COVID-19 Data Analysis Using Hadoop

ARUMAI STALIN MILTON 22BIT0677

SUDHARSANAN 22BIT0676

SARAVANAN 22BIT0663

Abstract

The COVID-19 pandemic has led to the generation of massive datasets involving infection rates, recoveries, fatalities, which traditional systems struggle to process efficiently. This project introduces a scalable big data solution using the Hadoop ecosystem to manage and analyse COVID-19 data. By utilizing HDFS for distributed storage, MapReduce for batch processing, Hive for structured querying, and the system uncovers valuable insights into pandemic trends. Data visualization is performed using Python libraries, providing clear and interactive representations of the analysis. Additionally, machine learning is applied using PySpark's Gradient Boosted Tree (GBT) regressor to predict future case trends. The proposed solution demonstrates the effectiveness of big data technologies in supporting data-driven decisions during health emergencies.

Introduction

The COVID-19 pandemic has led to an unprecedented surge in data related to infections, recoveries, deaths, and vaccinations across the globe. Traditional data processing techniques struggle to handle such vast datasets efficiently. To address this challenge, this project leverages **Hadoop and its ecosystem** to process and analyze COVID-19 data in a scalable manner. By integrating **HDFS, MapReduce, Hive, Apache Spark, and Apache Kafka**, we ensure efficient storage, batch processing, real-time analytics, and visualization of pandemic trends.

Applications

- Epidemic forecasting and alert systems

- Resource planning (hospital beds, ventilators, vaccines)
- Regional policy optimization based on case trends
- Early intervention and containment strategies

Significance

The integration of ML with Big Data enhances real-time decision-making and opens new frontiers for epidemic surveillance and proactive healthcare management.

Problem Definition

The primary challenge in COVID-19 data analysis is the **scalability and efficiency** of data processing. Large datasets consisting of daily case reports, and death counts require optimized handling. Traditional relational databases fail to process such data at scale, necessitating a big data approach. Key challenges include:

- Managing and storing large COVID-19 datasets.
- Processing batch and real-time data efficiently.
- Enabling structured querying and fast computations.
- Providing meaningful insights for policymakers and researchers.

Objectives

The main objectives of this project are:

- To collect and store COVID-19 data efficiently using **HDFS**.
- To process large-scale data using **MapReduce**.
- To enable structured data querying with **Hive**.
- To visualize insights using **Python library.**
- To predict using **PySpark.**

Data Collection

The dataset contains 49,068 records (rows).

The dataset is sourced from authoritative sources such as:

- **Kaggle repositories** – Community-provided structured datasets.

Dataset Attributes

- **Date (reporting date)**
- **Country/Region**
- **Confirmed cases**
- **Deaths**
- **Recovered**
- **Active cases**
- **WHO region**

Path: hdfs://localhost:9000/covid19_data/covid_19.csv

Literature Review

1. Literature Review 1: Forecasting COVID-19 Cases Using Machine Learning Algorithms

Reference: Chakraborty, I., & Ghosh, S. (2020). Real-time forecasts and risk assessment of novel coronavirus (COVID-19) cases: A data-driven analysis. Chaos, Solitons & Fractals.

Summary:
This study applied data-driven forecasting techniques such as ARIMA and Random Forests to predict COVID-19 spread. The authors emphasized the importance of temporal features like date and region in modeling. Their findings showed that ensemble methods, particularly Random Forests, provided more reliable predictions compared to simple statistical models.

Relevance to Our Project:
This aligns with our project's observation where Random Forests performed well, but Gradient Boosted Trees outperformed due to better error handling and sequential optimization.

2. Literature Review 2: Comparative Study of Regression Models on COVID-19 Dataset

Reference: Rustam, F., Reshi, A. A., & Mehmood, A. (2020). COVID-19 future forecasting using supervised machine learning models. IEEE Access.

Summary:
The paper compared multiple regression models—Linear Regression, Decision Tree, Random Forest, and Gradient Boosting—for COVID-19 case prediction. Gradient Boosted Trees achieved the best results in terms of RMSE and R^2. It was noted that

simpler models like Linear Regression underperformed due to the non-linearity in the pandemic data.

Relevance to Our Project:
Our findings are consistent with this paper, where Linear Regression yielded the lowest R^2 value, and GBT showed superior performance across all evaluation metrics.

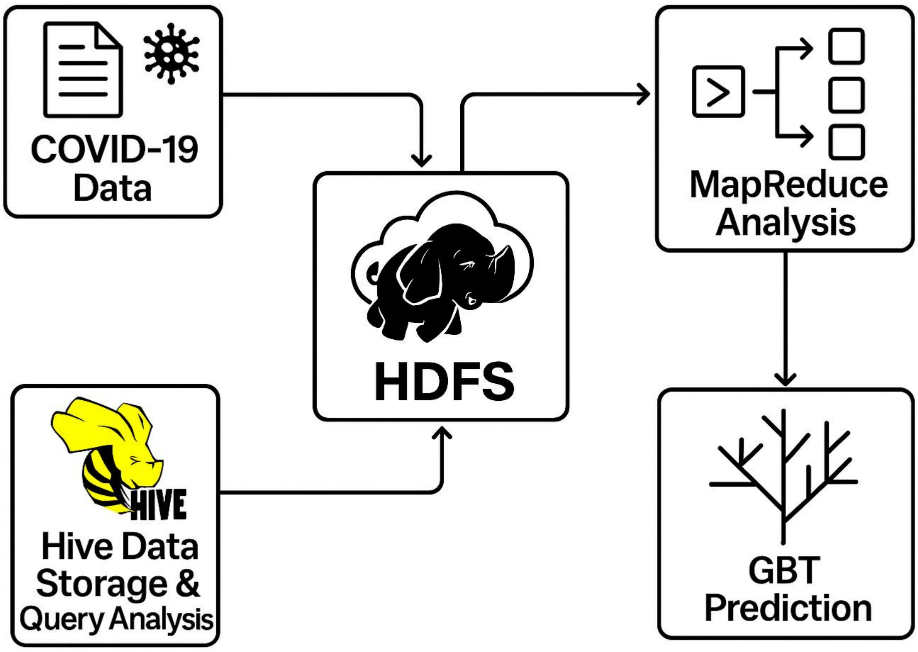
3. Literature Review 3: Big Data Frameworks for COVID-19 Analysis

Reference: Syed, A., & Zameer, A. (2021). A review on big data frameworks for COVID-19 analysis. Journal of King Saud University – Computer and Information Sciences.

Summary:
This review highlighted the role of Big Data ecosystems (Hadoop, HDFS, Spark, Hive) in pandemic tracking and modeling. It stressed the scalability and distributed nature of Spark for real-time data processing and ML integration. The review also recommended using Spark MLlib for massive-scale model training.

Relevance to Our Project:
Our architecture precisely follows this recommendation, using HDFS for storage and PySpark MLlib for scalable training and prediction—making it a real-world application of the proposed framework.

Architecture of the project



 Data Collection: Collects COVID-19 data from sources like health records, social media, and government reports.

 Data Storage (HDFS & Hive): Raw data is stored in **HDFS**, while **Hive** enables structured storage and querying.

 Preprocessing & Cleaning: Data is formatted, cleaned, and structured using **Hive queries**.

 Hadoop MapReduce (Processing Layer):

- **Mapper**: Converts structured data into key-value pairs for analysis.
- **Reducer**: Aggregates insights like infection trends and mortality rates.

 Feature Extraction: Extracts relevant features (date, deaths, confirmed, recovered) for predictions.

 Gradient Boosted Trees: Trained with processed data to predict.

 Model Evaluation: Accuracy and performance of the model are validated using test datasets.

 Query & Analysis (Hive): Hive is used to run complex queries for further insights.

 Visualization & Reporting: Dashboards display trends, predictions, and query results.

 Decision Support: Data-driven insights help policymakers in resource allocation and pandemic control.

Chosen Methodology: Among the tested models:

Decision Tree: Moderate performance, easy interpretability

Linear Regression: Poor performance due to non-linearity of data

Gradient Boosted Trees (GBT): High accuracy, handled region-wise complexity well

Methodology Chosen: Gradient Boosted Trees

Why GBT?

It yielded the lowest RMSE and MAE.

It achieved the highest R^2 value (0.9816), indicating strong predictive power.

Well-suited for tabular datasets with temporal and regional features.

Implementation Process

Data Storage Using HDFS

- The collected dataset is stored in **HDFS** for distributed storage.

Data Processing Using MapReduce

MapReduce **processes large datasets in parallel by extracting relevant fields and aggregating results to generate useful insights.**

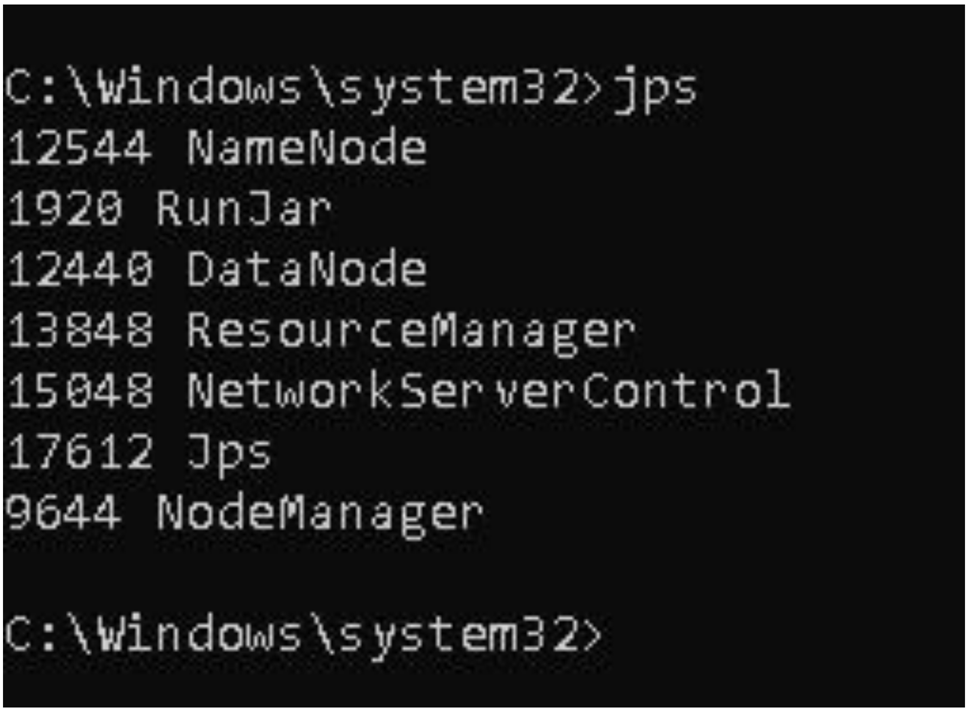
Querying with Hive

Apache Hive is used for structured querying, allowing for the retrieval of critical pandemic-related data such as total cases, country-wise trends, and vaccination statistics.

