AOL ANALYSIS WEATHER EVENTS

By: Nick Chandler, Luke Richard, Anirudh Sarda, Nancy Bou Kamel, Nataliia Remezova, Luisa Kalkert, Chiraag Mishra

Presentation Outline

1. Analysis Goals
2. External Dataset
3. Overall Database Schema (Pre Slicing)
4. The 5 Questions

Analysis Goals

1. Understand the characteristics of weather events of March to May 2006

&

2. Analyze how weather events impacted people's browsing behavior

External Dataset

Data-Source

<u>NOAA Storm Events Database</u>

- ❏ By the National Oceanic and Atmospheric Association (NOAA), specifically the National Weather Service (NWS)
- ❏ **Official Data**: NOAA as part of the US Department of Commerce offers scientifically accurate weather data

Storm Events Database

Weather event: A specific weather occurrence, like storms, floods, or heatwaves, confined to a particular time and place

Dimensions:

- ❑ Time (Begin and End)
- ❑ Event Type (e.g. Tornado, Drought, Hailstorm)
- ❑ Location
- ❑ Deaths and Injuries
- ❑ Monetary damage (Crops and Property)

```
CREATE TABLE AOL_SCHEMA.WEATHER_EVENTS (
    BEGIN_DATE_TIME TIMESTAMP,
    END_DATE_TIME TIMESTAMP,
    EVENT_TYPE VARCHAR(100),
    REGION VARCHAR(100),
    BEGIN_DAY INTEGER,
    END_DAY INTEGER,
    BEGIN_MONTH INTEGER,
    END_MONTH INTEGER,
    INJURIES_DIRECT INTEGER,
    INJURIES_INDIRECT INTEGER,
    DEATHS_DIRECT INTEGER,
    DEATHS_INDIRECT INTEGER,
    DAMAGE_PROPERTY INTEGER,
    DAMAGE_CROPS FLOAT,
    EPISODE_ID INT,
    EVENT_ID INT NOT NULL,
    PRIMARY KEY (EVENT_ID)
)
```

Preprocessing

```
def convert_abbreviated_string(value):
    # Check for 'k' (thousand), 'm' (million), 'b' (billion)
    if isinstance(value, str):
        print(value)
        if 'k' in value.lower():
            return float(value.replace('k', '').replace('K', '')) * 1000
        elif 'm' in value.lower():
            return float(value.replace('m', '').replace('M', '')) * 1000000
        elif 'b' in value.lower():
            return float(value.replace('b', '').replace('B', '')) * 1000000000
        else:
            # If no abbreviation, just return the float version of the number
            try:
                return float(value)
            except ValueError:
                return None  # or handle invalid strings as needed
    return value
```

