EARTHQUAKE ALERT CASE STUDY

By-:

Karunesh kr Pandey

Introduction

Earthquakes are among the most unpredictable and devastating natural disasters, often resulting in significant loss of life, damage to infrastructure, and long-term economic disruption. With the increasing availability of structured seismic data, it has become possible to analyze earthquake patterns more effectively and develop systems that support early warnings and disaster preparedness. This project focuses on the analytical exploration of a balanced earthquake alert dataset using modern data processing tools such as MySQL, Hive, and Sqoop, complemented by Python-based visualization.

The primary objective of this project is to uncover meaningful patterns and correlations between earthquake characteristics and their corresponding alert levels. By examining relationships between magnitude, depth, and intensity indicators, we aim to identify key factors that contribute to higher alert classifications. This analysis can support the development of predictive models and enhance the accuracy of alert systems used by disaster management authorities.

To achieve these goals, the project employs a hybrid data processing architecture. MySQL is used for initial data storage and structured querying, while Hive—built on top of the Hadoop ecosystem—is utilized for scalable analysis of large datasets. Sqoop serves as the bridge between MySQL and Hive, enabling seamless data transfer and integration. Additionally, Python libraries such as Pandas, Seaborn, and Plotly are used to visualize trends and present insights in an intuitive and impactful manner.

This case study demonstrates the practical application of big data tools in the domain of natural disaster analysis. By leveraging structured data and distributed computing, we aim to contribute to the broader goal of improving earthquake response strategies and public safety. The insights derived from this project can be valuable for researchers, government agencies, and emergency response teams working to mitigate the impact of seismic events.

Description of the Dataset

The earthquake alert dataset used in this project is a structured and balanced collection of seismic event records, designed to support analytical studies related to earthquake severity, intensity, and alert classification. Each record in the dataset represents a unique earthquake occurrence, capturing key attributes that describe its physical characteristics and impact level. The dataset is particularly valuable for disaster management, predictive modeling, and early warning system development.

The data is organized into a single table named earthquake_data, with each column representing a specific feature of the earthquake. The structure is as follows:

1. **Magnitude**
   - **Attribute**: magnitude
   - **Description**: Represents the intensity of the earthquake on the Richter scale. Higher values indicate stronger seismic activity and greater potential for damage.
2. **Depth**
   - **Attribute**: depth
   - **Description**: Indicates the depth at which the earthquake occurred, measured in kilometers. Shallow earthquakes (lower depth values) are generally more destructive at the surface.
3. **Community Internet Intensity (CDI)**
   - **Attribute**: cdi
   - **Description**: A crowd-sourced measure of perceived shaking intensity, collected from public reports. It reflects how strongly the earthquake was felt by individuals.
4. **Modified Mercalli Intensity (MMI)**
   - **Attribute**: mmi
   - **Description**: A scientific scale that quantifies the physical effects of the earthquake on people, buildings, and the environment. It ranges from weak to extreme shaking.
5. **Significance Score**

- **Attribute**: sig
- **Description**: A composite score that estimates the overall impact of the earthquake. It is calculated based on magnitude, population density, and other contributing factors.

6. **Alert Level**
   - **Attribute**: alert
   - **Description**: A categorical indicator of the earthquake's severity, classified into four levels:
   - green: Minimal impact
   - yellow: Moderate impact
   - orange: High impact
   - red: Severe impact

This dataset is balanced across all alert levels, ensuring equal representation of each category. This balance is critical for unbiased statistical analysis and effective training of machine learning models.

The relational integrity of the dataset allows for efficient querying and aggregation. For example, analysts can group earthquakes by alert level to calculate average magnitude or significance, filter events by depth range, or identify the most impactful earthquakes based on CDI or MMI scores.

Overall, the earthquake alert dataset provides a comprehensive foundation for seismic analysis. It enables researchers and decision-makers to explore correlations between earthquake features and alert classifications, assess risk levels, and enhance public safety through data-driven insights.

Project Scope

The scope of this project centers on the analytical exploration of earthquake events using a structured and balanced dataset. By leveraging modern data processing tools such as MySQL, Hive, and Sqoop, the project aims to uncover meaningful patterns in seismic activity and alert classifications. The dataset includes key attributes such as magnitude, depth, intensity measures (CDI and MMI), significance scores, and alert levels, which collectively provide a comprehensive view of each earthquake's impact.

This case study is designed to investigate how different earthquake characteristics influence the severity of alerts issued. Through scalable querying and statistical analysis, the project seeks to identify correlations between magnitude and depth, detect high-impact events, and evaluate the distribution of alert levels across various intensity ranges. The integration of MySQL and Hive via Sqoop enables efficient data handling and processing, while Python-based visualization tools help present insights in an intuitive and accessible format.

The broader scope includes supporting disaster preparedness efforts by providing data-driven insights that can inform early warning systems, resource allocation, and public safety strategies. The project also lays the foundation for future predictive modeling by establishing baseline relationships between seismic features and alert outcomes.