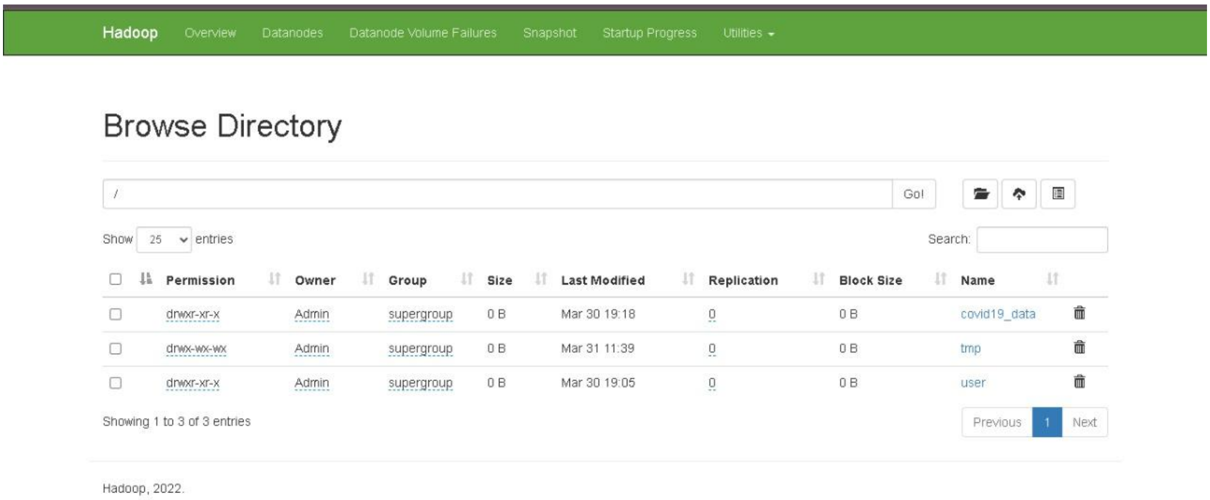
Results and Insights

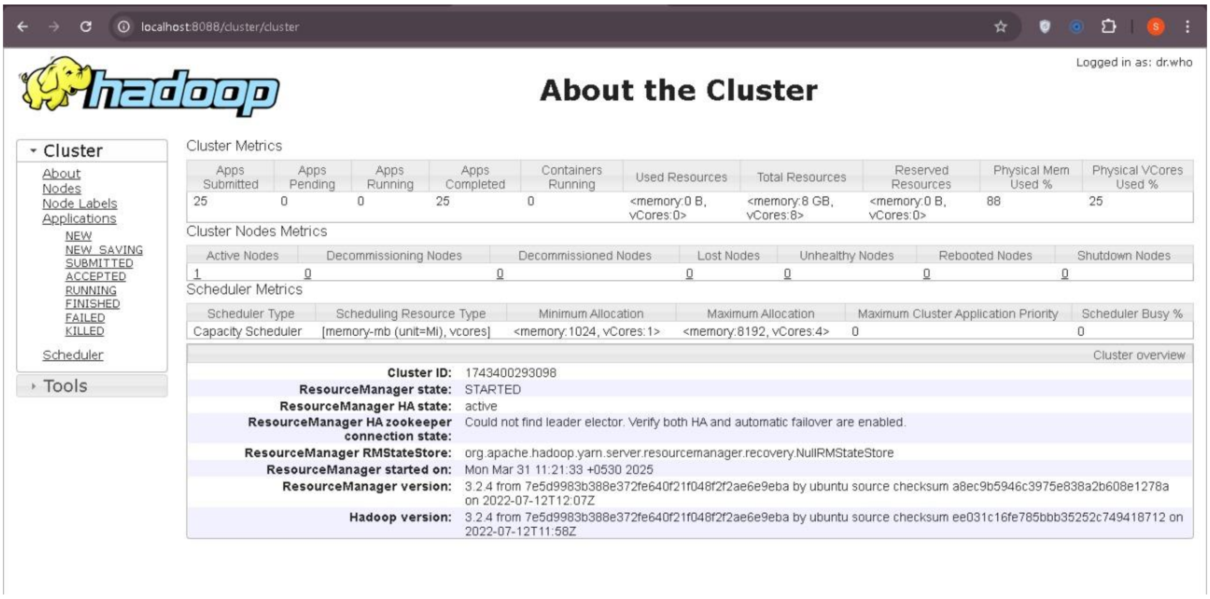
- **Trend Analysis:** Daily new cases and recoveries.
- **Geographical Impact:** Country-wise analysis of pandemic spread.
- **Effectiveness of Measures:** Impact of lockdowns and vaccinations.

Deamon processes

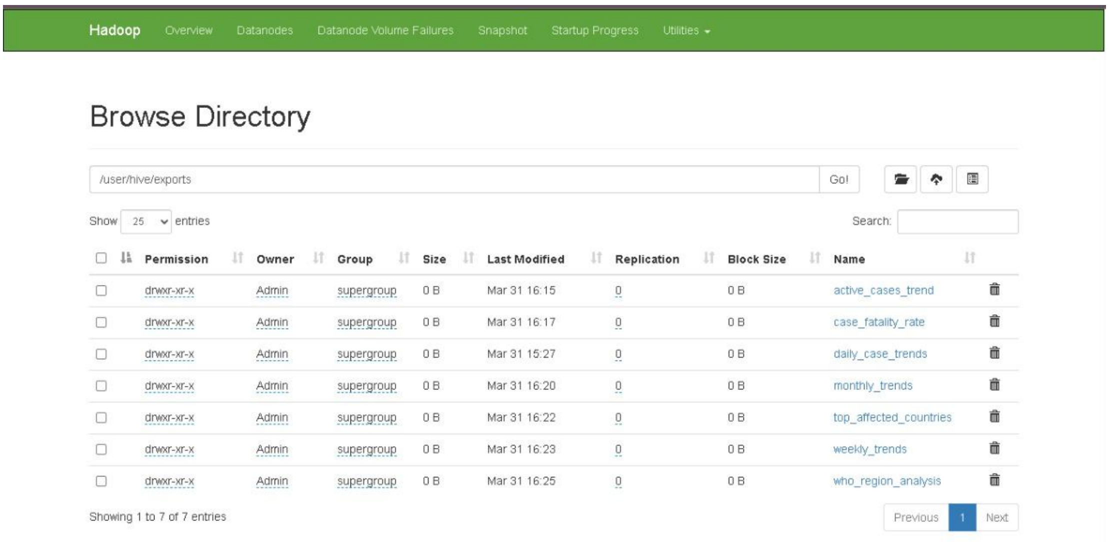


HDFS file upload




localhost:8088/cluster/cluster
Logged in as: dr.who
hadoop
About the Cluster
Cluster
About
Nodes
Node Labels
Applications
NEW
NEW_SAVING
SUBMITTED
ACCEPTED
RUNNING
FINISHED
FAILED
KILLED
Scheduler
Tools
Cluster Metrics
Apps Submitted | Apps Pending | Apps Running | Apps Completed | Containers Running | Used Resources | Total Resources | Reserved Resources | Physical Mem Used % | Physical VCores Used %
25 | 0 | 0 | 25 | 0 | <memory:0 B, vCores:0> | <memory:8 GB, vCores:8> | <memory:0 B, vCores:0> | 88 | 25
Cluster Nodes Metrics
Active Nodes | Decommissioning Nodes | Decommissioned Nodes | Lost Nodes | Unhealthy Nodes | Rebooted Nodes | Shutdown Nodes
1 | 0 | 0 | 0 | 0 | 0 | 0
Scheduler Metrics
Scheduler Type | Scheduling Resource Type | Minimum Allocation | Maximum Allocation | Maximum Cluster Application Priority | Scheduler Busy %
Capacity Scheduler | [memory-mb (unit=Mi), vcores] | <memory:1024, vCores:1> | <memory:8192, vCores:4> | 0 | 0
Cluster overview
Cluster ID: 1743400293098
ResourceManager state: STARTED
ResourceManager HA state: active
ResourceManager HA zookeeper connection state: Could not find leader elector. Verify both HA and automatic failover are enabled.
ResourceManager RMStateStore: org.apache.hadoop.yarn.server.resourcemanager.recovery.NullRMStateStore
ResourceManager started on: Mon Mar 31 11:21:33 +0530 2025
ResourceManager version: 3.2.4 from 7e5d9983b388e372fe640f21f048f2f2ae6e9eba by ubuntu source checksum a8ec9b5946c3975e838a2b608e1278a on 2022-07-12T12:07Z
Hadoop version: 3.2.4 from 7e5d9983b388e372fe640f21f048f2f2ae6e9eba by ubuntu source checksum ee031c16fe785bbb35252c749418712 on 2022-07-12T11:58Z

