JOMO KENYATTA UNIVERSITY
OF
AGRICULTURE & TECHNOLOGY

ADVANCED DB CAT

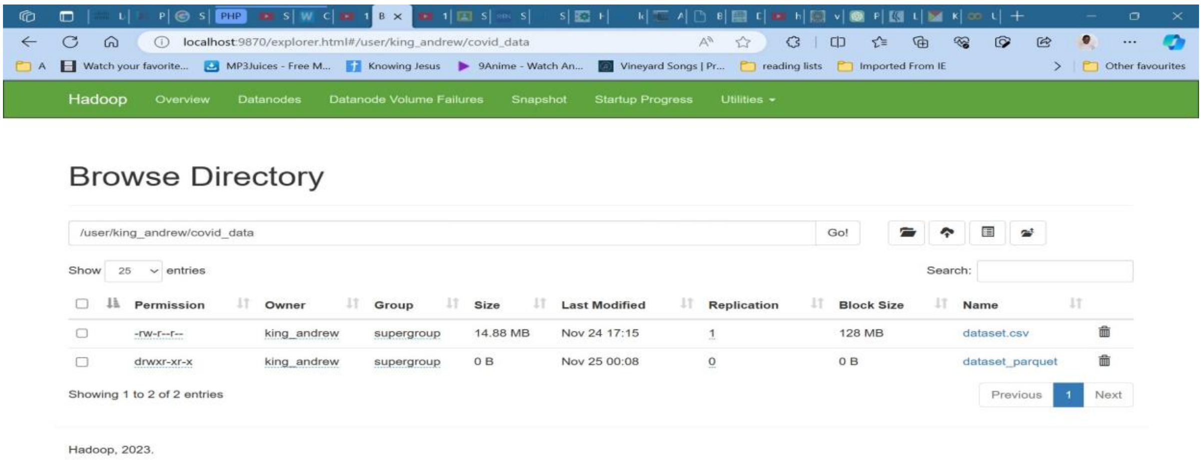
MARK NYANGADA – SCT222-0244/2020
JOSEPH OGURI – SCT222-0304/2020
VALENTINE WANJIKU - SCT222-0128/2020
ALPHONCE KOTIENO - SCT222-0181/2020

