Take a way cat.

MERCY NDIRITU	**SCT221-0474/2022**
DAVID MERITEI	**SCT221-0420/2022**
THOMAS MUTUKU	**SCT221-0674/2022**
BONFACE MUNENE	**SCT221-0401/2022**
SANDRA NYAMBURA	**SCT221-0410/2022**

We Compiled COVID-19 related Data(preferably data from Kenya) from this link

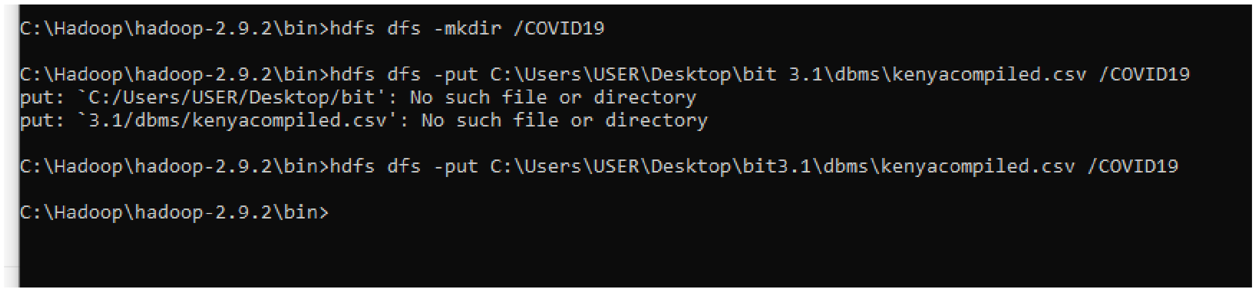
https://coronavirus.jhu.edu/

(i) Describe how the data was compiled in task 1 and include Screen captures of both code &and output) (3 Marks)

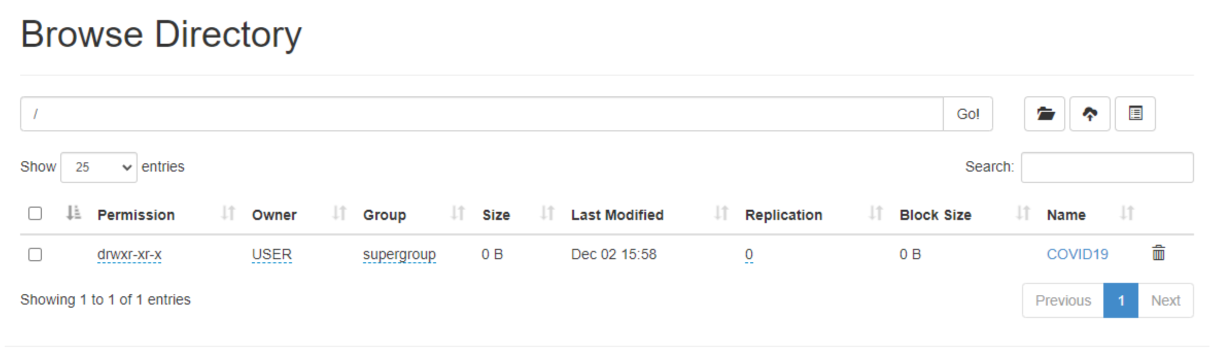
Data was compiled manually in excel by gathering all Kenya COVID19 data to one sheet

(ii) Describe how the data was ingested into Hadoop Data Lake and include screen shots. (3 Marks)

After making sure name node data node yarn are all working we made a directory in the bin and run the hdfs dfs command to fetch the data from our local disc c and the file is Kenya compiled which is a csv file



```
C:\Hadoop\hadoop-2.9.2\bin>hdfs dfs -mkdir /COVID19

C:\Hadoop\hadoop-2.9.2\bin>hdfs dfs -put C:\Users\USER\Desktop\bit 3.1\dbms\kenyacompiled.csv /COVID19
put: `C:/Users/USER/Desktop/bit': No such file or directory
put: `3.1/dbms/kenyacompiled.csv': No such file or directory

C:\Hadoop\hadoop-2.9.2\bin>hdfs dfs -put C:\Users\USER\Desktop\bit3.1\dbms\kenyacompiled.csv /COVID19

C:\Hadoop\hadoop-2.9.2\bin>
```