Hive tables


Hadoop
Overview
Datanodes
Datanode Volume Failures
Snapshot
Startup Progress
Utilities
Browse Directory
/user/hive/exports
Go!
Show 25 entries
Search:
Permission | Owner | Group | Size | Last Modified | Replication | Block Size | Name
drwxr-xr-x | Admin | supergroup | 0 B | Mar 31 16:15 | 0 | 0 B | active_cases_trend
drwxr-xr-x | Admin | supergroup | 0 B | Mar 31 16:17 | 0 | 0 B | case_fatality_rate
drwxr-xr-x | Admin | supergroup | 0 B | Mar 31 15:27 | 0 | 0 B | daily_case_trends
drwxr-xr-x | Admin | supergroup | 0 B | Mar 31 16:20 | 0 | 0 B | monthly_trends
drwxr-xr-x | Admin | supergroup | 0 B | Mar 31 16:22 | 0 | 0 B | top_affected_countries
drwxr-xr-x | Admin | supergroup | 0 B | Mar 31 16:23 | 0 | 0 B | weekly_trends
drwxr-xr-x | Admin | supergroup | 0 B | Mar 31 16:25 | 0 | 0 B | who_region_analysis
Showing 1 to 7 of 7 entries
Previous
1
Next

```
hive> show tables ;
2025-03-31T21:58:12,657 INFO [main] org.apache.hadoop.hive.conf.HiveConf - Using the default value passed in for log id: c88cbf10-625d-4409-9ed6-9c8253c55d3c
2025-03-31T21:58:12,668 INFO [main] org.apache.hadoop.hive.ql.session.SessionState - Updating thread name to c88cbf10-625d-4409-9ed6-9c8253c55d3c main
OK
active_cases_trend
case_fatality_rate
covid19
covid19_data
daily_case_trends
monthly_trends
top_affected_countries
weekly_trends
who_region_analysis
Time taken: 0.376 seconds, Fetched: 9 row(s)
2025-03-31T21:58:13,147 INFO [c88cbf10-625d-4409-9ed6-9c8253c55d3c main] org.apache.hadoop.hive.conf.HiveConf - Using the default value passed in for log id: c88cbf10-6
25d-4409-9ed6-9c8253c55d3c
2025-03-31T21:58:13,147 INFO [c88cbf10-625d-4409-9ed6-9c8253c55d3c main] org.apache.hadoop.hive.ql.session.SessionState - Resetting thread name to  main
hive>
```

Preprocessing

```
hive> SELECT COUNT(*) AS negative_values_count
    > FROM covid19_data
    > WHERE confirmed < 0 OR deaths < 0 OR recovered < 0;
2025-03-31T11:44:24,903 INFO [main] org.apache.hadoop.hive.conf.HiveConf - Using the default value passed in for log id: c88cbf10-625d-4409-9ed6-9c8253c55d3c
2025-03-31T11:44:24,903 INFO [main] org.apache.hadoop.hive.ql.session.SessionState - Updating thread name to c88cbf10-625d-4409-9ed6-9c8253c55d3c main
2025-03-31T11:44:25,208 INFO [c88cbf10-625d-4409-9ed6-9c8253c55d3c main] org.apache.hadoop.hive.common.FileUtils - Creating directory if it doesn't exist: hdfs://localh
ost:9000/tmp/hive/Admin/c88cbf10-625d-4409-9ed6-9c8253c55d3c/hive_2025-03-31_11-44-24_927_1938100302731786785-1/-mr-10001/.hive-staging_hive_2025-03-31_11-44-24_927_193
8100302731786785-1
Query ID = Admin_20250331114424_d840becb-c05e-47ca-bb86-ac746b615e6a
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1743400293098_0002, Tracking URL = http://Sudharsanan:8088/proxy/application_1743400293098_0002/
Kill Command = C:\hadoop\bin\mapred job  -kill job_1743400293098_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-03-31 11:44:43,061 Stage-1 map = 0%,  reduce = 0%
2025-03-31 11:44:52,589 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 5.999 sec
2025-03-31 11:45:02,123 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 11.591 sec
MapReduce Total cumulative CPU time: 11 seconds 591 msec
Ended Job = job_1743400293098_0002
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 11.591 sec   HDFS Read: 3321551 HDFS Write: 101 SUCCESS
Total MapReduce CPU Time Spent: 11 seconds 591 msec
OK
0
Time taken: 43.606 seconds, Fetched: 1 row(s)
2025-03-31T11:45:08,561 INFO [c88cbf10-625d-4409-9ed6-9c8253c55d3c main] org.apache.hadoop.hive.conf.HiveConf - Using the default value passed in for log id: c88cbf10-6
25d-4409-9ed6-9c8253c55d3c
2025-03-31T11:45:08,562 INFO [c88cbf10-625d-4409-9ed6-9c8253c55d3c main] org.apache.hadoop.hive.ql.session.SessionState - Resetting thread name to  main
hive>
```

```
hive> SELECT COUNT(*) FROM covid19_data
    > WHERE province_state IS NULL
    >    OR country_region IS NULL
    >    OR reported_date IS NULL
    >    OR confirmed IS NULL;
2025-03-31T11:39:10,458 INFO [main] org.apache.hadoop.hive.conf.HiveConf - Using the default value passed in for log id: c88cbf10-625d-4409-9ed6-9c8253c55d3c
2025-03-31T11:39:10,458 INFO [main] org.apache.hadoop.hive.ql.session.SessionState - Updating thread name to c88cbf10-625d-4409-9ed6-9c8253c55d3c main
2025-03-31T11:39:11,458 INFO [c88cbf10-625d-4409-9ed6-9c8253c55d3c main] org.apache.hadoop.hive.common.FileUtils - Creating directory if it doesn't exist: hdfs://localh
ost:9000/tmp/hive/Admin/c88cbf10-625d-4409-9ed6-9c8253c55d3c/hive_2025-03-31_11-39-10_479_1722502916930328771-1/-mr-10001/.hive-staging_hive_2025-03-31_11-39-10_479_1722
50291693032877-1
Query ID = Admin_20250331113910_c96f24de-a80a-45db-8017-2e430ff5d2cf
Total jobs = 1
Launching Job 1 out of 1
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
2025-03-31T11:39:12,708 INFO [c88cbf10-625d-4409-9ed6-9c8253c55d3c main] org.apache.hadoop.conf.Configuration.deprecation - mapred.submit.replication is deprecated. Ins
tead, use mapreduce.client.submit.file.replication
2025-03-31T11:39:14,699 INFO [c88cbf10-625d-4409-9ed6-9c8253c55d3c main] org.apache.hadoop.conf.Configuration.deprecation - yarn.resourcemanager.system-metrics-publishe
r.enabled is deprecated. Instead, use yarn.system-metrics-publisher.enabled
2025-03-31T11:39:15,165 INFO [c88cbf10-625d-4409-9ed6-9c8253c55d3c main] org.apache.hadoop.conf.Configuration - resource-types.xml not found
Starting Job = job_1743400293098_0001, Tracking URL = http://Sudharsanan:8088/proxy/application_1743400293098_0001/
Kill Command = C:\hadoop\bin\mapred job  -kill job_1743400293098_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-03-31 11:39:38,354 Stage-1 map = 0%,  reduce = 0%
2025-03-31 11:39:49,040 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.936 sec
2025-03-31 11:39:59,710 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 12.747 sec
MapReduce Total cumulative CPU time: 12 seconds 747 msec
Ended Job = job_1743400293098_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 12.747 sec   HDFS Read: 3321379 HDFS Write: 101 SUCCESS
Total MapReduce CPU Time Spent: 12 seconds 747 msec
OK
0
Time taken: 56.882 seconds, Fetched: 1 row(s)
2025-03-31T11:40:07,723 INFO [c88cbf10-625d-4409-9ed6-9c8253c55d3c main] org.apache.hadoop.hive.conf.HiveConf - Using the default value passed in for log id: c88cbf10-6
```

Analysation

Active case trends

```
hive> CREATE TABLE active_cases_trend AS
    > SELECT reported_date,
    >        SUM(active) AS total_active_cases
    > FROM covid19_data
    > GROUP BY reported_date
    > ORDER BY reported_date ASC;
2025-03-31T12:18:20,249 INFO [main] org.apache.hadoop.hive.conf.HiveConf - Using the default value passed in for log id: c88cbf10-625d-4409-9ed6-9c8253c55d3c
2025-03-31T12:18:20,249 INFO [main] org.apache.hadoop.hive.ql.session.SessionState - Updating thread name to c88cbf10-625d-4409-9ed6-9c8253c55d3c main
2025-03-31T12:18:20,322 INFO [c88cbf10-625d-4409-9ed6-9c8253c55d3c main] org.apache.hadoop.hive.common.FileUtils - Creating directory if it doesn't exist: hdfs://localh
ost:9000/user/hive/warehouse/.hive-staging_hive_2025-03-31_12-18-20_269_7183367165242453470-1
Query ID = Admin_20250331121820_24f11e58-d729-42f1-9a8d-1a019f77d5f3
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1743400293098_0015, Tracking URL = http://Sudharsanan:8088/proxy/application_1743400293098_0015/
Kill Command = C:\hadoop\bin\mapred job  -kill job_1743400293098_0015
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-03-31 12:18:42,605 Stage-1 map = 0%,  reduce = 0%
2025-03-31 12:18:54,327 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.372 sec
2025-03-31 12:19:05,896 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 11.261 sec
MapReduce Total cumulative CPU time: 11 seconds 261 msec
Ended Job = job_1743400293098_0015
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1743400293098_0016, Tracking URL = http://Sudharsanan:8088/proxy/application_1743400293098_0016/
Kill Command = C:\hadoop\bin\mapred job  -kill job_1743400293098_0016
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2025-03-31 12:19:36,490 Stage-2 map = 0%,  reduce = 0%
```