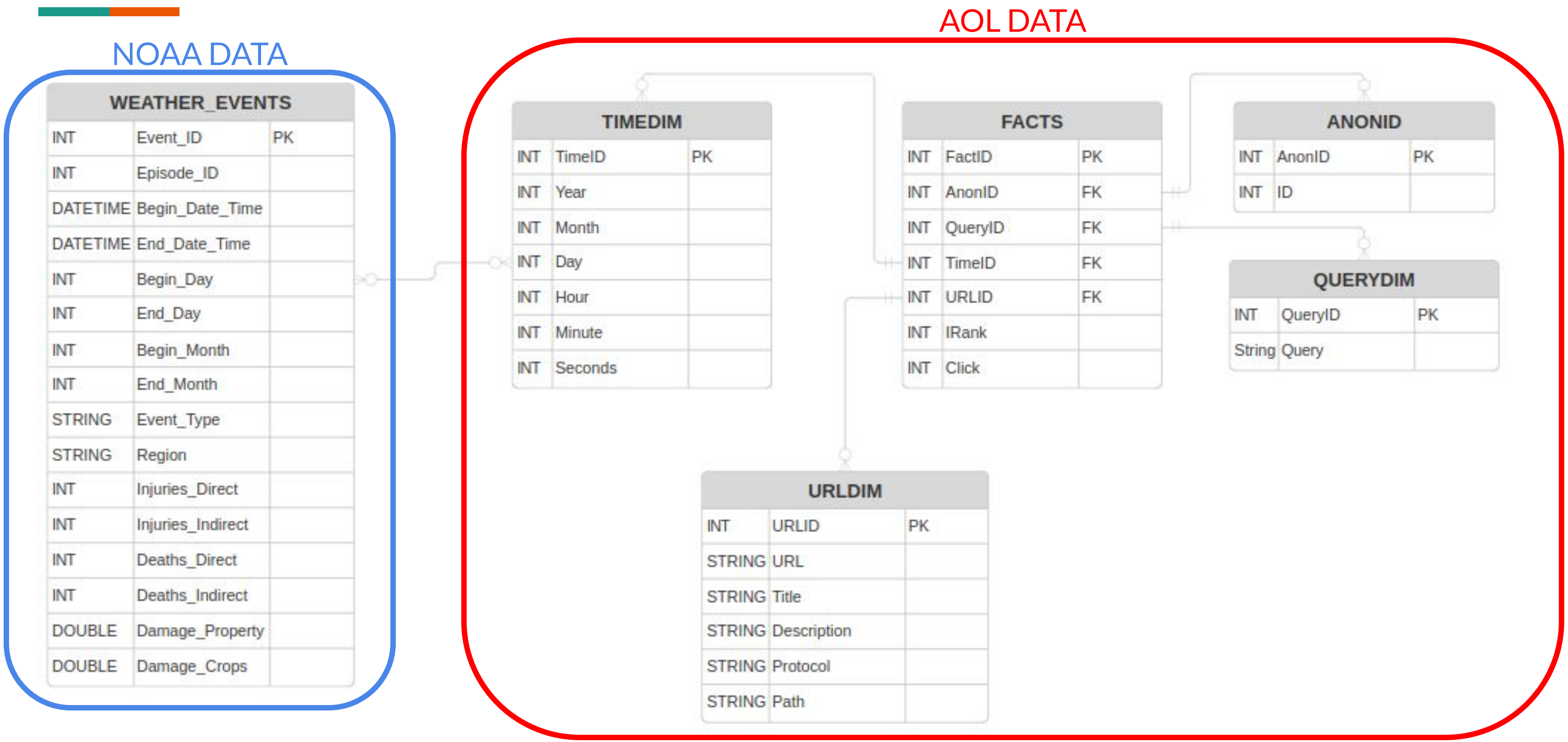
- ❑ Alter the format of the “counts of damages”

- ❑ Remove duplicate rows

```
# Replace NaN values with None
df = df.where(pd.notnull(df), None)
print(df.shape)

# Remove duplicates based on all columns except 'EVENT_ID'
columns_except_event_id = df.columns[df.columns != 'EVENT_ID']

# Drop duplicates based on all columns except 'EVENT_ID'
distinct_df = df.drop_duplicates(subset=columns_except_event_id)
```

