Group 5: Deadly & Damaging Storm Events

Ankur Bhattacharjee, Bimala Joshi, Cezane Karki, Isabella Pham, Kiran S. Kuchekar, Shivangi Vipulbhai Borad

University of North Texas

ADTA 5240: Harvesting, Storing and Retrieving Data

Dr. Schenita Floyd

April 29, 2025

Contents

Executive Summary

As a team of data engineers and data analysts hired by the American Red Cross, a nonprofit organization that helps the ones in need and focus to prepare well against the disaster, we work to improve the preparedness for the disaster created by storms. Our project aims to collect, process, analyze and interpret the required data for the further preparedness of this disaster.

For doing the analysis, we first collected two sets of data: static and streaming data. The static data was collected from National Oceanic and Atmospheric Administration's (NOAA) from 1950 till 2024. The streaming data was obtained from National Weather Service (NWS) API and was collected on 20th April, 2025 at various times. The data was stored on Google Cloud Platform (GCP) through buckets and Dataproc. The data was also cleaned using OpenRefine and was thus prepared for analysis.

The analysis was done through various tools like BigQuery, Apache Hive and Apache Spark. Spark was found to have faster processing time compared to Hive. The critical trends of storm, severity, impact on geographic locations and seasonality were affected by severe weather. It was also found through streaming data that there were inconsistencies in urgency and emergency alerts. Our analysis found out that a more predictive model, targeted resource management, and effective alert systems are required for effective management of storm disaster response and relief.

We recommend processing the mechanism of the predictive models, geographic risk mapping for storm and alert system management for the better preparedness of such disasters. This will help in quick mobilization of resources and effective implementation of the process. We recommend that data scientists work to make a predictive model for forecasting storm events, risk mapping, and better communication and alerts to the public.

Introduction

The American Red Cross is a network of volunteers and employees who are committed to providing help to the ones who need it. While providing help to those in need, they also prepare the community for impending disasters (American Red Cross, 2025). Many states regularly suffer from storm events such as tornadoes, hurricanes, and winter storms. Through the use of static and streaming (live) data, our team of data engineers were hired to construct a data pipeline and work alongside data scientists and analysts to provide actionable insight into how the Red Cross can better prepare for such incidents.

By framing our work process through the various stages of the data life cycle, we were able to work more efficiently and find workflow solutions that can be implemented later down the line. We believe that better preparedness and management of post-disaster scenarios could be achieved through this research. To that end, we are trying to help the American Red Cross with this project.

Work Process Overview

Generation

The data was generated from the storm incidents that occurred in different areas and at different times. Two data sets were extracted: one static and one streaming. The static dataset contains the data on storms from 1950 until 2024. The streaming dataset was collected on 20th of April, 2025 at different times, and the latest data was used in our analysis. Both sets of generated data were collected and stored in a GCP bucket for further analysis.

Collection

Streaming data was sourced from the National Weather Service (NWS) API. It was transferred to Google Cloud Platform (GCP) using python and transferred to a GCP bucket at different times.

Static data was obtained from the National Oceanic and Atmospheric Administration's (NOAA) Storm Events Database. The database contains storm event data from different years, and mentions the type of the event like thunderstorm wind, Tornado, winter storm, Hurricane or Typhoon. The beginning and the end time of each storm type is collected for the analysis. Similarly, it also contains the data of the beginning and ending of the storm.

Weather report streaming data contains the timely, up-to-date information, and includes observations such as the onset and end time of each recorded storm, as well as the severity of each storm. Response and urgency data were also collected as part of this dataset.

Processing

The data was processed (i.e. wrangled and cleaned) using OpenRefine. Null and unnecessary columns were removed using this application. Following the maxim “garbage in, garbage out”, we removed columns that were deemed unnecessary or contained too many null values. Further techniques such as text transformation and splitting individual cells which contained multiple values into distinct columns were also employed. After cleaning, each dataset was stored for future analysis.

OpenRefine

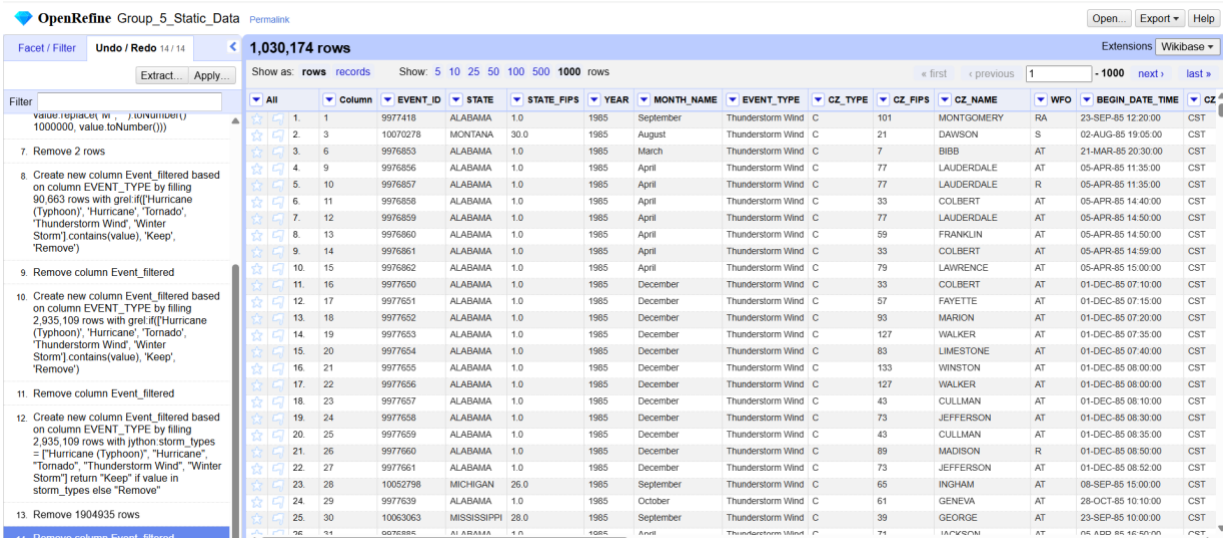


Figure 1.1. Static dataset loaded into OpenRefine. Screenshot taken by Ankur Bhattacharjee.

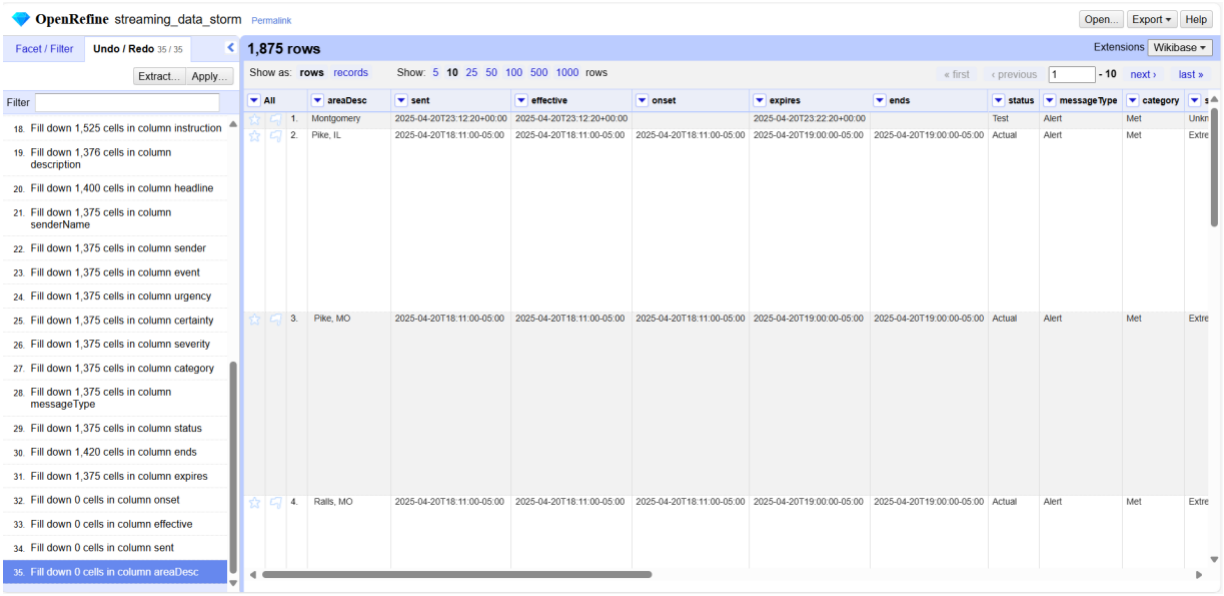


Figure 1.2. Streaming dataset loaded into OpenRefine. Screenshot taken by Ankur Bhattacharjee.

We used OpenRefine to clean and structure the raw data, organizing the static and streaming storm raw datasets into two separated projects. We applied various operations to refine the datasets, removing blank and erroneous values. While working with data, the golden rule remains: "Garbage in, garbage out".

The common OpenRefine operations used in our project are as follows:

i. Removing null value column

ii. Text Transformation

iii. Split multi-valued cell in column

iv. Fill down column

v. Remove null value instances

vi. Create new column based on desired event_type

Storage

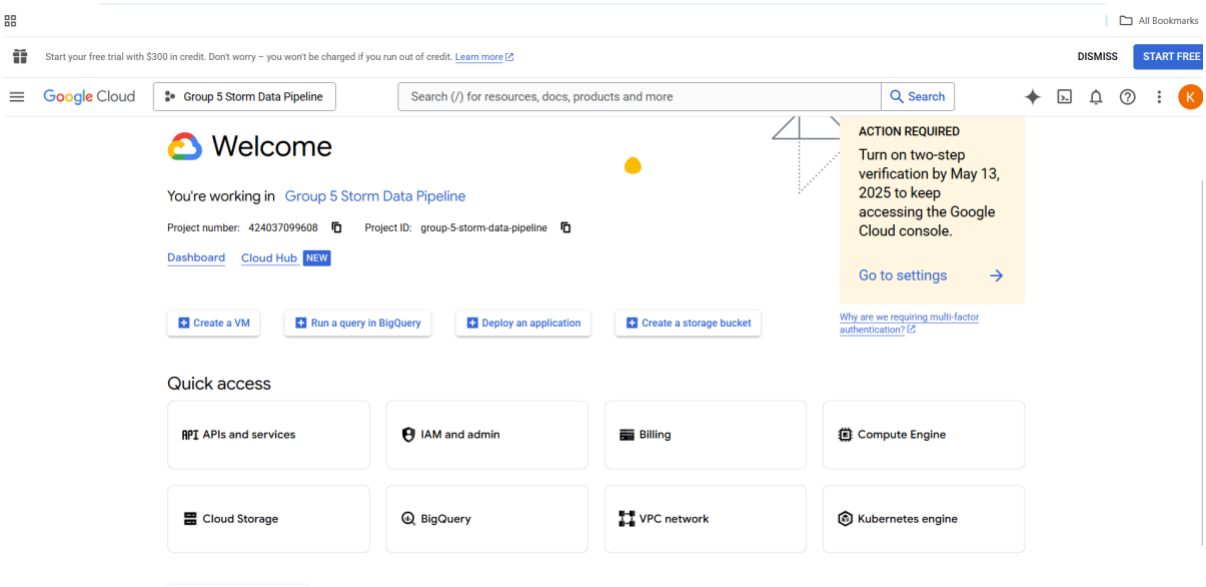


Figure 2. Screenshot taken by Kiran S. Kuchekar.

GCP Project

GCP was used to configure and process resources for the "Group 5 Storm Data Pipeline" in this project. The GCP console allowed us to manage cloud services like Compute Engine, BigQuery, and Apache sub-modules for a wide variety of data-related tasks. Our purpose for using GCP was to oversee a cloud setup in which we could execute queries in BigQuery, deploy apps, create buckets for storage, and control VMs to process and deal with storm-related information effectively.

The flexibility and scalability of GCP made it ideal for this project, as it involved a significant amount of big data. With services such as IAM for controlling access, we were able to maintain good performance as well as data security. Using this platform, we could streamline our workflow, automate deployments, and enable collaborative, cloud-native data analysis.

GCP Storage Bucket

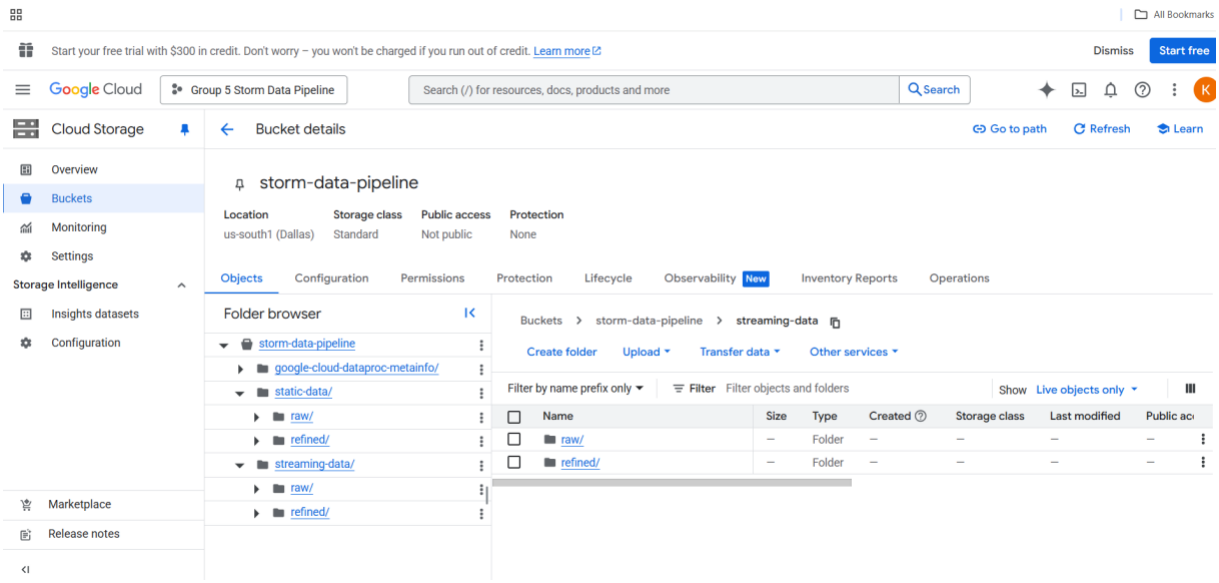


Figure 3. Screenshot taken by Kiran S. Kuchekar.

We used Google Cloud Storage for the project data pipeline. The service made it easier for us to manage and store extensive datasets in a well-structured manner in the cloud. All cleaned and processed data were uploaded to a bucket named "storm-data-pipeline". Streaming and static data were uploaded to separate folders, with subdirectories within each folder dedicated to the raw, original data and refined data, respectively. This enhanced our workflow by facilitating easy separation between processed and unprocessed data. We used Google Cloud Storage due to its scalability, seamless integration with other GCP services, and robust support for collaborative data engineering workflows.

Management

As previously stated data was stored in a GCP bucket with the proper roles assigned to each member through IAM. Specific IAM access was provided to each member of the team for further analysis work using specialized tools - such as Apache Hive and Spark - for enhanced security. In short, the data and its analysis progress was shared with everyone through GCP.

Analysis

GCP Dataproc

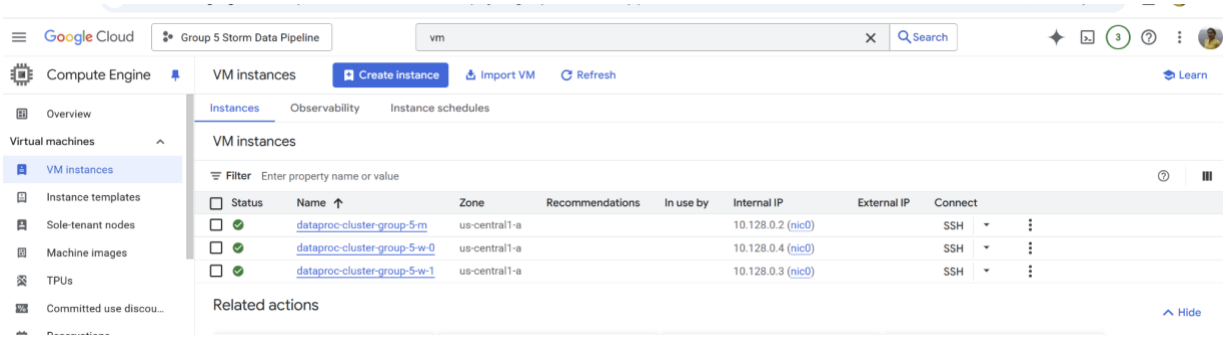


Figure 4. GCP Dataproc. Screenshot taken by Bimala Joshi.

We used GCP Dataproc, which simplified the job of running workloads of big data. To process storm data, Dataproc was used to create and manage the cluster, which consisted of master and worker virtual machine (VM) instances. These VMs will be used in the future to run data processing jobs through applications such as BigQuery, Apache Spark and HiveQL. The use of Compute Engine and Dataproc together will help with parallel processing large quantities of data, providing the scalability, and other cloud-based resources. This is fast and cost-effective, as it can be started and shut down quickly to reduce costs.

BigQuery: Streaming Data

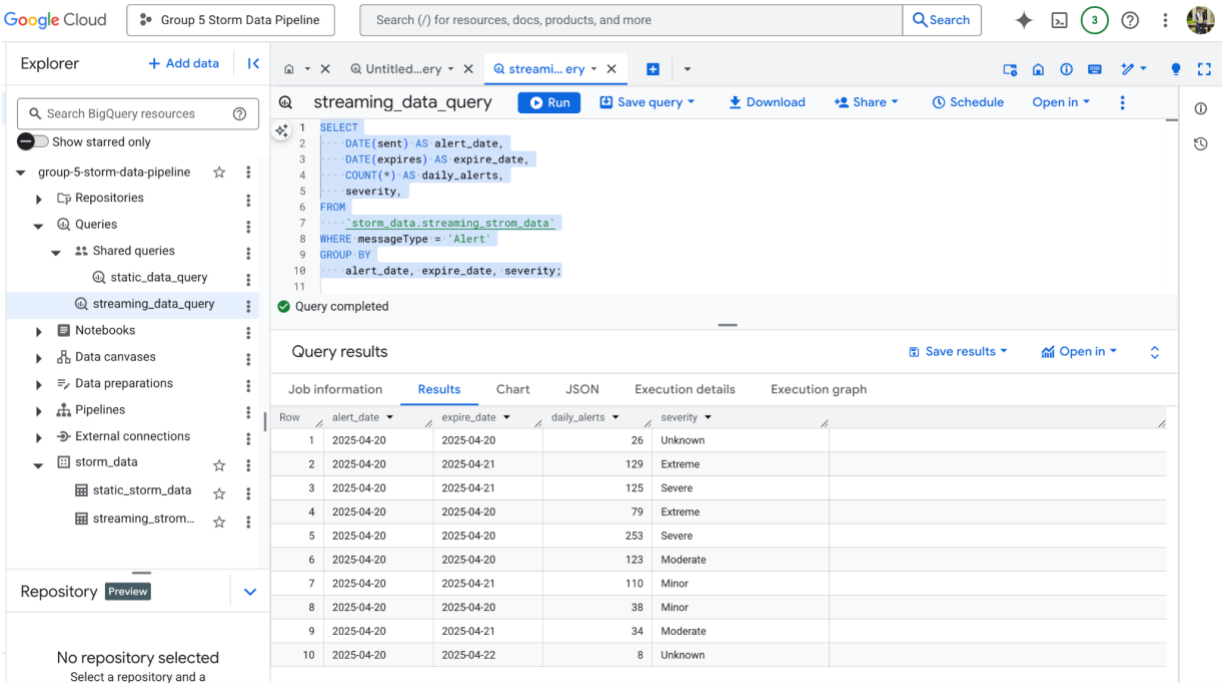


Figure 5.1. Big Query to get the number of daily alerts and severity between the alert and expiry date. Screenshot taken by Cezane Karki.