Case fatality rate

```
hive> CREATE TABLE case_fatality_rate AS
    > SELECT country_region,
    >        SUM(deaths) * 100.0 / SUM(confirmed) AS CFR_Percentage
    > FROM covid19_data
    > WHERE confirmed > 0
    > GROUP BY country_region
    > ORDER BY CFR_Percentage DESC;
2025-03-31T12:11:21,283 INFO [main] org.apache.hadoop.hive.conf.HiveConf - Using the default value passed in for log id: c88cbf10-625d-4409-9ed6-9c8253c55d3c
2025-03-31T12:11:21,283 INFO [main] org.apache.hadoop.hive.ql.session.SessionState - Updating thread name to c88cbf10-625d-4409-9ed6-9c8253c55d3c main
2025-03-31T12:11:21,353 INFO [c88cbf10-625d-4409-9ed6-9c8253c55d3c main] org.apache.hadoop.hive.common.FileUtils - Creating directory if it doesn't exist: hdfs://localh
ost:9000/user/hive/warehouse/.hive-staging_hive_2025-03-31_12-11-21_303_7645010681447861921-1
Query ID = Admin_20250331121121_47109e3a-5d9c-4e84-aa97-36d205c8462e
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1743400293098_0013, Tracking URL = http://Sudharsanan:8088/proxy/application_1743400293098_0013/
Kill Command = C:\hadoop\bin\mapred job  -kill job_1743400293098_0013
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-03-31 12:11:38,728 Stage-1 map = 0%,  reduce = 0%
2025-03-31 12:11:49,131 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.842 sec
2025-03-31 12:11:59,551 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 13.184 sec
MapReduce Total cumulative CPU time: 13 seconds 184 msec
Ended Job = job_1743400293098_0013
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1743400293098_0014, Tracking URL = http://Sudharsanan:8088/proxy/application_1743400293098_0014/
Kill Command = C:\hadoop\bin\mapred job  -kill job_1743400293098_0014
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2025-03-31 12:12:29,696 Stage-2 map = 0%,  reduce = 0%
```

Daily analysis trend

```
hive> CREATE TABLE daily_case_trends AS
    > SELECT reported_date,
    >        SUM(confirmed) AS total_confirmed,
    >        SUM(deaths) AS total_deaths,
    >        SUM(recovered) AS total_recovered
    > FROM covid19_data
    > GROUP BY reported_date
    > ORDER BY reported_date ASC;
2025-03-31T11:59:50,728 INFO [main] org.apache.hadoop.hive.conf.HiveConf - Using the default value passed in for log id: c88cbf10-625d-4409-9ed6-9c8253c55d3c
2025-03-31T11:59:50,728 INFO [main] org.apache.hadoop.hive.ql.session.SessionState - Updating thread name to c88cbf10-625d-4409-9ed6-9c8253c55d3c main
2025-03-31T11:59:50,841 INFO [c88cbf10-625d-4409-9ed6-9c8253c55d3c main] org.apache.hadoop.hive.common.FileUtils - Creating directory if it doesn't exist: hdfs://loc
ost:9000/user/hive/warehouse/.hive-staging_hive_2025-03-31_11-59-50_750_6432779928533042386-1
Query ID = Admin_20250331115950_4e7bc369-8dc1-4a24-a24d-6e4a9eef2b97
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1743400293098_0005, Tracking URL = http://Sudharsanan:8088/proxy/application_1743400293098_0005/
Kill Command = C:\hadoop\bin\mapred job  -kill job_1743400293098_0005
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-03-31 12:00:09,684 Stage-1 map = 0%,  reduce = 0%
2025-03-31 12:00:19,186 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.389 sec
2025-03-31 12:00:28,613 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 11.248 sec
MapReduce Total cumulative CPU time: 11 seconds 248 msec
Ended Job = job_1743400293098_0005
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1743400293098_0006, Tracking URL = http://Sudharsanan:8088/proxy/application_1743400293098_0006/
Kill Command = C:\hadoop\bin\mapred job  -kill job_1743400293098_0006
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2025-03-31 12:00:57,028 Stage-2 map = 0%,  reduce = 0%
```

Top affected countries

```
hive> CREATE TABLE top_affected_countries AS
    > SELECT country_region,
    >        SUM(confirmed) AS total_confirmed
    > FROM covid19_data
    > GROUP BY country_region
    > ORDER BY total_confirmed DESC
    > LIMIT 5;
2025-03-31T12:02:19,285 INFO [main] org.apache.hadoop.hive.conf.HiveConf - Using the default value passed in for log id: c88cbf10-625d-4409-9ed6-9c8253c55d3c
2025-03-31T12:02:19,286 INFO [main] org.apache.hadoop.hive.ql.session.SessionState - Updating thread name to c88cbf10-625d-4409-9ed6-9c8253c55d3c main
2025-03-31T12:02:19,373 INFO [c88cbf10-625d-4409-9ed6-9c8253c55d3c main] org.apache.hadoop.hive.common.FileUtils - Creating directory if it doesn't exist: hdfs://localh
ost:9000/user/hive/warehouse/.hive-staging_hive_2025-03-31_12-02-19_309_7637333473193430698-1
Query ID = Admin_20250331120219_6d384f29-a2c4-4af6-94b2-0b26c22be293
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1743400293098_0007, Tracking URL = http://Sudharsanan:8088/proxy/application_1743400293098_0007/
Kill Command = C:\hadoop\bin\mapred job  -kill job_1743400293098_0007
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-03-31 12:02:38,557 Stage-1 map = 0%,  reduce = 0%
2025-03-31 12:02:46,952 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 4.279 sec
2025-03-31 12:02:57,445 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 8.355 sec
MapReduce Total cumulative CPU time: 8 seconds 355 msec
Ended Job = job_1743400293098_0007
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
```

Monthly trends

```
hive> CREATE TABLE monthly_trends AS
    > SELECT YEAR(reported_date) AS year,
    >        MONTH(reported_date) AS month,
    >        SUM(confirmed) AS monthly_confirmed,
    >        SUM(deaths) AS monthly_deaths,
    >        SUM(recovered) AS monthly_recovered
    > FROM covid19_data
    > GROUP BY YEAR(reported_date), MONTH(reported_date)
    > ORDER BY year, month;
2025-03-31T12:08:25,339 INFO [main] org.apache.hadoop.hive.conf.HiveConf - Using the default value passed in for log id: c88cbf10-625d-4409-9ed6-9c8253c55d3c
2025-03-31T12:08:25,339 INFO [main] org.apache.hadoop.hive.ql.session.SessionState - Updating thread name to c88cbf10-625d-4409-9ed6-9c8253c55d3c main
2025-03-31T12:08:25,424 INFO [c88cbf10-625d-4409-9ed6-9c8253c55d3c main] org.apache.hadoop.hive.common.FileUtils - Creating directory if it doesn't exist: hdfs://localh
ost:9000/user/hive/warehouse/.hive-staging_hive_2025-03-31_12-08-25_361_5438186373701684782-1
Query ID = Admin_20250331120825_c358dbf0-9f3d-465e-bb7e-34a1f9201dc0
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1743400293098_0011, Tracking URL = http://Sudharsanan:8088/proxy/application_1743400293098_0011/
Kill Command = C:\hadoop\bin\mapred job  -kill job_1743400293098_0011
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-03-31 12:08:43,348 Stage-1 map = 0%,  reduce = 0%
2025-03-31 12:08:53,806 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.498 sec
2025-03-31 12:09:03,254 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 10.934 sec
MapReduce Total cumulative CPU time: 10 seconds 934 msec
Ended Job = job_1743400293098_0011
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1743400293098_0012, Tracking URL = http://Sudharsanan:8088/proxy/application_1743400293098_0012/
```

Weekly trends

```
hive> CREATE TABLE weekly_trends AS
    > SELECT YEAR(reported_date) AS year,
    >        WEEKOFYEAR(reported_date) AS week,
    >        SUM(confirmed) AS weekly_confirmed,
    >        SUM(deaths) AS weekly_deaths,
    >        SUM(recovered) AS weekly_recovered
    > FROM covid19_data
    > GROUP BY YEAR(reported_date), WEEKOFYEAR(reported_date)
    > ORDER BY year, week;
2025-03-31T12:05:13,076 INFO [main] org.apache.hadoop.hive.conf.HiveConf - Using the default value passed in for log id: c88cbf10-625d-4409-9ed6-9c8253c55d3c
2025-03-31T12:05:13,077 INFO [main] org.apache.hadoop.hive.ql.session.SessionState - Updating thread name to c88cbf10-625d-4409-9ed6-9c8253c55d3c main
2025-03-31T12:05:13,160 INFO [c88cbf10-625d-4409-9ed6-9c8253c55d3c main] org.apache.hadoop.hive.common.FileUtils - Creating directory if it doesn't exist: hdfs://localh
ost:9000/user/hive/warehouse/.hive-staging_hive_2025-03-31_12-05-13_097_8596597523892300736-1
Query ID = Admin_20250331120513_417bff5e-17af-4ef8-a07d-782e20671613
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1743400293098_0009, Tracking URL = http://Sudharsanan:8088/proxy/application_1743400293098_0009/
Kill Command = C:\hadoop\bin\mapred job  -kill job_1743400293098_0009
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-03-31 12:05:30,928 Stage-1 map = 0%,  reduce = 0%
2025-03-31 12:05:40,400 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 6.029 sec
2025-03-31 12:05:49,838 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 11.278 sec
MapReduce Total cumulative CPU time: 11 seconds 278 msec
Ended Job = job_1743400293098_0009
Launching Job 2 out of 2
Number of reduce tasks determined at compile time: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1743400293098_0010, Tracking URL = http://Sudharsanan:8088/proxy/application_1743400293098_0010/
```

Exporting tables as csv files

```
C:\Windows\system32>hdfs dfs -get /user/hive/exports/active_cases_trend C:/exp

C:\Windows\system32>hdfs dfs -get /user/hive/exports/case_fatality_rate C:/exp

C:\Windows\system32>hdfs dfs -get /user/hive/exports/monthly_trends C:/exp

C:\Windows\system32>hdfs dfs -get /user/hive/exports/top_affected_countries C:/exp

C:\Windows\system32>hdfs dfs -get /user/hive/exports/weekly_trends C:/exp

C:\Windows\system32>hdfs dfs -get /user/hive/exports/who_region_analysis C:/exp

C:\Windows\system32>
```