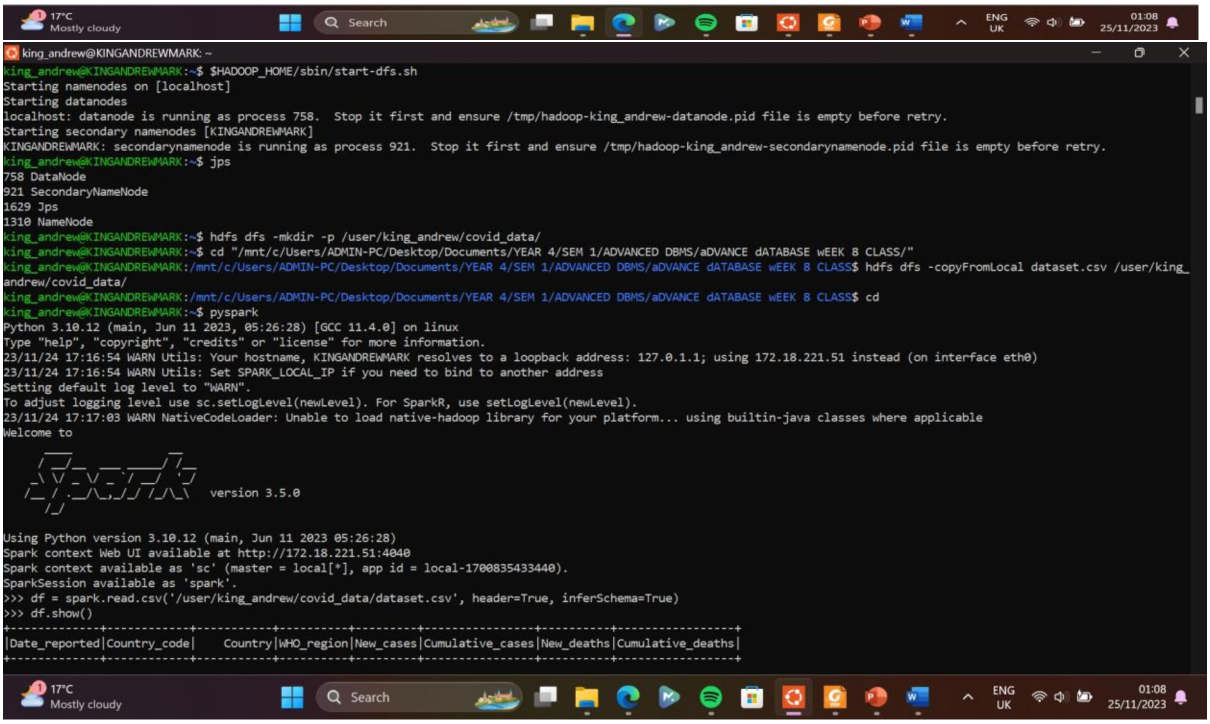
NOVEMBER 24, 2023

1. **Describe how the data was compiled in task 1 and include Screen captures of both code and output**

   Data was compiled through hadoop's core components: Hadoop Distributed File System (HDFS) for distributed storage and MapReduce for processing. Data was stored in smaller blocks across a cluster using HDFS, allowing parallel storage and retrieval. In the MapReduce programming model, data was processed and compiled through the Map and Reduce phases, enabling parallel and distributed computation

2. **Describe how the data was ingested into Hadoop data lake and include screenshots**