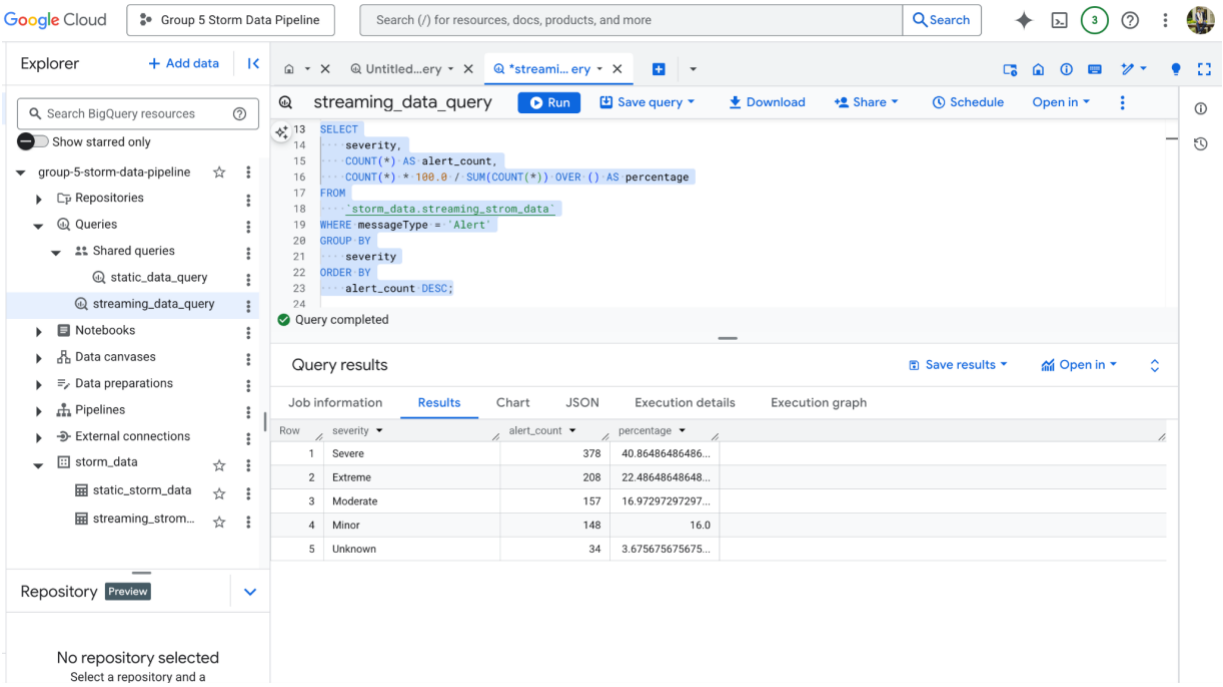


Figure 5.2. Big Query to get the severity level and its alert count along with the percentage of each severity level out of the total number of alerts. Screenshot taken by Cezane Karki.

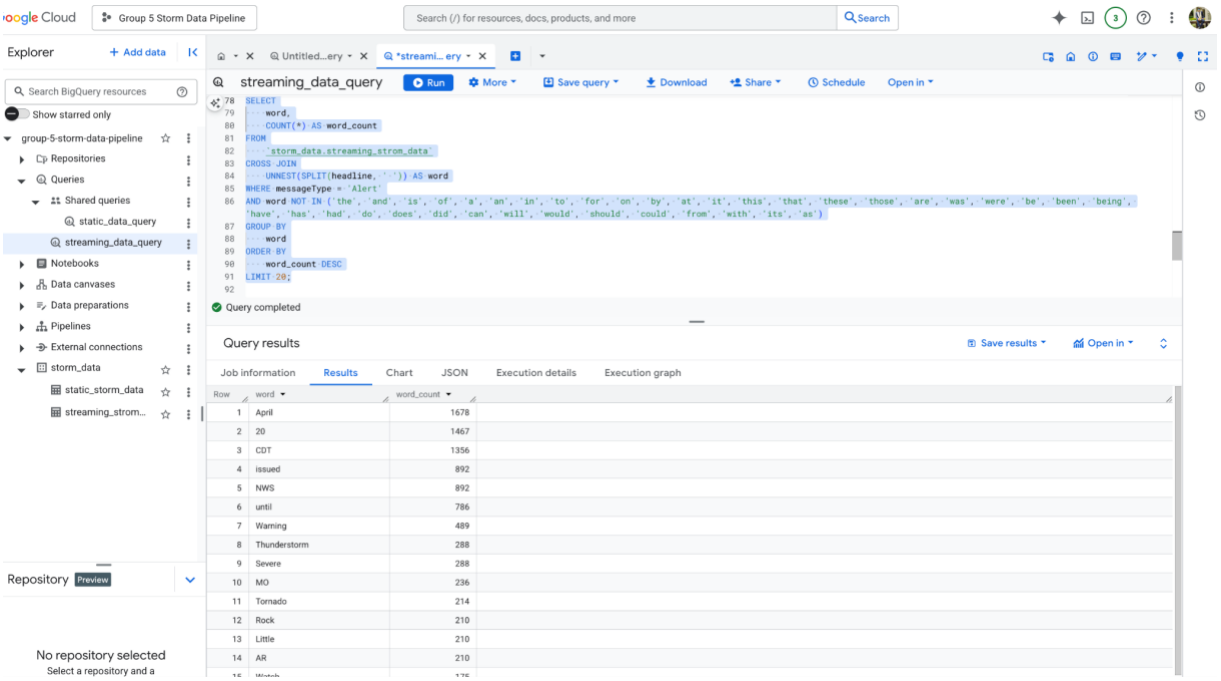


Figure 5.3. Big Query to get the word counts used in the headlines of the data. Screenshot taken by Cezane Karki.

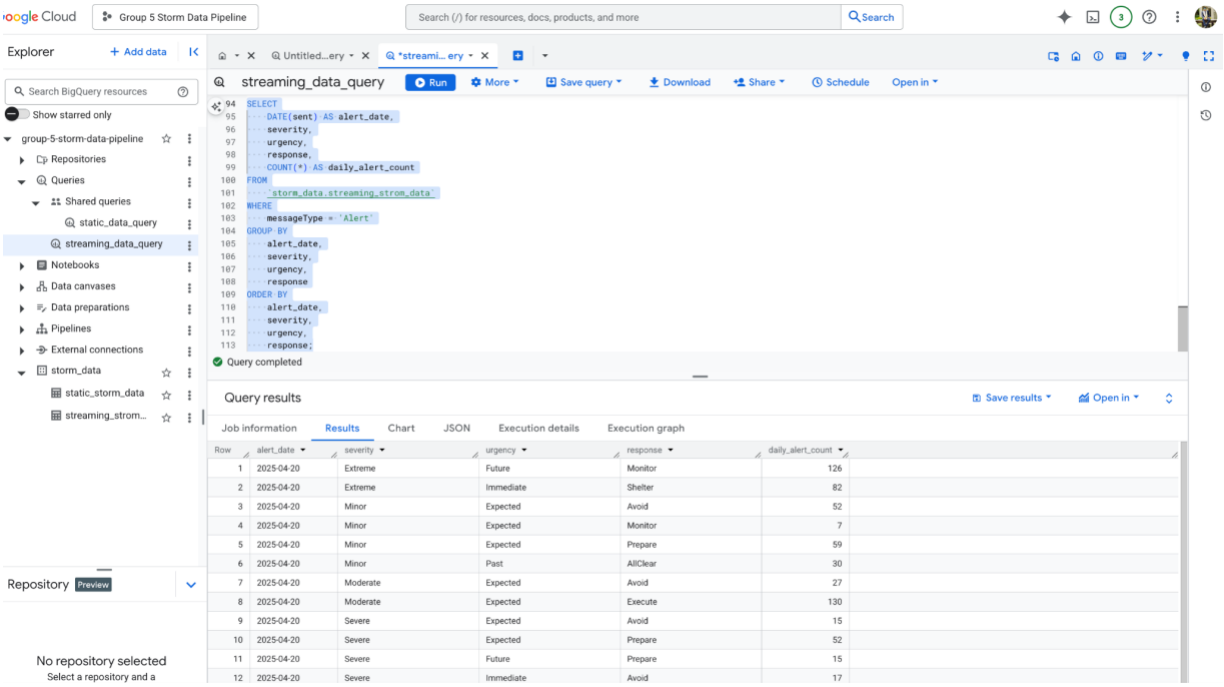


Figure 5.4. Big Query to get the number of storm alerts for each date, broken down by severity, urgency, and response. Screenshot taken by Cezane Karki.

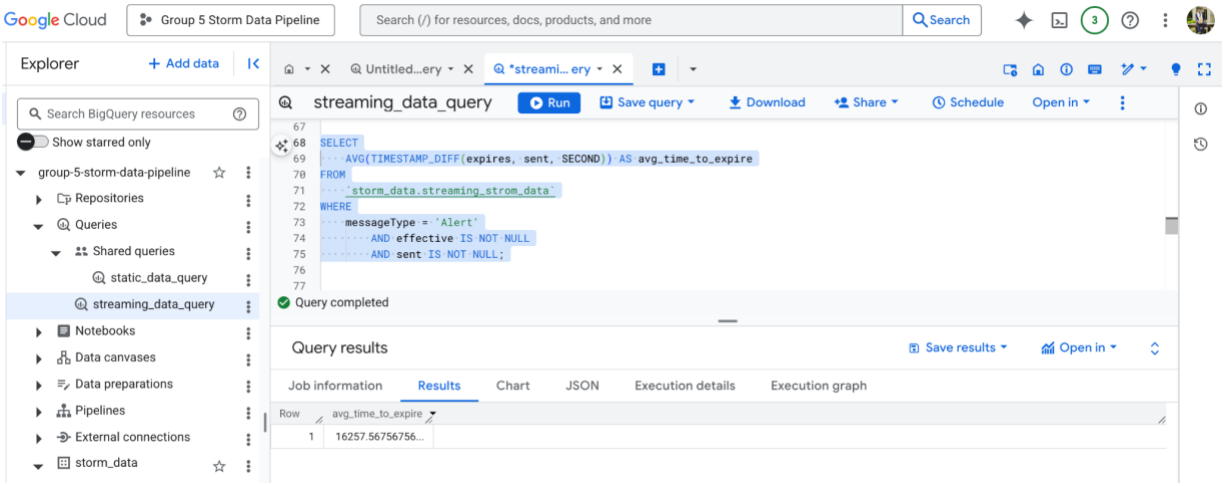


Figure 5.5. Big Query to get the average time for an alert to expire. Screenshot taken by Cezane Karki.

BigQuery: Static Data

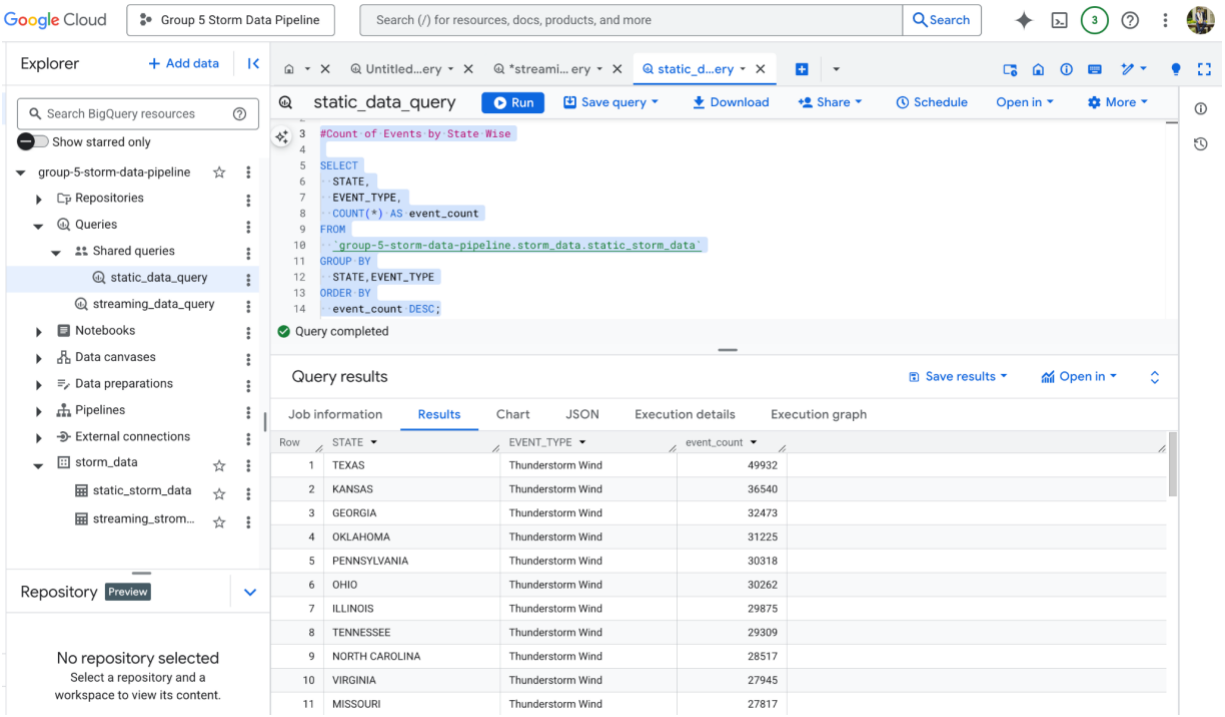


Figure 6.1. Big Query to get the number of storm events for each state and event type, ordered by the event count in descending order. Screenshot taken by Cezane Karki.

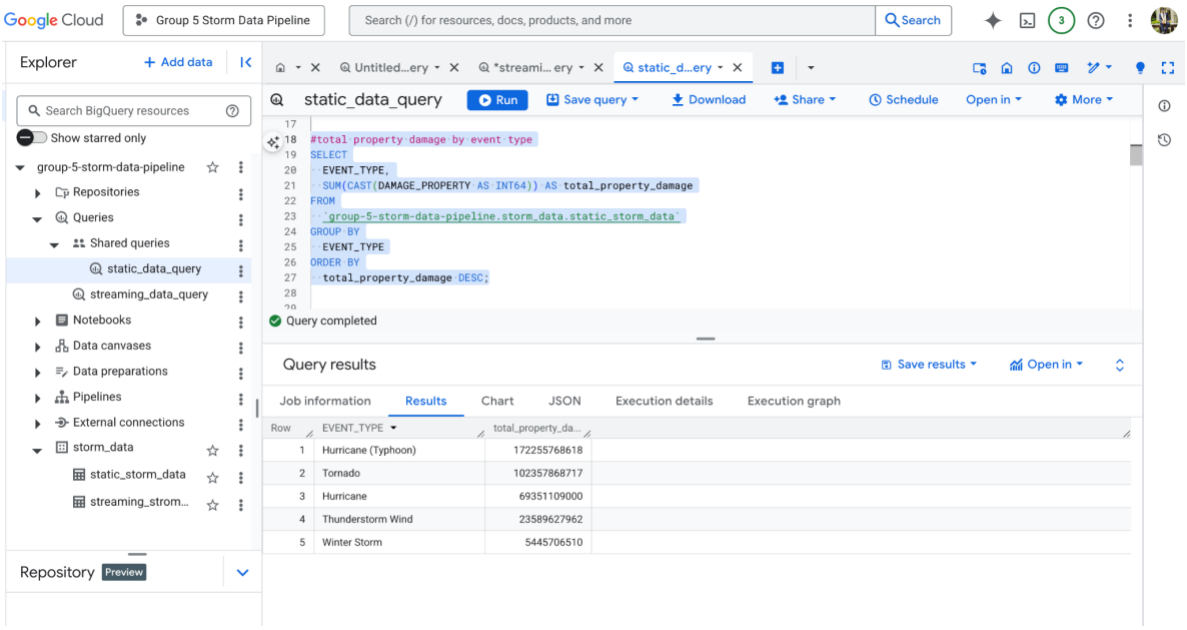


Figure 6.2. Big Query to get the total damaged property grouped by the event type. Screenshot taken by Cezane Karki.

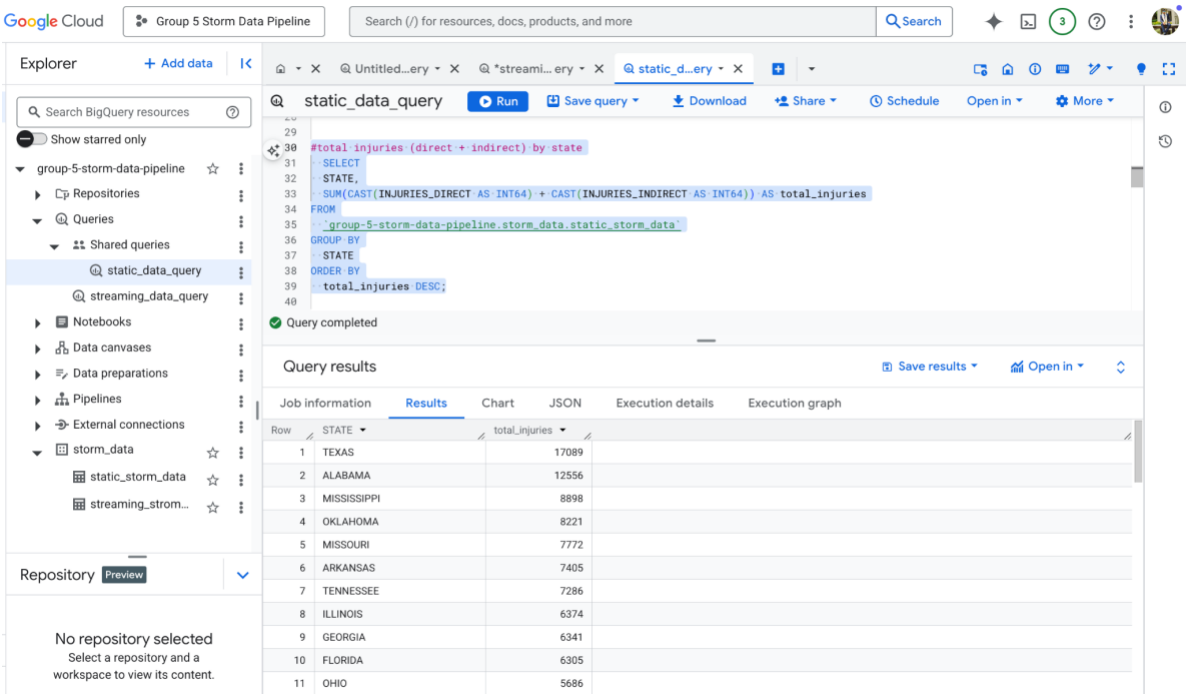


Figure 6.3. Big Query to get the total number of injuries in each state. Screenshot taken by Cezane Karki.

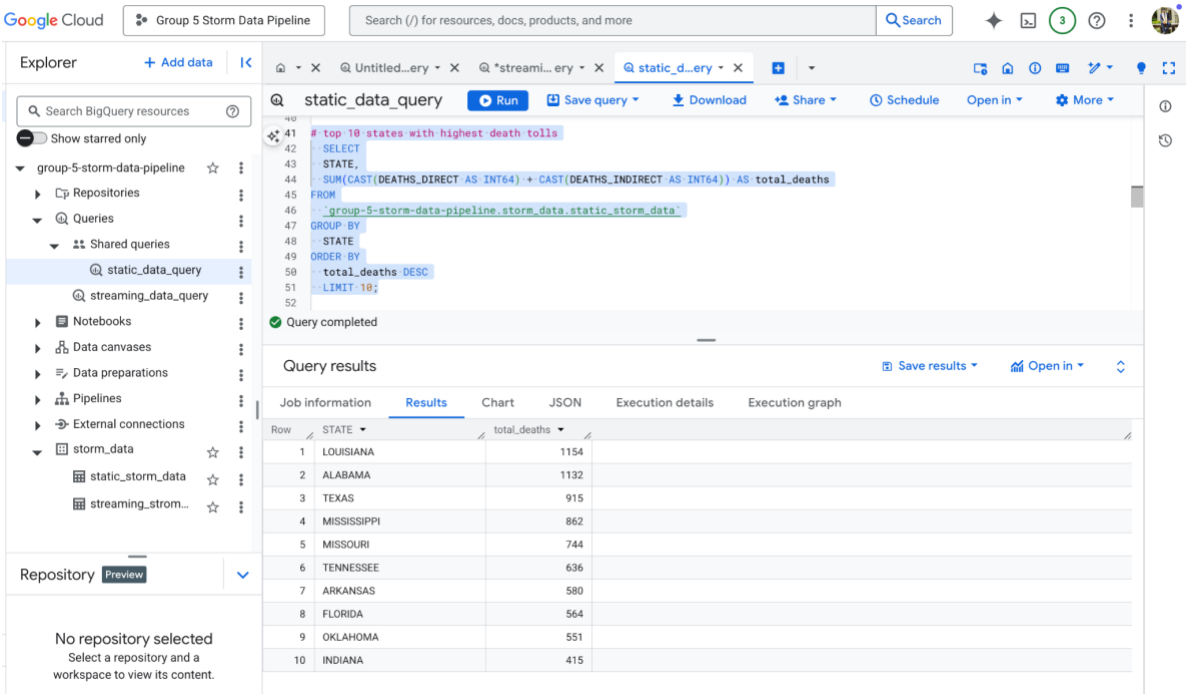


Figure 6.4. Big Query to get the total number of deaths in each state. Screenshot taken by Cezane Karki.