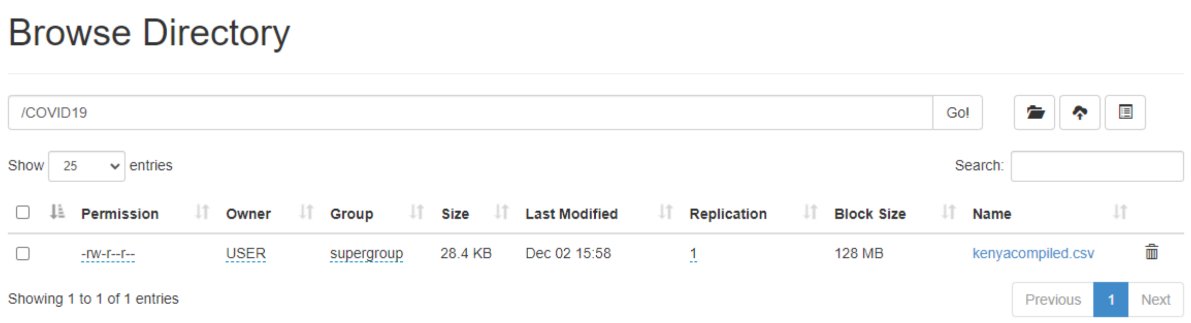
Browse Directory

/ Go!

Show 25 entries — Search:

	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
☐	drwxr-xr-x	USER	supergroup	0 B	Dec 02 15:58	0	0 B	COVID19

Showing 1 to 1 of 1 entries — Previous 1 Next



Browse Directory

/COVID19 Go!

Show 25 entries — Search:

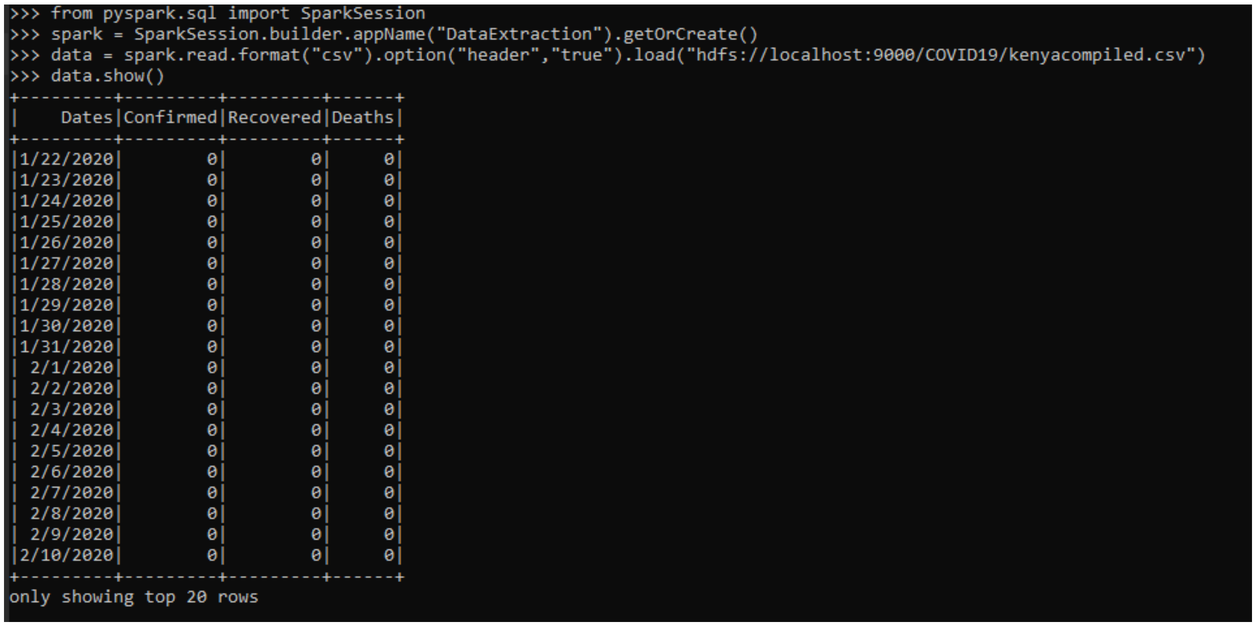
	Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
☐	-rw-r--r--	USER	supergroup	28.4 KB	Dec 02 15:58	1	128 MB	kenyacompiled.csv