Project Goals

1. **Analyze Alert Distribution**
   Examine how earthquake events are distributed across different alert levels (green, yellow, orange, red) to understand the frequency and severity of seismic activity.
2. **Evaluate Magnitude and Depth Correlation**
   Investigate the relationship between earthquake magnitude and depth to determine how these factors jointly influence alert classification and potential surface impact.
3. **Identify High-Risk Events**
   Filter and highlight earthquakes with extreme values in magnitude, CDI, MMI, or significance score to identify events that pose the greatest threat.
4. **Perform Scalable Querying Using Hive**
   Utilize Hive's distributed processing capabilities to execute complex queries on large datasets, enabling efficient aggregation, filtering, and grouping operations.
5. **Integrate Data Using Sqoop**
   Employ Sqoop to transfer structured data from MySQL into the Hadoop ecosystem, ensuring seamless integration and accessibility for Hive-based analysis.
6. **Detect Missing or Incomplete Records**
   Run validation queries to identify records with missing or null alert values, contributing to data quality assessment and cleaning.
7. **Visualize Trends and Patterns**
   Use Python libraries such as Pandas, Seaborn, and Plotly to create visual representations of key insights, including alert distribution, intensity comparisons, and top-impact events.
8. **Support Disaster Preparedness and Decision-Making**
   Provide actionable insights that can assist government agencies, researchers, and emergency response teams in improving earthquake response strategies and public safety measures.

Tools and Working Environment

This project utilizes a combination of powerful data processing and analysis tools to manage, query, and visualize earthquake data efficiently. The selected tools are well-suited for handling structured datasets, performing scalable queries, and generating meaningful insights. The working environment integrates relational databases, distributed computing frameworks, and visualization libraries to support a comprehensive analytical workflow.

1. **MySQL**
    - **Description**: MySQL is an open-source relational database management system (RDBMS) known for its reliability and ease of use. It supports structured query language (SQL) for managing and manipulating relational data.
    - **Role in Project**: In this case study, MySQL serves as the initial storage platform for the earthquake dataset. It allows for basic data validation, filtering, and aggregation using SQL queries. MySQL is used to prepare the data before transferring it to the Hadoop ecosystem for large-scale analysis.
2. **Hive**
    - **Description**: Hive is a data warehousing tool built on top of the Hadoop framework. It provides a SQL-like query language called HiveQL, which enables users to perform queries on large datasets stored in the Hadoop Distributed File System (HDFS).
    - **Role in Project**: Hive is used to execute scalable queries on the earthquake dataset after it is imported into HDFS. It supports complex operations such as grouping, filtering, and statistical aggregation, making it ideal for analyzing seismic patterns and alert distributions across thousands of records.
3. **Sqoop**
    - **Description**: Sqoop is a command-line interface tool designed for transferring data between relational databases and Hadoop. It supports both import and export operations, enabling seamless integration between MySQL and Hive.
    - **Role in Project**: Sqoop is used to import the earthquake dataset from MySQL into Hive. This integration allows the project to leverage the

strengths of both systems—structured querying in MySQL and distributed processing in Hive—without manual data migration.

4. **Hadoop Ecosystem**
   - **Description**: Hadoop is an open-source framework for distributed storage and processing of large datasets. It includes components such as HDFS for storage and MapReduce for parallel computation.
   - **Role in Project**: The Hadoop ecosystem provides the underlying infrastructure for Hive. It ensures that large volumes of earthquake data can be stored and processed efficiently across multiple nodes, supporting high-performance analytics.
5. **Python (Pandas, Seaborn, Plotly)**
   - **Description**: Python is a versatile programming language widely used for data analysis and visualization. Libraries such as Pandas, Seaborn, and Plotly offer powerful tools for data manipulation and graphical representation.
   - **Role in Project**: Python is used to visualize trends and patterns in the earthquake dataset. It helps present insights such as alert distribution, magnitude-depth relationships, and top-impact events in an intuitive and interactive format.

Hive Query Analysis

After importing the earthquake dataset into Hive using Sqoop, a series of HiveQL queries were executed to perform large-scale analysis on the seismic data. Hive, built on top of the Hadoop ecosystem, enabled efficient querying of the dataset stored in the Hadoop Distributed File System (HDFS). The queries were designed to extract meaningful insights related to earthquake characteristics, alert levels, and intensity patterns. Below is a detailed explanation of each query and its analytical purpose:

```
hive> create table earth_data(magnitude string,depth int,cdi int,mmi int,sig int
,alert int) Row format delimited fields terminated by ',';
OK
Time taken: 1.085 seconds
```

```
hive> LOAD DATA INPATH '/user/hive/warehouse/earthquake_data.csv' INTO TABLE earth_data;
Loading data to table mcads.earth_data
Table mcads.earth_data stats: [numFiles=1, totalSize=43662]
OK
Time taken: 0.316 seconds
```