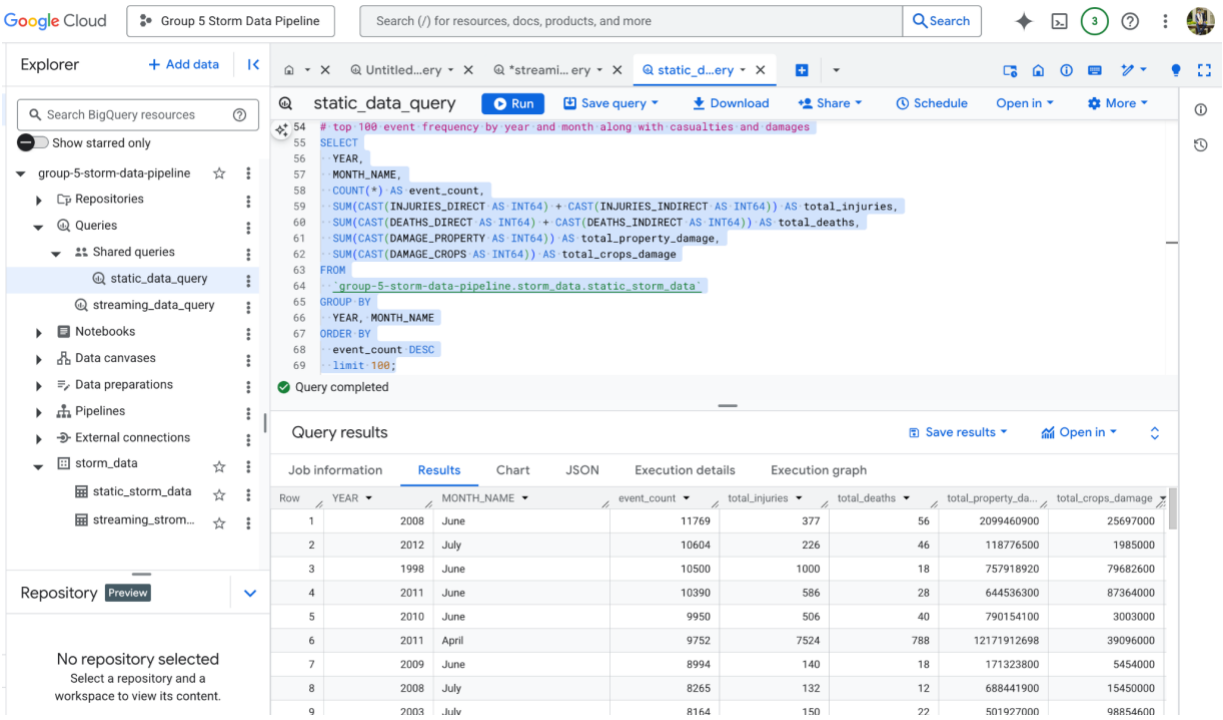


Figure 6.5. Big Query to get the total number of storm events, injuries, deaths, property damage, and crop damage for each year and month. Screenshot taken by Cezane Karki.

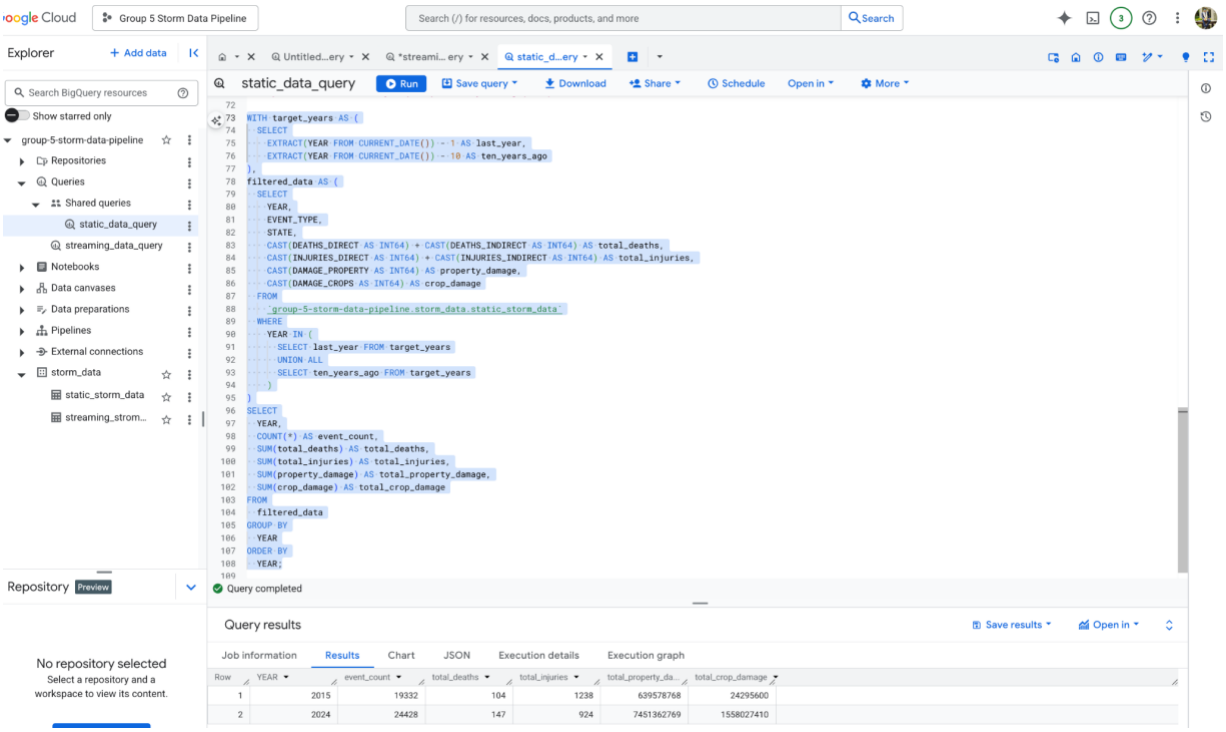


Figure 6.6. Big Query to get the storm data for the last year and prior 10 years for the calculation of total number of events, deaths, injuries, property damage, and crop damage for each year (2024 and 2015). Screenshot taken by Cezane Karki.

We used Google BigQuery to deeply analyze the storm dataset. We queried the real storm alert data in real-time to retrieve trends in frequency, severity, urgency, and types of response. SQL queries were written to aggregate the alerts over date, flag them as severe and urgent, and determine the average time from when an alert was made until when it lapsed. We also extracted the most common kinds of weather occurrences and extracted the most common words from alert titles to find dominant themes in storm communication. The high-velocity and scalable nature of BigQuery enables us to efficiently query this vast dataset, which supports the capacity to derive sound conclusions regarding public weather warning behavior and communication trends in terms of warnings.

Apache Hive: Streaming Data

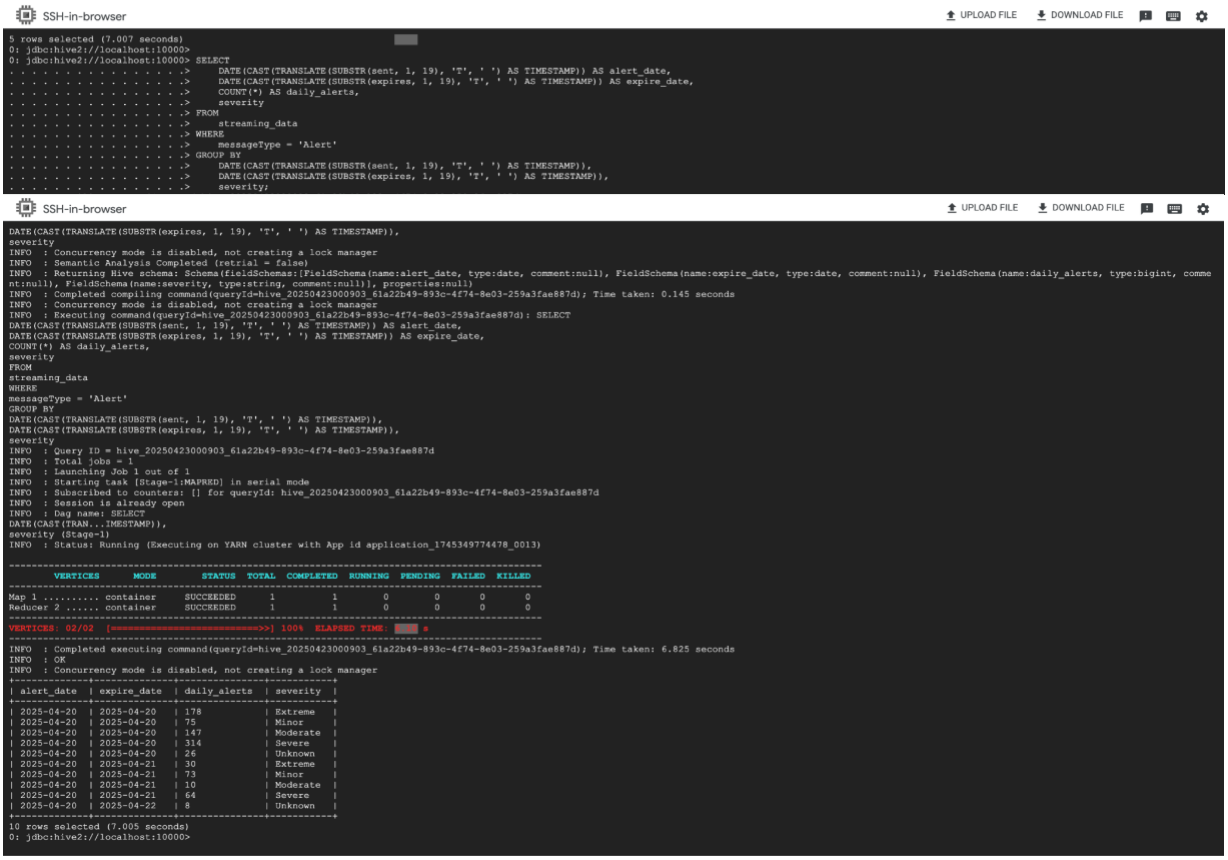


Figure 7.1. HiveQL query to identify the number of cyclone alerts received on a daily basis by severity and an indication of alert frequency with the help of impacted timestamp conversion which would help in aggregation over time. Screenshots taken by Shivangi Borad.

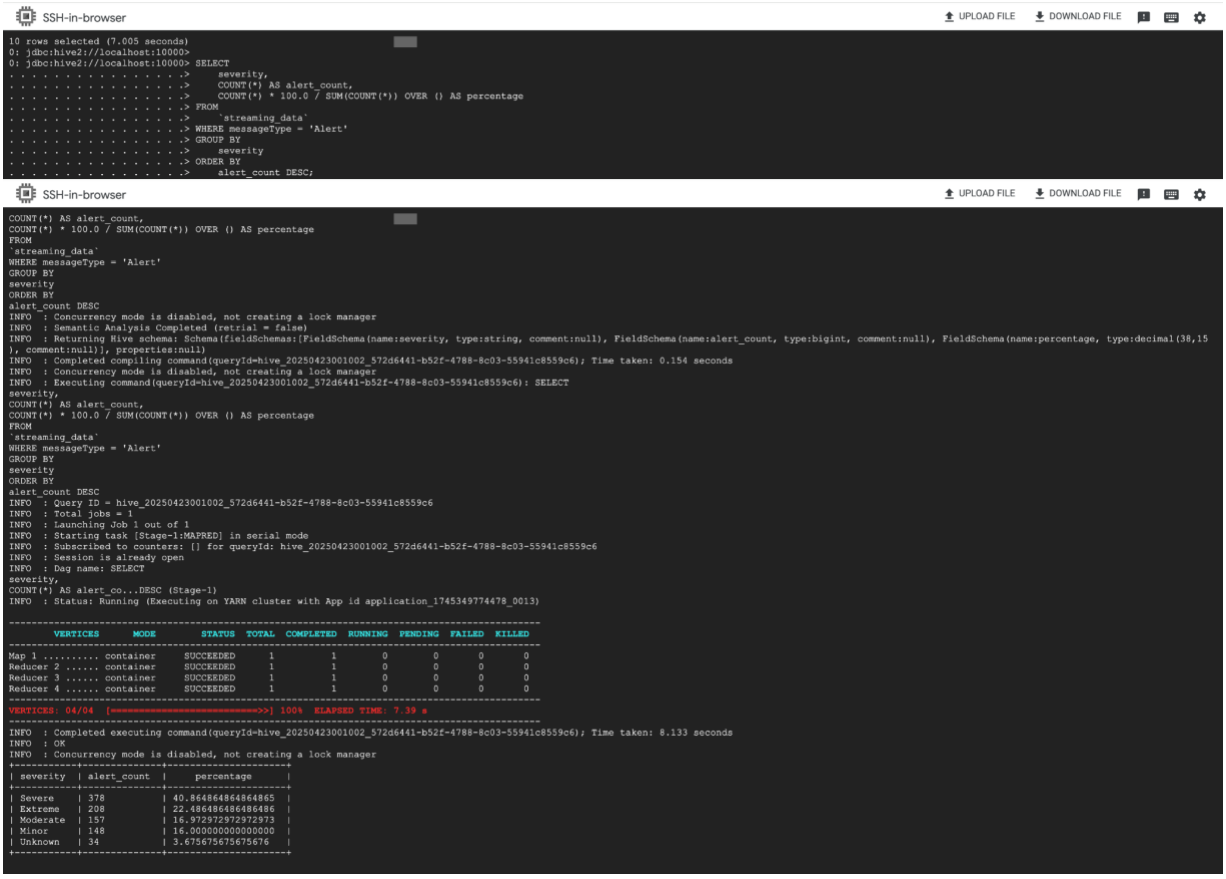


Figure 7.2. HiveQL query for estimating the number and percentage of alerts at each severity scale, which includes grouping by severity for indicating the most often-observed alert types generated during a cyclone event. Screenshot taken by Shivangi Borad.

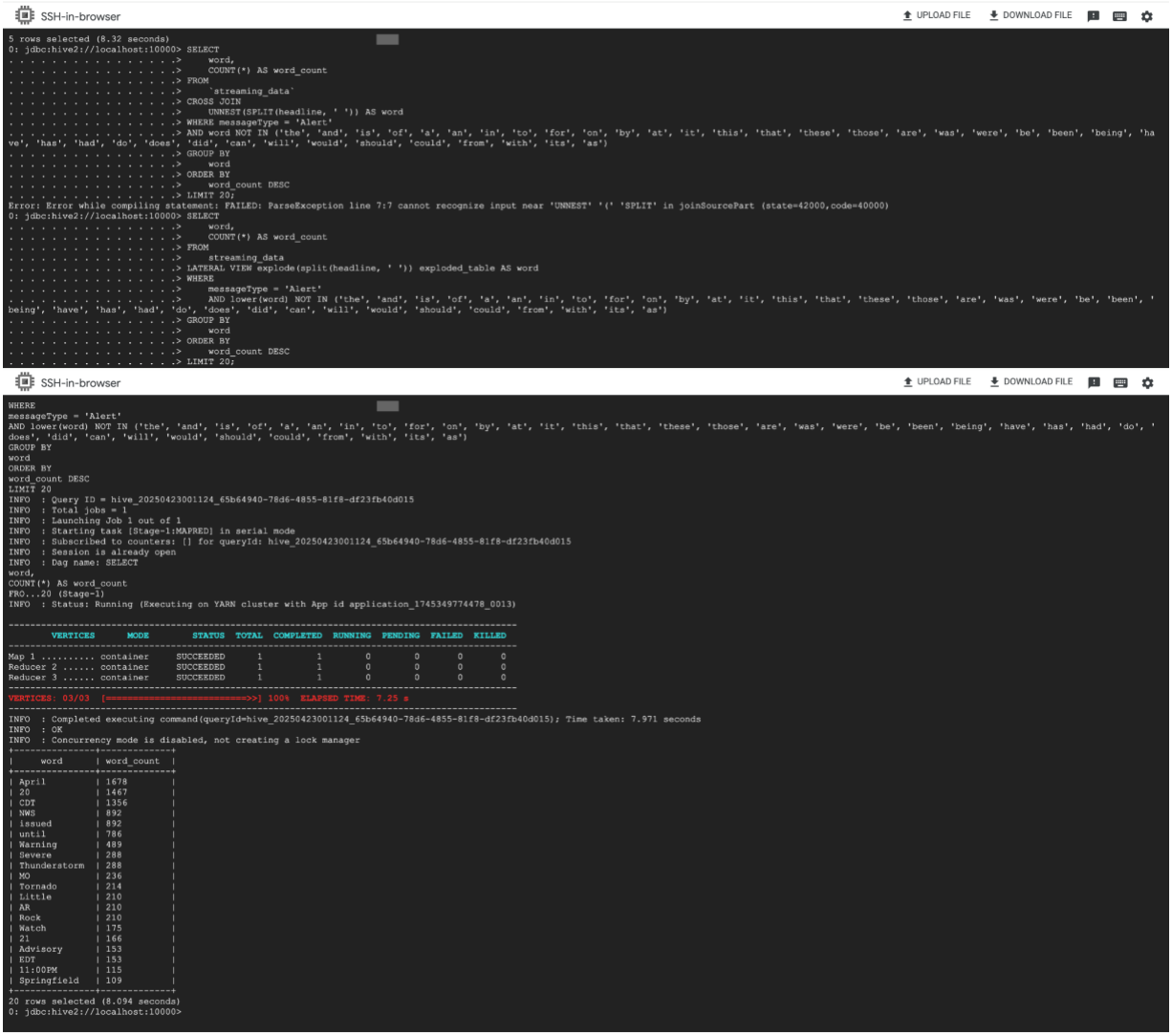


Figure 7.3. HiveQL query extracting and counting the most repeated words in alert headlines filtering out stopwords for highlighting important terms in public understanding warnings. Screenshot taken by Shivangi Borad.

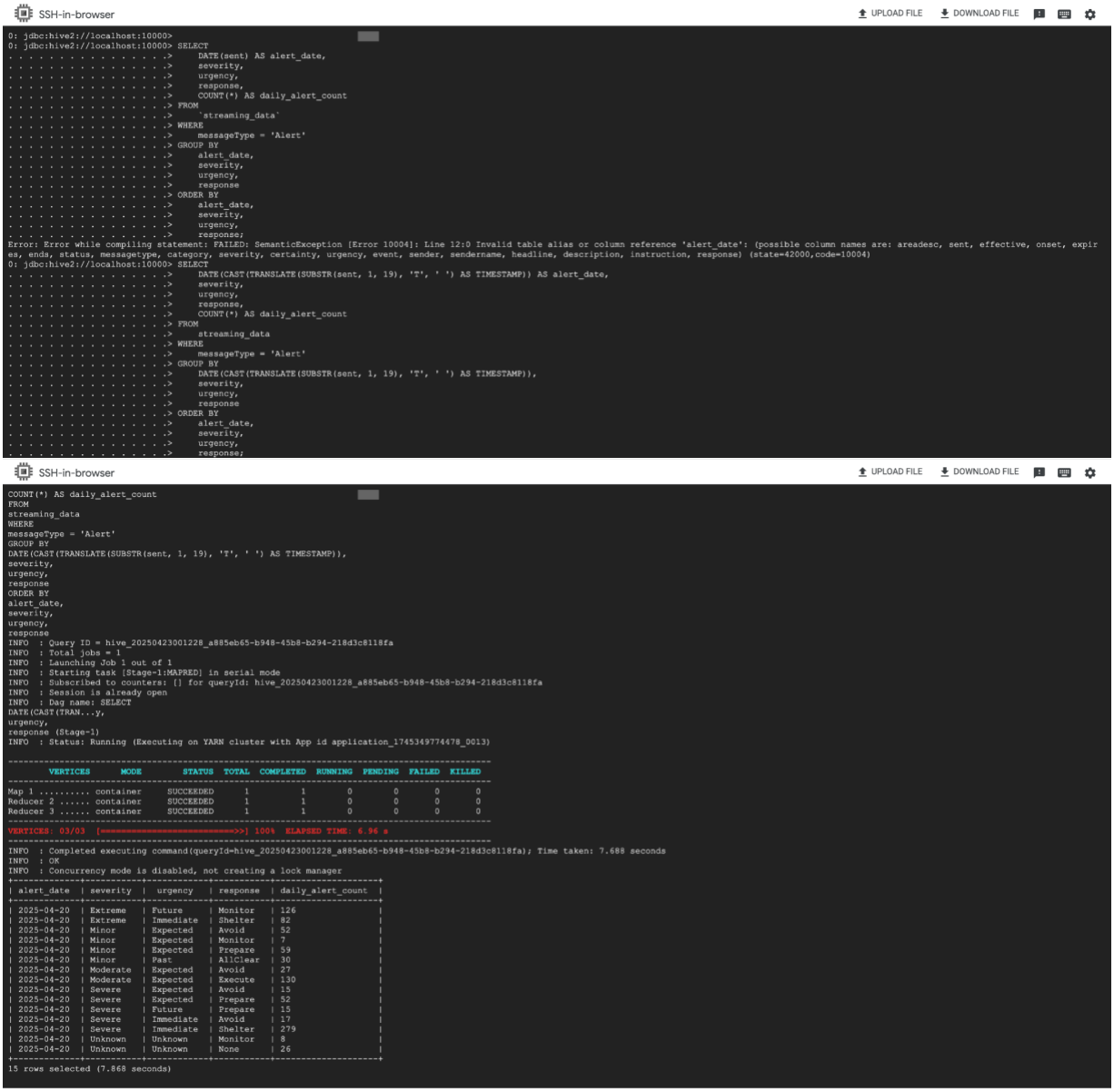
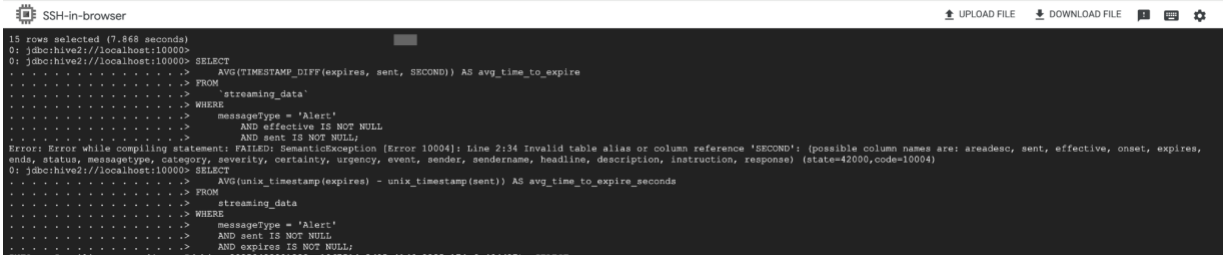


Figure 7.4. HiveQL Query: Grouping of the data by date, severity, urgency, and response type of cyclone alerts for day-by-day study of how these characteristics change with the increase or decrease in danger levels. Screenshot taken by Shivangi Borad.