3. **Describe how data was extracted using pyspark and include associated screen shots**

```
!pip install -q findspark
!pip install pyspark
```

```
Requirement already satisfied: pyspark in /usr/local/lib/python3.10/dist-packages (3.5.0)
Requirement already satisfied: py4j==0.10.9.7 in /usr/local/lib/python3.10/dist-packages (from pyspark) (0.10.9.7)
```

```
import os
os.environ["JAVA_HOME"] = "/usr/lib/jvm/java-8-openjdk-amd64"
os.environ["SPARK_HOME"] = "/content/spark-3.5.0-bin-hadoop3"

import findspark
findspark.init()
```

```
from pyspark.sql import SparkSession

spark = SparkSession.builder \
        .master("local") \
        .appName("Hands-on PySpark on Google Colab") \
        .getOrCreate()
```

```
!pwd
!ls
```

```
/content
dataset.csv  sample_data  spark-3.5.0-bin-hadoop3  spark.tgz
```

```
spark_data = spark.read.format('csv').load("/content/dataset.csv")
```

```
spark_data.show(5, truncate=False)
```

```
+------------+------------+-----------+----------+---------+-----------------+----------+-----------------+
|_c0         |_c1         |_c2        |_c3       |_c4      |_c5              |_c6       |_c7              |
+------------+------------+-----------+----------+---------+-----------------+----------+-----------------+
|Date_reported|Country_code|Country    |WHO_region|New_cases|Cumulative_cases|New_deaths|Cumulative_deaths|
|2020-01-03  |AF          |Afghanistan|EMRO      |0        |0                |0         |0                |
|2020-01-04  |AF          |Afghanistan|EMRO      |0        |0                |0         |0                |
|2020-01-05  |AF          |Afghanistan|EMRO      |0        |0                |0         |0                |
|2020-01-06  |AF          |Afghanistan|EMRO      |0        |0                |0         |0                |
+------------+------------+-----------+----------+---------+-----------------+----------+-----------------+
only showing top 5 rows
```

```
spark_data = spark.read.format('csv').options(header='true').load("/content/dataset.csv")
spark_data.show(5, truncate=False)
```

```
+-------------+------------+-----------+----------+---------+----------------+----------+-----------------+
|Date_reported|Country_code|Country    |WHO_region|New_cases|Cumulative_cases|New_deaths|Cumulative_deaths|
+-------------+------------+-----------+----------+---------+----------------+----------+-----------------+
|2020-01-03   |AF          |Afghanistan|EMRO      |0        |0               |0         |0                |
|2020-01-04   |AF          |Afghanistan|EMRO      |0        |0               |0         |0                |
|2020-01-05   |AF          |Afghanistan|EMRO      |0        |0               |0         |0                |
|2020-01-06   |AF          |Afghanistan|EMRO      |0        |0               |0         |0                |
|2020-01-07   |AF          |Afghanistan|EMRO      |0        |0               |0         |0                |
+-------------+------------+-----------+----------+---------+----------------+----------+-----------------+
only showing top 5 rows
```

```
spark_data.printSchema()
```

```
root
 |-- Date_reported: string (nullable = true)
 |-- Country_code: string (nullable = true)
 |-- Country: string (nullable = true)
 |-- WHO_region: string (nullable = true)
 |-- New_cases: string (nullable = true)
 |-- Cumulative_cases: string (nullable = true)
 |-- New_deaths: string (nullable = true)
 |-- Cumulative_deaths: string (nullable = true)
```

```
kenya_data_filtered_rows = kenya_data.filter(~(col("New_cases") ==0) & ~(col("Cumulative_cases") ==0)& ~(col("New_deaths") ==0)& ~(col("Cumulative_deaths") ==0))
```

```
kenya_data_filtered_rows.show()
```

```
+-------------+------------+-------+----------+---------+----------------+----------+-----------------+
|Date_reported|Country_code|Country|WHO_region|New_cases|Cumulative_cases|New_deaths|Cumulative_deaths|
+-------------+------------+-------+----------+---------+----------------+----------+-----------------+
|   2020-03-27|          KE|  Kenya|      AFRO|        3|              28|         4|                4|
|   2020-04-02|          KE|  Kenya|      AFRO|       22|              81|         2|                6|
|   2020-04-13|          KE|  Kenya|      AFRO|        6|             197|         1|                8|
|   2020-04-14|          KE|  Kenya|      AFRO|       11|             208|         1|                9|
|   2020-04-17|          KE|  Kenya|      AFRO|        9|             234|         1|               10|
|   2020-04-19|          KE|  Kenya|      AFRO|       16|             262|         1|               11|
|   2020-04-20|          KE|  Kenya|      AFRO|        8|             270|         2|               13|
|   2020-04-30|          KE|  Kenya|      AFRO|       10|             384|         1|               14|
|   2020-05-01|          KE|  Kenya|      AFRO|       12|             396|         2|               16|
|   2020-05-02|          KE|  Kenya|      AFRO|       15|             411|         4|               20|
|   2020-05-03|          KE|  Kenya|      AFRO|       24|             435|         1|               21|
|   2020-05-04|          KE|  Kenya|      AFRO|       30|             465|         2|               23|
|   2020-05-07|          KE|  Kenya|      AFRO|       47|             582|         3|               26|
|   2020-05-08|          KE|  Kenya|      AFRO|       25|             607|         3|               29|
|   2020-05-10|          KE|  Kenya|      AFRO|       28|             649|         1|               30|
|   2020-05-11|          KE|  Kenya|      AFRO|       23|             672|         2|               32|
|   2020-05-12|          KE|  Kenya|      AFRO|       28|             700|         1|               33|
|   2020-05-13|          KE|  Kenya|      AFRO|       15|             715|         3|               36|
|   2020-05-14|          KE|  Kenya|      AFRO|       22|             737|         4|               40|
|   2020-05-15|          KE|  Kenya|      AFRO|       21|             758|         2|               42|
+-------------+------------+-------+----------+---------+----------------+----------+-----------------+
only showing top 20 rows
```

```
[ ] spark_data = spark.read.format('csv').options(header='true', inferSchema='true').load("/content/dataset.csv")
    spark_data.show(5, truncate=False)
    spark_data.printSchema()

+------------+------------+-----------+----------+---------+----------------+----------+-----------------+
|Date_reported|Country_code|Country    |WHO_region|New_cases|Cumulative_cases|New_deaths|Cumulative_deaths|
+------------+------------+-----------+----------+---------+----------------+----------+-----------------+
|2020-01-03   |AF          |Afghanistan|EMRO      |0        |0               |0         |0                |
|2020-01-04   |AF          |Afghanistan|EMRO      |0        |0               |0         |0                |
|2020-01-05   |AF          |Afghanistan|EMRO      |0        |0               |0         |0                |
|2020-01-06   |AF          |Afghanistan|EMRO      |0        |0               |0         |0                |
|2020-01-07   |AF          |Afghanistan|EMRO      |0        |0               |0         |0                |
+------------+------------+-----------+----------+---------+----------------+----------+-----------------+
only showing top 5 rows

root
 |-- Date_reported: date (nullable = true)
 |-- Country_code: string (nullable = true)
 |-- Country: string (nullable = true)
 |-- WHO_region: string (nullable = true)
 |-- New_cases: integer (nullable = true)
 |-- Cumulative_cases: integer (nullable = true)
 |-- New_deaths: integer (nullable = true)
 |-- Cumulative_deaths: integer (nullable = true)
```

```
[ ] spark_data.count()

335118
```

```
[ ] df = spark.read.csv("dataset.csv", header=True, inferSchema=True)
```

```
[ ] kenya_data = df.filter(df["country"] == "Kenya")
```

```
[ ] kenya_data.show(5)

+-------------+------------+-------+----------+---------+----------------+----------+-----------------+
|Date_reported|Country_code|Country|WHO_region|New_cases|Cumulative_cases|New_deaths|Cumulative_deaths|
+-------------+------------+-------+----------+---------+----------------+----------+-----------------+
|   2020-01-03|          KE|  Kenya|      AFRO|        0|               0|         0|                0|
|   2020-01-04|          KE|  Kenya|      AFRO|        0|               0|         0|                0|
|   2020-01-05|          KE|  Kenya|      AFRO|        0|               0|         0|                0|
|   2020-01-06|          KE|  Kenya|      AFRO|        0|               0|         0|                0|
|   2020-01-07|          KE|  Kenya|      AFRO|        0|               0|         0|                0|
+-------------+------------+-------+----------+---------+----------------+----------+-----------------+
only showing top 5 rows
```