1. Total number of earthquakes

SELECT COUNT(*) FROM earthquake_data;

```
hive> SELECT COUNT(*) FROM earth_data;
Query ID = cloudera_20251029231111_c640d5bc-8f27-43e3-828d-8284f67d2263
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_1761803663889_0001, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761803663889
_0001/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761803663889_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-10-29 23:11:56,859 Stage-1 map = 0%,  reduce = 0%
2025-10-29 23:12:02,200 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.72 sec
2025-10-29 23:12:08,502 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.59 sec
MapReduce Total cumulative CPU time: 1 seconds 590 msec
Ended Job = job_1761803663889_0001
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 1.59 sec   HDFS Read: 51301 HDFS Write: 5 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 590 msec
OK
1301
Time taken: 21.756 seconds, Fetched: 1 row(s)
```

2. Average magnitude

SELECT AVG(magnitude) FROM earthquake_data;

```
hive> SELECT AVG(magnitude) FROM earth_data;
Query ID = cloudera_20251029231414_6373aa7b-22e5-4c55-9867-613524d345ba
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_1761803663889_0002, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761803663889
_0002/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761803663889_0002
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-10-29 23:14:34,577 Stage-1 map = 0%,  reduce = 0%
2025-10-29 23:14:39,809 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.66 sec
2025-10-29 23:14:46,046 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.49 sec
MapReduce Total cumulative CPU time: 1 seconds 490 msec
Ended Job = job_1761803663889_0002
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 1.49 sec   HDFS Read: 51913 HDFS Write: 18 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 490 msec
OK
7.061007692307706
Time taken: 19.426 seconds, Fetched: 1 row(s)
```

3. Average significance (sig) grouped by magnitude range

CASE WHEN magnitude < 4 THEN 'Low’ WHEN magnitude BETWEEN 4 AND 6 THEN 'Moderate' ELSE SELECT 'High' END AS magnitude_category, AVG(sig) AS avg_significance FROM earth_data GROUP BY CASE WHEN magnitude < 4 THEN 'Low' WHEN magnitude BETWEEN 4 AND 6 THEN 'Moderate' ELSE 'High' END;

```
hive> SELECT
    >   CASE
    >     WHEN magnitude < 4 THEN 'Low'
    >     WHEN magnitude BETWEEN 4 AND 6 THEN 'Moderate'
    >     ELSE 'High'
    >   END AS magnitude_category,
    >   AVG(sig) AS avg_significance
    > FROM earth_data
    > GROUP BY
    >   CASE
    >     WHEN magnitude < 4 THEN 'Low'
    >     WHEN magnitude BETWEEN 4 AND 6 THEN 'Moderate'
    >     ELSE 'High'
    >   END;
Query ID = cloudera_20251029233232_8220a18e-0f25-450c-954d-950a12a17e23
Total jobs = 1
Launching Job 1 out of 1
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_1761803663889_0010, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761803663889
_0010/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761803663889_0010
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-10-29 23:32:25,778 Stage-1 map = 0%,  reduce = 0%
2025-10-29 23:32:30,969 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.96 sec
2025-10-29 23:32:37,207 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.84 sec
MapReduce Total cumulative CPU time: 1 seconds 840 msec
Ended Job = job_1761803663889_0010
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 1.84 sec   HDFS Read: 54443 HDFS Write: 24 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 840 msec
OK
High    -9.749230769230769
Time taken: 18.119 seconds, Fetched: 1 row(s)
```

4. Count of earthquakes by alert level

SELECT alert, COUNT(*) FROM earthquake_data GROUP BY alert;

```
hive> SELECT alert, COUNT(*) FROM earth_data GROUP BY alert;
Query ID = cloudera_20251029231717_a2c7e061-a2dd-4576-b14b-1117e96ba29e
Total jobs = 1
Launching Job 1 out of 1
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_1761803663889_0003, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761803663889
_0003/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761803663889_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-10-29 23:17:26,172 Stage-1 map = 0%,  reduce = 0%
2025-10-29 23:17:32,422 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.68 sec
2025-10-29 23:17:38,689 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.53 sec
MapReduce Total cumulative CPU time: 1 seconds 530 msec
Ended Job = job_1761803663889_0003
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 1.53 sec   HDFS Read: 51686 HDFS Write: 8 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 530 msec
OK
NULL    1301
Time taken: 19.507 seconds, Fetched: 1 row(s)
```

5. Maximum and minimum depth

SELECT MAX(depth), MIN(depth) FROM earth_data;

```
hive> SELECT MAX(depth) AS max_depth, MIN(depth) AS min_depth FROM earth_data;
Query ID = cloudera_20251029232121_f05b60dc-9303-48a5-b406-3b2c91950e84
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_1761803663889_0006, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761803663889
_0006/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761803663889_0006
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-10-29 23:21:57,431 Stage-1 map = 0%,  reduce = 0%
2025-10-29 23:22:01,672 Stage-1 map = 100%,  reduce = 0%
2025-10-29 23:22:07,966 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.51 sec
MapReduce Total cumulative CPU time: 1 seconds 510 msec
Ended Job = job_1761803663889_0006
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 1.51 sec   HDFS Read: 51993 HDFS Write: 6 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 510 msec
OK
670     2
Time taken: 18.278 seconds, Fetched: 1 row(s)
```

