**Lukas saidim - SCT222-0137/2020**

**Nelson kamau - Sct222-0301/2020**

**Kahura Gedion karanja - Sct222-0155/2020**

**Kibet collins kemboi - SCT222-0147/2020**

**Francis Peter Thuo - Sct222-0126/2020**

**DBMS CAT**

**Question 1**

Covid-19 is a contagious disease caused by Virus known as Sars-Cov-2, which hit the entire world in the year 2020.

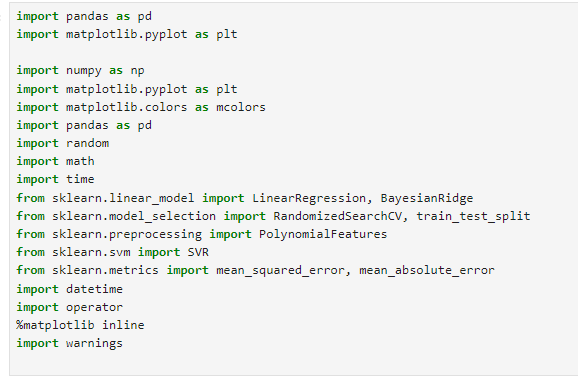
This task aims to study the Covid-19 Statistics with respect to active cases, confirmed cases, total deaths and so on across the world. Further, it analyses the countries which have been badly hit by the virus and some important ratios to find out about the deaths caused by Covid-19 virus.

The Dataset comprises of six .csv files which contain information about the Covid-19 status across different countries and across the world. The data has been collected from the World Health Organization website and from several other government sources and then compiled thereafter. The six data files are explained below:

* 'country\_wise.csv': Raw data comprising of Covid-19 statistics (total cases, active cases, recovered cases, total deaths) across different countries/regions as defined by WHO.
* 'day\_wise\_country.csv': Day-wise and country-wise Covid-19 statistics.
* 'day\_wise\_worldwide.csv': Day-wise Covid-19 Statistics for from January-2020 to July-2020 across the world.
* 'east\_africa\_covid\_data.csv': cumulative number of coronavirus (COVID-19) cases in East Africa.
* 'kenya\_county\_cases.csv': Cumulative number of confirmed coronavirus (COVID-19) cases in Kenya counties.
* 'kenya\_daily\_cases.csv': Total confirmed cases daily in Kenya.

Data for COVID-19 is compiled by downloading information from various sources such as Worldometer, Africa Open Data, and Kenya Open Data. The data is be obtained in different formats, including XMLs then later converted to CSVs.

In the Jupyter Notebook environment, we compile and examine the data to prepare for analysis. The initial steps involve setting up the notebook on a local machine and importing the necessary modules.



**NB**

1*.* ***import pandas as pd***: This imports the Pandas library and aliases it as `pd`. Pandas is a powerful data manipulation and analysis library for Python. It provides data structures like DataFrame for efficient data handling.

2. ***import matplotlib.pyplot as plt***: This imports the `pyplot` module from the Matplotlib library and aliases it as `plt`. Matplotlib is a popular plotting library in Python, and `pyplot` provides a convenient interface for creating various types of plots.

3. ***import numpy as np***: This imports the NumPy library and aliases it as `np`. NumPy is a numerical computing library for Python, providing support for large, multi-dimensional arrays and matrices, along with mathematical functions to operate on these.

4. ***import matplotlib.colors as mcolors***: This imports the `colors` module from Matplotlib and aliases it as `mcolors`. It provides a collection of named colors and color maps that can be used in plotting.

6. ***`import random***`: This imports the `random` module, which provides functions for generating random numbers and performing random operations.

7***. `import math***`: This imports the built-in `math` module, providing mathematical functions and constants.

8. `***import time`:*** This imports the built-in `time` module, which provides various time-related functions.

9. ***`from sklearn.linear\_model import LinearRegression, BayesianRidge***`: This imports specific regression models from the scikit-learn library. Scikit-learn is a machine learning library for Python, and here you're importing linear regression and Bayesian Ridge regression models.

10. ***`from sklearn.model\_selection import RandomizedSearchCV, train\_test\_split`:*** This imports classes for model selection, including `RandomizedSearchCV` for hyperparameter tuning and `train\_test\_split` for splitting data into training and testing sets.

11***. `from sklearn.preprocessing import PolynomialFeatures`:*** This imports a class for generating polynomial features. It is often used in combination with linear regression to model non-linear relationships.

12***. `from sklearn.svm import SVR`:*** This imports the Support Vector Regression (SVR) model from scikit-learn, which is a regression algorithm based on support vector machines.

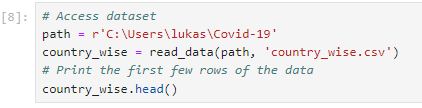
13. `***from sklearn.metrics import mean\_squared\_error, mean\_absolute\_error`:*** This imports functions for evaluating regression model performance, including mean squared error and mean absolute error.

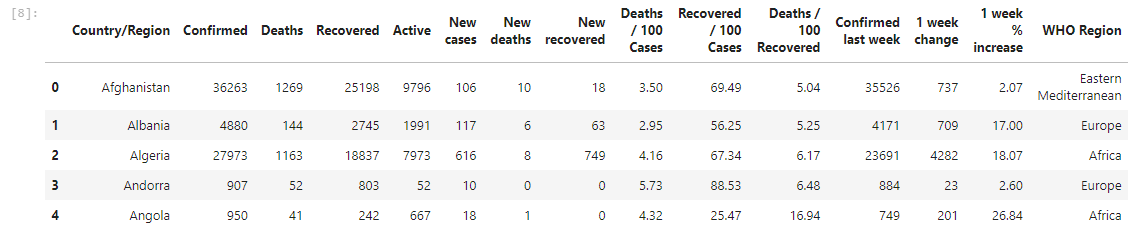
14***. `import datetime***`: This imports the built-in `datetime` module, which provides classes for working with dates and times.