4. Describe pre-processing tasks/techniques used to prepare the data (include screenshots) and give reasons to justify your choices

1. Collection and loading data into python: this is the first step in the data analysis process. Python with libraries like pandas, provide powerful tools for data manipulation and analysis. Using pandas to load data allows for efficient handling of datasets making it easy to perform various operations.
2. Isnull and unique values: isnull identifies and handles missing values (NaN values). This is essential because many machine learning algorithms cannot handle missing values. Unique values checks unique values in categorical columns helping in understanding the diversity of data. Useful for identifying cardinality of categorical features and deciding on encoding strategies. It assists in identifying potential issues like typos or inconsistencies in categorical data.
3. Filtering only values from Kenya: is essential when you are interested in analysing or modelling data for that specific entity. It helps to focus the analysis, gain insights into country-specific trends and is crucial for tasks like building country specific predictive models or generating reports tailored to a particular region.
4. Getting recovered by subtracting deaths from cases: assumes that individuals who aren't active cases are recovered. It provides a quick estimate and is useful for certain types of analyses, especially when detailed recovery data is not available.

```
df.describe()
```

	New_cases	Cumulative_cases	New_deaths	Cumulative_deaths
count	3.351180e+05	3.351180e+05	335118.000000	3.351180e+05
mean	2.303699e+03	1.535676e+06	20.827846	1.845291e+04
std	3.851160e+04	6.943032e+06	164.849599	7.687598e+04
min	-6.507900e+04	0.000000e+00	-3520.000000	0.000000e+00
25%	0.000000e+00	2.779000e+03	0.000000	2.100000e+01
50%	0.000000e+00	3.800800e+04	0.000000	4.110000e+02
75%	1.420000e+02	4.482015e+05	1.000000	5.938000e+03
max	6.966046e+06	1.034368e+08	43994.000000	1.138309e+06

```
[ ] df.isnull()
```

	Date_reported	Country_code	Country	WHO_region	New_cases	Cumulative_cases	New_deaths	Cumulative_deaths
0	False	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...	...
335113	False	False	False	False	False	False	False	False
335114	False	False	False	False	False	False	False	False
335115	False	False	False	False	False	False	False	False
335116	False	False	False	False	False	False	False	False
335117	False	False	False	False	False	False	False	False