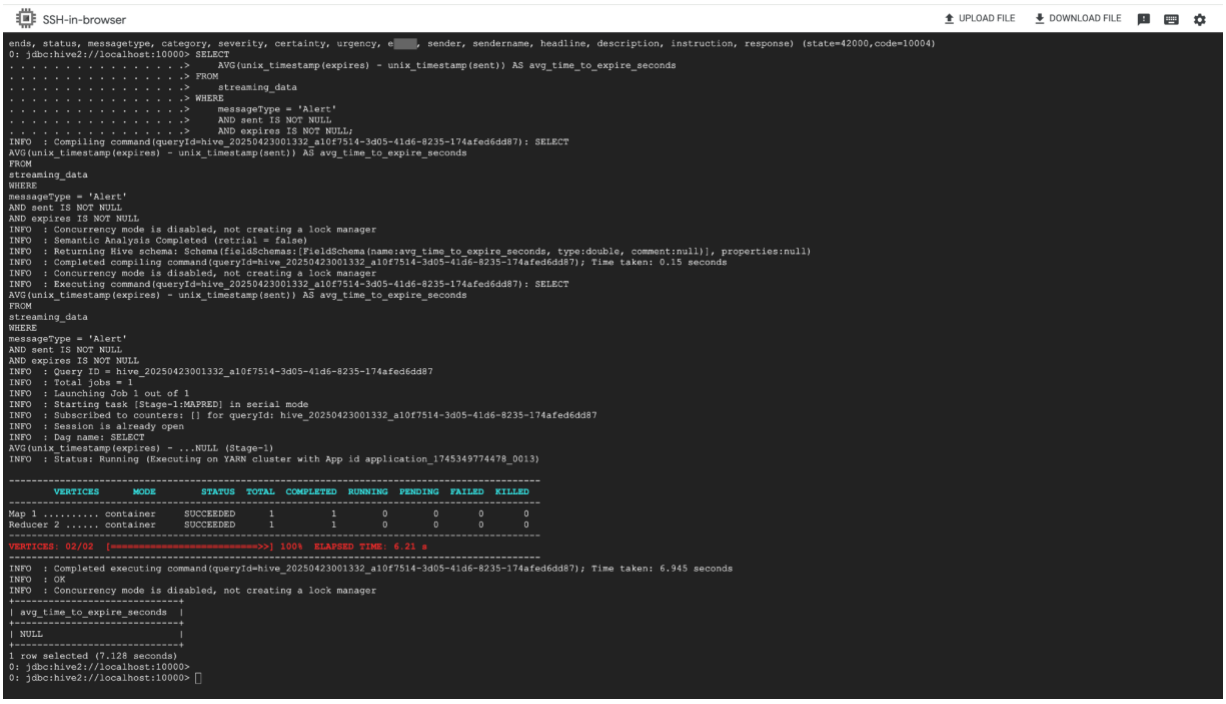


Figure 7.5. HiveQL query computes the average time elapsed between the sending of an alert and its expiration. This query aims to assess the entire timing and quality of the data but it has not returned any result. Screenshot taken by Shivangi Borad.

We used Apache Hive to run queries on cyclone alert information that is stored in our cluster. Using Hive, we were able to sort and examine the data with relatively simple SQL-like queries. Our queries included timestamp conversion, grouping, aggregations, and filtering, and enabled us to glean rich information on alert frequency, severity, urgency, and wording. This is helpful for us in transforming alert data into meaningful summaries for better situational awareness and disaster planning.

Apache Spark: Streaming Data

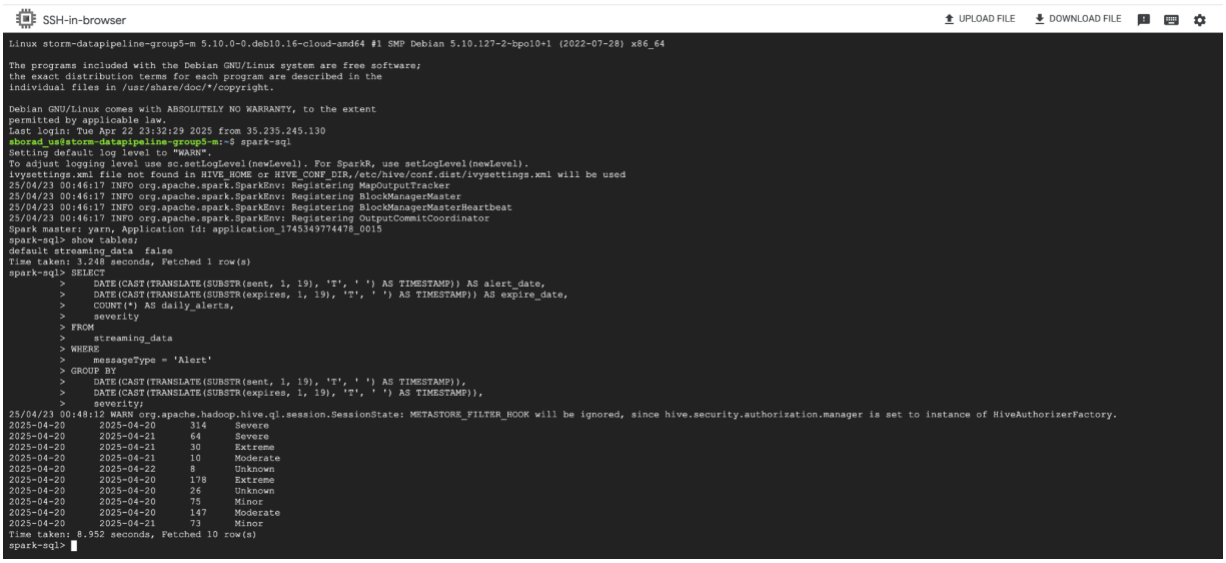


Figure 8.1. The Spark SQL query is grouped by day and performs aggregations on cyclone alerts according to their level of severity so as to allow for distributed computation for fast analysis on the alert trends as they occur. Screenshot taken by Shivangi Borad.

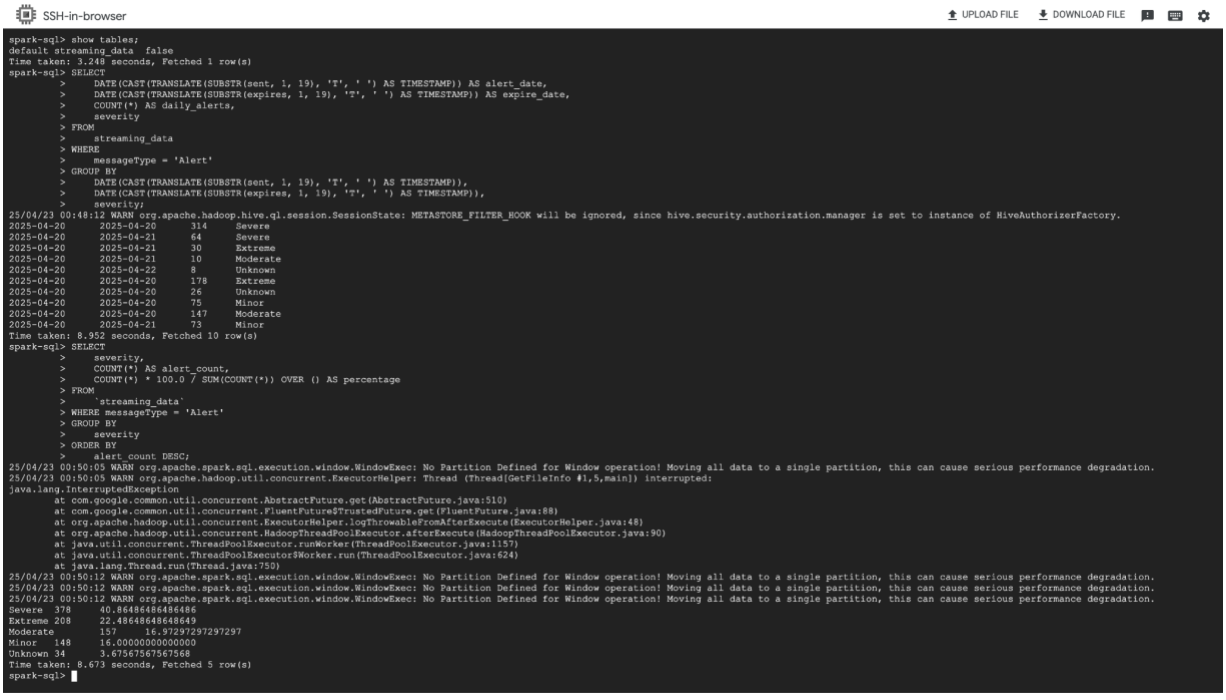


Figure 8.2. A Spark SQL query counts the alert notifications and calculates their percentages based on severity levels via parallel processing to help indicate the highly represented cyclone alert categories. Screenshot taken by Shivangi Borad.

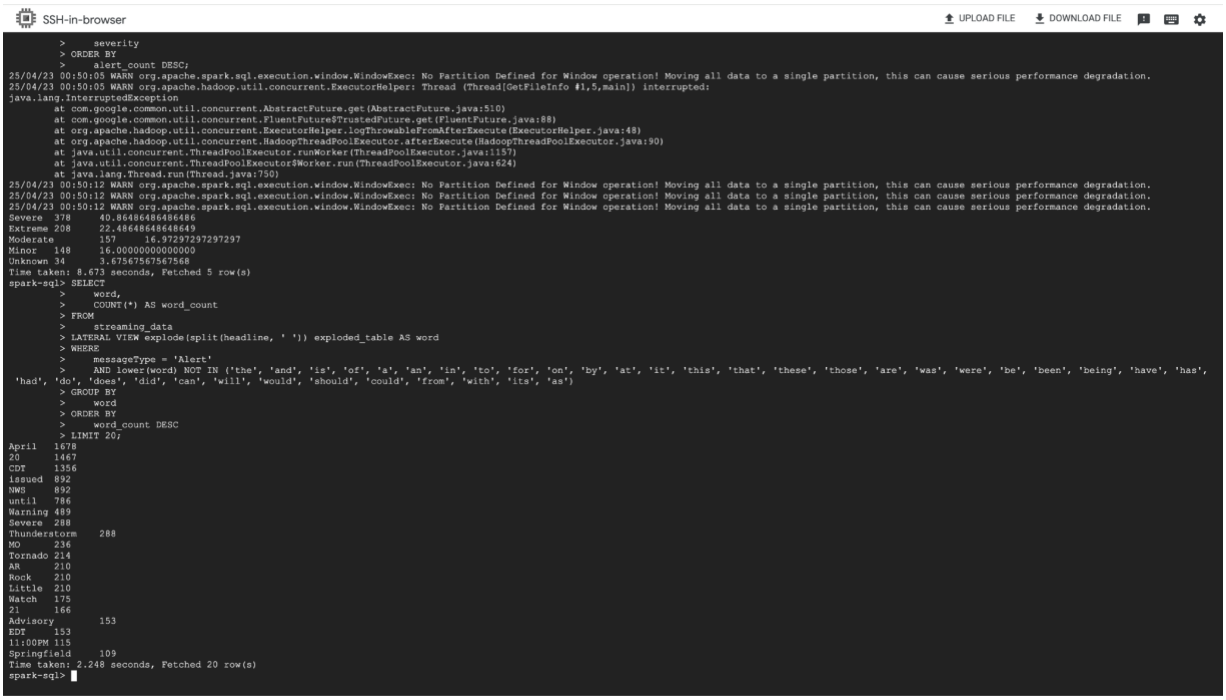


Figure 8.3. The Spark SQL query transforms the alert headlines, removes stopwords, and finds the most common important words, aiding the fast and scalable extraction of keywords from stream data. Screenshot taken by Shivangi Borad.

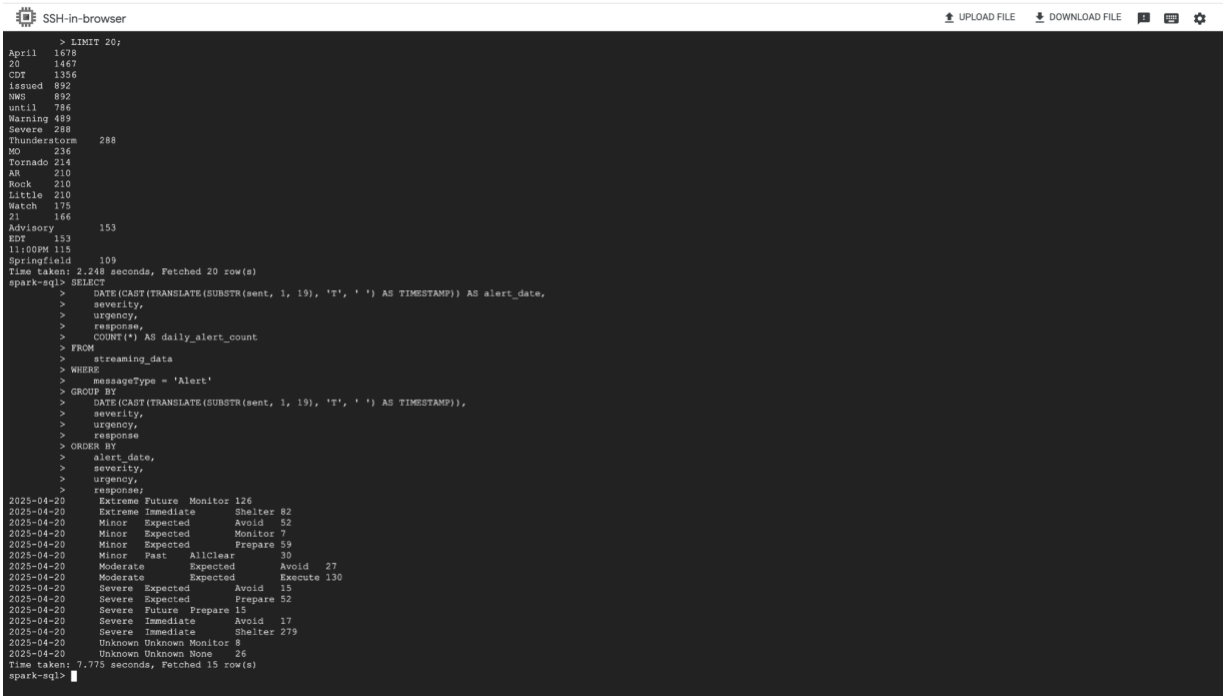


Figure 8.4. A Spark SQL query groups the alert data by several fields such as date, severity, urgency, and response type, employing Spark's distributed platform to detect the daily changes in alerting strategies. Screenshot taken by Shivangi Borad.

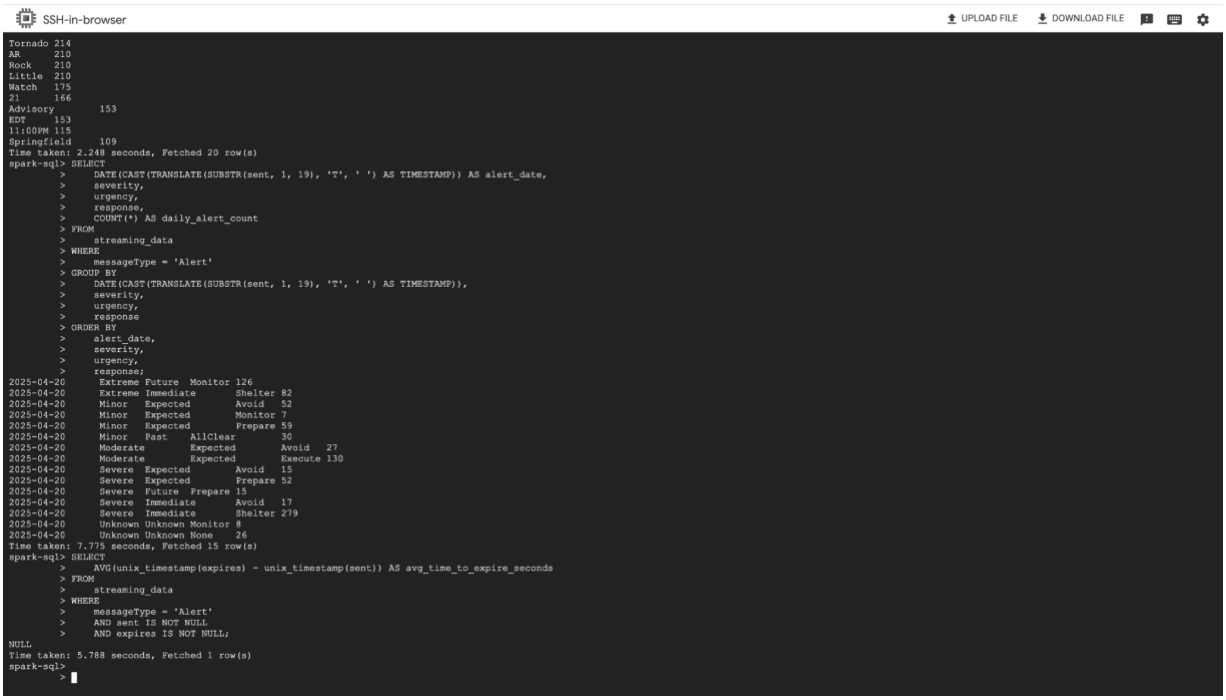


Figure 8.5. A Spark SQL query determines the average time lag between the alert being issued and its expiration with an effort to analyze the performance and timing of their warning system through the advantage of speed provided by Spark. Screenshot taken by Shivangi Borad.

We used the Apache Spark SQL to efficiently handle alert data and effectively process massive amounts of data. Using Spark SQL, we performed operations like timestamps and group aggregations on a large scale. This aspect improved our data processing, enabling us to compute repetitive word counts from alert information and determine message lengths. With Spark SQL, we had a seamless workflow despite high levels of data.

Apache Hive: Static Data

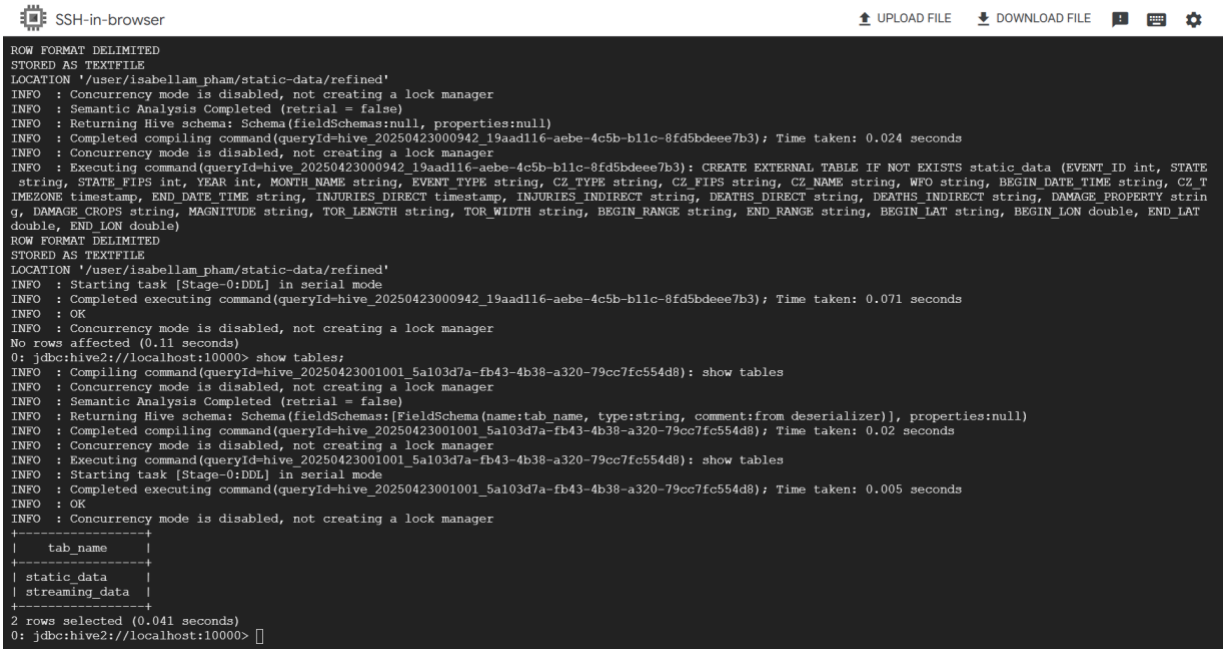


Figure 9.1. Creation of an Apache Hive table for static data. Screenshot taken by Isabella Pham.