15. ***`import operator`:*** This imports the `operator` module, which provides functions that implement built-in operations as functions.

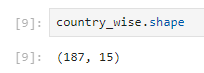
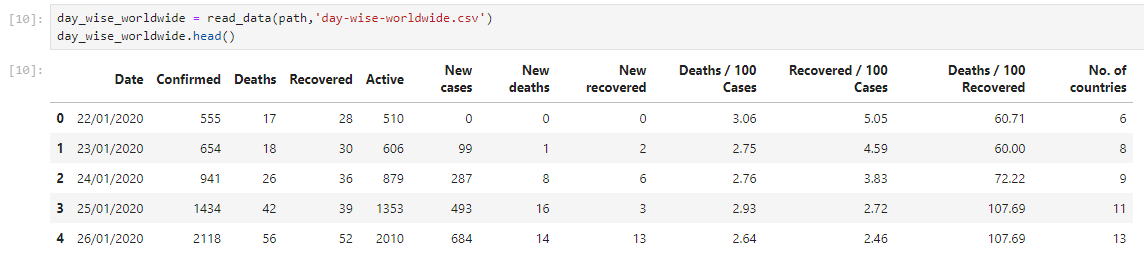
16. ***`%matplotlib inline`:*** This is a Jupyter Notebook magic command that ensures that Matplotlib plots are displayed inline in the notebook rather than in a separate window.

By using the Pandas library to read a CSV file named 'country\_wise.csv' from the specified path, and then it prints the first few rows of the loaded data.





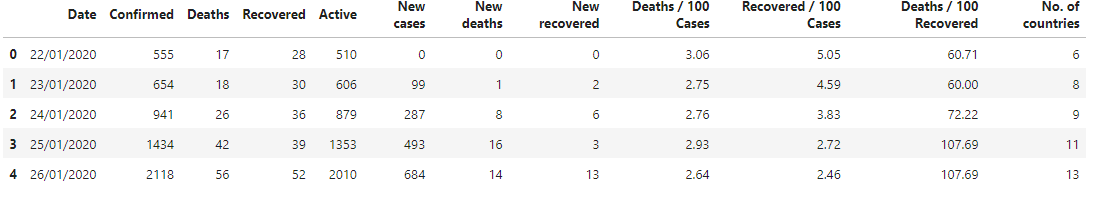
The *country\_wise.shape* command is used to retrieve the dimensions of the DataFrame country\_wise. Specifically, it returns a tuple representing the number of rows and columns in the DataFrame.



**Day wise (New cases, deaths, recovered, deaths/100, recovered/100)**

This line uses the read\_data() function from the Pandas library to read data from a CSV file. The path variable represents the file path or URL from which the CSV file is read, and 'day-wise-worldwide.csv' is the name of the CSV file.

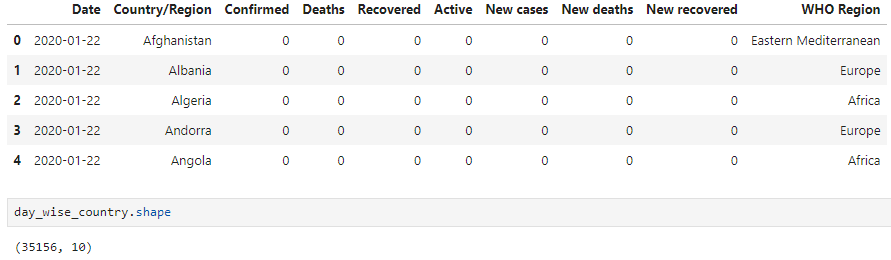




**Day wise for each country**

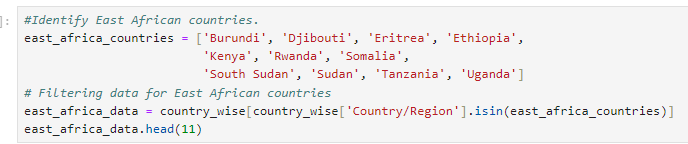
We are going to read another dataset named 'day-wise-country.csv'. Each row represents a specific observation at a particular date for a given country or region.

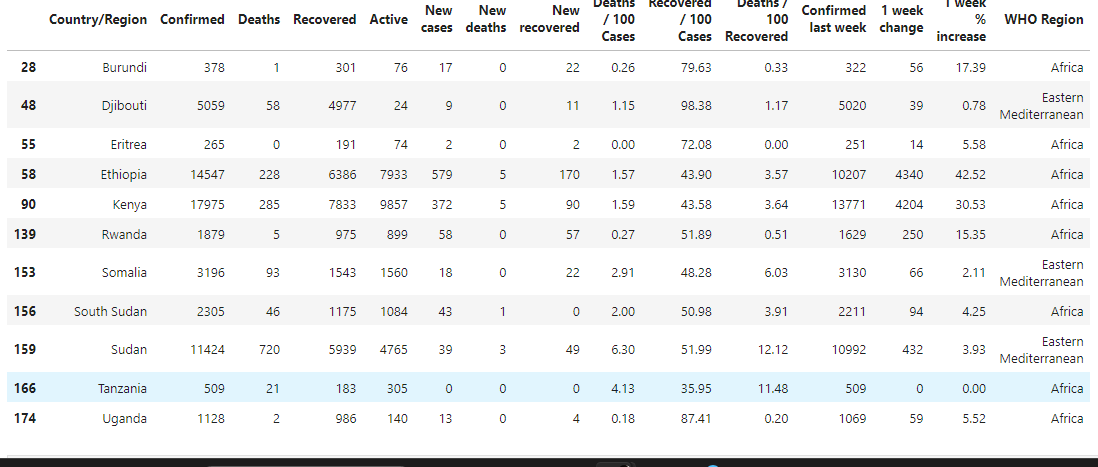




**East African countries**

We identify and filter data for East African countries from a DataFrame named country wise.