Database Schema

Database-Schema

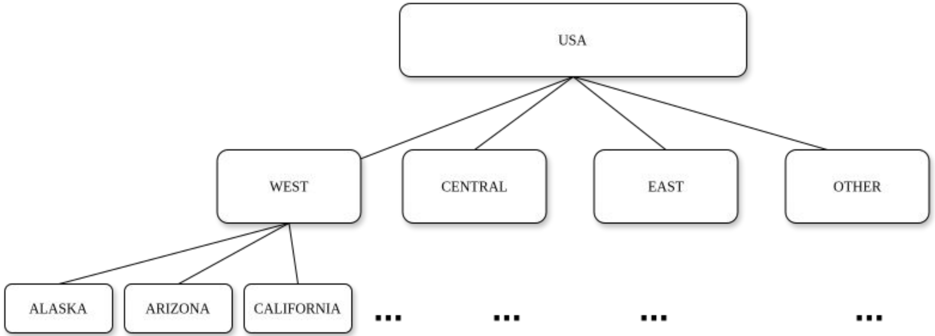


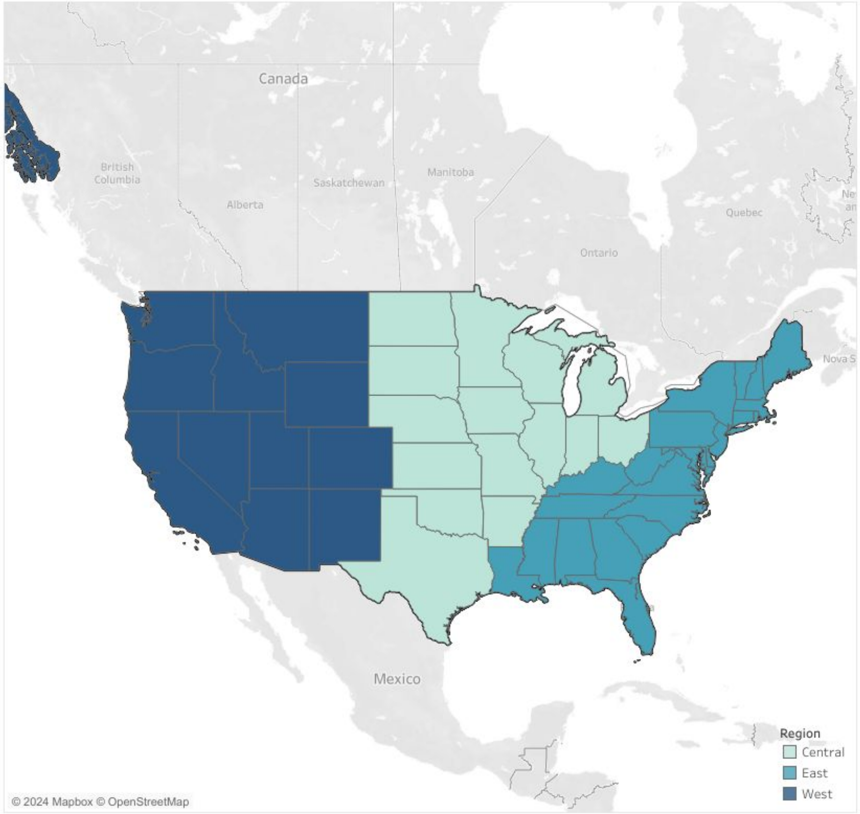
Our Questions

Question 1:
When and where have weather events been most destructive?

Spatial Hierarchy

- We group the US states into **4 Regions:** West, Central, East and Other (e.g. Guam, Puerto Rico or American Samoa)