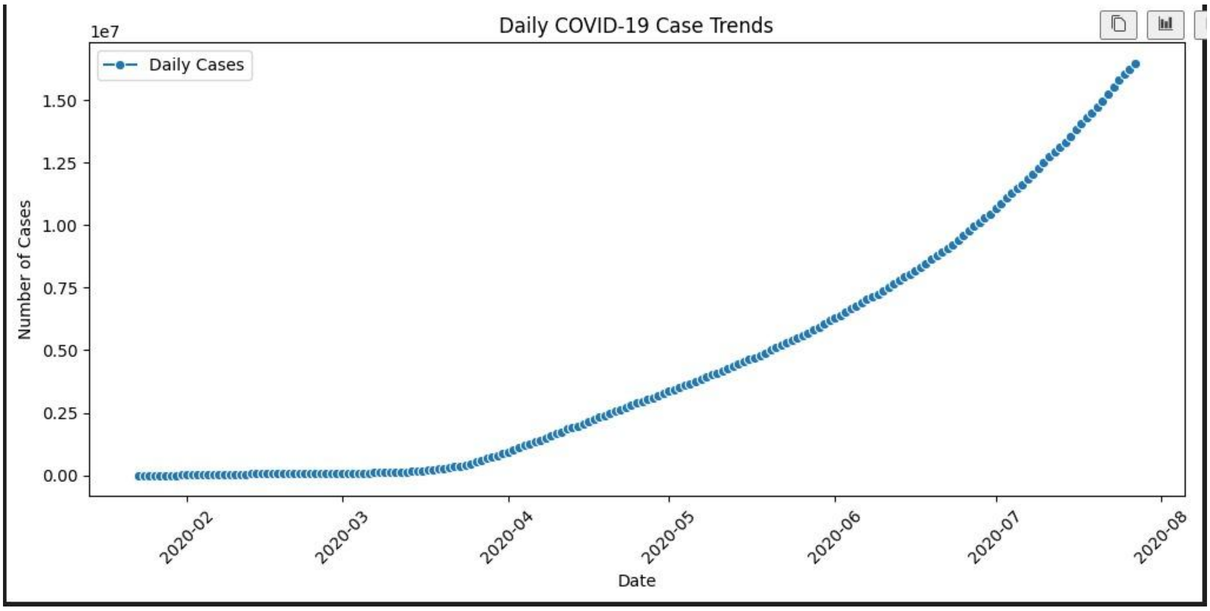
Data visualization

```python
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

[1] ✓ 11.2s Python

```python
df = pd.read_csv(r"C:\exp\daily_case_trends\daily_case_trends.csv")  # Assuming it's comma-separated
df.columns = ["date", "cases", "deaths", "recovered"]
df['date'] = pd.to_datetime(df['date'], errors='coerce')
plt.figure(figsize=(12, 5))
sns.lineplot(data=df, x="date", y="cases", marker="o", label="Daily Cases")
plt.xticks(rotation=45)
plt.title("Daily COVID-19 Case Trends")
plt.xlabel("Date")
plt.ylabel("Number of Cases")
plt.legend()
plt.show()
```

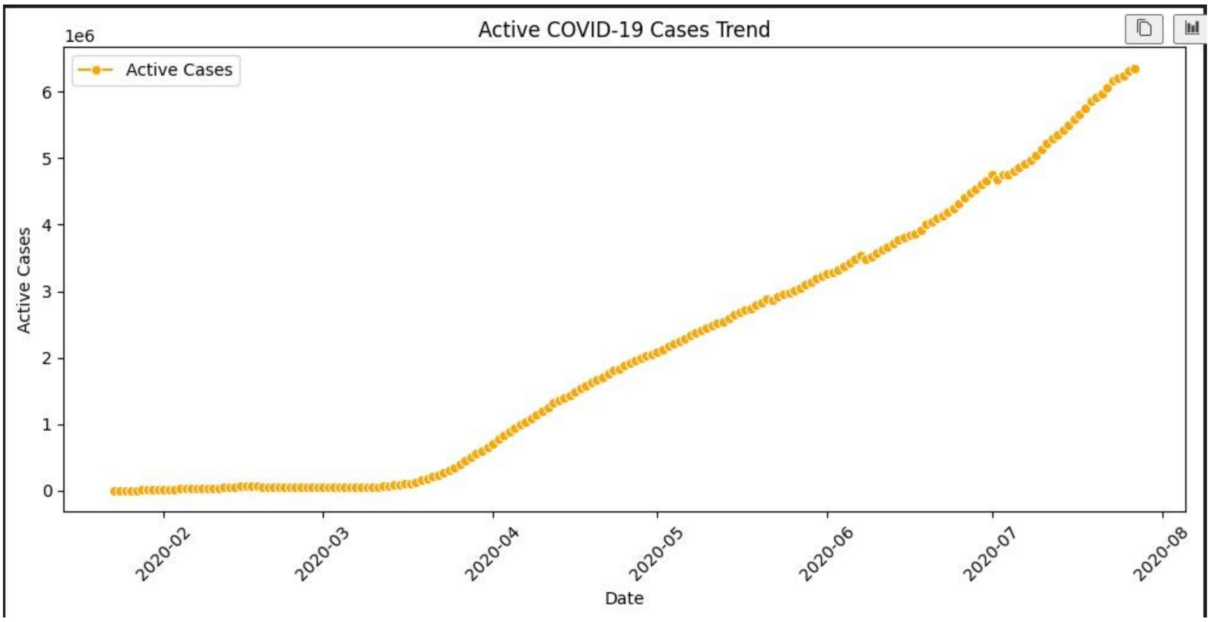
[2] ✓ 0.9s Python




```python
df = pd.read_csv(r"C:\exp\active_cases_trend\active_cases_trend.csv")
df.columns = ["date", "active_cases"]
df['date'] = pd.to_datetime(df['date'], errors='coerce')

plt.figure(figsize=(12, 5))
sns.lineplot(data=df, x="date", y="active_cases", marker="o", label="Active Cases", color="orange")
plt.xticks(rotation=45)
plt.title("Active COVID-19 Cases Trend")
plt.xlabel("Date")
plt.ylabel("Active Cases")
plt.legend()
plt.show()
```

✓ 0.8s

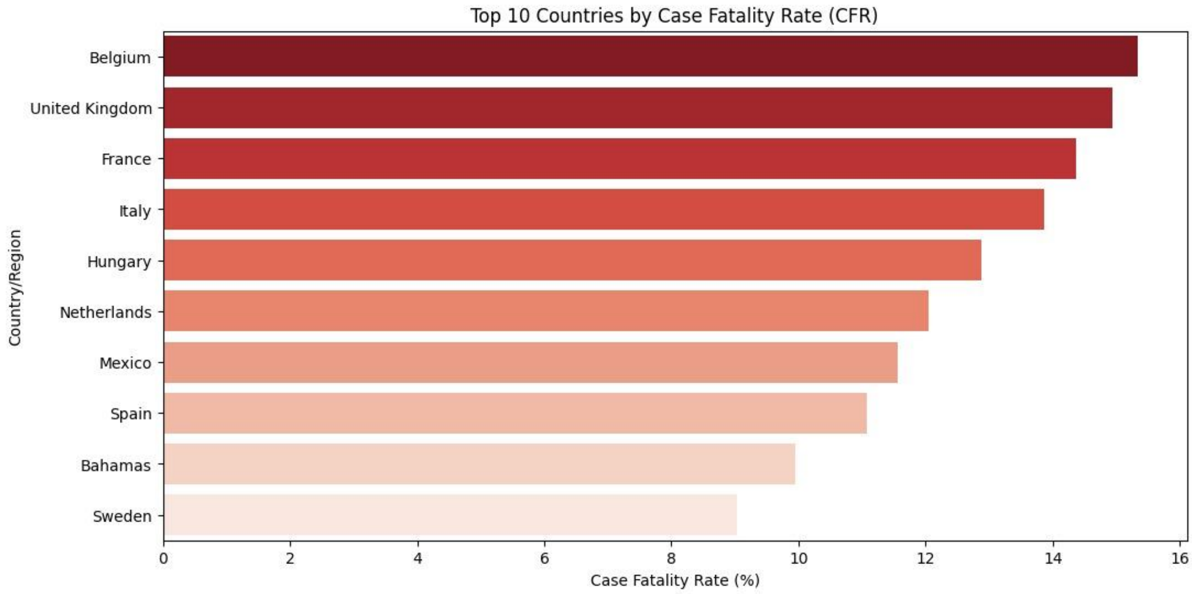
Python




```python
df = pd.read_csv(r"C:\exp\case_fatality_rate\case_fatality_rate.csv")
df.columns = ["country_region", "cfr_percentage"]

# Sort by CFR values for better visualization
df = df.sort_values(by="cfr_percentage", ascending=False).head(10)  # Top 10 countries