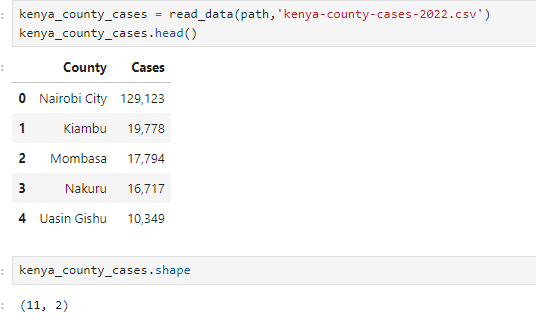




Then save the filtered data for East African countries to a new CSV file named 'east-africa-covid-data.csv'.



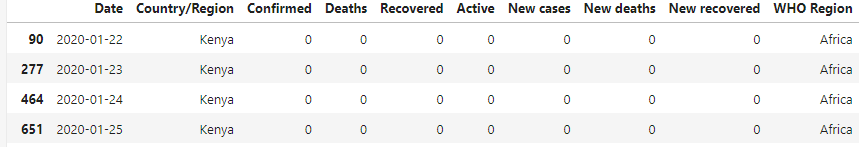
**Kenya cases in counties**



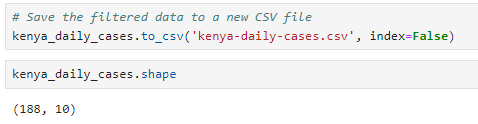
The data provides information on the number of COVID-19 cases reported in various counties in Kenya. Each row corresponds to a specific county, and the 'Cases' column indicates the reported number of cases in that county. The 'County' column holds the names of the counties.

**Kenya daily cases**

The resulting kenya\_daily\_cases DataFrame should have the same columns as day\_wise\_country, and it will include only the rows where the country is 'Kenya'. To specifically get information on daily COVID-19 metrics like Confirmed, Deaths, Recovered, Active, New cases, New deaths, and New recovered, you can look at the relevant columns in the DataFrame. The structure of these columns should represent the daily changes in those metrics for Kenya.



Save to a new csv file



**Question 2 – ingest data into Hadoop**

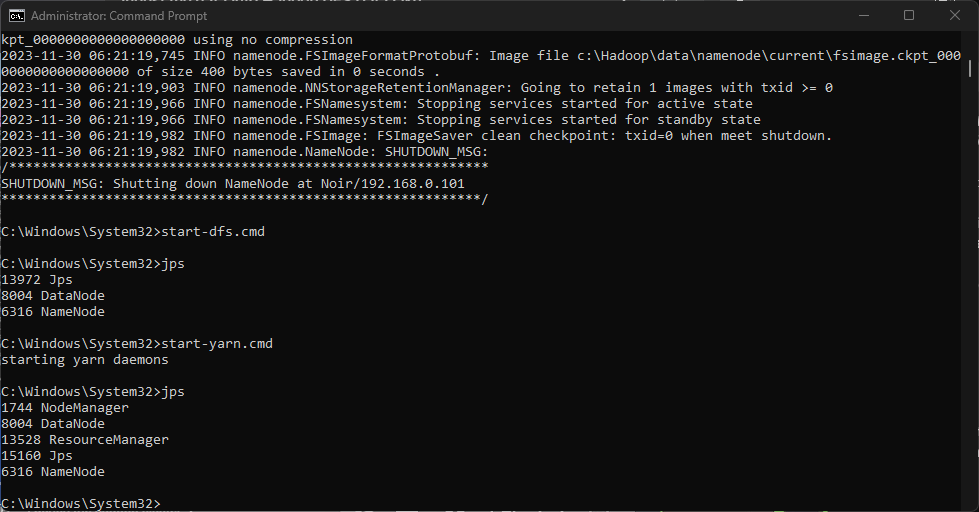
Starting Hadoop involves initiating the various Hadoop daemons and services. Hadoop is typically deployed in a distributed environment, and starting it requires a set of commands.

Start Hadoop Distributed File System (HDFS):