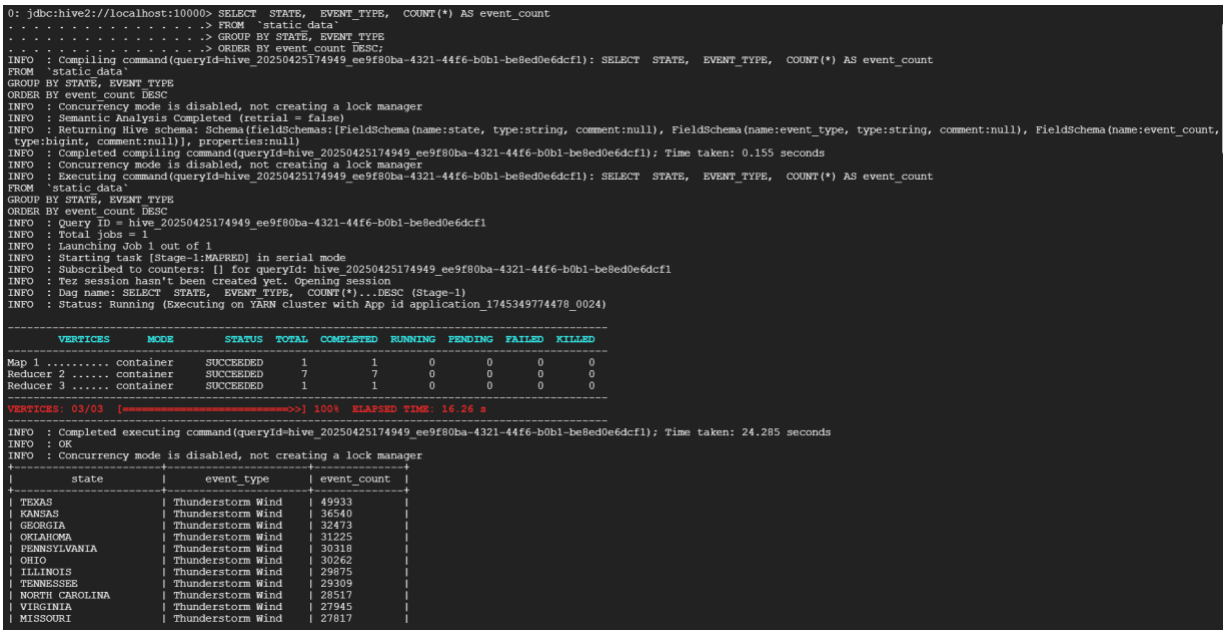


Figure 9.2. HiveQL query requesting data on the number of storm events per storm event type, grouped by storm type and state and ordered by the number of events in descending order. Screenshot taken by Isabella Pham. (Not Pictured: total elapsed time 24.554 seconds)

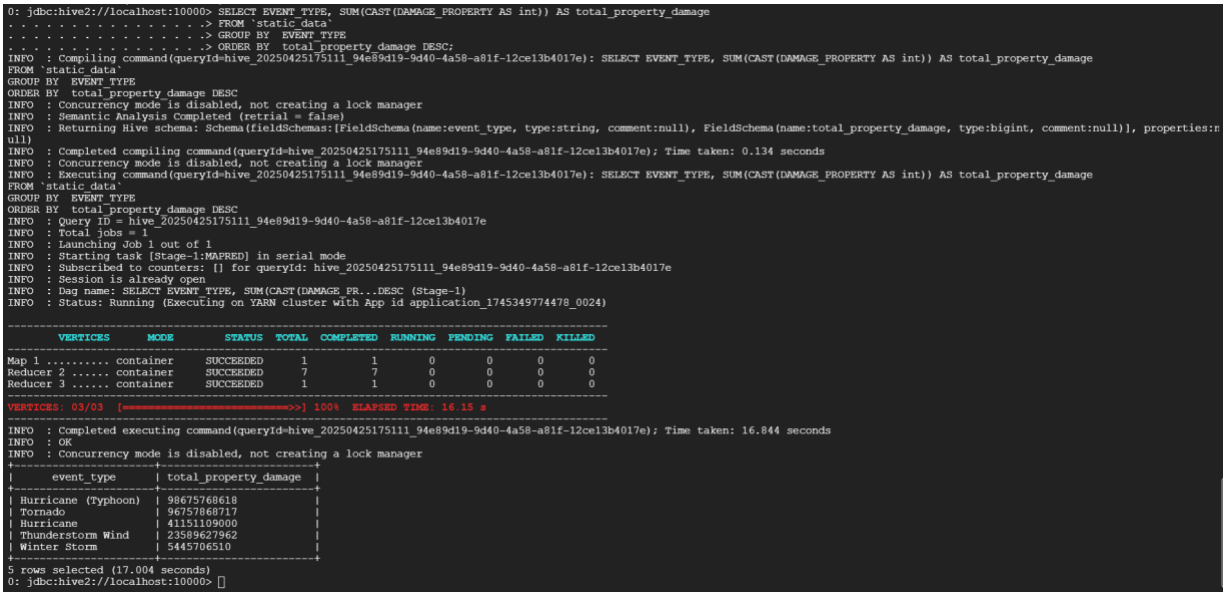


Figure 9.3. HiveQL query summarizing the total cost of (estimated) property damage in USD, grouped by storm event type and organized in descending order by damage amount. Screenshot taken by Isabella Pham.

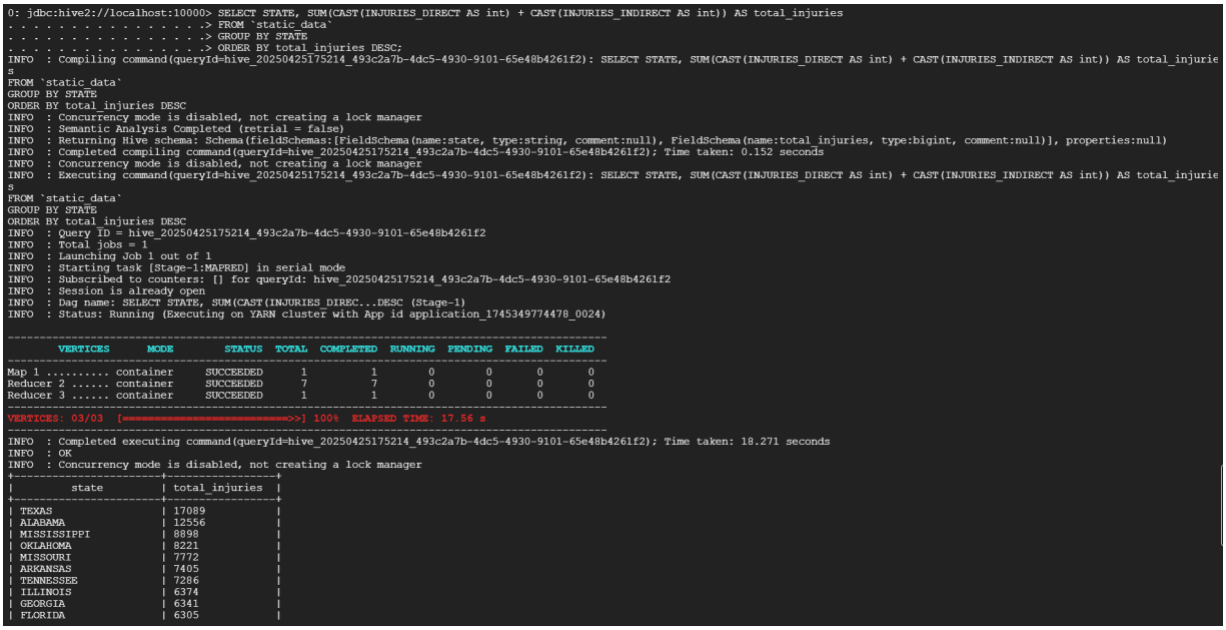


Figure 9.4. HiveQL query summarizing the total number of injuries (both direct and indirect) caused by severe storms; grouped by state and organized in descending order by number of injuries. Screenshot taken by Isabella Pham. (Not Pictured: total elapsed time 18.463 seconds)

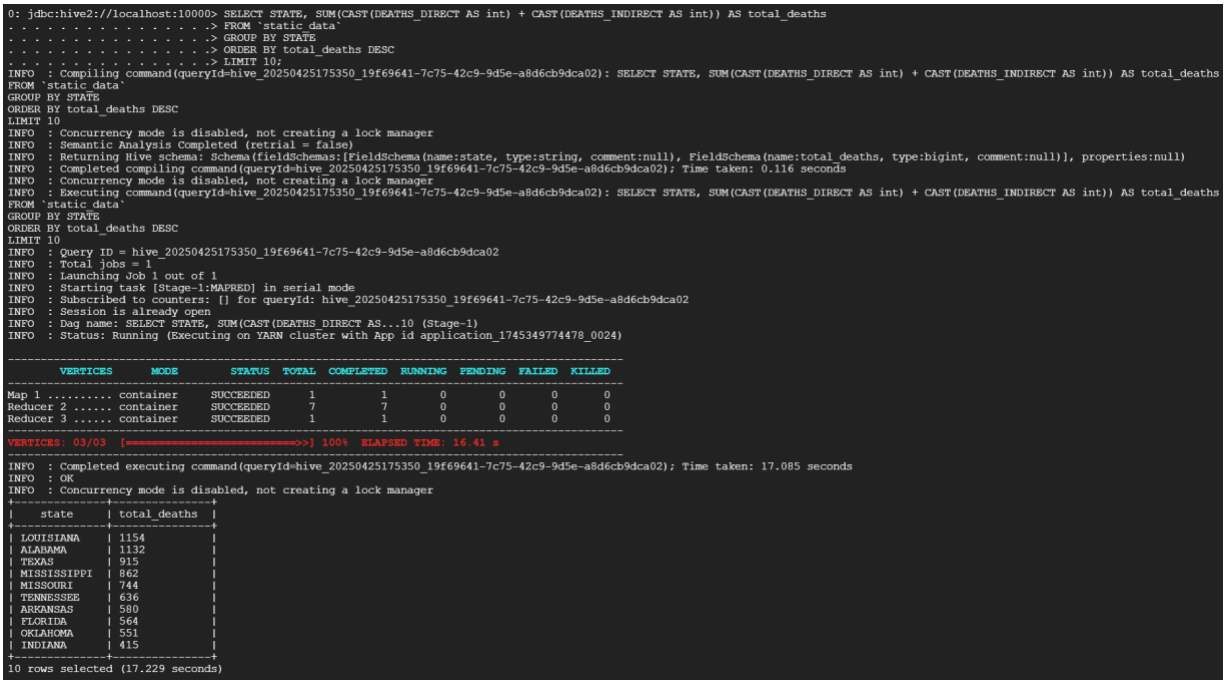


Figure 9.5. HiveQL query requesting a list of the 10 U.S. states with the highest death tolls caused by severe storms; organized in descending order by the number of deaths. Screenshot taken by Isabella Pham.

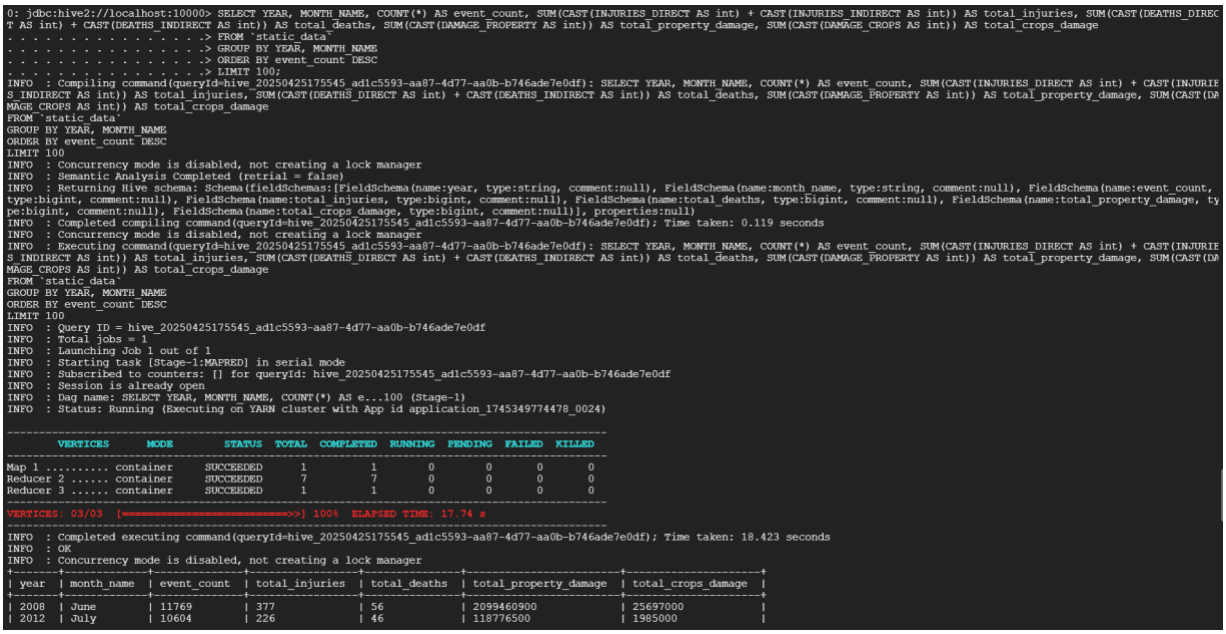


Figure 9.6. HiveQL query requesting a list of the top 100 dates (by year and month) in which the most storm events occurred, along with associated data relating to the total number of injuries, deaths, the (estimated) value property damage, and the (estimated) value of crop damage. All crop and property damages are measured in USD. Screenshot taken by Isabella Pham. (Not Pictured: total elapsed time 18.571 seconds)

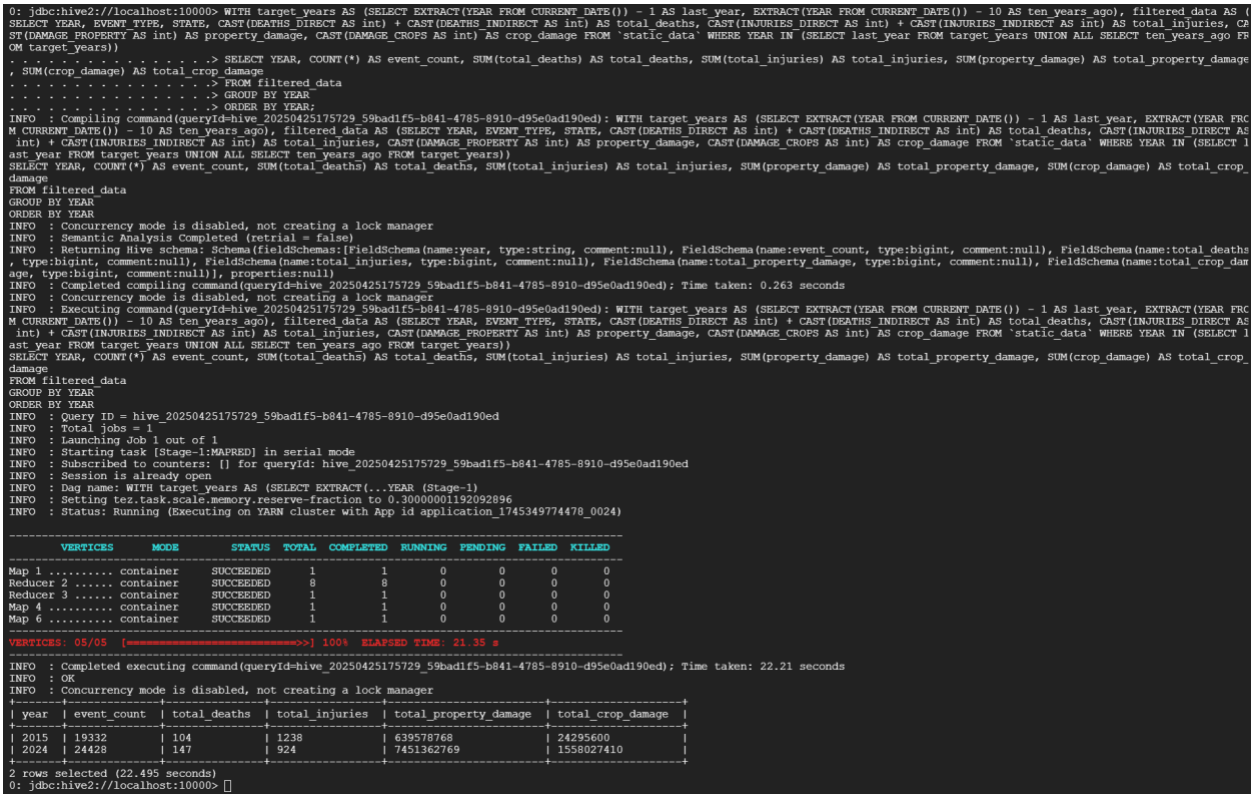
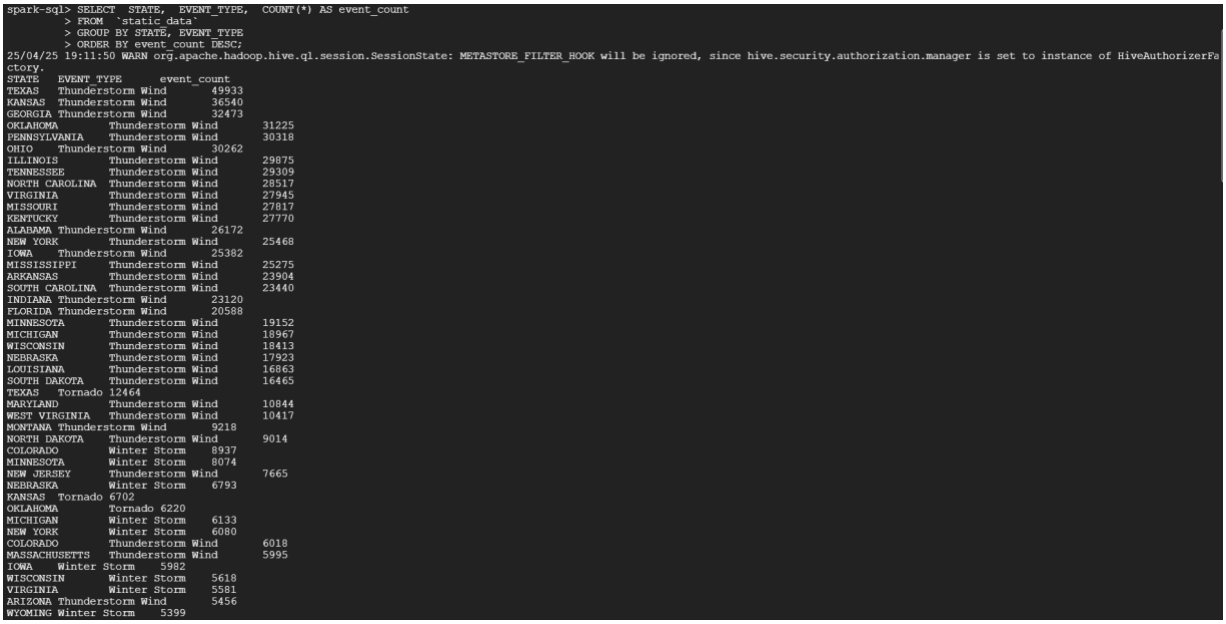


Figure 9.7. HiveQL query comparing summary data between two years roughly a decade apart, 2024 and 2015, starting with data from 2015. All (estimated) values of property and crop damage are measured in USD. Screenshot taken by Isabella Pham.

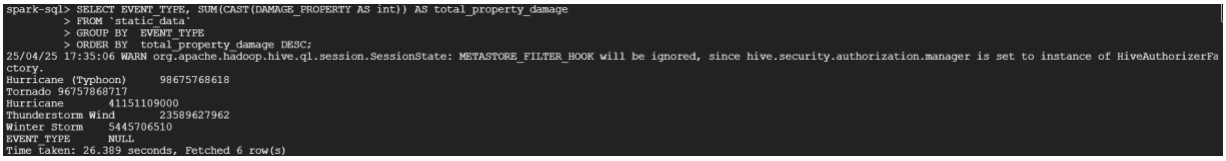
HiveQL was used to effectively extract, transform, and summarize large quantities of storm event structured data, with a specific focus on data measuring impact zones and the aftereffects of dangerous U.S.-based storms (e.g. fatalities, injuries, property damage, etc.). For the purposes of optimizing legibility and organization, explicit data types and column mappings - which aligned the Hive table with the structure of the data source - were defined using HiveQL. Beyond this, we are able to tailor queries to the unique needs of the Red Cross through the application of both HiveQL and Apache Spark, which will prove especially useful for future visualization and data analysis efforts.