Showing 1 to 1 of 1 entries — Previous 1 Next

(iii) Describe how data was extracted using pyspark and include associated screen shots (3 Marks)

You typically leverage the Spark ecosystem's capabilities to read data stored in Hadoop. Start by initializing a PySpark session

Use PySpark's DataFrame API to read data from HDFS and specified HDFS path check the loaded data by displaying a sample of the DataFrame.



```
>>> from pyspark.sql import SparkSession
>>> spark = SparkSession.builder.appName("DataExtraction").getOrCreate()
>>> data = spark.read.format("csv").option("header","true").load("hdfs://localhost:9000/COVID19/kenyacompiled.csv")
>>> data.show()
+---------+---------+---------+------+
|    Dates|Confirmed|Recovered|Deaths|
+---------+---------+---------+------+
|1/22/2020|        0|        0|     0|
|1/23/2020|        0|        0|     0|
|1/24/2020|        0|        0|     0|
|1/25/2020|        0|        0|     0|
|1/26/2020|        0|        0|     0|
|1/27/2020|        0|        0|     0|
|1/28/2020|        0|        0|     0|
|1/29/2020|        0|        0|     0|
|1/30/2020|        0|        0|     0|
|1/31/2020|        0|        0|     0|
| 2/1/2020|        0|        0|     0|
| 2/2/2020|        0|        0|     0|
| 2/3/2020|        0|        0|     0|
| 2/4/2020|        0|        0|     0|
| 2/5/2020|        0|        0|     0|
| 2/6/2020|        0|        0|     0|
| 2/7/2020|        0|        0|     0|
| 2/8/2020|        0|        0|     0|
| 2/9/2020|        0|        0|     0|
|2/10/2020|        0|        0|     0|
+---------+---------+---------+------+
only showing top 20 rows
```

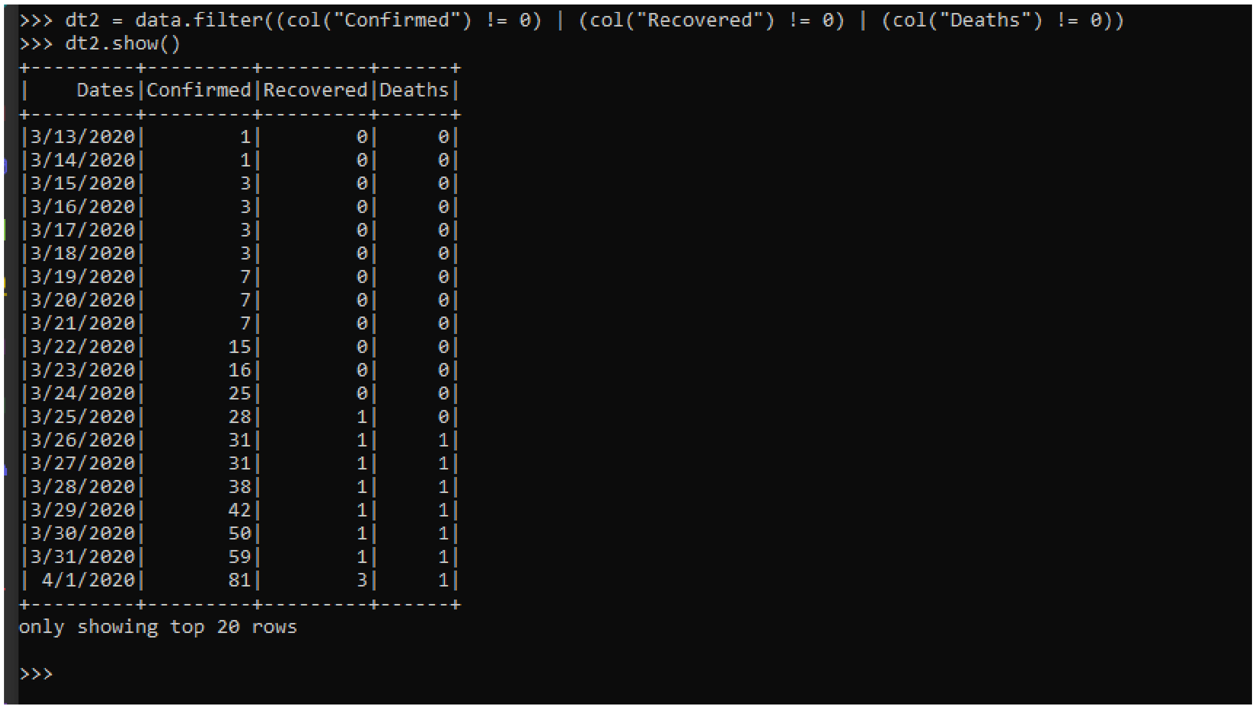
(iv) Describe pre-processing tasks/techniques used to prepare the data (include screen shots) and give reason (s) to justify your choices (3 Marks).