6. Earthquakes with magnitude > 7.0

SELECT * FROM earthquake_data WHERE magnitude > 7.0;

```
hive> SELECT * FROM earth_data WHERE magnitude > 7.0 LIMIT 10;
OK
7.30    37      5       5       65      NULL
7.60    26      9       8       7       NULL
7.60    116     8       8       -59     NULL
7.20    236     7       5       108     NULL
7.30    41      9       8       93      NULL
7.30    165     8       7       -103    NULL
7.30    14      9       5       86      NULL
7.50    126     8       8       70      NULL
7.30    527     4       4       58      NULL
7.10    6       1       4       8       NULL
Time taken: 0.058 seconds, Fetched: 10 row(s)
```

7. Group by alert and average magnitude

SELECT alert, AVG(magnitude) FROM earthquake_data GROUP BY alert;

```
hive> SELECT alert, AVG(magnitude) AS avg_magnitude FROM earth_data GROUP BY alert;
Query ID = cloudera_20251029232525_acf11e30-f34e-4c0e-be93-51219795b9c1
Total jobs = 1
Launching Job 1 out of 1
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_1761803663889_0007, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761803663889
_0007/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761803663889_0007
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-10-29 23:25:28,164 Stage-1 map = 0%,  reduce = 0%
2025-10-29 23:25:33,397 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.7 sec
2025-10-29 23:25:39,637 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.56 sec
MapReduce Total cumulative CPU time: 1 seconds 560 msec
Ended Job = job_1761803663889_0007
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 1.56 sec   HDFS Read: 52219 HDFS Write: 21 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 560 msec
OK
NULL    7.061007692307706
Time taken: 18.138 seconds, Fetched: 1 row(s)
```

8. Earthquakes sorted by significance

SELECT * FROM earthquake_data ORDER BY sig DESC LIMIT 10;

```
hive> SELECT * FROM earth_data ORDER BY sig DESC LIMIT 10;
Query ID = cloudera_20251029232626_9c1bf0a0-a460-4195-b709-01c4709e2df7
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_1761803663889_0008, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761803663889
_0008/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761803663889_0008
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-10-29 23:26:37,134 Stage-1 map = 0%,  reduce = 0%
2025-10-29 23:26:42,366 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.72 sec
2025-10-29 23:26:47,578 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.57 sec
MapReduce Total cumulative CPU time: 1 seconds 570 msec
Ended Job = job_1761803663889_0008
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 1.57 sec   HDFS Read: 51954 HDFS Write: 192 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 570 msec
OK
7.80    664     6       3       127     NULL
7.60    22      0       7       121     NULL
7.50    145     7       7       121     NULL
6.90    9       9       6       120     NULL
7.48    142     7       6       120     NULL
7.49    145     7       6       119     NULL
6.90    10      9       9       119     NULL
6.70    40      7       7       118     NULL
6.60    57      7       6       118     NULL
6.90    9       8       8       118     NULL
Time taken: 18.167 seconds, Fetched: 10 row(s)
```

9. Top 5 earthquakes with highest CDI (Community Internet Intensity)

SELECT *FROM earth_data ORDER BY cdi DESC LIMIT 5;

```
hive> SELECT *
    > FROM earth_data
    > ORDER BY cdi DESC
    > LIMIT 5;
Query ID = cloudera_20251029233333_fb2b854e-4aea-4da2-87f1-cfa49ccdc0e2
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_1761803663889_0011, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761803663889
_0011/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761803663889_0011
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-10-29 23:33:36,103 Stage-1 map = 0%,  reduce = 0%
2025-10-29 23:33:41,348 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.74 sec
2025-10-29 23:33:47,579 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.62 sec
MapReduce Total cumulative CPU time: 1 seconds 620 msec
Ended Job = job_1761803663889_0011
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 1.62 sec   HDFS Read: 51947 HDFS Write: 91 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 620 msec
OK
7.21    12      9       8       31      NULL
6.71    11      9       8       16      NULL
7.10    25      9       8       70      NULL
7.90    94      9       7       -52     NULL
6.91    12      9       8       16      NULL
Time taken: 18.151 seconds, Fetched: 5 row(s)
```

10. Count of earthquakes with missing or null alert values

SELECT COUNT(*) AS missing_alerts FROM earth_data WHERE alert IS NULL OR alert = '';

```
hive> SELECT COUNT(*) AS missing_alerts
    > FROM earth_data
    > WHERE alert IS NULL OR alert = '';
Query ID = cloudera_20251029234040_bae4a921-fcc4-4d58-b744-08b5da8f2398
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_1761803663889_0012, Tracking URL = http://quickstart.cloudera:8088/proxy/application_1761803663889
_0012/
Kill Command = /usr/lib/hadoop/bin/hadoop job  -kill job_1761803663889_0012
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2025-10-29 23:40:33,864 Stage-1 map = 0%,  reduce = 0%
2025-10-29 23:40:40,117 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 0.82 sec
2025-10-29 23:40:46,356 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 1.65 sec
MapReduce Total cumulative CPU time: 1 seconds 650 msec
Ended Job = job_1761803663889_0012
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 1.65 sec   HDFS Read: 52273 HDFS Write: 5 SUCCESS
Total MapReduce CPU Time Spent: 1 seconds 650 msec
OK
1301
```