Apache Spark: Static Data



```
spark-sql> SELECT  STATE,  EVENT_TYPE,  COUNT(*) AS event_count
         > FROM  `static_data`
         > GROUP BY STATE, EVENT_TYPE
         > ORDER BY event_count DESC;
25/04/25 19:11:50 WARN org.apache.hadoop.hive.ql.session.SessionState: METASTORE_FILTER_HOOK will be ignored, since hive.security.authorization.manager is set to instance of HiveAuthorizerFa
ctory.
STATE   EVENT_TYPE      event_count
TEXAS   Thunderstorm Wind       49933
KANSAS  Thunderstorm Wind       36540
GEORGIA Thunderstorm Wind       32473
OKLAHOMA        Thunderstorm Wind       31225
PENNSYLVANIA    Thunderstorm Wind       30318
OHIO    Thunderstorm Wind       30262
ILLINOIS        Thunderstorm Wind       29875
TENNESSEE       Thunderstorm Wind       29309
NORTH CAROLINA  Thunderstorm Wind       28517
VIRGINIA        Thunderstorm Wind       27945
MISSOURI        Thunderstorm Wind       27817
KENTUCKY        Thunderstorm Wind       27770
ALABAMA Thunderstorm Wind       26172
NEW YORK        Thunderstorm Wind       25468
IOWA    Thunderstorm Wind       25382
MISSISSIPPI     Thunderstorm Wind       25275
ARKANSAS        Thunderstorm Wind       23904
SOUTH CAROLINA  Thunderstorm Wind       23440
INDIANA Thunderstorm Wind       23120
FLORIDA Thunderstorm Wind       20588
MINNESOTA       Thunderstorm Wind       19152
MICHIGAN        Thunderstorm Wind       18967
WISCONSIN       Thunderstorm Wind       18413
NEBRASKA        Thunderstorm Wind       17923
LOUISIANA       Thunderstorm Wind       16863
SOUTH DAKOTA    Thunderstorm Wind       16465
TEXAS   Tornado 12464
MARYLAND        Thunderstorm Wind       10844
WEST VIRGINIA   Thunderstorm Wind       10417
MONTANA Thunderstorm Wind       9218
NORTH DAKOTA    Thunderstorm Wind       9014
COLORADO        Winter Storm    8937
MINNESOTA       Winter Storm    8074
NEW JERSEY      Thunderstorm Wind       7665
NEBRASKA        Winter Storm    6793
KANSAS  Tornado 6702
OKLAHOMA        Tornado 6220
MICHIGAN        Winter Storm    6133
NEW YORK        Winter Storm    6080
COLORADO        Thunderstorm Wind       6018
MASSACHUSETTS   Thunderstorm Wind       5995
IOWA    Winter Storm    5982
WISCONSIN       Winter Storm    5618
VIRGINIA        Winter Storm    5581
ARIZONA Thunderstorm Wind       5456
WYOMING Winter Storm    5399
```

Figure 10.1. Spark SQL query extracting data regarding the total number of events per storm type, grouped by state and organized in descending order. Screenshot taken by Isabella Pham. (Not Pictured: total elapsed time 23.685 seconds)



```
spark-sql> SELECT EVENT_TYPE, SUM(CAST(DAMAGE_PROPERTY AS int)) AS total_property_damage
         > FROM `static_data`
         > GROUP BY  EVENT_TYPE
         > ORDER BY  total_property_damage DESC;
25/04/25 17:35:06 WARN org.apache.hadoop.hive.ql.session.SessionState: METASTORE_FILTER_HOOK will be ignored, since hive.security.authorization.manager is set to instance of HiveAuthorizerFa
ctory.
Hurricane (Typhoon)     98675768618
Tornado 96757868717
Hurricane       41151109000
Thunderstorm Wind       23589627962
Winter Storm    5445706510
EVENT_TYPE      NULL
Time taken: 26.389 seconds, Fetched 6 row(s)
```

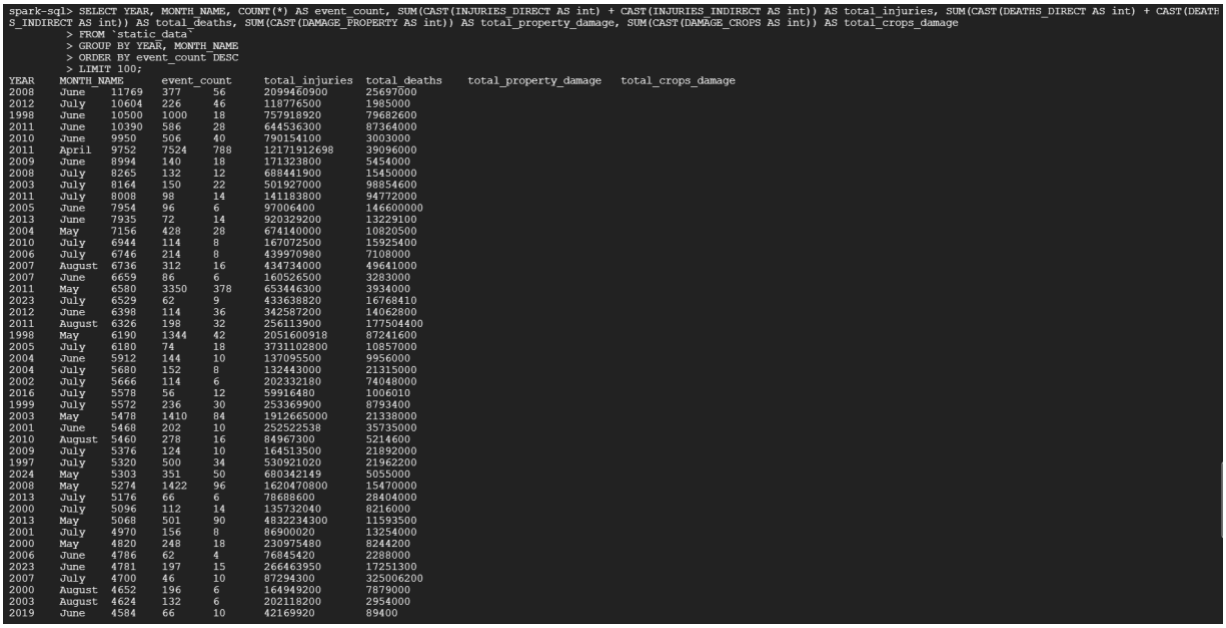
Figure 10.2. Spark SQL query summarizing the total value of (estimated) property damage in USD per storm event type, organized in descending order according to damage value. Screenshot taken by Isabella Pham.

```
spark-sql> SELECT STATE, SUM(CAST(INJURIES_DIRECT AS int) + CAST(INJURIES_INDIRECT AS int)) AS total_injuries
         > FROM `static_data`
         > GROUP BY STATE
         > ORDER BY total_injuries DESC;
STATE   total_injuries
TEXAS   17089
ALABAMA 12556
MISSISSIPPI     8898
OKLAHOMA        8221
MISSOURI        7772
ARKANSAS        7405
TENNESSEE       7286
ILLINOIS        6374
GEORGIA 6341
FLORIDA 6305
OHIO    5686
INDIANA 5542
KENTUCKY        5065
NORTH CAROLINA  4580
MICHIGAN        4274
KANSAS  4108
LOUISIANA       3887
IOWA    3578
PENNSYLVANIA    2580
MINNESOTA       2549
WISCONSIN       2482
SOUTH CAROLINA  2135
MASSACHUSETTS   1998
VIRGINIA        1784
NEBRASKA        1591
UTAH    1269
NEW YORK        1101
GUAM    970
SOUTH DAKOTA    950
CONNECTICUT     829
MARYLAND        775
NEW JERSEY      673
COLORADO        655
NORTH DAKOTA    551
ARIZONA 536
WYOMING 513
WASHINGTON      375
WEST VIRGINIA   364
CALIFORNIA      339
```

Figure 10.3. Spark SQL query summarizing the total number of injuries (both direct and indirect) caused by severe storms; organized by state and in descending order according to the number of injuries per state. Screenshot taken by Isabella Pham. (Not Pictured: total elapsed time 18.170 seconds)

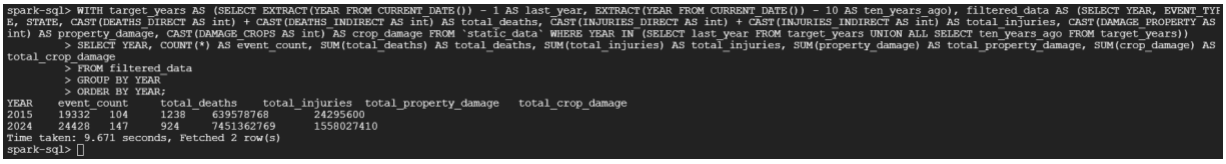
```
spark-sql> SELECT STATE, SUM(CAST(DEATHS_DIRECT AS int) + CAST(DEATHS_INDIRECT AS int)) AS total_deaths
         > FROM `static_data`
         > GROUP BY STATE
         > ORDER BY total_deaths DESC
         > LIMIT 10;
25/04/25 19:18:44 WARN org.apache.hadoop.util.concurrent.ExecutorHelper: Thread (Thread[GetFileInfo #1,5,main]) interrupted:
java.lang.InterruptedException
        at com.google.common.util.concurrent.AbstractFuture.get(AbstractFuture.java:510)
        at com.google.common.util.concurrent.FluentFuture$TrustedFuture.get(FluentFuture.java:88)
        at org.apache.hadoop.util.concurrent.ExecutorHelper.logThrowableFromAfterExecute(ExecutorHelper.java:48)
        at org.apache.hadoop.util.concurrent.HadoopThreadPoolExecutor.afterExecute(HadoopThreadPoolExecutor.java:90)
        at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1157)
        at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
        at java.lang.Thread.run(Thread.java:750)
STATE   total_deaths
LOUISIANA       1154
ALABAMA 1132
TEXAS   915
MISSISSIPPI     862
MISSOURI        744
TENNESSEE       636
ARKANSAS        580
FLORIDA 564
OKLAHOMA        551
INDIANA 415
Time taken: 8.502 seconds, Fetched 10 row(s)
spark-sql>
```

Figure 10.4. Spark SQL query requesting a list of the 10 U.S. states with the highest number of deaths caused by severe storms; organized in descending order according to the number of deaths per state. Screenshot taken by Isabella Pham.



```
spark-sql> SELECT YEAR, MONTH_NAME, COUNT(*) AS event_count, SUM(CAST(INJURIES_DIRECT AS int) + CAST(INJURIES_INDIRECT AS int)) AS total_injuries, SUM(CAST(DEATHS_DIRECT AS int) + CAST(DEATH
S_INDIRECT AS int)) AS total_deaths, SUM(CAST(DAMAGE_PROPERTY AS int)) AS total_property_damage, SUM(CAST(DAMAGE_CROPS AS int)) AS total_crops_damage
         > FROM `static_data`
         > GROUP BY YEAR, MONTH_NAME
         > ORDER BY event_count DESC
         > LIMIT 100;
YEAR    MONTH_NAME      event_count     total_injuries  total_deaths    total_property_damage   total_crops_damage
2008    June    11769   377     56      2099460900      25697000
2012    July    10604   226     46      118776500       1985000
1998    June    10500   1000    18      757918920       79682600
2011    June    10390   586     28      644536300       87364000
2010    June    9950    506     40      790154100       3003000
2011    April   9752    7524    788     12171912698     39096000
2009    June    8994    140     18      171323800       5454000
2008    July    8265    132     12      688441900       15450000
2003    July    8164    150     22      501927000       98854600
2011    July    8008    98      14      141183800       94772000
2005    June    7954    96      6       97006400        146600000
2013    June    7935    72      14      920329200       13229100
2004    May     7156    428     28      674140000       10820500
2010    July    6944    114     8       167072500       15925400
2006    July    6746    214     8       439970980       7108000
2007    August  6736    312     16      434734000       49641000
2007    June    6659    86      6       160526500       3283000
2011    May     6580    3350    378     653446300       3934000
2023    July    6529    62      9       433638820       16768410
2012    June    6398    114     36      342587200       14062800
2011    August  6326    198     32      256113900       177504400
1998    May     6190    1344    42      2051600918      87241600
2005    July    6180    74      18      3731102800      10857000
2004    June    5912    144     10      137095500       9956000
2004    July    5680    152     8       132443000       21315000
2002    July    5666    114     6       202332180       74048000
2016    July    5578    56      12      59916480        1006010
1999    July    5572    236     30      253369900       8793400
2003    May     5478    1410    84      1912665000      21338000
2001    June    5460    202     10      252522538       35735000
2010    August  5460    278     16      84967300        5214600
2009    July    5376    124     10      164513500       21892000
1997    July    5320    500     34      530921020       21962200
2024    May     5303    351     50      680342149       5055000
2008    May     5274    1422    96      1620470800      15470000
2013    July    5176    66      6       78688600        28404000
2000    July    5096    112     14      135732040       8216000
2013    May     5068    501     90      4832234300      11593500
2001    July    4970    156     8       86900020        13254000
2000    May     4820    248     18      230975480       8244200
2006    June    4786    62      4       76845420        2288000
2023    June    4781    197     15      266463950       17251300
2007    July    4700    46      10      87294300        325006200
2000    August  4652    196     6       164949200       7879000
2003    August  4624    132     6       202118200       2954000
2019    June    4584    66      10      42169920        89400
```

Figure 10.5. Spark SQL query requesting a list of 100 dates (by year and month) in which the most storm events occurred, including associated data relating to the number of injuries, deaths, (estimated) value of property damages, and (estimated) value of crop damages. All crop and property damages are measured in USD. Screenshot taken by Isabella Pham. (Not Pictured: total elapsed time 9.264 seconds)



```
spark-sql> WITH target_years AS (SELECT EXTRACT(YEAR FROM CURRENT_DATE()) - 1 AS last_year, EXTRACT(YEAR FROM CURRENT_DATE()) - 10 AS ten_years_ago), filtered_data AS (SELECT YEAR, EVENT_TYP
E, STATE, CAST(DEATHS_DIRECT AS int) + CAST(DEATHS_INDIRECT AS int) AS total_deaths, CAST(INJURIES_DIRECT AS int) + CAST(INJURIES_INDIRECT AS int) AS total_injuries, CAST(DAMAGE_PROPERTY AS
int) AS property_damage, CAST(DAMAGE_CROPS AS int) AS crop_damage FROM `static_data` WHERE YEAR IN (SELECT last_year FROM target_years UNION ALL SELECT ten_years_ago FROM target_years))
         > SELECT YEAR, COUNT(*) AS event_count, SUM(total_deaths) AS total_deaths, SUM(total_injuries) AS total_injuries, SUM(property_damage) AS total_property_damage, SUM(crop_damage) AS
total_crop_damage
         > FROM filtered_data
         > GROUP BY YEAR
         > ORDER BY YEAR;
YEAR    event_count     total_deaths    total_injuries  total_property_damage   total_crop_damage
2015    19332   104     1238    639578768       24295600
2024    24428   147     924     7451362769      1558027410
Time taken: 9.671 seconds, Fetched 2 row(s)
spark-sql>
```

Figure 10.6. Spark SQL query comparing summary data from two years roughly a decade apart (2024 and 2015), starting with data from 2015. All (estimated) values of property and crop damage are measured in USD. Screenshot taken by Isabella Pham.

Similar to our prior applications of HiveQL, Spark SQL was used to parse, transform, and summarize large quantities of structured data relating to severe storm events documented within the United States in a highly efficient manner. Specific focus was placed on data measuring impact zones and the lingering, measurable effects/damages caused by severe storms, such as fatalities, injuries, estimated property damage, and estimated crop damages. Queried data is returned in a simple format which can be easily be applied to future visualization and data analysis efforts, and has the potential to be used to train machine learning algorithms, which can be used to predict the potential after effects of future storms.

Hive & Spark Comparisons

Static Dataset

Query #	Time Elapsed	
	Apache Hive	**Apache Spark**
1	24.554 seconds	23.685 seconds
2	17.004 seconds	26.389 seconds
3	18.463 seconds	18.170 seconds
4	17.229 seconds	8.502 seconds
5	18.571 seconds	9.264 seconds
6	22.495 seconds	9.671 seconds
Average Total Time	***19.72 seconds***	***15.95 seconds***

Streaming Dataset

Query #	Time Elapsed	
	Apache Hive	**Apache Spark**
1	7.005 seconds	8.952 seconds
2	8.133 seconds	8.673 seconds
3	8.094 seconds	2.248 seconds
4	7.868 seconds	7.775 seconds
5	7.128 seconds	5.788 seconds
Average Total	***7.65 seconds***	***6.69 seconds***

Time		

Comparison Conclusion

Apache Spark was consistently faster than Apache Hive when evaluating both static and streaming datasets, with a total average processing time of 11.74 seconds for Spark and 14.23 seconds for Hive. This is likely because of the differences in core functionalities between the two Hadoop sub-modules: While Hive is a scalable data warehousing solution that stores data on a computer's data storage device (i.e. its disk) - similar to how data is stored in a relational database management system - and processes each query by reading from and writing to the storage device, Spark is a framework with no dedicated storage; instead, it loads data into memory (RAM), processes everything in-memory, and only writes the final results to the computer's storage. (Berman, 2019). Since Hive is dependent on slower disk-based operations, while Spark processes data directly into memory through Spark, Spark achieves faster processing speed as compared to Hive.

Overall, the static and streaming queries were constructed using similar aggregations and groupings. However, static data queries resulted in longer query times due to the sheer size of the static dataset. The smaller size of the streaming dataset - combined with the computing power of Spark - resulted in Spark surpassing Hive entirely in terms of processing times. Based on these findings, we have concluded that the architecture of Spark made it better suited for both real-time and batch processing in this project.

Machine Learning (BONUS)

Damage Forecast For the Year 2025

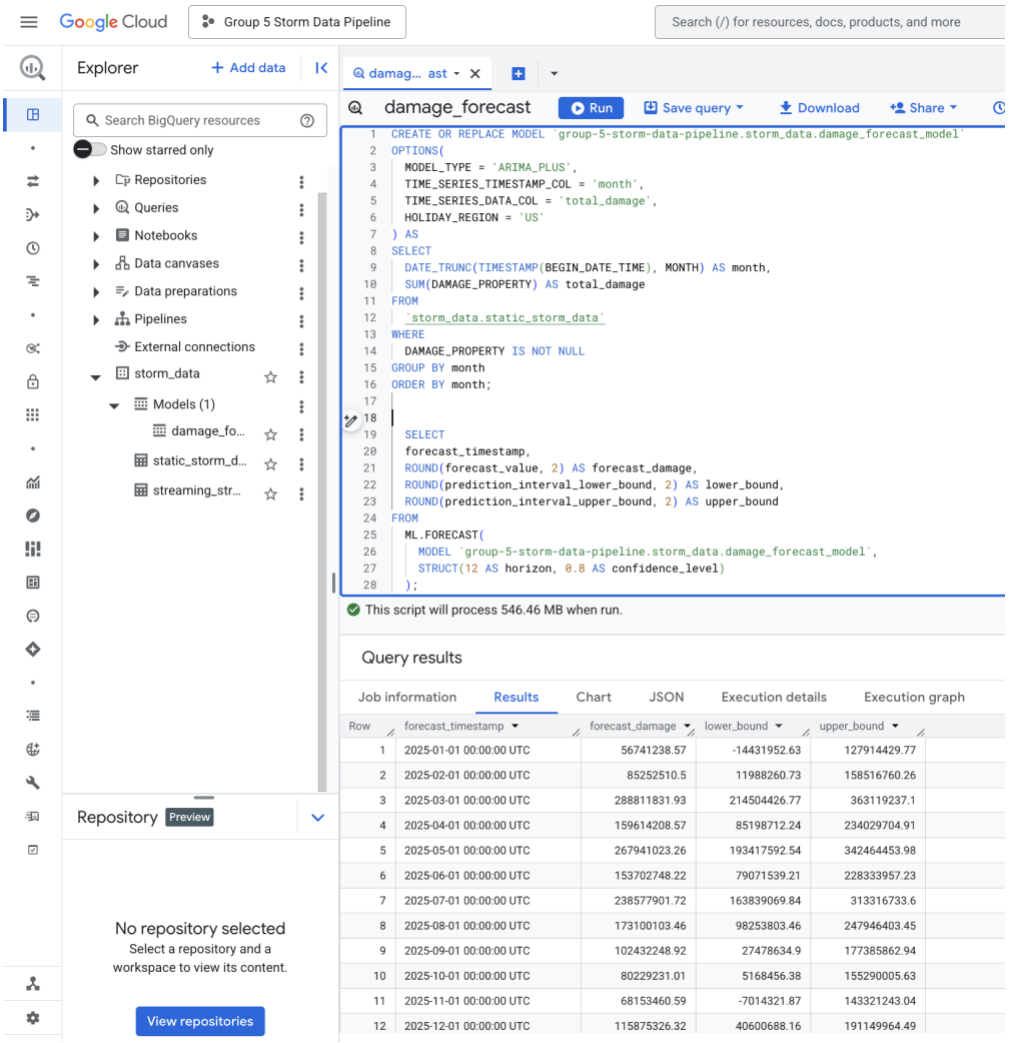


Figure 11. Screenshot taken by Cezane Karki.

We used Google Big Query to train the machine learning model to forecast damage that might occur in the year 2025 by seeing the damage pattern from the year 1950 to present. This tool helps to provide the future information about the damages and potential upper and lower bound of the damages which can be useful for planning and preparation for the potential storm impact. Similar to this, looking at the patterns of the storm and its occurrence time, potential forecasts can be done for the storm type, impact and size that might occur in the future.