335118 rows × 8 columns

```
[ ] df.isnull().value_counts()
```

```
Date_reported  Country_code  Country  WHO_region  New_cases  Cumulative_cases  New_deaths  Cumulative_deaths
False          False         False    False       False      False             False       False                333704
               True          False    False       False      False             False       False                  1414
dtype: int64
```

```
[ ] print(df.columns)
```

```
Index(['Date_reported', 'Country_code', 'Country', 'WHO_region', 'New_cases',
       'Cumulative_cases', 'New_deaths', 'Cumulative_deaths'],
      dtype='object')
```

```
[ ] unique_values = df['Country_code'].unique()
    print(unique_values)
```

```
['AF' 'AL' 'DZ' 'AS' 'AD' 'AO' 'AI' 'AG' 'AR' 'AM' 'AW' 'AU' 'AT' 'AZ'
 'BS' 'BH' 'BD' 'BB' 'BY' 'BE' 'BZ' 'BJ' 'BM' 'BT' 'BO' 'XA' 'BA' 'BW'
 'BR' 'VG' 'BN' 'BG' 'BF' 'BI' 'CV' 'KH' 'CM' 'CA' 'KY' 'CF' 'TD' 'CL'
 'CN' 'CO' 'KM' 'CG' 'CK' 'CR' 'CI' 'HR' 'CU' 'CW' 'CY' 'CZ' 'KP' 'CD'
 'DK' 'DJ' 'DM' 'DO' 'EC' 'EG' 'SV' 'GQ' 'ER' 'EE' 'SZ' 'ET' 'FK' 'FO'
 'FJ' 'FI' 'FR' 'GF' 'PF' 'GA' 'GM' 'GE' 'DE' 'GH' 'GI' 'GR' 'GL' 'GD'
 'GP' 'GU' 'GT' 'GG' 'GN' 'GW' 'GY' 'HT' 'VA' 'HN' 'HU' 'IS' 'IN' 'ID'
 'IR' 'IQ' 'IE' 'IM' 'IL' 'IT' 'JM' 'JP' 'JE' 'JO' 'KZ' 'KE' 'KI' 'XK'
 'KW' 'KG' 'LA' 'LV' 'LB' 'LS' 'LR' 'LY' 'LI' 'LT' 'LU' 'MG' 'MW' 'MY'
 'MV' 'ML' 'MT' 'MH' 'MQ' 'MR' 'MU' 'YT' 'MX' 'FM' 'MC' 'MN' 'ME' 'MS'
 'MA' 'MZ' 'MM' nan 'NR' 'NP' 'NL' 'NC' 'NZ' 'NI' 'NE' 'NG' 'NU' 'MK' 'MP'
 'NO' 'PS' 'OM' ' ' 'PK' 'PW' 'PA' 'PG' 'PY' 'PE' 'PH' 'PN' 'PL' 'PT' 'PR'
 'QA' 'KR' 'MD' 'RE' 'RO' 'RU' 'RW' 'XC' 'BL' 'SH' 'KN' 'LC' 'MF' 'PM'
 'VC' 'WS' 'SM' 'ST' 'SA' 'SN' 'RS' 'SC' 'SL' 'SG' 'XB' 'SX' 'SK' 'SI'
 'SB' 'SO' 'ZA' 'SS' 'ES' 'LK' 'SD' 'SR' 'SE' 'CH' 'SY' 'TJ' 'TH' 'GB'
 'TL' 'TG' 'TK' 'TO' 'TT' 'TN' 'TR' 'TM' 'TC' 'TV' 'UG' 'UA' 'AE' 'TZ'
 'US' 'VI' 'UY' 'UZ' 'VU' 'VE' 'VN' 'WF' 'YE' 'ZM' 'ZW']
```

```
filtered_df = df[df['Country_code'] == 'KE']
print(filtered_df)
```

```
        Date_reported Country_code Country WHO_region  New_cases  \
154126    2020-01-03           KE   Kenya       AFRO          0
154127    2020-01-04           KE   Kenya       AFRO          0
154128    2020-01-05           KE   Kenya       AFRO          0
154129    2020-01-06           KE   Kenya       AFRO          0
154130    2020-01-07           KE   Kenya       AFRO          0
...              ...          ...     ...        ...        ...
155535    2023-11-12           KE   Kenya       AFRO          0
155536    2023-11-13           KE   Kenya       AFRO          0
155537    2023-11-14           KE   Kenya       AFRO          0
155538    2023-11-15           KE   Kenya       AFRO          0
155539    2023-11-16           KE   Kenya       AFRO          0

        Cumulative_cases  New_deaths  Cumulative_deaths
154126                 0           0                  0
154127                 0           0                  0
154128                 0           0                  0
154129                 0           0                  0
154130                 0           0                  0
...                  ...         ...                ...
155535            344070           0               5689
155536            344070           0               5689
155537            344070           0               5689
155538            344070           0               5689
155539            344070           0               5689

[1414 rows x 8 columns]
```

```
# Assuming df is your DataFrame
df['Cumulative_recovered'] = df['Cumulative_cases'] - df['Cumulative_deaths']

# Print the updated DataFrame
print(df)
```

```
       Date_reported Country_code      Country WHO_region  New_cases  \
0         2020-01-03           AF  Afghanistan       EMRO          0
1         2020-01-04           AF  Afghanistan       EMRO          0
2         2020-01-05           AF  Afghanistan       EMRO          0
3         2020-01-06           AF  Afghanistan       EMRO          0
4         2020-01-07           AF  Afghanistan       EMRO          0
...              ...          ...          ...        ...        ...
335113    2023-11-12           ZW     Zimbabwe       AFRO          0
335114    2023-11-13           ZW     Zimbabwe       AFRO          0
335115    2023-11-14           ZW     Zimbabwe       AFRO          0
335116    2023-11-15           ZW     Zimbabwe       AFRO          0
335117    2023-11-16           ZW     Zimbabwe       AFRO          0