Data cleaning - Removing data with zero values is a common data cleaning task, especially when dealing with numerical data

1.The data we used had a couple of zero values

2.Avoiding Skewing of Statistical Measures:

3. Division by zero is undefined and can lead to errors or infinite values**.**



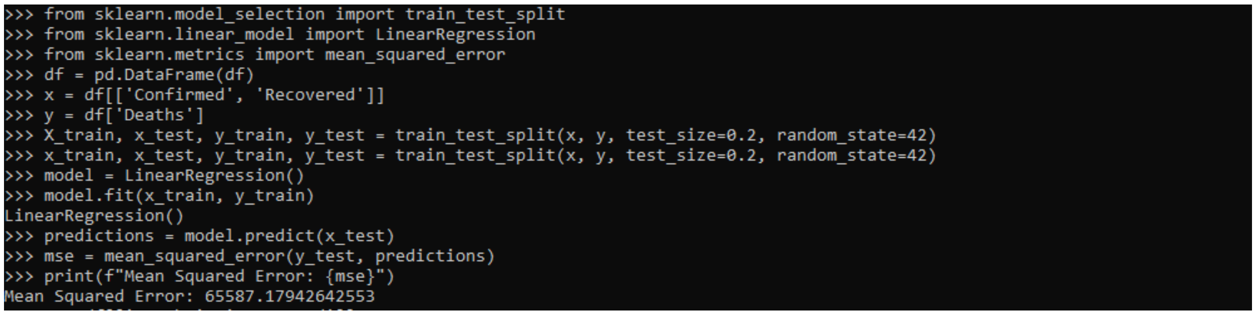
```
>>> dt2 = data.filter((col("Confirmed") != 0) | (col("Recovered") != 0) | (col("Deaths") != 0))
>>> dt2.show()
+---------+---------+---------+------+
|    Dates|Confirmed|Recovered|Deaths|
+---------+---------+---------+------+
|3/13/2020|        1|        0|     0|
|3/14/2020|        1|        0|     0|
|3/15/2020|        3|        0|     0|
|3/16/2020|        3|        0|     0|
|3/17/2020|        3|        0|     0|
|3/18/2020|        3|        0|     0|
|3/19/2020|        7|        0|     0|
|3/20/2020|        7|        0|     0|
|3/21/2020|        7|        0|     0|
|3/22/2020|       15|        0|     0|
|3/23/2020|       16|        0|     0|
|3/24/2020|       25|        0|     0|
|3/25/2020|       28|        1|     0|
|3/26/2020|       31|        1|     1|
|3/27/2020|       31|        1|     1|
|3/28/2020|       38|        1|     1|
|3/29/2020|       42|        1|     1|
|3/30/2020|       50|        1|     1|
|3/31/2020|       59|        1|     1|
| 4/1/2020|       81|        3|     1|
+---------+---------+---------+------+
only showing top 20 rows

>>>
```