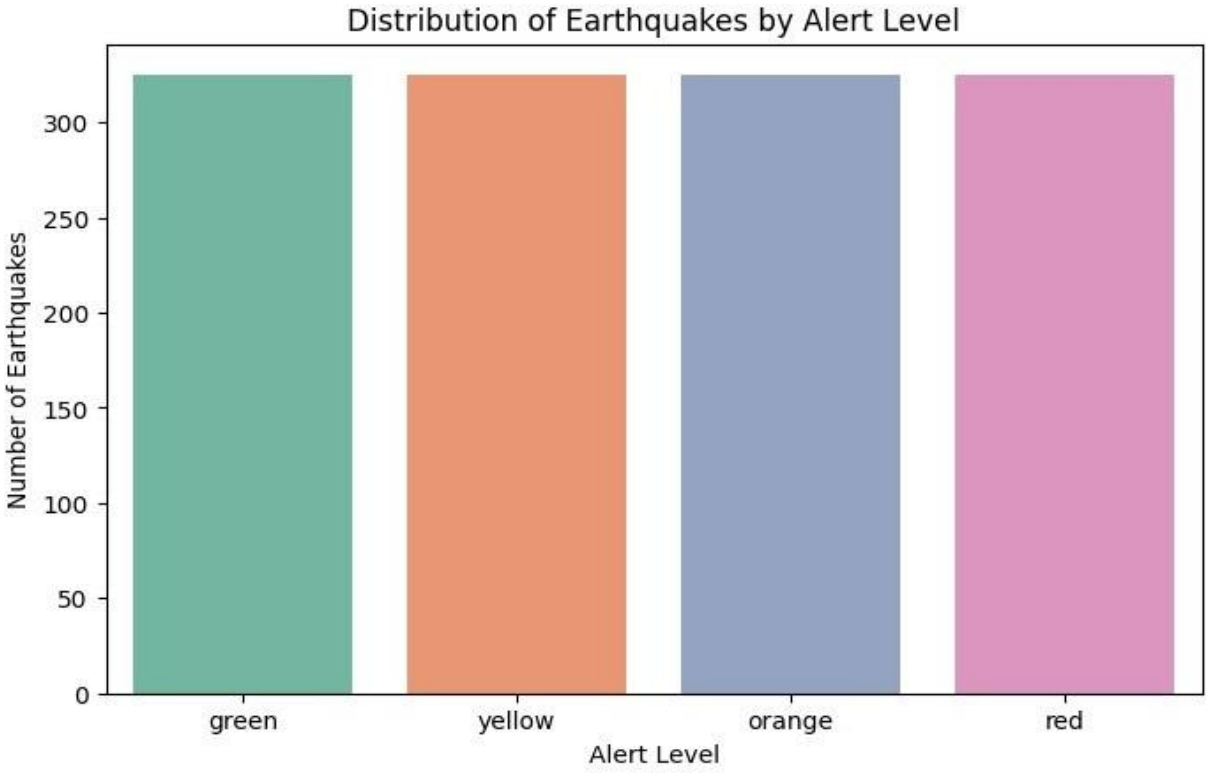
Data Visualization

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

%matplotlib inline

df = pd.read_csv("earthquake_alert_balanced_dataset.csv") print(df.shape)
print(df.columns)