# Plot
plt.figure(figsize=(12, 6))
sns.barplot(data=df, x="cfr_percentage", y="country_region", hue="country_region", palette="Reds_r")
plt.xlabel("Case Fatality Rate (%)")
plt.ylabel("Country/Region")
plt.title("Top 10 Countries by Case Fatality Rate (CFR)")
plt.show()
```

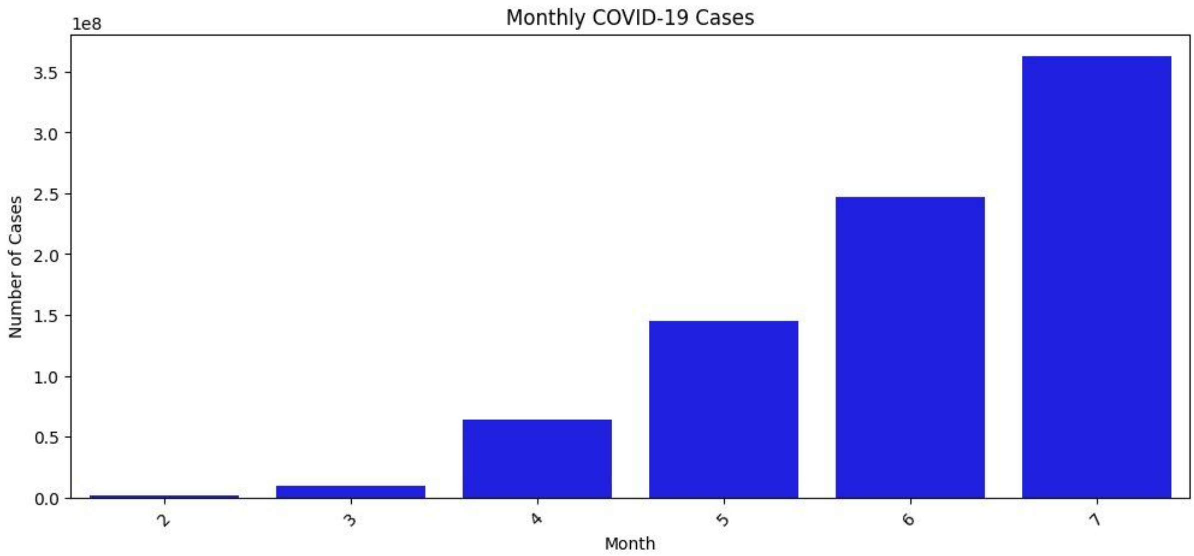
✓ 1.0s

Python



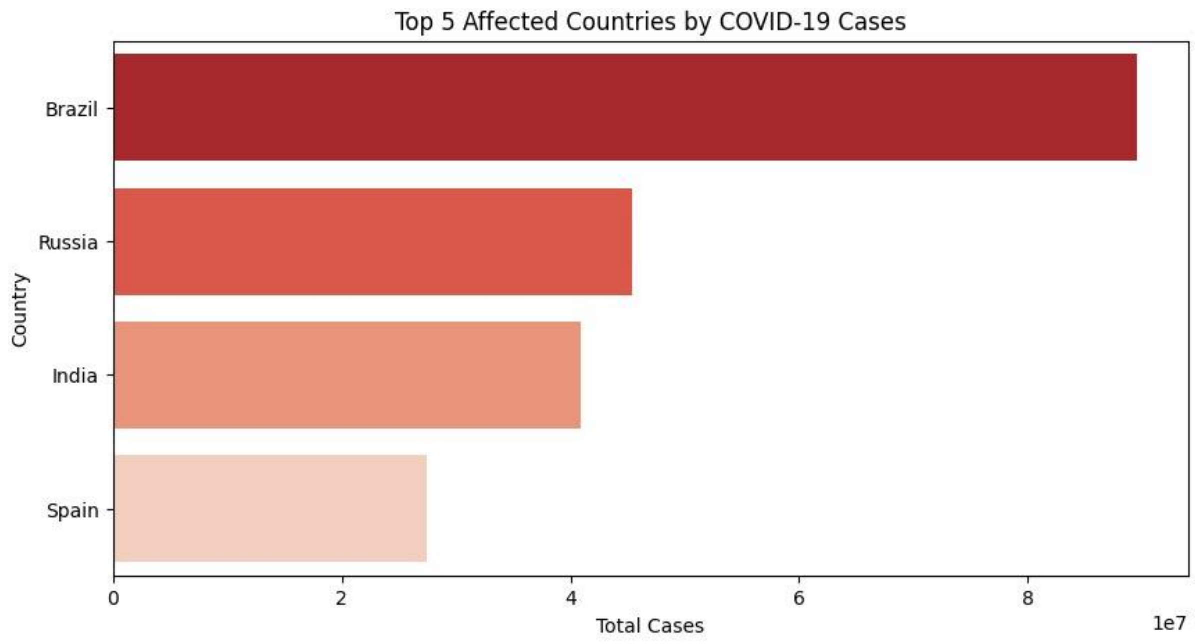

```
df = pd.read_csv(r"C:\exp\monthly_trends\monthly_trends.csv")
df.columns = ["year","month", "cases", "deaths", "recovered"]

plt.figure(figsize=(12, 5))
sns.barplot(data=df, x="month", y="cases", color="blue")
plt.xticks(rotation=45)
plt.title("Monthly COVID-19 Cases")
plt.xlabel("Month")
plt.ylabel("Number of Cases")
plt.show()
```

✓ 0.5s Python



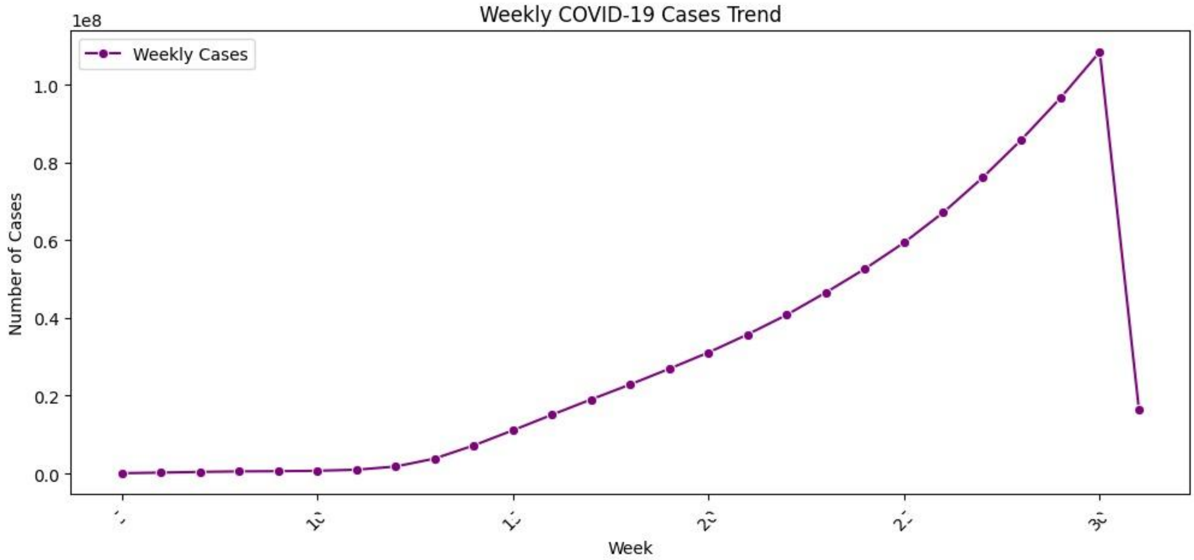

```
df = pd.read_csv(r"C:\exp\top_affected_countries\top_affected_countries.csv")
df.columns = ["country", "total_cases"]

plt.figure(figsize=(10, 5))
sns.barplot(data=df, x="total_cases", y="country", hue="country", palette="Reds_r")
plt.title("Top 5 Affected Countries by COVID-19 Cases")
plt.xlabel("Total Cases")
plt.ylabel("Country")
plt.show()
```

✓ 1.8s Python



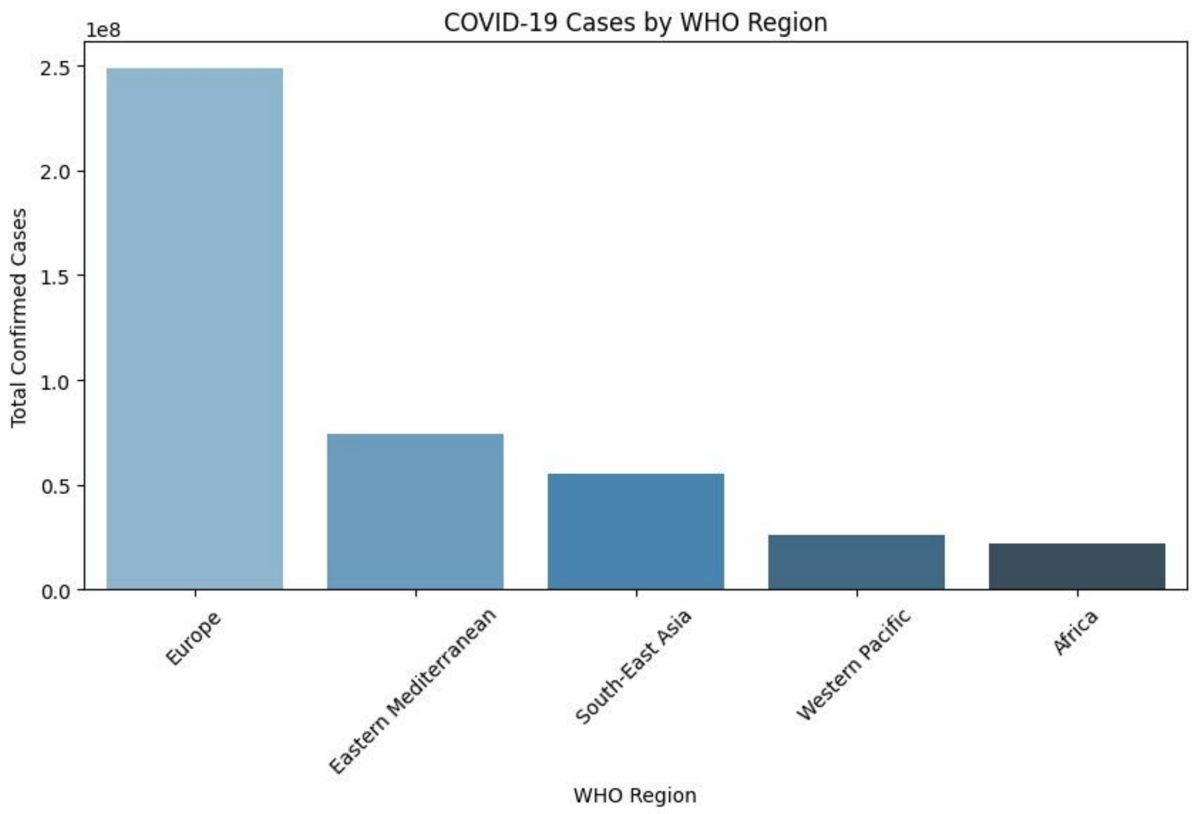