v) Test results and interpretation

Prediction model

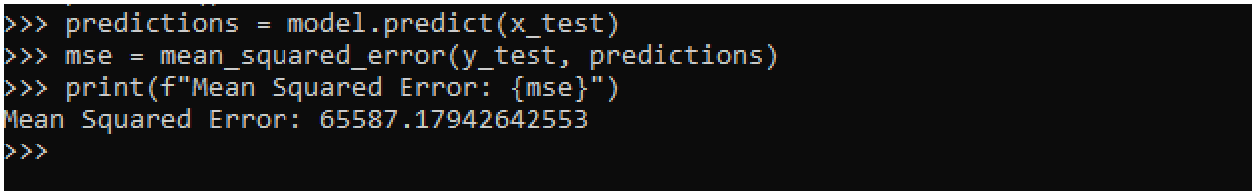
Test results show the model is working well and predicts the number of death cases or Mortality rate against number of confirmed cases and the recovered cases

We tested using Mean squared error and root mean



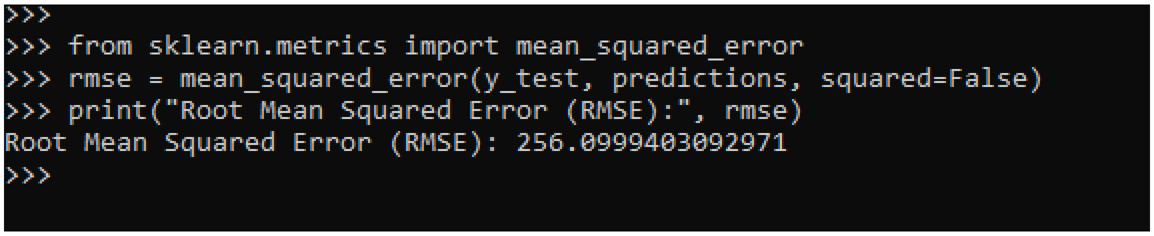
```
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.linear_model import LinearRegression
>>> from sklearn.metrics import mean_squared_error
>>> df = pd.DataFrame(df)
>>> x = df[['Confirmed', 'Recovered']]
>>> y = df['Deaths']
>>> X_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
>>> x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
>>> model = LinearRegression()
>>> model.fit(x_train, y_train)
LinearRegression()
>>> predictions = model.predict(x_test)
>>> mse = mean_squared_error(y_test, predictions)
>>> print(f"Mean Squared Error: {mse}")
Mean Squared Error: 65587.17942642553
```

Mean squared error



```
>>> predictions = model.predict(x_test)
>>> mse = mean_squared_error(y_test, predictions)
>>> print(f"Mean Squared Error: {mse}")
Mean Squared Error: 65587.17942642553
>>>
```

Root mean

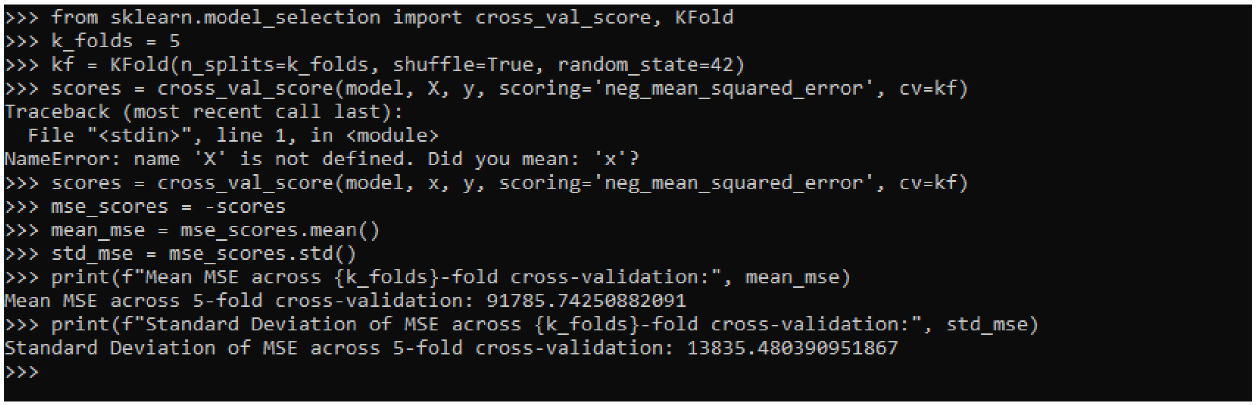


```
>>>
>>> from sklearn.metrics import mean_squared_error
>>> rmse = mean_squared_error(y_test, predictions, squared=False)
>>> print("Root Mean Squared Error (RMSE):", rmse)
Root Mean Squared Error (RMSE): 256.0999403092971
>>>
```

vi) Validation results and interpretation

We validated the model to perform k-fold cross-validation for a regression model using scikit-learn's cross_val_score and KFold classes.