(1300, 6)
```

Q What is the distribution of earthquakes across different alert levels?

Purpose: To understand how many events fall under each severity category (green, yellow, orange, red).

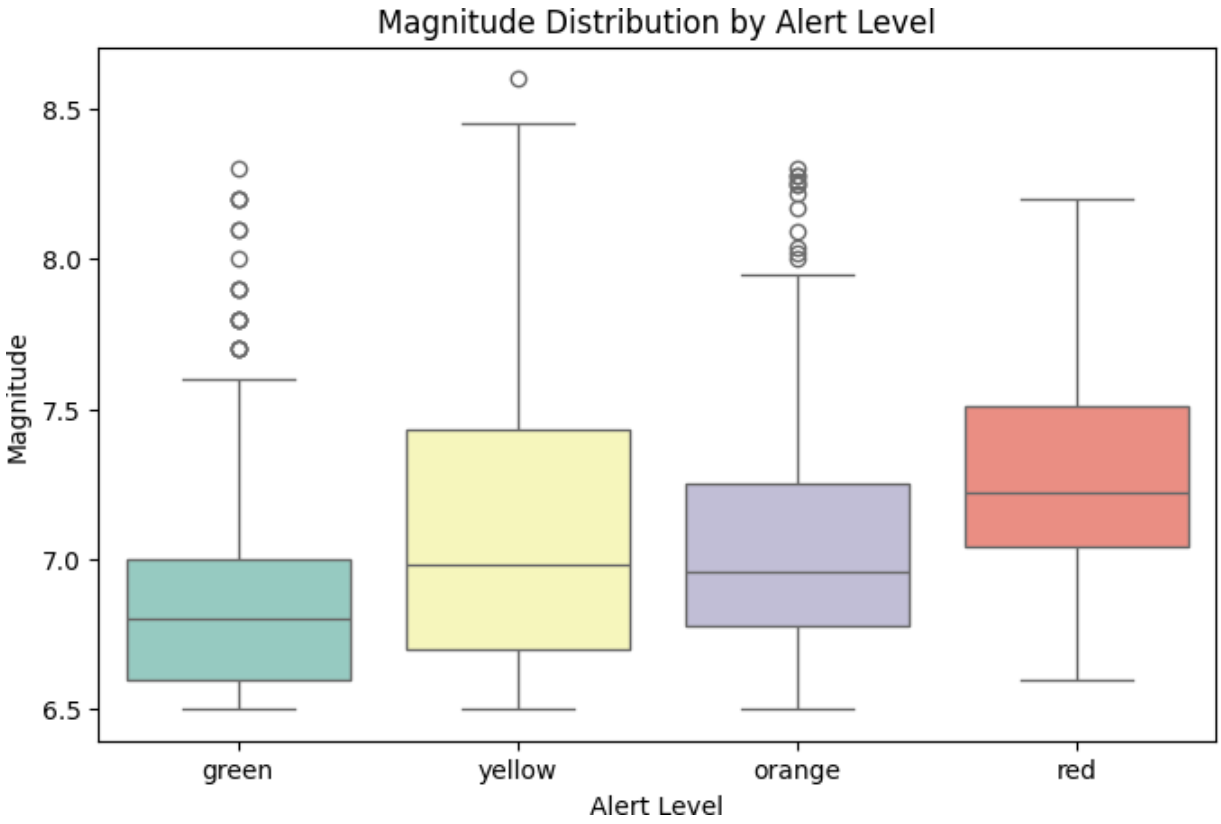
```
plt.figure(figsize=(8,5))
sns.countplot(x='alert', hue='alert', data=df, palette='Set2', legend=False)
plt.title("Distribution of Earthquakes by Alert Level") plt.xlabel("Alert Level")
plt.ylabel("Number of Earthquakes") plt.show()
```



Q How does earthquake magnitude vary across alert levels?

Purpose: To compare the intensity of earthquakes (magnitude) across different alert categories— green, yellow, orange, and red. This helps us understand whether higher alert levels are consistently associated with stronger earthquakes.

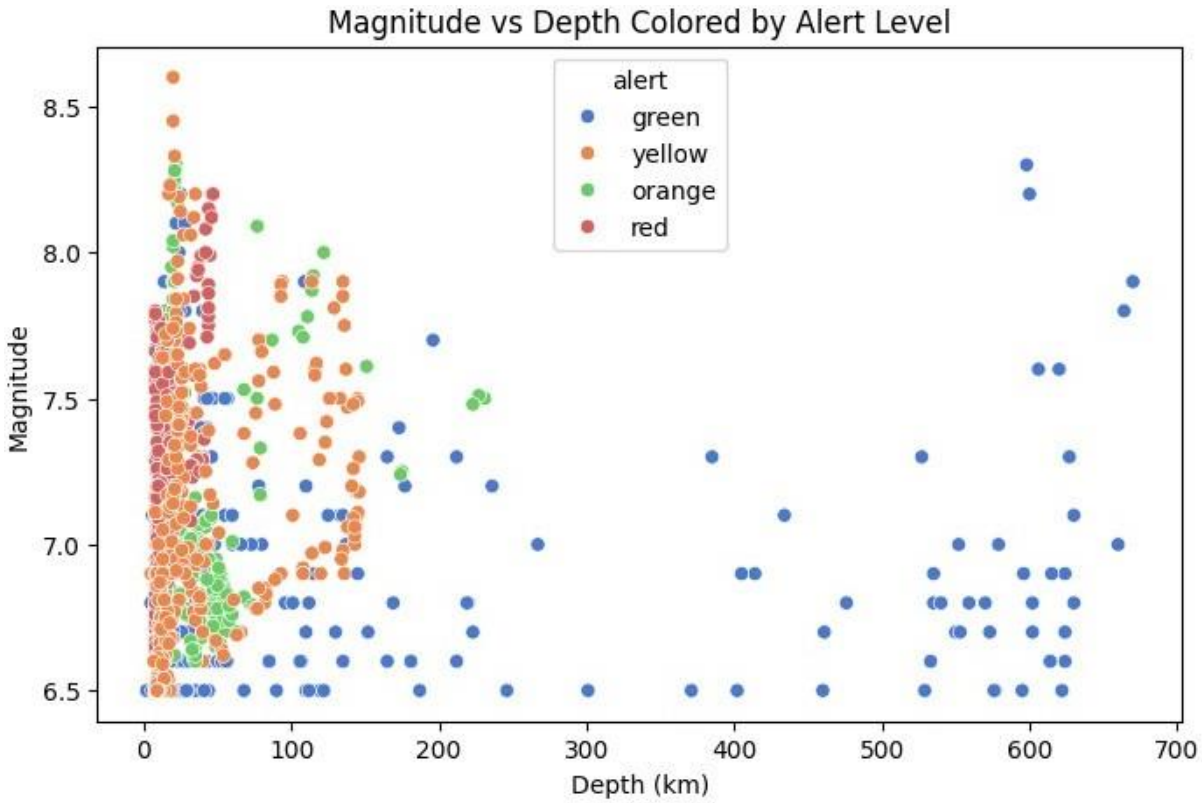
```
plt.figure(figsize=(8,5))
sns.boxplot(x='alert', y='magnitude', hue='alert', data=df, palette='Set3', dodge=False,
legend=False) plt.title("Magnitude Distribution by Alert Level") plt.xlabel("Alert Level")
plt.ylabel("Magnitude") plt.show()
```



Q: Is there a relationship between depth and magnitude of earthquakes?

Purpose:To analyze whether deeper earthquakes tend to be stronger or weaker, and how they are distributed across alert levels.

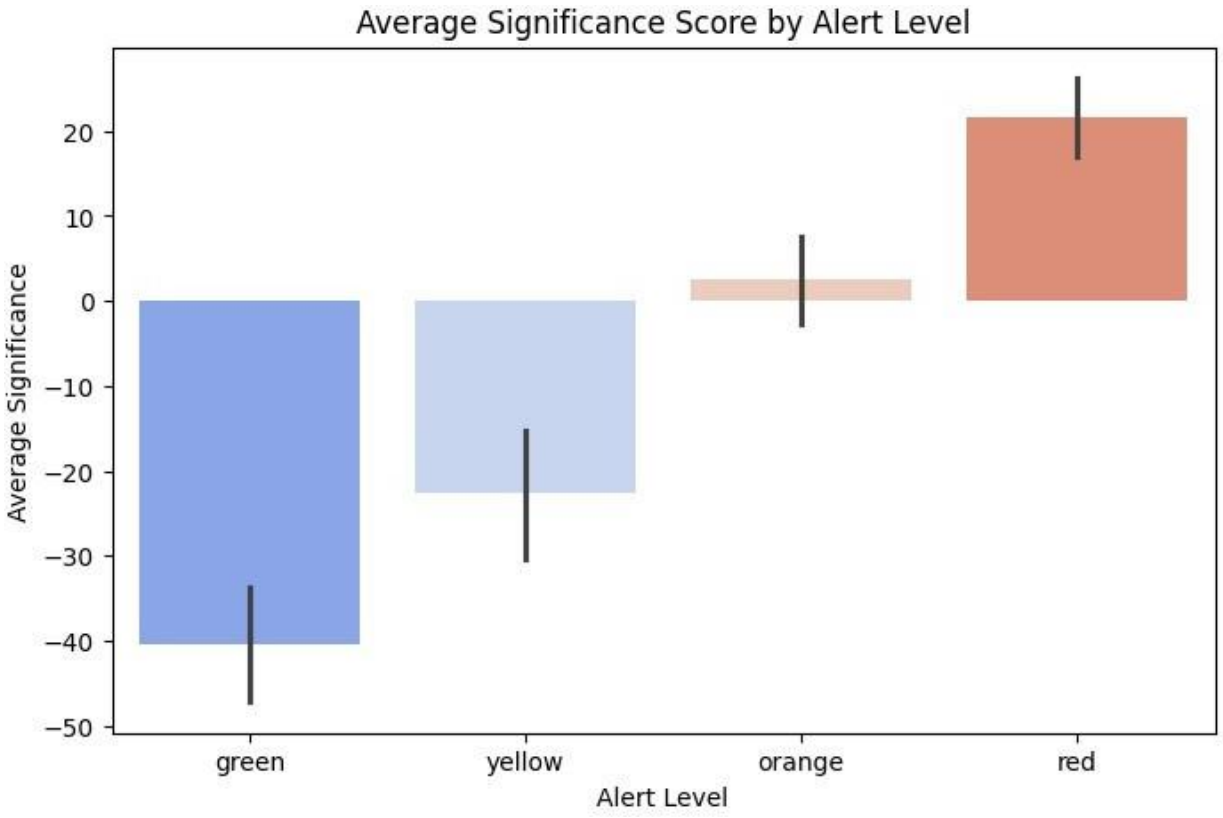
```
plt.figure(figsize=(8,5))
sns.scatterplot(x='depth', y='magnitude', hue='alert', data=df, palette='muted')
plt.title("Magnitude vs Depth Colored by Alert Level") plt.xlabel("Depth (km)")
plt.ylabel("Magnitude") plt.show()
```



Q: Which alert level has the highest average significance score?

Purpose:To identify which alert category (green, yellow, orange, red) is linked to the most impactful earthquakes based on the sig score