```python
df = pd.read_csv(r"C:\exp\weekly_trends\weekly_trends.csv")

# Rename columns correctly
df.columns = ["year", "week", "weekly_confirmed", "weekly_deaths", "weekly_recovered"]

plt.figure(figsize=(12, 5))
sns.lineplot(data=df, x="week", y="weekly_confirmed", marker="o", color="purple", label="Weekly Cases")
plt.xticks(rotation=45)
plt.title("Weekly COVID-19 Cases Trend")
plt.xlabel("Week")
plt.ylabel("Number of Cases")
plt.legend()
plt.show()
```
✓ 0.3s

Python



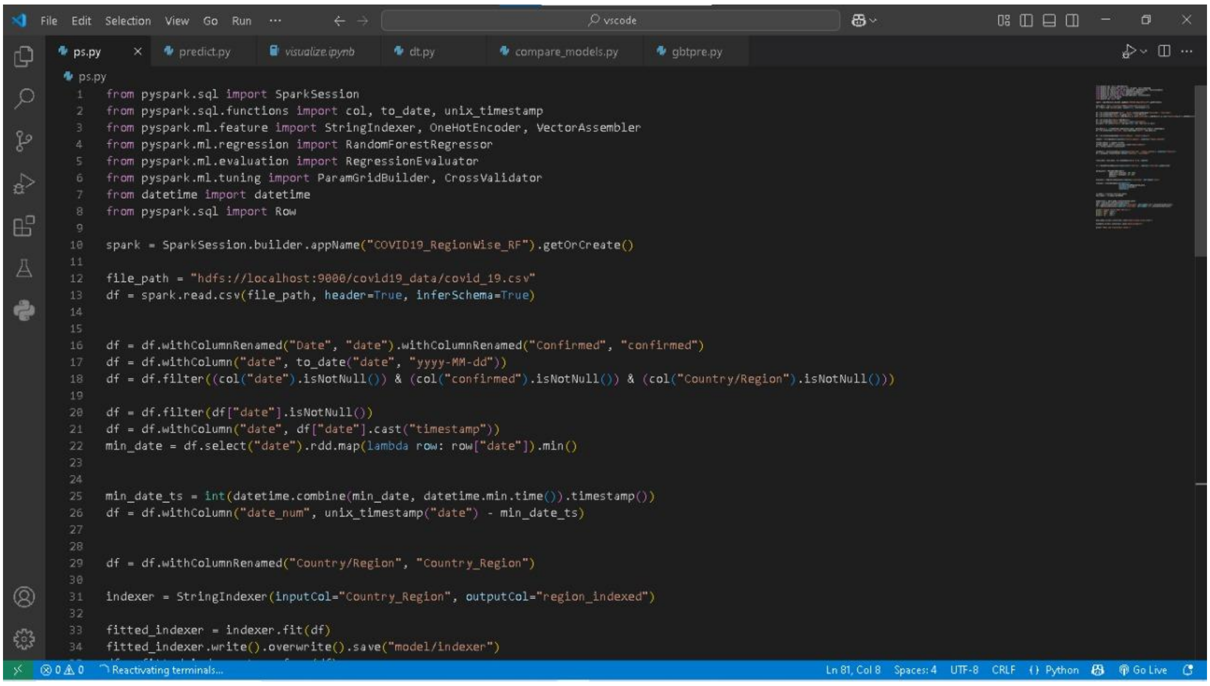

```python
df = pd.read_csv(r"C:\exp\who_region_analysis\who_region_analysis.csv")

# Correct column names based on Hive schema
df.columns = ["who_region", "total_confirmed", "total_deaths", "total_recovered"]

plt.figure(figsize=(10, 5))
sns.barplot(data=df, x="who_region", y="total_confirmed", palette="Blues_d")
plt.xticks(rotation=45)
plt.title("COVID-19 Cases by WHO Region")
plt.xlabel("WHO Region")
plt.ylabel("Total Confirmed Cases")
plt.show()
```

✓ 0.7s Python



ML prediction

Random forest



```python
assembler = VectorAssembler(inputCols=["date_num", "region_indexed"], outputCol="features")
df = assembler.transform(df).select("features", "confirmed")



train_data, test_data = df.randomSplit([0.8, 0.2], seed=42)


rf = RandomForestRegressor(featuresCol="features", labelCol="confirmed",maxBins=256)


param_grid = (ParamGridBuilder()
              .addGrid(rf.numTrees, [20, 50])
              .addGrid(rf.maxDepth, [5, 10])
              .build())

evaluator = RegressionEvaluator(labelCol="confirmed", metricName="rmse")

crossval = CrossValidator(estimator=rf,
                          estimatorParamMaps=param_grid,
                          evaluator=evaluator,
                          numFolds=3)


cv_model = crossval.fit(train_data)
best_model = cv_model.bestModel


predictions = best_model.transform(test_data)
rmse = evaluator.evaluate(predictions)
mae = RegressionEvaluator(labelCol="confirmed", metricName="mae").evaluate(predictions)
r2 = RegressionEvaluator(labelCol="confirmed", metricName="r2").evaluate(predictions)

print(f" Random Forest Model Metrics:")
```

```python
predictions = best_model.transform(test_data)
rmse = evaluator.evaluate(predictions)
mae = RegressionEvaluator(labelCol="confirmed", metricName="mae").evaluate(predictions)
r2 = RegressionEvaluator(labelCol="confirmed", metricName="r2").evaluate(predictions)

print(f" Random Forest Model Metrics:")
print(f" RMSE: {rmse}")
print(f" MAE : {mae}")
print(f" R²  : {r2}")


best_model.write().overwrite().save("model/random_forest_model")

assembler.write().overwrite().save("model/assembler")

print(" Model and transformers saved.")
```

```
C:\Users\Admin> "C:\Users\Admin\AppData\Local\Programs\Python\Python310\python.exe" "E:\vscode\ps.py"
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/04/11 18:52:59 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
25/04/11 18:53:36 WARN DAGScheduler: Broadcasting large task binary with size 1533.4 KiB
25/04/11 18:53:37 WARN DAGScheduler: Broadcasting large task binary with size 2.3 MiB
25/04/11 18:53:52 WARN DAGScheduler: Broadcasting large task binary with size 1466.6 KiB
25/04/11 18:53:54 WARN DAGScheduler: Broadcasting large task binary with size 2.3 MiB
25/04/11 18:53:57 WARN DAGScheduler: Broadcasting large task binary with size 3.6 MiB
25/04/11 18:54:01 WARN DAGScheduler: Broadcasting large task binary with size 5.5 MiB
25/04/11 18:54:15 WARN DAGScheduler: Broadcasting large task binary with size 1500.6 KiB
25/04/11 18:54:16 WARN DAGScheduler: Broadcasting large task binary with size 2.2 MiB
25/04/11 18:54:29 WARN DAGScheduler: Broadcasting large task binary with size 1466.3 KiB
25/04/11 18:54:31 WARN DAGScheduler: Broadcasting large task binary with size 2.3 MiB
25/04/11 18:54:33 WARN DAGScheduler: Broadcasting large task binary with size 3.7 MiB
25/04/11 18:54:36 WARN DAGScheduler: Broadcasting large task binary with size 5.7 MiB
25/04/11 18:54:52 WARN DAGScheduler: Broadcasting large task binary with size 1531.0 KiB
25/04/11 18:54:53 WARN DAGScheduler: Broadcasting large task binary with size 2.3 MiB
25/04/11 18:55:08 WARN DAGScheduler: Broadcasting large task binary with size 1481.7 KiB
25/04/11 18:55:10 WARN DAGScheduler: Broadcasting large task binary with size 2.4 MiB
25/04/11 18:55:12 WARN DAGScheduler: Broadcasting large task binary with size 3.7 MiB
25/04/11 18:55:16 WARN DAGScheduler: Broadcasting large task binary with size 5.6 MiB
□ Random Forest Model Metrics:
□□ RMSE: 40738.55704105764
□□ MAE : 11710.39468121
□□ R²  : 0.8791791287694141
□ Model and transformers saved.

C:\Users\Admin>SUCCESS: The process with PID 3912 (child process of PID 10432) has been terminated.
SUCCESS: The process with PID 10432 (child process of PID 12332) has been terminated.
SUCCESS: The process with PID 12332 (child process of PID 4608) has been terminated.
```

predict.py

```python
from pyspark.sql import SparkSession, Row
from pyspark.ml.feature import StringIndexerModel, VectorAssembler
from pyspark.ml.regression import RandomForestRegressionModel
from datetime import datetime


spark = SparkSession.builder.appName("COVID19_RegionWise_Predict").getOrCreate()


best_model = RandomForestRegressionModel.load("model/random_forest_model")
indexer = StringIndexerModel.load("model/indexer")


min_date = datetime(2020, 1, 22)


input_date_str = "2025-06-01"
input_region = "Brazil"


input_date = datetime.strptime(input_date_str, "%Y-%m-%d")
input_date_num = int((input_date - min_date).total_seconds())


input_df = spark.createDataFrame([Row(date_num=input_date_num, **{"Country/Region": input_region})])


input_df = input_df.withColumnRenamed("Country/Region", "Country_Region")


input_df = indexer.transform(input_df)


assembler = VectorAssembler(inputCols=["date_num", "region_indexed"], outputCol="features")
```

```
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/04/11 19:06:35 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
+-----------------+
|       prediction|
+-----------------+
|963649.7982141869|
+-----------------+


C:\Users\Admin>SUCCESS: The process with PID 14520 (child process of PID 15036) has been terminated.
SUCCESS: The process with PID 15036 (child process of PID 2280) has been terminated.
SUCCESS: The process with PID 2280 (child process of PID 13036) has been terminated.