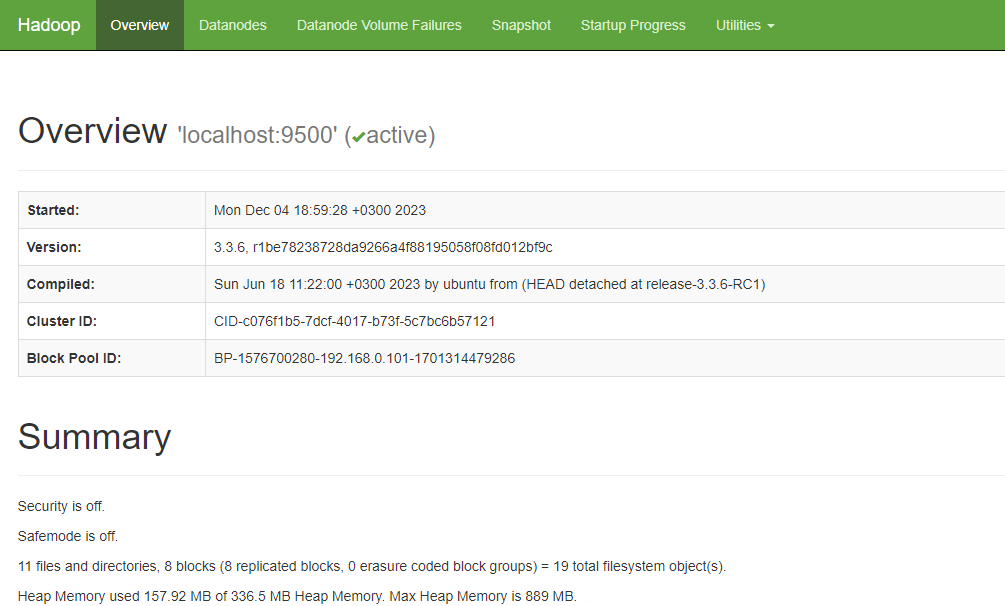
* Open a terminal
* Use the following command to start HDFS: *on windows*
* start-dfs.cmd
* This command starts the NameNode and DataNode daemons.

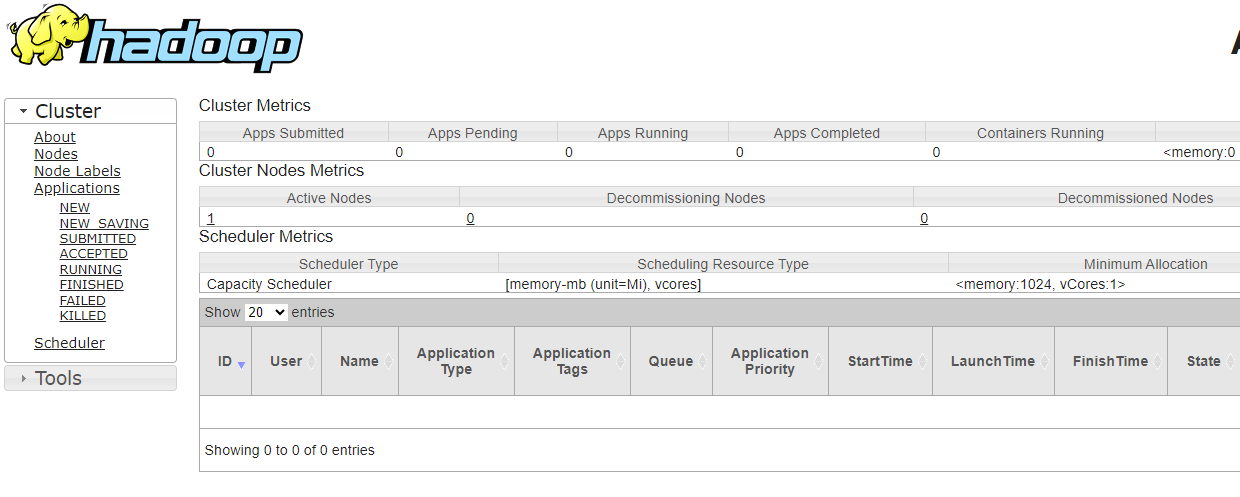
Start Yet Another Resource Negotiator (YARN):

* start-yarn.cmd
* This command starts ResourceManager and NodeManager daemons.
* Verify Hadoop Processes:
* After starting HDFS and YARN, you can check the status of the Hadoop processes by accessing the Hadoop web interfaces.

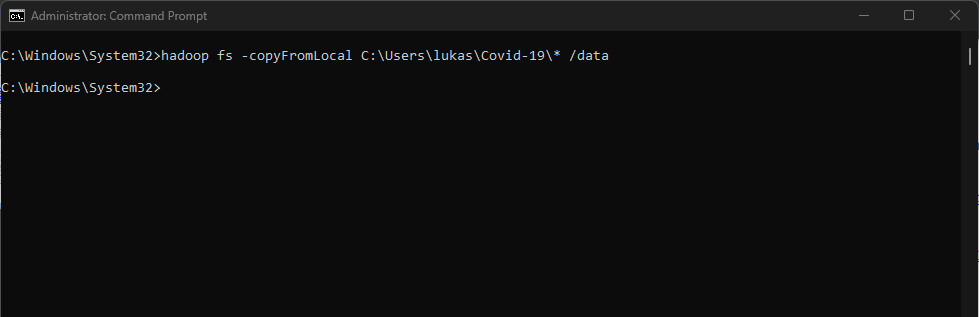


* Open a web browser and navigate to the following URLs:
* HDFS NameNode: <http://localhost:5007> instead we used: 9201 due to conflict with Jupyter notebook
* YARN ResourceManager: <http://localhost:8088>



