ylabel('New Cases')
plt.title("task2")
```

```
Out[85]: Text(0.5, 1.0, 'task2')
```



TASK 3: -Compare the new cases and death cases day-wise on multi-line graph, mark the legends and properly label and title the graph.

In [41]:
```
#task 3
plt.figure(figsize=(20,6))
plt.plot(d,a)
plt.plot(d,b)
plt.xlabel("date")
plt.ylabel("new_deaths and new_cases")
plt.title("task 3")
plt.legend(["new_cases","new_deaths"]);
```



TASK 4: - Display how the new cases and new tests are related day-wise on multi-line graph, mark the legends and properly label and title the graph.

In [87]:
```
#task 4
plt.figure(figsize=(20,6))
plt.plot(d,a)
plt.plot(d,c)
plt.xlabel("new_cases and new_tests")
plt.ylabel("date")
plt.title("task 4")
plt.legend(["new_cases","new_tests"]);
```