        Cumulative_cases  New_deaths  Cumulative_deaths  Cumulative_recovered
0                      0           0                  0                     0
1                      0           0                  0                     0
2                      0           0                  0                     0
3                      0           0                  0                     0
4                      0           0                  0                     0
...                  ...         ...                ...                   ...
335113            265821           0               5720                260101
335114            265821           0               5720                260101
335115            265821           0               5720                260101
335116            265821           0               5720                260101
335117            265821           0               5720                260101

[335118 rows x 9 columns]
```

```
df = df[(df != 0).any(axis=1)]

print(df)
```

```
       Date_reported Country_code      Country WHO_region  New_cases  \
0         2020-01-03           AF  Afghanistan       EMRO          0
1         2020-01-04           AF  Afghanistan       EMRO          0
2         2020-01-05           AF  Afghanistan       EMRO          0
3         2020-01-06           AF  Afghanistan       EMRO          0
4         2020-01-07           AF  Afghanistan       EMRO          0
...              ...          ...          ...        ...        ...
335113    2023-11-12           ZW     Zimbabwe       AFRO          0
335114    2023-11-13           ZW     Zimbabwe       AFRO          0
335115    2023-11-14           ZW     Zimbabwe       AFRO          0
335116    2023-11-15           ZW     Zimbabwe       AFRO          0
335117    2023-11-16           ZW     Zimbabwe       AFRO          0

        Cumulative_cases  New_deaths  Cumulative_deaths  Cumulative_recovered
0                      0           0                  0                     0
1                      0           0                  0                     0
2                      0           0                  0                     0
3                      0           0                  0                     0
4                      0           0                  0                     0
...                  ...         ...                ...                   ...
335113            265821           0               5720                260101
335114            265821           0               5720                260101
335115            265821           0               5720                260101
335116            265821           0               5720                260101
335117            265821           0               5720                260101

[335118 rows x 9 columns]
```

```
df = df.drop(columns=['Date_reported', 'Country_code', 'Country', 'WHO_region', 'New_cases'])

print(df)
```

```
[ ]
df = df.drop(columns=['Date_reported', 'Country_code', 'Country', 'WHO_region', 'New_cases'])

print(df)
```

```
        Cumulative_cases  New_deaths  Cumulative_deaths  Cumulative_recovered
0                      0           0                  0                     0
1                      0           0                  0                     0
2                      0           0                  0                     0
3                      0           0                  0                     0
4                      0           0                  0                     0
...                  ...         ...                ...                   ...
335113            265821           0               5720                260101
335114            265821           0               5720                260101
335115            265821           0               5720                260101
335116            265821           0               5720                260101
335117            265821           0               5720                260101

[335118 rows x 4 columns]
```

```
[ ] print(df.columns)
```

```
Index(['Cumulative_cases', 'New_deaths', 'Cumulative_deaths',
       'Cumulative_recovered'],
      dtype='object')
```

```
[ ] # Assuming df is your DataFrame
    df = df.drop(columns=['New_deaths'])
```

```
[ ] print(df.columns)
```

```
Index(['Cumulative_cases', 'Cumulative_deaths', 'Cumulative_recovered'], dtype='object')
```

```
df
```