```
plt.figure(figsize=(8,5))
sns.barplot(x='alert', y='sig', hue='alert', data=df, estimator='mean', palette='coolwarm', dodge=False, legend=False) plt.title("Average Significance Score by Alert Level") plt.xlabel("Alert Level")
plt.ylabel("Average Significance") plt.show()
```

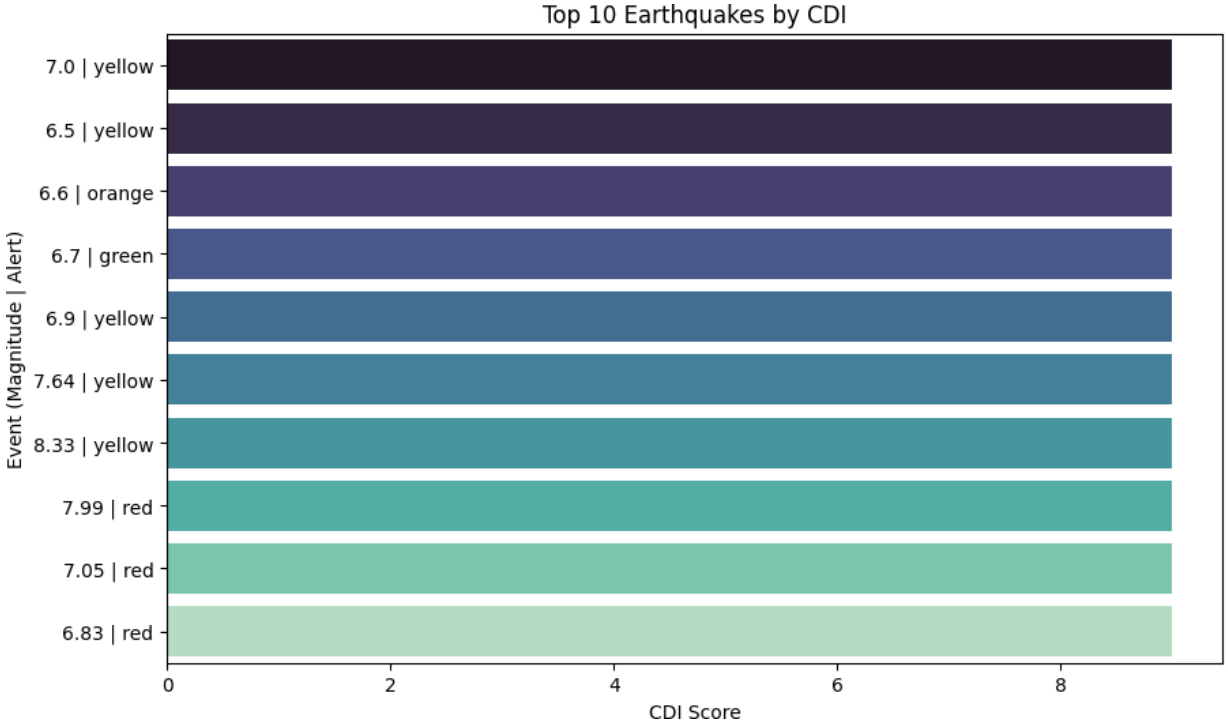


Q: What are the top 10 earthquakes with the highest CDI (Community Internet Intensity)?

- **Purpose:To identify the most strongly felt earthquakes based on public reports (cdi score). Useful for understanding real-world impact.**