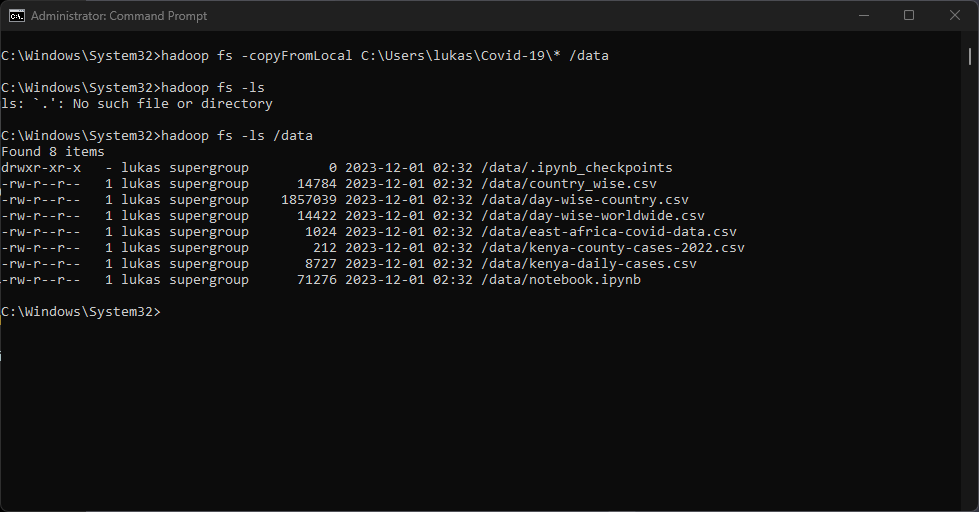


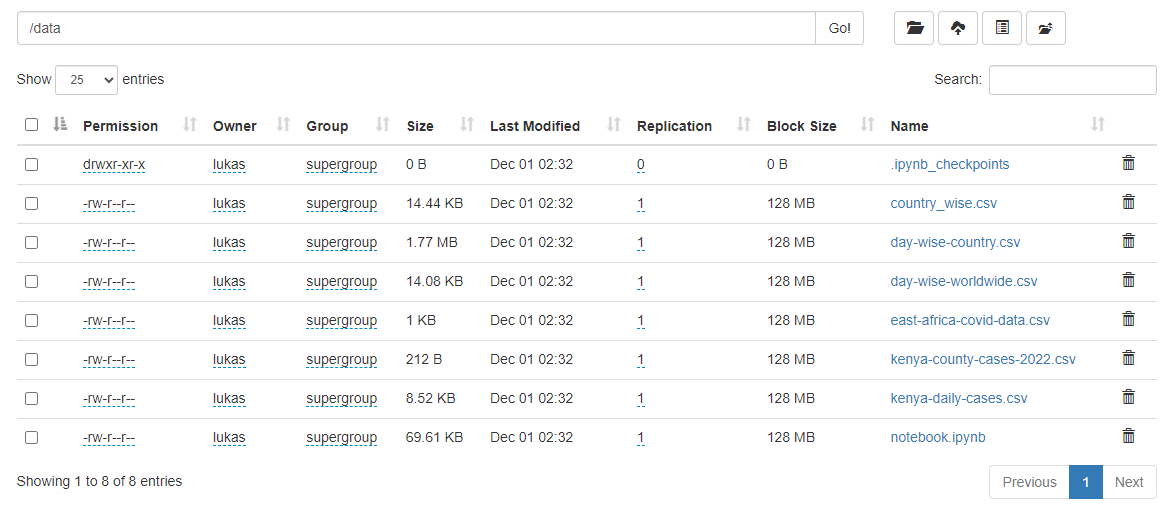
Ingesting data into Hadoop involves transferring data from external sources into the Hadoop Distributed File System (HDFS) so that it can be processed and analyzed by Hadoop-based applications.



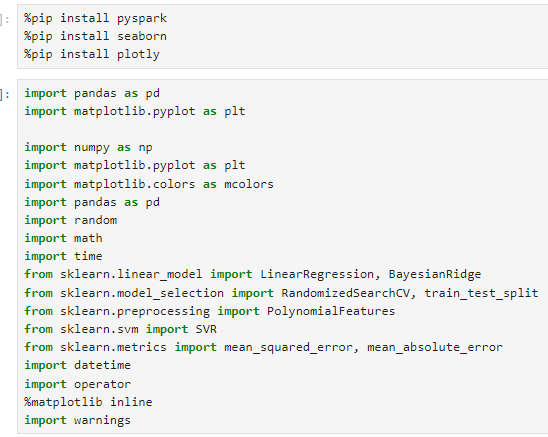
The Hadoop command-line interface provides commands to interact with HDFS, including copying files.

After ingesting data, you can use Hadoop commands or web interfaces to check if the data has been successfully loaded into HDFS.

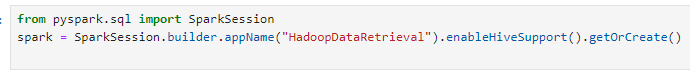




## **Use pyspark package to extract the data from the data lake**



These commands will install the specified Python packages (pyspark, seaborn, and plotly) using the Python package manager, pip.



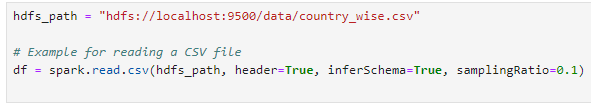
spark = *SparkSession.builder.appName("HadoopDataRetrieval").enableHiveSupport().getOrCreate():* This line creates a Spark session. Here's what each part of the chain does:

**SparkSession.builder:** Starts the process of building a SparkSession.

**appName("HadoopDataRetrieval**"): Sets a user-defined name for the application, which will be displayed in the Spark web UI.