	Cumulative_cases	Cumulative_deaths	Cumulative_recovered
0	0	0	0
1	0	0	0
2	0	0	0
3	0	0	0
4	0	0	0
...	...	...	...
335113	265821	5720	260101
335114	265821	5720	260101
335115	265821	5720	260101
335116	265821	5720	260101
335117	265821	5720	260101

335118 rows × 3 columns

```
[ ] df = df[(df != 0).all(axis=1)]
```

```
[ ] df
```

	Cumulative_cases	Cumulative_deaths	Cumulative_recovered
81	40	1	39
82	42	1	41
83	74	1	73
84	74	1	73
85	80	2	78
...	...	...	...
335113	265821	5720	260101
335114	265821	5720	260101
335115	265821	5720	260101
335116	265821	5720	260101
335117	265821	5720	260101

283577 rows × 3 columns

5. Test results and interpretations and Validation results and interpretations

** a) Number of Death cases or Mortality rate **

```
# Step 1: Select features and target variable
features = df[['Cumulative_cases', 'Cumulative_recovered']]
target = df['Cumulative_deaths']
```

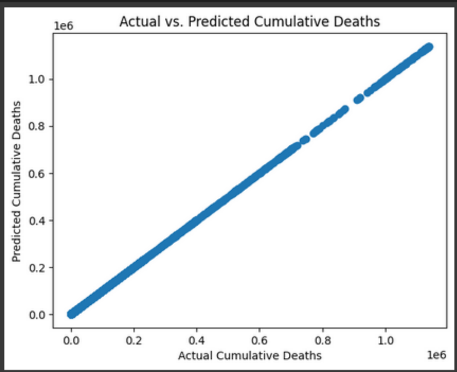
```
# Step 2: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(features, target, test_size=0.2, random_state=42)

# Step 3: Create and train the linear regression model
model = LinearRegression()
model.fit(X_train, y_train)
```

```
LinearRegression
LinearRegression()
```

```
# Step 4: Make predictions on the test set
y_pred = model.predict(X_test)

# Step 5: Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R-squared: {r2}')
```

```
Mean Squared Error: 6.559615166134512e-18
R-squared: 1.0
```

```
# Step 6: Visualize the predictions
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Cumulative Deaths')
plt.ylabel('Predicted Cumulative Deaths')
plt.title('Actual vs. Predicted Cumulative Deaths')
plt.show()
```



b) Predicting the Number of Confirmed Cases:

```
# Step 1: Select features and target variable
features_b = df[['Cumulative_deaths', 'Cumulative_recovered']]
target_b = df['Cumulative_cases']
```

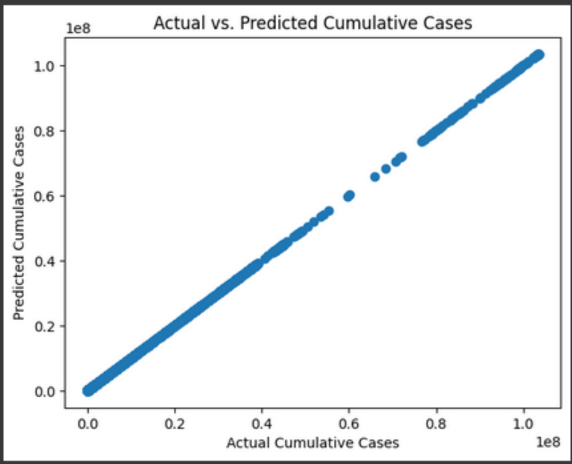
```
# Step 2: Split the data into training and testing sets
X_train_b, X_test_b, y_train_b, y_test_b = train_test_split(features_b, target_b, test_size=0.2, random_state=42)

# Step 3: Create and train the linear regression model
model_b = LinearRegression()
model_b.fit(X_train_b, y_train_b)
```

```
LinearRegression
LinearRegression()
```

```
# Step 4: Make predictions on the test set
y_pred_b = model_b.predict(X_test_b)

# Evaluate the model
mse_b = mean_squared_error(y_test_b, y_pred_b)
r2_b = r2_score(y_test_b, y_pred_b)

print(f'Mean Squared Error (Confirmed Cases): {mse_b}')
print(f'R-squared (Confirmed Cases): {r2_b}')
```

```
Mean Squared Error (Confirmed Cases): 5.932711230795542e-17
R-squared (Confirmed Cases): 1.0
```

```python
#  Visualize the predictions
plt.scatter(y_test_b, y_pred_b)
plt.xlabel('Actual Cumulative Cases')
plt.ylabel('Predicted Cumulative Cases')
plt.title('Actual vs. Predicted Cumulative Cases')
plt.show()
```



Predicting the Number of Recovery Cases or Recovery Rate:

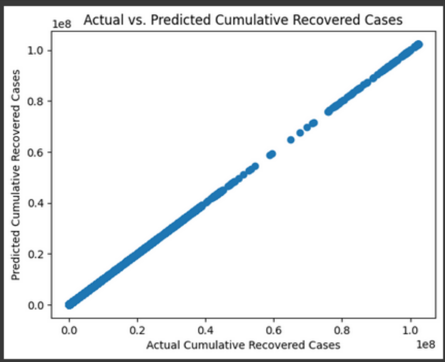
```python
#  Select features and target variable
features_c = df[['Cumulative_cases', 'Cumulative_deaths']]
target_c = df['Cumulative_recovered']
```

```python
#  Split the data into training and testing sets
X_train_c, X_test_c, y_train_c, y_test_c = train_test_split(features_c, target_c, test_size=0.2, random_state=42)

#  Create and train the linear regression model
model_c = LinearRegression()
model_c.fit(X_train_c, y_train_c)

#  Make predictions on the test set
y_pred_c = model_c.predict(X_test_c)
```

```python
#  Evaluate the model
mse_c = mean_squared_error(y_test_c, y_pred_c)
r2_c = r2_score(y_test_c, y_pred_c)

print(f'Mean Squared Error (Recovery Cases): {mse_c}')
print(f'R-squared (Recovery Cases): {r2_c}')
```

```
Mean Squared Error (Recovery Cases): 1.9915921893881187e-18
R-squared (Recovery Cases): 1.0
```

```
# Visualize the predictions
plt.scatter(y_test_c, y_pred_c)
plt.xlabel('Actual Cumulative Recovered Cases')
plt.ylabel('Predicted Cumulative Recovered Cases')
plt.title('Actual vs. Predicted Cumulative Recovered Cases')
plt.show()
```



MSE (Mean Squared Error):

MSE measures the average squared difference between the predicted values and the actual values. Lower MSE values indicate better model performance. In your results, you have extremely low MSE values (on the order of 1e-18 and 1e-17). This suggests that the predicted values are very close to the actual values, indicating a very accurate model. R-squared (Coefficient of Determination):

**R-squared* * is a statistical measure of how well the regression predictions approximate the real data points. It ranges from 0 to 1, where 1 indicates a perfect fit. Your R-squared values are 1.0, which is the highest possible value. This means that your model perfectly predicts the target variable based on the given features. Essentially, the model explains 100% of the variability in the target variable.

This regression models (for predicting Cumulative Cases, Cumulative Deaths, Cumulative Recovered, etc.) are performing exceptionally well based on these evaluation metrics. The low MSE values and R-squared values of 1.0 suggest that the models are accurately capturing the patterns in the data and making very precise predictions. However, it's essential to consider the context of your data and the nature of your predictive task to ensure that the model's performance aligns with your expectations and requirements. Additionally, be cautious of overfitting, especially if you have a small dataset.

6. Potential applications of the interpreted results

1. Public Health Policy Planning:

    - Mortality Rate Analysis: Understanding the mortality rate provides critical insights into the severity of the disease. Governments and health organizations can use this information to plan and implement public health policies, allocate resources effectively, and prepare healthcare systems for potential surges in cases.
    - Confirmed Cases Analysis: Analyzing the confirmed cases helps in assessing the overall disease burden. This information is valuable for resource allocation, determining testing and healthcare infrastructure needs, and understanding the scale of the outbreak.
    - Recovery Rate Analysis: Monitoring the recovery rate is crucial for evaluating the effectiveness of healthcare interventions. High recovery rates can indicate the success of treatment protocols and guide decisions on patient care.

2. Resource Allocation and Preparedness:

   - Mortality Rate Forecasting: Predictive analysis can be used to forecast mortality rates based on current trends. This information assists in proactively allocating medical resources, including hospital beds, ventilators, and medical personnel, to regions at higher risk.
   - Scenario Planning: Understanding potential future scenarios based on predictive analysis helps authorities plan for different outcomes. This includes estimating the number of hospitalizations, ICU admissions, and overall healthcare needs.

3. Community Awareness and Education:

   - Communication Strategies: Interpreted results can inform public communication strategies. Health authorities can use mortality rates, confirmed cases, and recovery rates to communicate the seriousness of the situation, promote preventive measures, and provide realistic expectations to the public.
   - Targeted Interventions: Knowing the mortality rate and distribution of confirmed cases allows for targeted interventions in specific communities or demographics. This can include focused testing campaigns, vaccination drives, and educational programs.

4. Research and Development:

   - Vaccine and Treatment Research: Predictive analysis can guide research efforts, helping prioritize vaccine development and treatment strategies. It can identify areas with high mortality rates where interventions are urgently needed.

- Epidemiological Studies: Understanding the patterns of confirmed cases and recovery rates contributes to epidemiological studies. Researchers can explore factors influencing the spread of the disease and identify characteristics associated with higher recovery rates.

5. International Collaboration:
   - Global Response Coordination: Sharing interpreted results internationally facilitates collaboration in response efforts. Countries with successful strategies can share insights and support regions facing challenges.
   - Policy Harmonization: Consistent data interpretation enables better coordination of policies and strategies across borders. This is especially important in the context of a global pandemic where international cooperation is essential.