Visualizations (BONUS)

Static Dataset

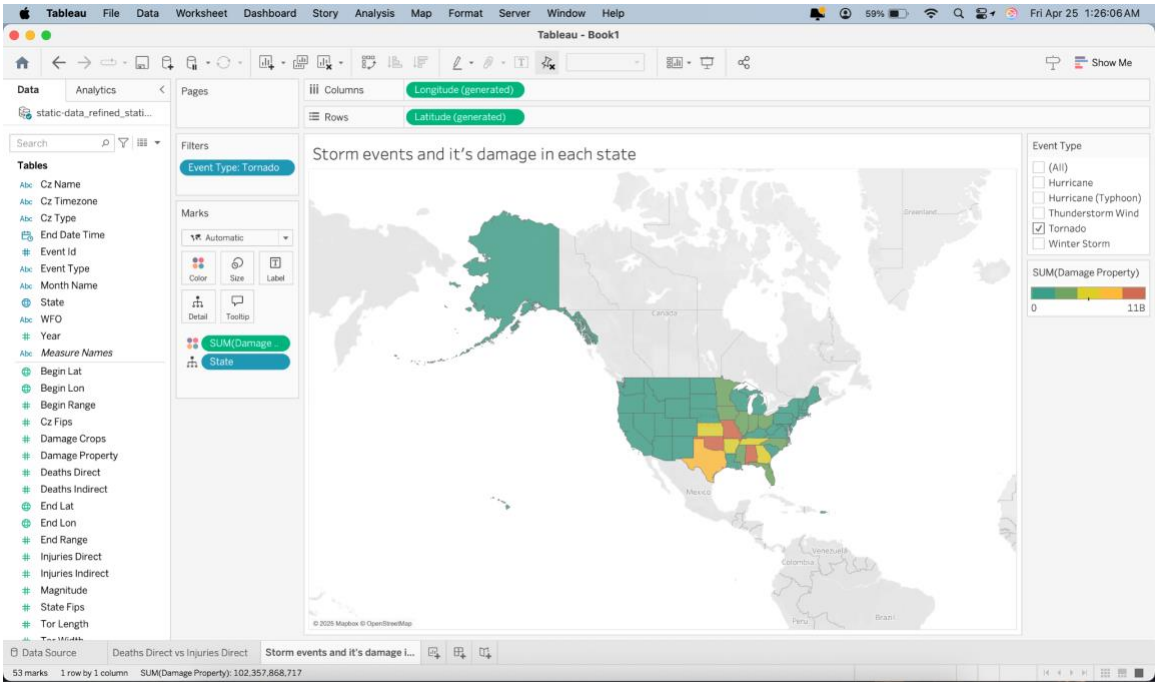


Figure 12.1. Screenshot taken by Cezane Karki.

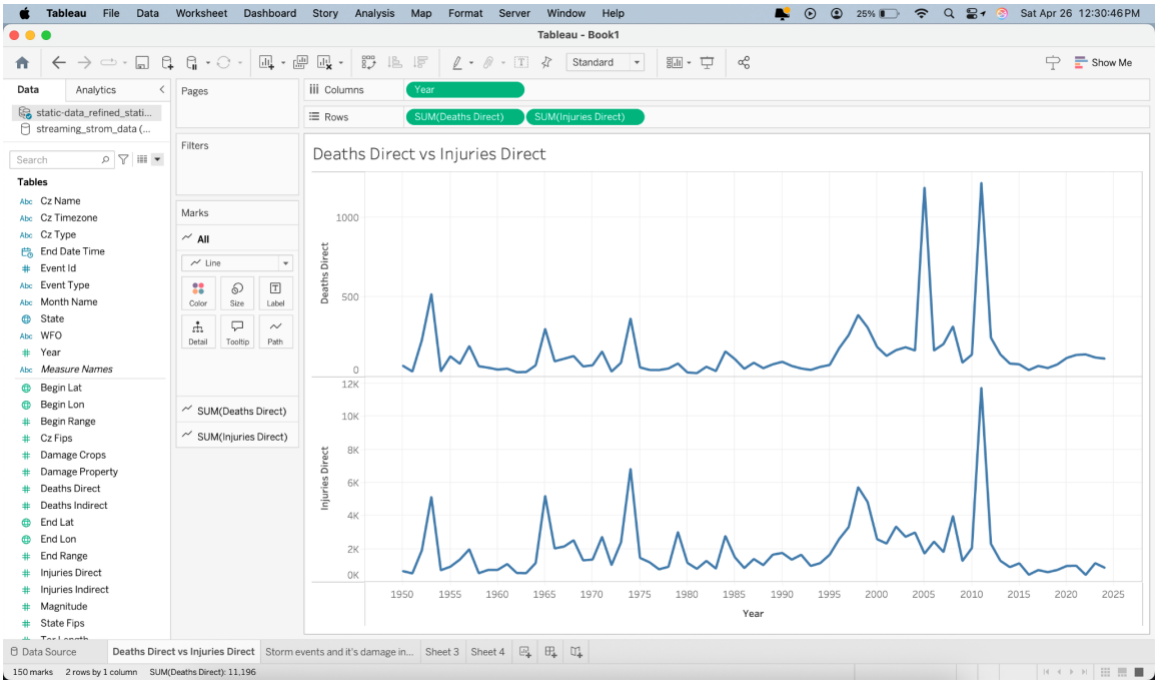


Figure 12.2. Screenshot taken by Cezane Karki.

Streaming Dataset

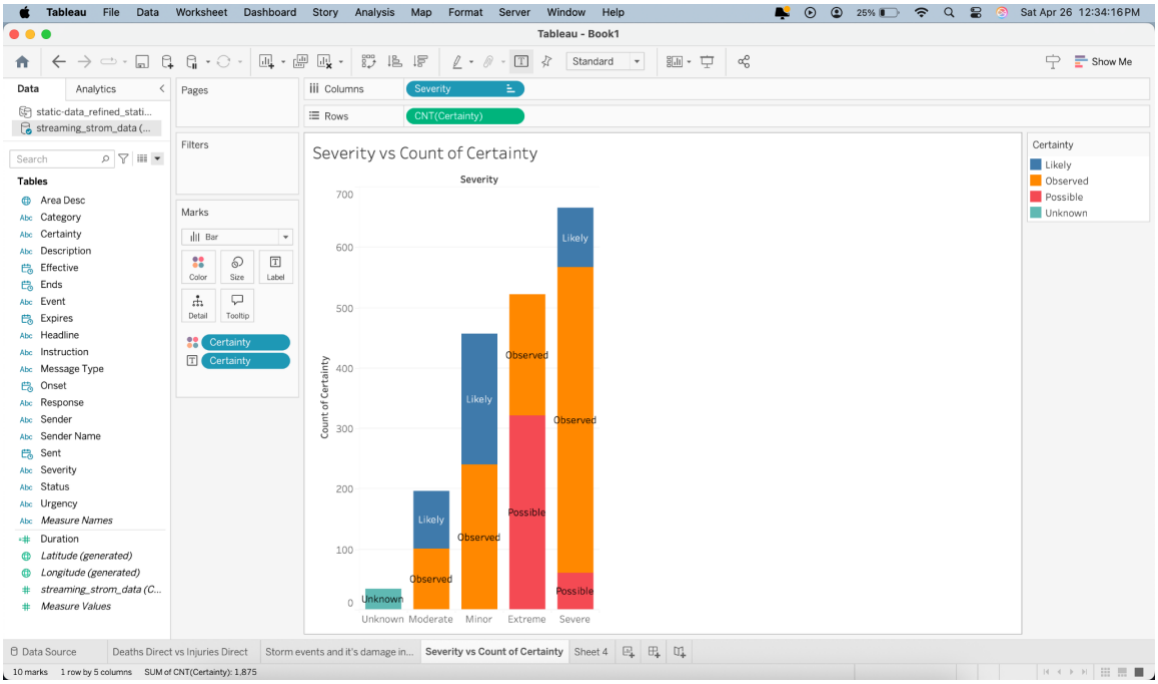


Figure 13.1. Screenshot taken by Cezane Karki.

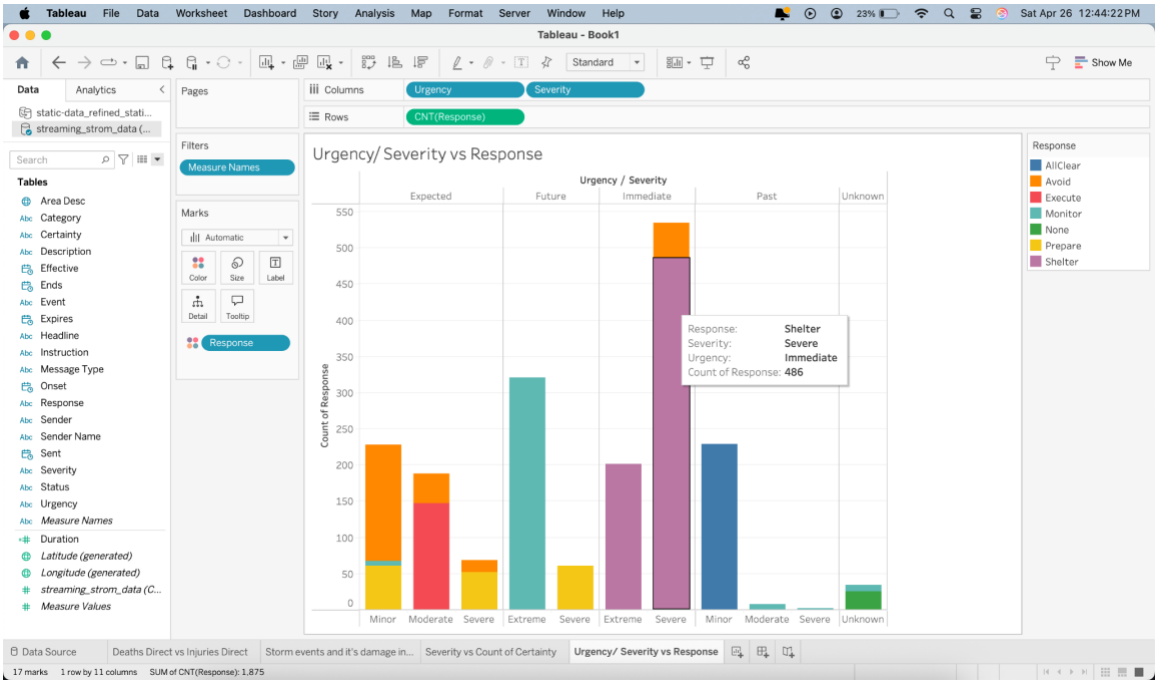


Figure 13.2. Screenshot taken by Cezane Karki.

Interpretation (BONUS)

Our interpretation of storm event data processed through Big Query, Apache Hive, and Apache Spark has thrown up several actionable recommendations essential to the disaster response efforts of the American Red Cross. The static data analysis (from 1950 to 2024) realistically identified states most frequently impacted by storms. Texas, Kansas, and Oklahoma have the highest recorded number of impacts, of storm events, deaths, and crop damages, and property damages. (Figure 6.5 & 12.1)Some storm types such as tornadoes and thunderstorms contributed significantly to injuries and deaths over the decades, highlighting the vulnerabilities in different regions.

The streaming data acquired on April 20, 2025 indicated a distinct difference in the urgency and severity labeling of storm alerts across different parts of the United States. For example, high-urgency alerts have been given out most frequently by the states of Florida, Alabama, and Louisiana for tornadoes, severe thunderstorms, and flash floods. Word frequency analysis pointed to predominance of certain emergency communication words such as "Warning," "Severe," "Tornado," and "Flash Flood," suggesting the many hazards needing immediate attention by the public. Stream data analysis emphasized the need for better standardization and clarity in messaging practices, such that affected populations interpret correctly the level of threat and response.

The machine learning functionality and visualizations further enhanced our understanding and analysis of the data and its interpretation. Predictive damage forecasts with BigQuery machine learning functionality suggested 2025 might again trend upwards in property damage from previous years; this would be consistent with general predictions about climate change. The visualizations built from static data and the streaming data have further communicated trends like the frequency of different storm events, the severity distribution, and daily alert patterns clearly. These machine learning and visualization tools have enabled us to better present complex relationships in the data and make insights into actions for the American Red Cross.

Overall, the interpretation of our findings will help the American Red Cross in identifying the vulnerable areas, improving the channels of emergency communication, and initiating future more predictive, data-driven efforts at disaster response.

Conclusion

Discussion

In this project we utilized the data life cycle approach to derive a complete picture of the nature of the storm phenomenon. Using a systematic approach that included the generation, collection, processing, storage, management, and analysis of data, it revealed meaningful patterns in storms and created a strong framework to support real-time decision-making for disaster relief by the American Red Cross.

The joining of static data from 1950 to 2024 with streaming data gathered on April 20, 2025, facilitated comparative analysis between historical perspectives and real-time data, further elevating the validity and applicability of the findings. The use of tools such as OpenRefine, Apache Spark, Hive, and Google BigQuery was essential in effectively cleaning the data, transforming it, and conducting in-depth analytical processing. The coordination between classical SQL-based analysis and modern big data processing software was key in handling the magnitude and frequency of the storm data (Hashem et al., 2015).

The BigQuery analysis also yielded major trends, such as the number and kind of storms, the level of urgency, and the changing patterns of response over time. By classifying alert data in severity and geographic location categories, the areas having most at-risk of severe weather events were clear. Further, the ability of Apache Hive and Spark to derive insights from vast data make it better suited for making faster decisions. Other similar projects have used the tools of NLP to extract insights from real-time data such as social media and official announcements in crisis (Imran et al., 2014).

One of the most evident omissions to emerge in the analysis was the lack of tropical cyclone data, something that could compromise the accuracy of the forecasts in areas hit hardest by such storms. Another challenge we faced while completing this project was the dataset. Most of the columns of the dataset required to be altered to be suitable for the tools that we used in the processing stage. Also, the size of the dataset created hurdles with OpenRefine, as it interfered with the performance of the tool and its response while working. Hence, we had to spend additional time on planning of data management before initiating the actual process.

To conclude, by merging historical and live storm data using high-performance tools like BigQuery, Apache Hive, and Apache Spark, this project offered an end-to-end analytical system

to better prepare and respond for the American Red Cross. The enhanced visuals, derived from static and streaming data, compellingly illustrated significant patterns, like storm type occurrences, intensity trends, and emergency message development, that would remain elusive from raw data. These graphical tools summarized complex datasets to insightful information, proving that visuals are an important element to enhance public communications strategies and detection of at-risk areas. Furthermore, the Spark versus Hive comparison unearthed Spark's competitiveness regarding processing time, crucial for up-to-the-minute analysis for alerts. Having said that, Hive would be a better choice from an economic viewpoint as the processing time difference between the two services was minor compared to the size of the dataset. By making big data for storm events accessible and understandable, this project makes it easier for future data scientists and analysts to develop more predictive, concise, and user-friendly disaster management for the Red Cross organization.

Recommendations

1. Forecasting Storm Effects Using Predictive Modeling

Based on an in-depth analysis of historical storm data from 1950 to 2024, as well as real-time data from April 2025, the American Red Cross should establish a partnership with data scientists to develop machine learning models for the prediction of the types of storms, their severity, and response needs. The patterns observed in storm frequencies, seasonal patterns, and severity distributions suggest potential for predictive analytics for management of storms. With the use of methods such as deep learning and time series forecasting, early warning systems can be improved, opening the door to proactive emergency planning (Li, Li, et al., 2020).

Expected Impact: Improves disaster anticipation and resource readiness, reducing emergency response times.

2. Geospatial Risk Mapping for Resource Deployment

Visualizations based on static and streaming data show geographic clusters characterized by high storm intensity, high rates of injury, and high property losses. These observations can help the Red Cross prioritize resource allocation by generating dynamic heatmaps and clustering models that are updated in real time (Hashem et al., 2015). Pre-deployment of critical supplies, such as emergency shelters, medical supplies, and rescue personnel, to high-risk areas, can help improve the effectiveness of response activities.

Expected Impact: Enhances resource effectiveness and increases community resilience in high-risk areas.

3. Real-Time, Automated Alert Systems

Since Spark and BigQuery have shown success handling streaming alert data, the Red Cross should create a real-time monitoring dashboard. This dashboard would systematically filter alerts according to severity, urgency, and response type, supplemented with visualization aids like graphical frequency trends and urgency heatmaps (Imran, et al., 2014). It would enable the decision-making processes of field officers to be faster, data-driven, enabling the latter to prepare societies proactively for approaching storms.

Expected Impact: It provides real-time situational awareness, enabling faster, more accurate responses as disasters occur.

4. Optimization of Public Messaging

Analysis of often repeated key terms "Warning," "Severe," and "Tornado" from streaming alerts holds great promise for improving public communication. The Red Cross needs to concentrate efforts towards creating standardized alert messages that utilize explicit and imperative templates. Through testing the templates with groups of people of different backgrounds, we can provide assurance that messages get quickly understood and evince appropriate public actions, thereby reducing fatalities as well as misunderstandings of disasters (Imran et al., 2014).

Expected Impact: Increases public awareness and compliance in response to emergency notifications, thus creating better safety outcomes.

Supporting Documentation

Data Sheet

We used two data sets - one static and one streaming - of data for the analysis of severe storms based in the United States.

The static dataset, Storm Events Database, was obtained from the National Oceanic and Atmospheric Administration's (NOAA). The dataset contains data from 1950 until 2024, and focuses on the location, storm type/severity, and impact (financial and otherwise) of each documented storm. We chose this dataset because it is accessible (i.e. free for public use), the data are both formatted consistently and despite spanning several decades, the quality of each observation is generally reliable.

The streaming data was obtained from the National Weather Service (NWS) API. The live data was collected through the API on the 20th of April, 2025 at different times. Collected data contained information about the severity, onset, and offset of recorded storms. We chose this data because we believe that its use can help the Red Cross to predict severe storms and prepare affected communities ahead of time, as to minimize damages and casualties. The data is also current (i.e. up-to-date), timely, and - similarly to the streaming dataset obtained from NOAA - publicly accessible.

Meeting Notes

Meeting 1

Meeting Information	**Date:** April 15, 2025 **Times:** ● Start: 1:40 PM ● End: 3:52 PM **Attendees:** ● Shivangi Vipulbhai Borad ● Isabella Pham ● Kiran Kuchekar ● Ankur Bhattacharjee ● Cezane Karki **Note-Taker:** ● Isabella Pham
Notes	● Bimala wasn't able to meet due to a personal emergency, but was kept updated via WhatsApp message and assisted with searching for datasets ● In order to facilitate collaborative work, the following were created and shared among the group: ○ Shared Google Drive folder ○ Google Document ○ Google Slides presentation ○ Jira dashboard

- Isabella updated and shared project overview notes (taken during class) with the group
- Discussed group availability for the week and scheduled the next meeting (April 18)

DATASET RESEARCH/DISCUSSION

- Discussed and researched potential datasets
  - Group ran into issues with certain datasets being unavailable, with datasets being of inadequate size/quality, and with datasets containing useful but not completely relevant information
- Located a static dataset on U.S.-based storm events from the National Centers for Environmental Information
  - Discussed quality of dataset, and agreed that it was high-quality (easily-accessible, consistent, accurate, and current)
  - Discovered that the dataset contained a wide range of data, *except* for enough data on tropical cyclones
- Agreed that using the above dataset would be ideal if our use case/prompt included other types of storm events

PLANNING

- Discussed (but didn't fully solidify):
  - An outline for our plan of action
  - How work might be divided
- Wrote an email to Dr. Schenita Floyd regarding the following:
  - Potentially expanding the scope of our use case/prompt to include other types of storm events (e.g. tornadoes, hurricanes)
  - Document format guidelines

Decisions

- **Static Dataset:** Storm Events Database
  - Contains storm event information across the U.S. (from 1950 to 2024)
  - Data will be filtered to only include tropical storms, hurricanes, and tornados
- **Streaming Dataset:** No decision made yet
  - Cezane and Ankur agreed to take shared responsibility for setting up APIs
- **Use Case:** Group agreed to expand our use case/prompt to include other storm events (with Dr. Floyd's permission)
- **Next Meeting:**
  - April 18, 2025 - Willis Library
  - Approximate meeting time: 1:00 PM - 5:00 PM

Action Items

Item #	*Description*	*Assigned To*	*Due Date*	*Status*
1	Email Dr. Floyd	Isabella Pham, Kiran Kuchekar, Shivangi Borad	04/15/2025	Completed
2	Search for datasets	Whole Group	04/18/2025	Ongoing
3	Thoroughly check static dataset for quality + compile data	Cezane Karki	04/18/2025	Ongoing
4	Identify an appropriate streaming dataset for use	Cezane Karki, Ankur Bhattacharjee	04/18/2025	Ongoing
5	Choose a non-profit organization and determine why we are gathering data for them	Whole Group	04/18/2025	Ongoing
6	Create GCP project and assign access permissions	Kiran S. Kuchekar	04/18/2025	Not Started
7	Create GCP bucket & upload raw static data	Kiran S. Kuchekar	04/18/2025	Not Started