**enableHiveSupport():** Enables Hive support, allowing you to interact with Hive using Spark.

**getOrCreate():** Gets an existing SparkSession or creates a new one if it doesn't exist.

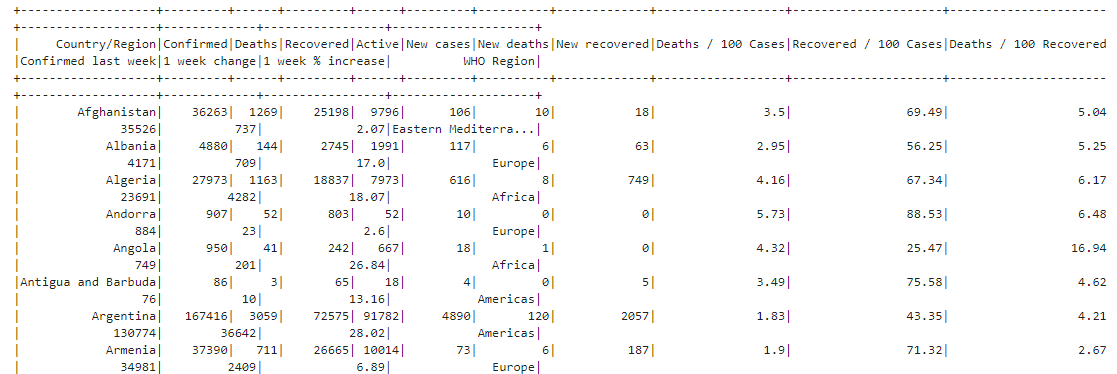




Using PySpark to read a CSV file from HDFS (Hadoop Distributed File System) into a DataFrame named df.

*hdfs\_path = "hdfs://localhost:9500/data/country\_wise.csv"*

This line defines the HDFS path to the CSV file you want to read. The path is specified as a URL starting with "hdfs://" followed by the hostname and port number of the HDFS Namenode, and then the path to the file.





This line reads the full CSV dataset into a new PySpark DataFrame called ***full\_df.***

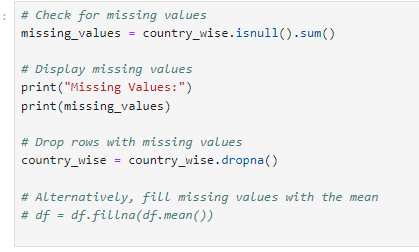
option("header", "true"): Specifies that the first row of the CSV file contains column names.

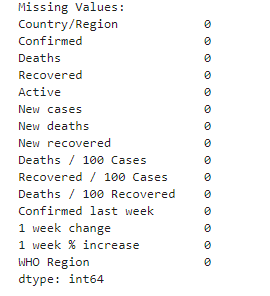
samplingRatio=1: Reads the entire dataset for schema inference since the sampling ratio is set to 1.

## **Choose appropriate techniques to Pre- process the extracted data**

**Handling Missing Values:**

Check for missing values and decide how to handle them. Common methods include dropping rows with missing values, filling missing values with the mean or median, or using more advanced imputation techniques.





**Data Types Conversion**

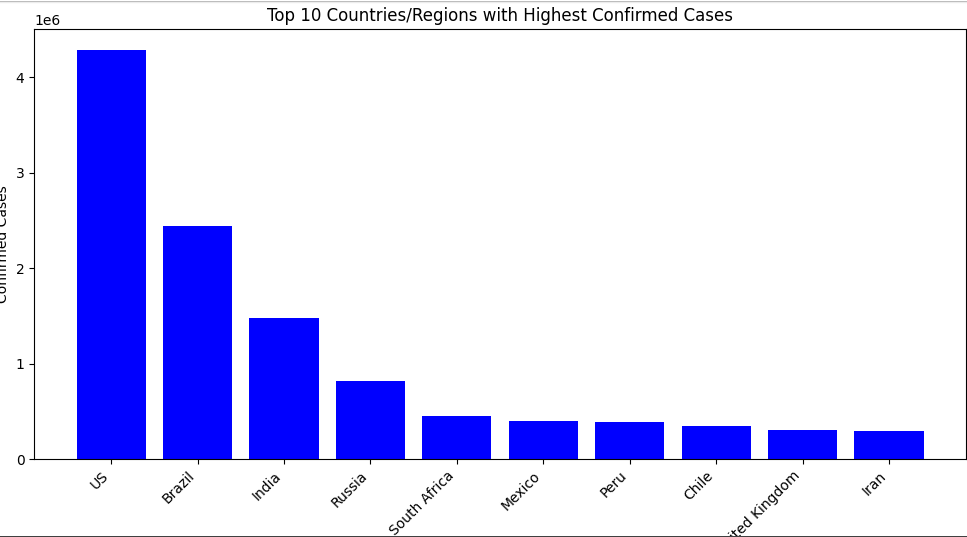
Ensure that columns have the correct data types. Convert columns to appropriate types if necessary.