- The mean MSE provides an estimate of the overall predictive accuracy of the model, with lower values indicating better performance.
- The standard deviation of the MSE gives insights into the variability of the model's performance across different folds. A lower standard deviation suggests more consistent performance.



```
>>> from sklearn.model_selection import cross_val_score, KFold
>>> k_folds = 5
>>> kf = KFold(n_splits=k_folds, shuffle=True, random_state=42)
>>> scores = cross_val_score(model, X, y, scoring='neg_mean_squared_error', cv=kf)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
NameError: name 'X' is not defined. Did you mean: 'x'?
>>> scores = cross_val_score(model, x, y, scoring='neg_mean_squared_error', cv=kf)
>>> mse_scores = -scores
>>> mean_mse = mse_scores.mean()
>>> std_mse = mse_scores.std()
>>> print(f"Mean MSE across {k_folds}-fold cross-validation:", mean_mse)
Mean MSE across 5-fold cross-validation: 91785.74250882091
>>> print(f"Standard Deviation of MSE across {k_folds}-fold cross-validation:", std_mse)
Standard Deviation of MSE across 5-fold cross-validation: 13835.480390951867
>>>
```

vii) Potential application of interpreted results

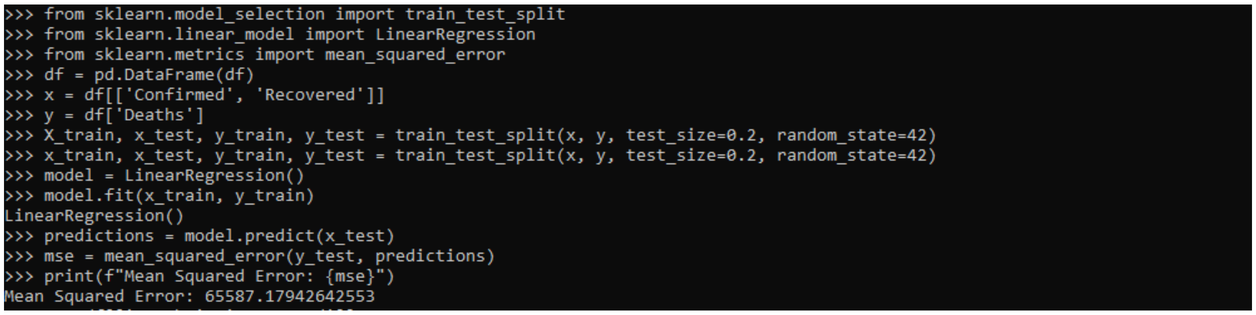
Mean Squared Error (MSE):

65587.17942642553

- **Interpretation:**
    - The MSE represents the average squared difference between the predicted and actual values of the number of deaths in the test set.
    - A lower MSE indicates better predictive performance. In this case, the model's predictions, on average, have a squared error of approximately 65587.18.

Health Application:

- The linear regression model, trained on features Confirmed Cases and Recovered Cases, attempts to predict the number of deaths.
- The MSE, being a measure of prediction accuracy, suggests how well the model is capturing the variability in death counts based on the provided features.



```
>>> from sklearn.model_selection import train_test_split
>>> from sklearn.linear_model import LinearRegression
>>> from sklearn.metrics import mean_squared_error
>>> df = pd.DataFrame(df)
>>> x = df[['Confirmed', 'Recovered']]
>>> y = df['Deaths']
>>> X_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
>>> x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
>>> model = LinearRegression()
>>> model.fit(x_train, y_train)
LinearRegression()
>>> predictions = model.predict(x_test)
>>> mse = mean_squared_error(y_test, predictions)
>>> print(f"Mean Squared Error: {mse}")
Mean Squared Error: 65587.17942642553
```