	8	Format presentation document and create presentation outline	Isabella Pham	04/22/2025	Not Started

Meeting 2

Meeting Information	**Date:** April 18, 2025 **Times:** ● Start: 2:14 PM ● End: 4:33 PM **Attendees:** ● Shivangi Vipulbhai Borad ● Isabella Pham ● Kiran Kuchekar ● Bimala Joshi ● Cezane Karki ● Ankur Bhattacharjee **Note-Taker:** ● Isabella Pham
Notes	● Added tasks and assigned tasks on the Jira project dashboard ● Identified two potential streaming data sources and agreed to contact Dr. Floyd for approval: ○ Weather.Gov (National Weather Service): API Web Service ○ WxData.com: Storm Reports API ● Created the following: ○ GCP project (added all group members as principals) ○ GCP storage bucket ○ Directory structure within storage bucket (separate containers for raw and refined data for both streaming and static data sets) ● Combed through the static dataset & discussed how to clean it (i.e. which columns to delete, which to keep, etc.) ● Discussed presentation format/theme and outline ● Discussed which non-profits we can use ● Ran into issues with running data from the static dataset through OpenRefine ○ Meeting ended without a clear solution
Decisions	● Non-Profit ○ Chose the American Red Cross for their humanitarian and disaster relief efforts ○ Determined that - within the scope of the project prompt - we were hired to build a pipeline for data which can be used to predict storm events for more efficient and effective disaster aid ● Chose two potential streaming datasets (pending approval from Dr. Floyd) ● Scheduled next meetings: ○ Meeting 3: ■ April 22 (Tuesday) ■ 12:00 PM - Undetermined End Time ■ Curry Hall, Rom 204 ○ Meeting 4: ■ April 25 (Friday) ■ 2:30 PM - 4:30 PM ■ Willis Library, Room 250G

Action Items

Item #	*Description*	*Assigned To*	*Due Date*	*Status*
1	Thoroughly check static dataset for quality + compile data	Cezane Karki, Ankur Bhattacharjee	04/18/2025	Completed
2	Email Dr. Floyd	Isabella Pham	04/18/2025	Completed

	3	Identify an appropriate streaming dataset for use	Cezane Karki, Ankur Bhattacharjee	04/18/2025	Completed
	4	Thoroughly check static dataset for quality + compile data	Cezane Karki	04/18/2025	Completed
	5	Choose a non-profit organization and determine why we are gathering data for them	Whole Group	04/18/2025	Completed
	6	Create GCP project and assign access permissions	Kiran S. Kuchekar	04/18/2025	Completed
	7	Create GCP bucket & upload raw static data	Kiran S. Kuchekar	04/18/2025	Completed
	8	Format presentation document and create presentation outline	Isabella Pham	04/22/2025	Ongoing
	9	Clean/refine data from static data source using OpenRefine	Ankur Bhattacharjee	04/22/2025	Ongoing
	10	Clean/refine data from streaming data source using OpenRefine	Ankur Bhattacharjee	04/22/2025	Not Started
	11	Create cluster using GCP Dataproc	Bimala Joshi	04/22/2025	Not Started
	12	Write SQL queries in GCP BigQuery for the static dataset	Cezane Karki	04/22/2025	Not Started
	13	Write SQL queries in GCP BigQuery for the streaming dataset	Cezane Karki	04/22/2025	Not Started

Meeting 3

Meeting Information	**Date:** April 22, 2025 **Times:** ● Start: 12:08 PM ● End: 8:07 PM **Attendees:** ● Shivangi Vipulbhai Borad ● Isabella Pham ● Kiran Kuchekar ● Bimala Joshi ● Cezane Karki ● Ankur Bhattacharjee **Note-Taker:** ● Isabella Pham
Notes	● Solved previous issues with OpenRefine & finished cleaning both static and streaming data ● Created presentation outline, began writing speaker notes, & completed multiple slides ● Uploaded refined streaming and static data to GCP bucket ● Created cluster using GCP Dataproc + VM instances (1 manager node, 2 worker nodes) ● Wrote multiple queries using GCP BigQuery for both streaming and static data ● Copied the refined versions of both streaming and static datasets to HDFS and created tables for each ● Wrote and sent queries for the streaming data table using Spark SQL

	● Began writing and sending queries for the static data table using HiveQL
Decisions	● N/A

Action Items

Item #	*Description*	*Assigned To*	*Due Date*	*Status*
1	Format presentation document and create presentation outline	Isabella Pham, Shivangi Vipulbhai Borad	04/22/2025	Completed
2	Clean/refine static data using OpenRefine	Ankur Bhattacharjee	04/22/2025	Completed
3	Set up APIs for streaming data	Cezane Karki, Ankur Bhattacharjee	04/22/2025	Completed
4	Clean/refine streaming data using OpenRefine	Ankur Bhattacharjee	04/22/2025	Completed
5	Create cluster using GCP Dataproc	Bimala Joshi	04/22/2025	Completed
6	Write SQL queries in GCP BigQuery for the static dataset	Cezane Karki	04/22/2025	Completed
7	Write SQL queries in GCP BigQuery for the streaming dataset	Cezane Karki	04/22/2025	Completed
8	Copy static data from GCP bucket to HDFS; create table	Isabella Pham	04/22/2025	Completed
9	Copy streaming data from GCP bucket to HDFS; create table	Shivangi Vipulbhai Borad	04/22/2025	Completed
10	Query data using Spark SQL queries	Shivangi Vipulbhai Borad	04/22/2025	Completed
11	Query data using HiveQL queries	Isabella Pham	04/25/2025	Ongoing
12	Update document outline & fill in documentation	Bimala Joshi, Group	04/25/2025	Ongoing
13	Format, revise, and edit document before submission	Isabella Pham, Group	04/28/2025	Not Started

Meeting 4

Meeting Information	**Date:** April 26, 2025 **Times:** ● Start: 9:00 PM ● End: 10:40 PM **Attendees:** ● Shivangi Vipulbhai Borad ● Isabella Pham

	● Kiran Kuchekar ● Bimala Joshi ● Cezane Karki ● Ankur Bhattacharjee **Note-Taker:** ● Isabella Pham
Notes	● Reviewed document to: ○ Update and change written content according to group consensus ○ Ensure that all written content and formatting was both factual and correct ○ Ensure that all group members are on the same page ● Updated document contents according to group discussion and agreement ● Assigned presentation slides to group members
Decisions	● Assigned presentation slides to group members ● Agreed to meet at 9 PM on Monday evening to practice presenting

Action Items

Item #	*Description*	*Assigned To*	*Due Date*	*Status*
1	Query data using HiveQL queries	Isabella Pham	04/25/2025	Completed
2	Update document outline & fill in documentation	Bimala Joshi, Group	04/25/2025	Completed
3	Format, revise, and edit document before submission	Isabella Pham	04/28/2025	Completed
4	Complete comparison chart & interpretation for Apache Hive and Spark	Isabella Pham, Shivangi Vipulbhai Borad	04/28/2025	Completed
5	Create table of contents	Isabella Pham	04/28/2025	Completed
6	Complete & upload data visualizations	Cezane Karki	04/28/2025	Completed
7	Complete written content for the “Interpretation” section	Shivangi Vipulbhai Borad	04/28/2025	Completed
8	Complete writing discussion and actionable recommendations	Kiran S. Kuchekar	04/28/2025	Completed
9	Complete written content for the conclusion (discussion & recommendations)	Kiran S. Kuchekar	04/28/2025	Ongoing
10	Complete data sheet	Bimala Joshi	04/28/2025	Completed
11	Update slides according to contents of the written document	Whole Group	04/28/2025	Ongoing
12	Assign presentation slides	Whole Group	04/27/2025	Completed

Meeting 5

<table>
<tr><td>Meeting Information</td><td colspan="5"><u>Date:</u> April 28, 2025
<u>Times:</u>
● Start: 9:00 PM
● End: 10:09 PM
<u>Attendees:</u>
● Shivangi Vipulbhai Borad
● Isabella Pham
● Kiran Kuchekar
● Bimala Joshi
● Cezane Karki
● Ankur Bhattacharjee
<u>Note-Taker:</u>
● Isabella Pham</td></tr>
<tr><td>Notes</td><td colspan="5">● Practiced presenting (with a specific focus on remaining within 8 minutes)
● Reviewed/double-checked presentation and written document for content and any errors</td></tr>
<tr><td>Decisions</td><td colspan="5">● N/A</td></tr>
<tr><td rowspan="3">Action Items</td><td>Item #</td><td>Description</td><td>Assigned To</td><td>Due Date</td><td>Status</td></tr>
<tr><td>1</td><td>Complete written content for the conclusion (discussion & recommendations)</td><td>Kiran S. Kuchekar</td><td>04/28/2025</td><td>Completed</td></tr>
<tr><td>2</td><td>Update slides according to contents of the written document</td><td>Whole Group</td><td>04/28/2025</td><td>Completed</td></tr>
</table>

Access & Permission Policy

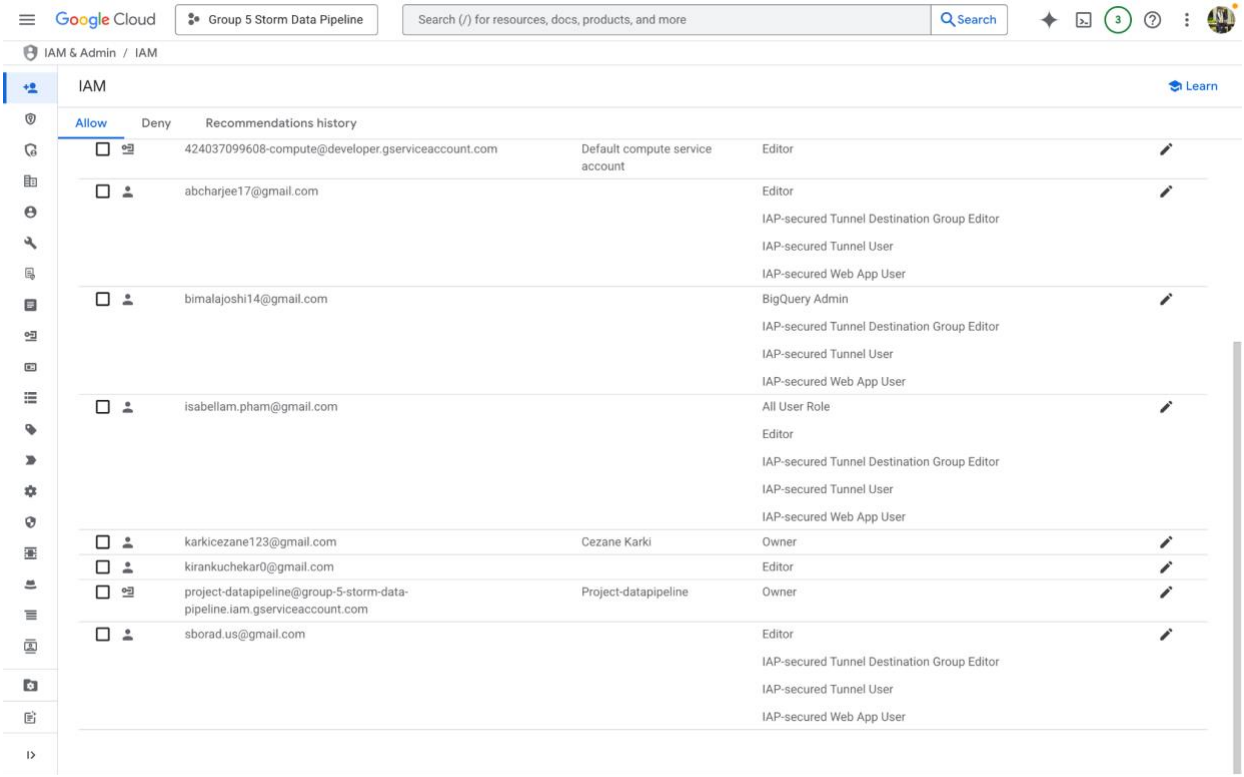


Figure 14. IAM policy for all permitted users.

Jira Dashboard (BONUS)

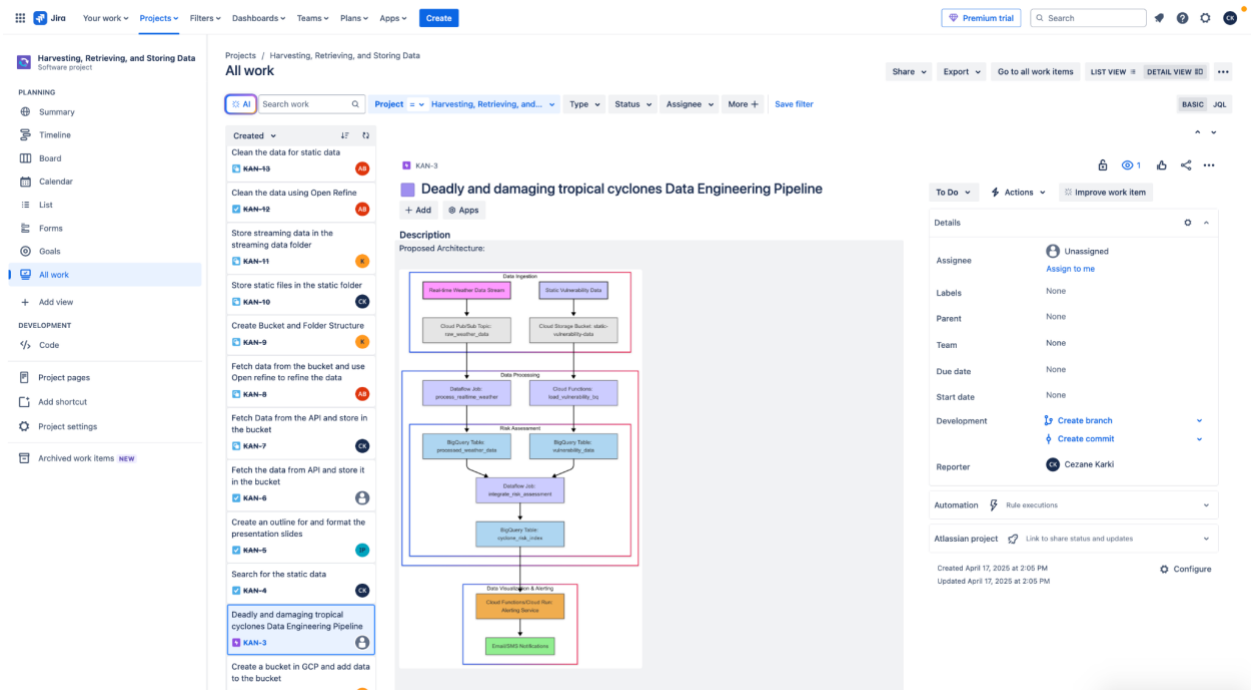


Figure 15.1 Jira dashboard (Project Epic).

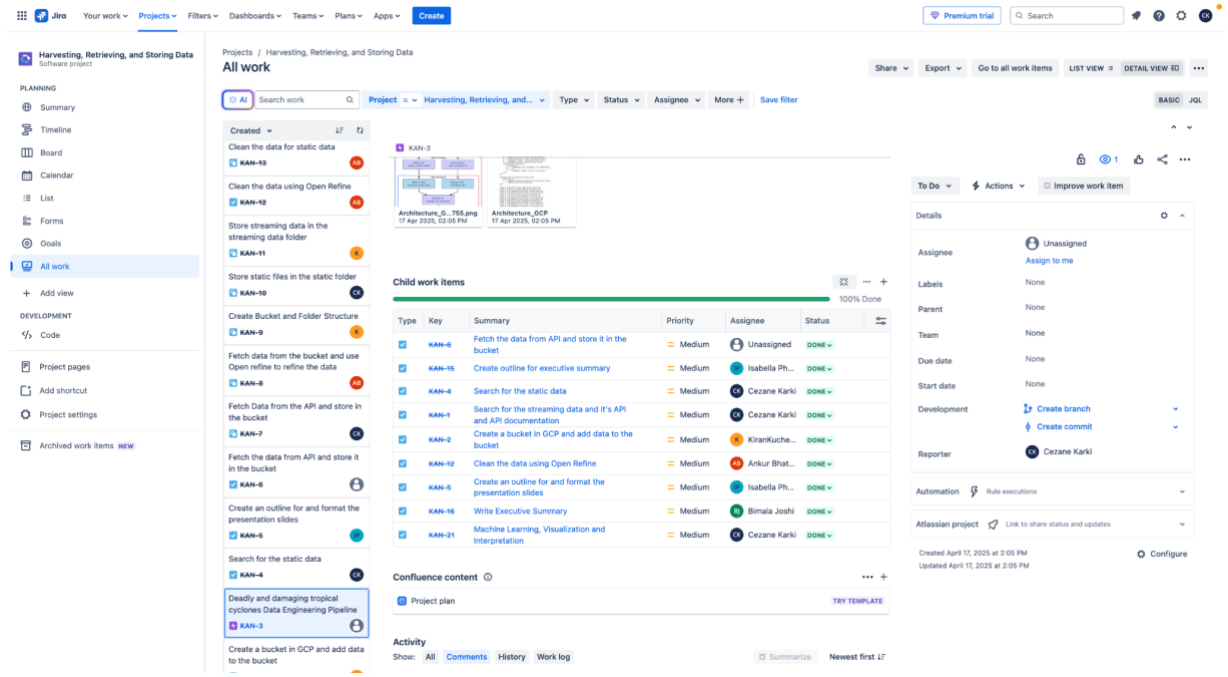


Figure 15.2 Jira dashboard (Task assigned to Epic).

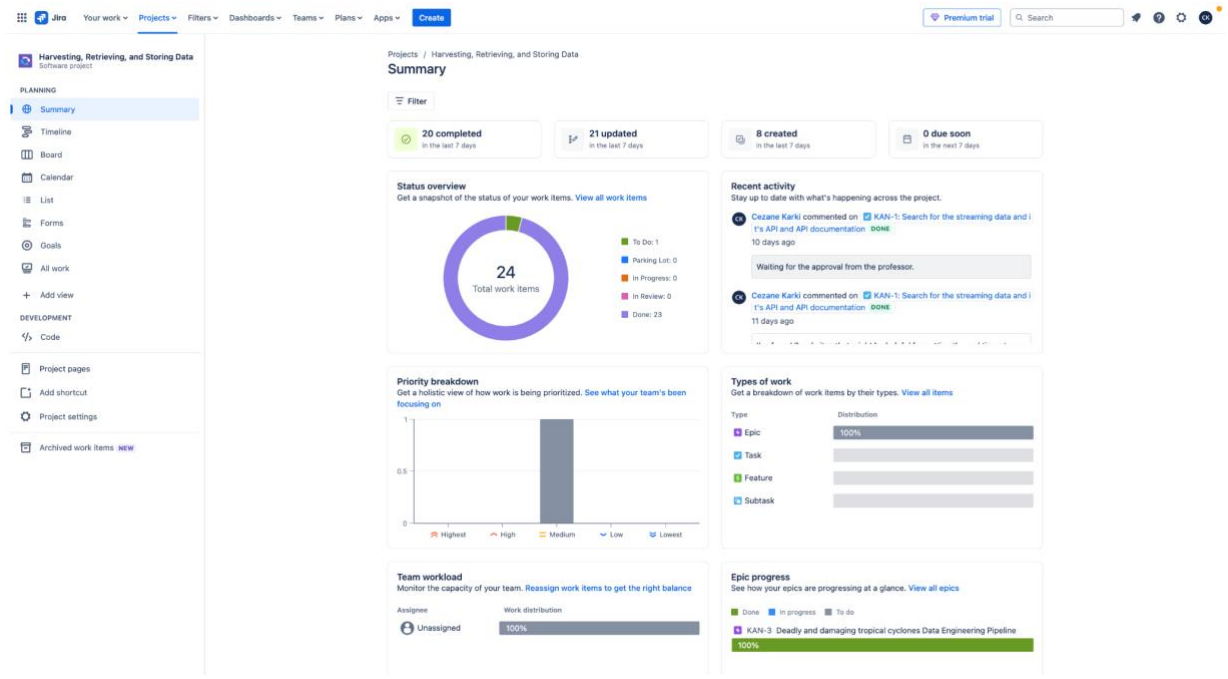


Figure 15.3 Jira dashboard (Summary of the Project).

References

American Red Cross. (2025). *About us*. https://www.redcross.org/about-us.html

Berman, D. (2019, August 5). *Comparing Apache Hive vs. Spark*. Logz.Io. https://logz.io/blog/hive-vs-spark/

Hashem, I. A. T., Yaqoob, I., Anuar, N. B., Mokhtar, S., Gani, A., & Khan, S. U. (2015). The rise of "big data" on cloud computing: Review and open research issues. *Information Systems*, 47, 98–115. https://doi.org/10.1016/j.is.2014.07.006

Imran, M., Castillo, C., Lucas, J., Meier, P., & Vieweg, S. (2014). AIDR: Artificial Intelligence for Disaster Response. *Proceedings of the 23rd International Conference on World Wide Web (WWW '14 Companion)*, 159–162. https://doi.org/10.1145/2567948.2577034

Li, Z., Li, W., Wang, Y., & Duan, Y. (2020). A survey of deep learning-based human activity recognition in disaster response. *Pattern Recognition Letters, 129*, 120–126. https://doi.org/10.3390/rs11091068

Shimizu. (2014, December 9). *American Red Cross logo vector*. BrandLogos. https://brandlogos.net/american-red-cross-25705.html