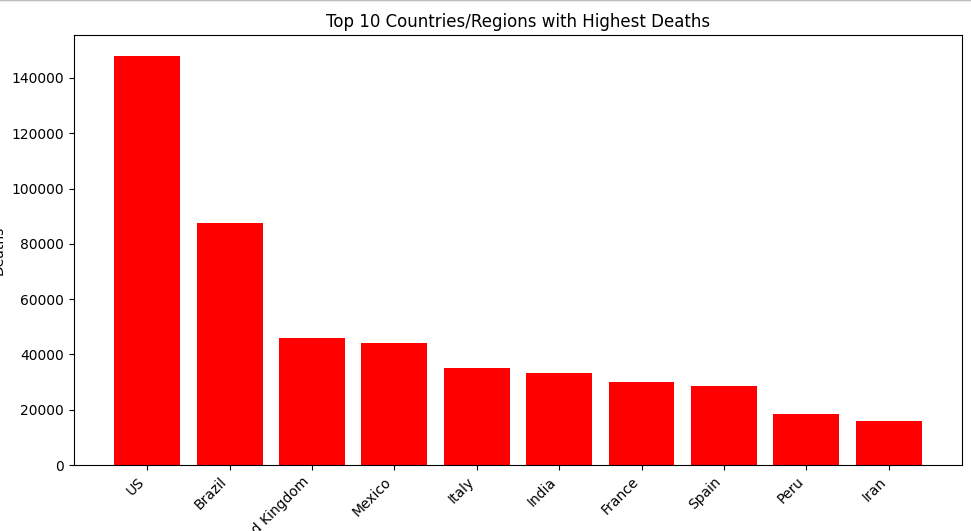


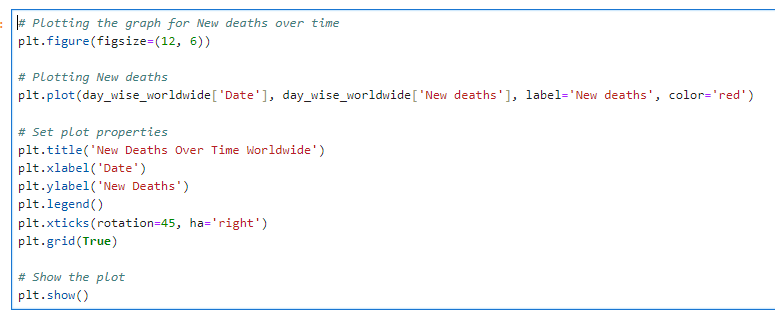
**Exploratory Data Analysis (EDA)**

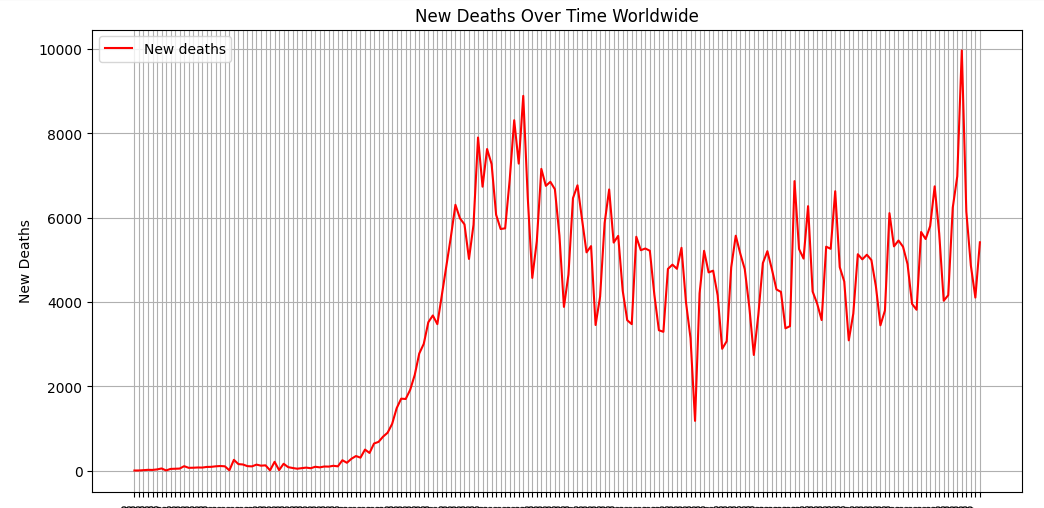
Explore the data to gain insights and identify patterns. Use visualizations such as histograms, box plots, and scatter plots.