```
top_cdi = df.sort_values(by='cdi', ascending=False).head(10)

 # Add a new column for labels (e.g., magnitude + alert) top_cdi['label'] =
 top_cdi['magnitude'].astype(str) + " | " + top_cdi['alert']

plt.figure(figsize=(10,6))

sns.barplot(x='cdi', y='label', data=top_cdi, orient='h', palette='mako')

plt.title("Top 10 Earthquakes by CDI") plt.xlabel("CDI
Score") plt.ylabel("Event (Magnitude | Alert)") plt.show()

C:\Users\Karunesh\AppData\Local\Temp\ipykernel_17484\1620549304.py:7: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
```

```
removed in v0.14.0. Assign the `y` variable to `hue` and set

`legend=False` for the same effect.

    sns.barplot(x='cdi', y='label', data=top_cdi, orient='h', palette='mako')
```


Top 10 Earthquakes by CDI
7.0 | yellow
6.5 | yellow
6.6 | orange
6.7 | green
6.9 | yellow
7.64 | yellow
8.33 | yellow
7.99 | red
7.05 | red
6.83 | red
Event (Magnitude | Alert)
0
2
4
6
8
CDI Score

Conclusion

The visual analysis of the earthquake alert dataset reveals distinct patterns across alert levels, magnitude, depth, and public impact metrics. The distribution plot shows that lower alert levels (green and yellow) are more frequent, while higher alert levels (orange and red) are comparatively rare but significantly more intense. Boxplot analysis confirms that earthquakes categorized under red alerts tend to have higher magnitudes, indicating a direct correlation between alert severity and earthquake strength.

The scatter plot between depth and magnitude suggests that impactful earthquakes are not limited to shallow depths; however, many high-magnitude events occur closer to the surface, amplifying their destructive potential. The average significance score across alert levels further reinforces that red alerts are associated with the most consequential seismic events. Lastly, the top 10 earthquakes ranked by CDI (Community Internet Intensity) highlight that public perception and felt intensity do not always align with technical magnitude, emphasizing the importance of community-based reporting in disaster response.

Overall, these visual insights provide a comprehensive understanding of how technical parameters and public impact metrics interact across different alert levels, aiding in better preparedness and response strategies.