C:\Users\Admin> "C:\Users\Admin\AppData\Local\Programs\Python\Python310\python.exe" "E:\vscode\predict.py"
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/04/11 19:11:27 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
+-----------------+
|       prediction|
+-----------------+
|963649.7982141869|
+-----------------+


C:\Users\Admin>SUCCESS: The process with PID 11316 (child process of PID 14844) has been terminated.
SUCCESS: The process with PID 14844 (child process of PID 18776) has been terminated.
SUCCESS: The process with PID 18776 (child process of PID 6156) has been terminated.


C:\Users\Admin> "C:\Users\Admin\AppData\Local\Programs\Python\Python310\python.exe" "E:\vscode\predict.py"
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/04/11 19:12:48 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
+------------------+
|        prediction|
+------------------+
|1865384.3670600846|
+------------------+


C:\Users\Admin>SUCCESS: The process with PID 6208 (child process of PID 7084) has been terminated.
SUCCESS: The process with PID 7084 (child process of PID 21824) has been terminated.
SUCCESS: The process with PID 21824 (child process of PID 15904) has been terminated.
```

Comparison

compare_models.py

```python
from pyspark.sql import SparkSession
from pyspark.sql.functions import col, to_date, unix_timestamp
from pyspark.ml.feature import StringIndexer, VectorAssembler
from pyspark.ml.regression import DecisionTreeRegressor, LinearRegression, GBTRegressor
from pyspark.ml.evaluation import RegressionEvaluator
from datetime import datetime


spark = SparkSession.builder.appName("COVID19_RegionWise_ModelComparison").getOrCreate()

file_path = "hdfs://localhost:9000/covid19_data/covid_19.csv"
df = spark.read.csv(file_path, header=True, inferSchema=True)


df = df.withColumnRenamed("Date", "date").withColumnRenamed("Confirmed", "confirmed")
df = df.withColumn("date", to_date("date", "yyyy-MM-dd"))
df = df.filter((col("date").isNotNull()) & (col("confirmed").isNotNull()) & (col("Country/Region").isNotNull()))
df = df.withColumnRenamed("Country/Region", "Country_Region")
df = df.withColumn("date", df["date"].cast("timestamp"))

min_date = df.select("date").rdd.map(lambda row: row["date"]).min()
min_date_ts = int(datetime.combine(min_date, datetime.min.time()).timestamp())
df = df.withColumn("date_num", unix_timestamp("date") - min_date_ts)


indexer = StringIndexer(inputCol="Country_Region", outputCol="region_indexed")
df = indexer.fit(df).transform(df)


assembler = VectorAssembler(inputCols=["date_num", "region_indexed"], outputCol="features")
df = assembler.transform(df).select("features", "confirmed")


train_data, test_data = df.randomSplit([0.8, 0.2], seed=42)
```

```python
evaluator_rmse = RegressionEvaluator(labelCol="confirmed", metricName="rmse")
evaluator_mae = RegressionEvaluator(labelCol="confirmed", metricName="mae")
evaluator_r2 = RegressionEvaluator(labelCol="confirmed", metricName="r2")

def evaluate_model(name, predictions):
    rmse = evaluator_rmse.evaluate(predictions)
    mae = evaluator_mae.evaluate(predictions)
    r2 = evaluator_r2.evaluate(predictions)
    print(f"\n🔍 {name} Evaluation:")
    print(f"➡️  RMSE : {rmse:.2f}")
    print(f"➡️  MAE  : {mae:.2f}")
    print(f"➡️  R²   : {r2:.4f}")

dt = DecisionTreeRegressor(featuresCol="features", labelCol="confirmed", maxBins=200)
dt_model = dt.fit(train_data)
dt_predictions = dt_model.transform(test_data)
evaluate_model("Decision Tree Regressor", dt_predictions)


lr = LinearRegression(featuresCol="features", labelCol="confirmed")
lr_model = lr.fit(train_data)
lr_predictions = lr_model.transform(test_data)
evaluate_model("Linear Regression", lr_predictions)


gbt = GBTRegressor(featuresCol="features", labelCol="confirmed", maxIter=50, maxBins=200)
gbt_model = gbt.fit(train_data)
gbt_predictions = gbt_model.transform(test_data)
evaluate_model("Gradient Boosted Trees", gbt_predictions)

model_save_path = "model/gbt"
gbt_model.write().overwrite().save(model_save_path)
print("✅ GBT model saved to HDFS.")
```

```
C:\Users\Admin>"C:\Users\Admin\AppData\Local\Programs\Python\Python310\python.exe" "E:\vscode\compare_models.py"
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/04/14 23:55:16 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.

□ Decision Tree Regressor Evaluation:
□□ RMSE : 25237.19
□□ MAE  : 7620.21
□□ R²   : 0.9536
25/04/14 23:56:15 WARN Instrumentation: [965665ad] regParam is zero, which might cause numerical instability and overfitting.
25/04/14 23:56:16 WARN InstanceBuilder: Failed to load implementation from:dev.ludovic.netlib.blas.JNIBLAS
25/04/14 23:56:16 WARN InstanceBuilder: Failed to load implementation from:dev.ludovic.netlib.lapack.JNILAPACK

□ Linear Regression Evaluation:
□□ RMSE : 115867.10
□□ MAE  : 30149.38
□□ R²   : 0.0226

□ Gradient Boosted Trees Evaluation:
□□ RMSE : 15891.17
□□ MAE  : 3871.61
□□ R²   : 0.9816
```

GBT

gbtpre.py

```python
from pyspark.sql import SparkSession, Row
from pyspark.ml.feature import StringIndexerModel, VectorAssembler
from pyspark.ml.regression import GBTRegressionModel
from datetime import datetime


spark = SparkSession.builder.appName("COVID19_RegionWise_GBT_Predict").getOrCreate()

gbt_model = GBTRegressionModel.load("model/gbt")
indexer = StringIndexerModel.load("model/indexer")

min_date = datetime(2020, 1, 22)


input_date_str = "2021-06-01"
input_region = "India"


input_date = datetime.strptime(input_date_str, "%Y-%m-%d")
input_date_num = int((input_date - min_date).total_seconds())


input_df = spark.createDataFrame([Row(date_num=input_date_num, **{"Country/Region": input_region})])


input_df = input_df.withColumnRenamed("Country/Region", "Country_Region")


input_df = indexer.transform(input_df)


assembler = VectorAssembler(inputCols=["date_num", "region_indexed"], outputCol="features")
input_df = assembler.transform(input_df)
```

gbtpre.py

```python
input_df = spark.createDataFrame([Row(date_num=input_date_num, **{"Country/Region": input_region})])


input_df = input_df.withColumnRenamed("Country/Region", "Country_Region")


input_df = indexer.transform(input_df)


assembler = VectorAssembler(inputCols=["date_num", "region_indexed"], outputCol="features")
input_df = assembler.transform(input_df)


prediction = gbt_model.transform(input_df)
prediction.select("prediction").show()
```

Command Prompt

```
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/04/13 19:49:01 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
25/04/13 19:49:37 WARN InstanceBuilder: Failed to load implementation from:dev.ludovic.netlib.blas.JNIBLAS
+-----------------+
|       prediction|
+-----------------+
|2434535.075327458|
+-----------------+


C:\Users\Admin>SUCCESS: The process with PID 14740 (child process of PID 9720) has been terminated.
SUCCESS: The process with PID 9720 (child process of PID 16648) has been terminated.
SUCCESS: The process with PID 16648 (child process of PID 7028) has been terminated.


C:\Users\Admin>"C:\Users\Admin\AppData\Local\Programs\Python\Python310\python.exe" "E:\vscode\gbtpre.py"
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/04/13 19:51:28 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
25/04/13 19:51:55 WARN InstanceBuilder: Failed to load implementation from:dev.ludovic.netlib.blas.JNIBLAS
+-----------------+
|       prediction|
+-----------------+
|1473598.990221896|
+-----------------+


C:\Users\Admin>SUCCESS: The process with PID 21520 (child process of PID 19016) has been terminated.
SUCCESS: The process with PID 19016 (child process of PID 16504) has been terminated.
SUCCESS: The process with PID 16504 (child process of PID 1564) has been terminated.


C:\Users\Admin>"C:\Users\Admin\AppData\Local\Programs\Python\Python310\python.exe" "E:\vscode\gbtpre.py"
Setting default log level to "WARN".
To adjust logging level use sc.setLogLevel(newLevel). For SparkR, use setLogLevel(newLevel).
25/04/13 19:53:03 WARN Utils: Service 'SparkUI' could not bind on port 4040. Attempting port 4041.
25/04/13 19:53:28 WARN InstanceBuilder: Failed to load implementation from:dev.ludovic.netlib.blas.JNIBLAS
+-----------------+
|       prediction|
+-----------------+
|1473598.990221896|
+-----------------+


C:\Users\Admin>SUCCESS: The process with PID 11260 (child process of PID 9268) has been terminated.
SUCCESS: The process with PID 9268 (child process of PID 12224) has been terminated.
```

Model Evaluation & Results

Each model was evaluated using three metrics:

- **RMSE (Root Mean Squared Error)**
- **MAE (Mean Absolute Error)**
- **R^2 (Coefficient of Determination)**

Model	**RMSE**	**MAE**	**R^2**
Decision Tree	25,237.19	7,620.21	0.9536
Linear Regression	115,867.1	30,149.38	0.0226

Model	RMSE	MAE	R^2
Random Forest Regressor	40,337.24	11,016.59	0.8815
Gradient Boosted Trees	**15,891.17**	**3,871.61**	**0.9816**

Conclusion

This project successfully demonstrates a **scalable Hadoop-based solution** for COVID-19 data analysis. By integrating **HDFS, MapReduce, Hive, and PySpark**, we ensure efficient data processing and real-time tracking. The results provide meaningful insights for researchers and policymakers to analyse pandemic trends effectively. Future work includes integrating **machine learning models** for predictive analytics and enhancing forecasting accuracy for future pandemics.