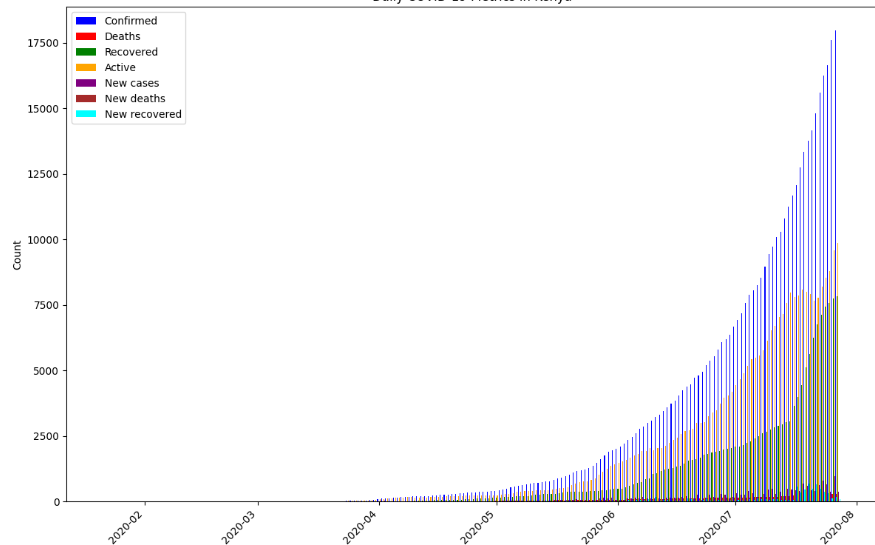




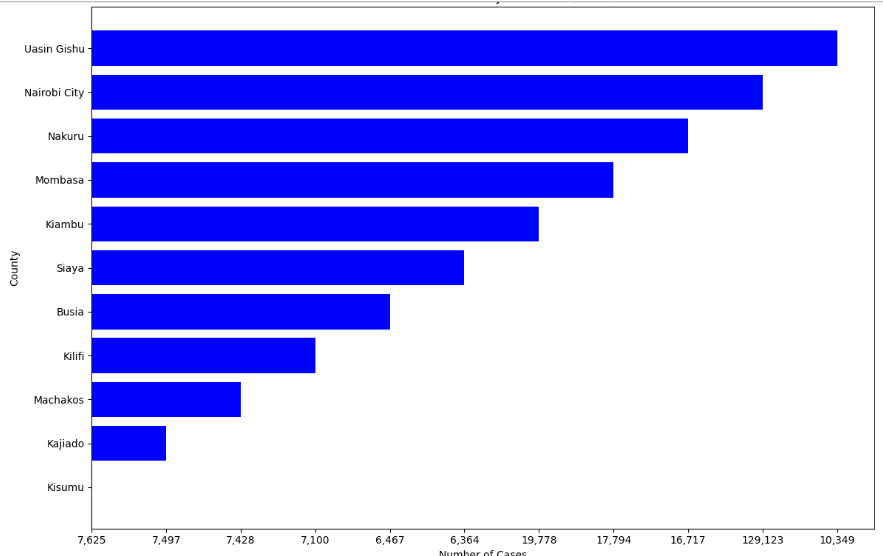




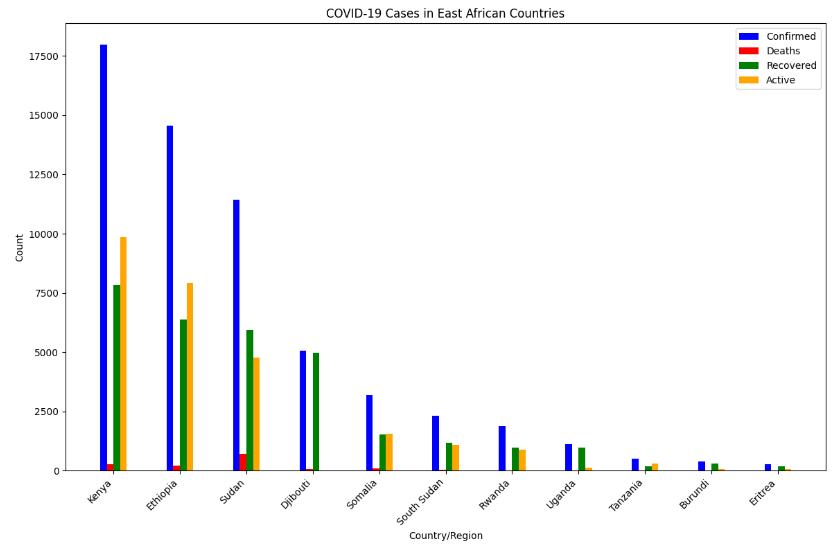




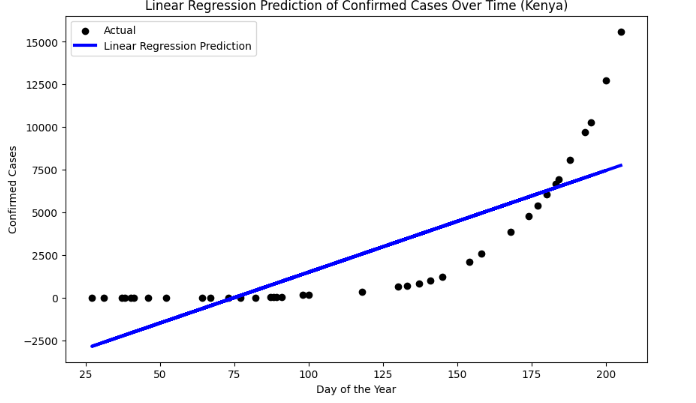












This example uses a linear regression model based on the day of the year to predict the number of confirmed cases for Kenya. The code also calculates the Mean Squared Error as an evaluation metric.

**Potential applications of the interpreted results**

The interpreted results from the time series analysis and predictions of COVID-19 cases in Kenya can be applied in various ways to support decision-making, public health efforts, and resource allocation. Here are potential applications of the interpreted results:

1. **Resource Planning and Allocation** - Governments and healthcare authorities can use the predictions to plan for and allocate resources such as hospital beds, ventilators, medical personnel, and medications in anticipation of future needs.

2. **Policy Decision Support** - Policy-makers can use the insights gained from the analysis to make informed decisions on public health measures, lockdowns, travel restrictions, and vaccination campaigns.

3**. Early Warning Systems** - The predictions can serve as part of an early warning system, helping authorities take proactive measures to prevent or mitigate potential surges in COVID-19 cases.

4. **Healthcare Capacity Planning** - Hospitals and healthcare facilities can use the predictions to plan for surges in patient numbers, ensuring they have the necessary capacity and resources to handle increased demand.

5. **Public Awareness and Communication** - Communicating the predictions to the public can help raise awareness and encourage adherence to preventive measures during periods of higher risk.

6. **Vaccination Campaign Planning** - The predictions can aid in planning and optimizing vaccination campaigns by identifying periods of higher risk and targeting specific demographics or regions.

7. **Research and Epidemiological Studies** - Researchers can use the interpreted results to further investigate patterns and trends, contributing to epidemiological studies that deepen our understanding of the virus's behaviour.

8. **International Collaboration** - Sharing the predictions with international health organizations and neighboring countries can facilitate collaboration and information exchange, especially in regions with interconnected healthcare systems.

9. **Data-Driven Decision-Making** - The results support a data-driven approach to decision-making, enabling stakeholders to make evidence-based choices and respond more effectively to the evolving situation.

10. **Public Health Messaging** - Tailoring public health messaging based on predicted trends can help communicate risks and preventive measures more effectively to the public.

Predictions are inherently uncertain, and various factors can influence the accuracy of the model. Therefore, the results should be used as a supportive tool alongside other data and expert judgment in the decision-making process. Regular model evaluation and validation against new data are crucial to maintaining the model's relevance and accuracy over time.