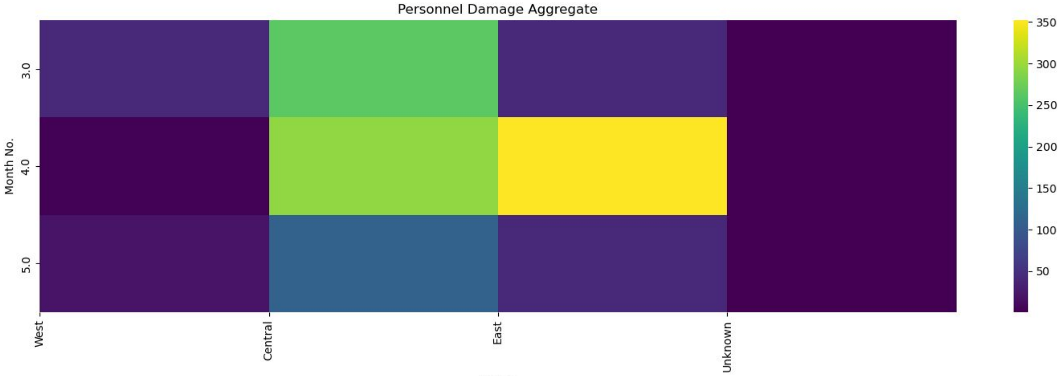


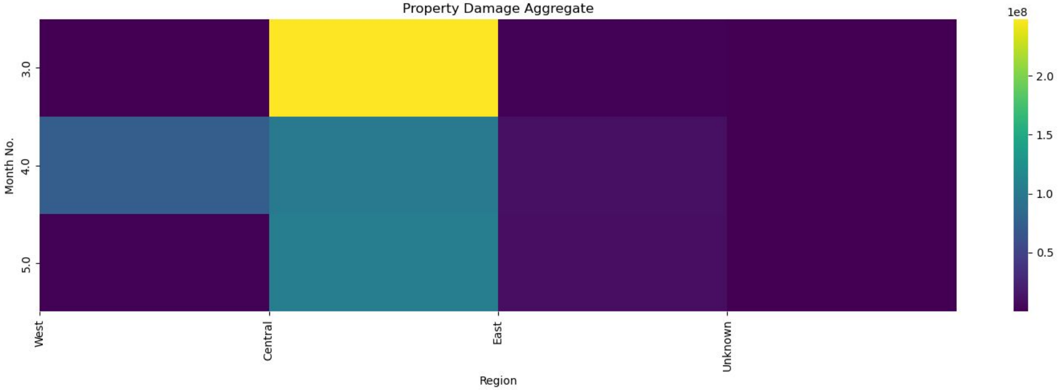


Query

```sql
WITH NewWeatherData AS (
    SELECT
        MONTH(BEGIN_DATE_TIME) AS BEGIN_MON,
        REGION,
        CASE
            -- West Region
            WHEN STATE_FIPS IN ('02', '04', '06', '08', '15', '16', '30', '32', '35', '41', '49', '53', '56')
            THEN 'West'
            -- Central Region
            WHEN STATE_FIPS IN ('05', '17', '18', '19', '20', '26', '27', '29', '31', '38', '39', '46', '48', '55')
            THEN 'Central'
            -- East Region
            WHEN STATE_FIPS IN ('01', '09', '10', '11', '12', '13', '21', '22', '23', '24', '25', '28', '33', '34', '36', '37', '40', '42', '44' '45', '47', '50',
'51', '54')
            THEN 'East'
            ELSE 'Unknown'
        END AS TRISECTION,
        COALESCE (INJURIES_DIRECT, 0) + COALESCE (INJURIES_INDIRECT, 0) + COALESCE(DEATHS_DIRECT, 0) + COALESCE(DEATHS_INDIRECT, 0) AS HUMAN_DAMAGE,
        COALESCE(DAMAGE_PROPERTY, 0) + COALESCE (DAMAGE_CROPS, 0) AS NON_HUMAN_DAMAGE
    FROM
        AOL_SCHEMA.WEATHER_EVENTS
    WHERE
        MONTH(BEGIN_DATE_TIME) >= 3.0
SELECT
    BEGIN_MON,
    REGION,
    TRISECTION,
    SUM(HUMAN_DAMAGE) AS TOTAL_HUMAN_DAMAGE,
    SUM(NON_HUMAN_DAMAGE) AS TOTAL_NON_HUMAN_DAMAGE
FROM
    NewWeatherData
GROUP BY
    CUBE (BEGIN_MON, REGION, TRISECTION)
HAVING
    SUM(HUMAN_DAMAGE) > 0
    AND SUM(NON_HUMAN_DAMAGE) > 0
ORDER BY
    BEGIN_MON,
    TRISECTION,
    REGION;
```

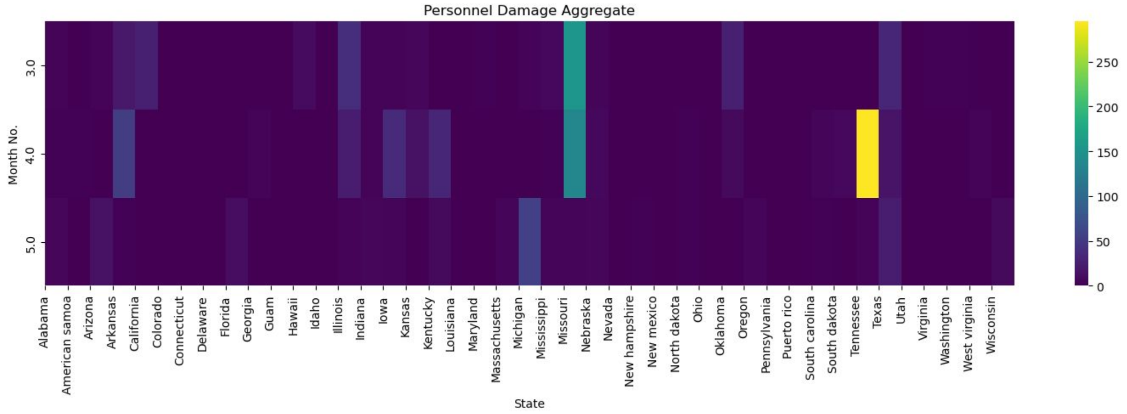
Region Aggregates Heatmaps

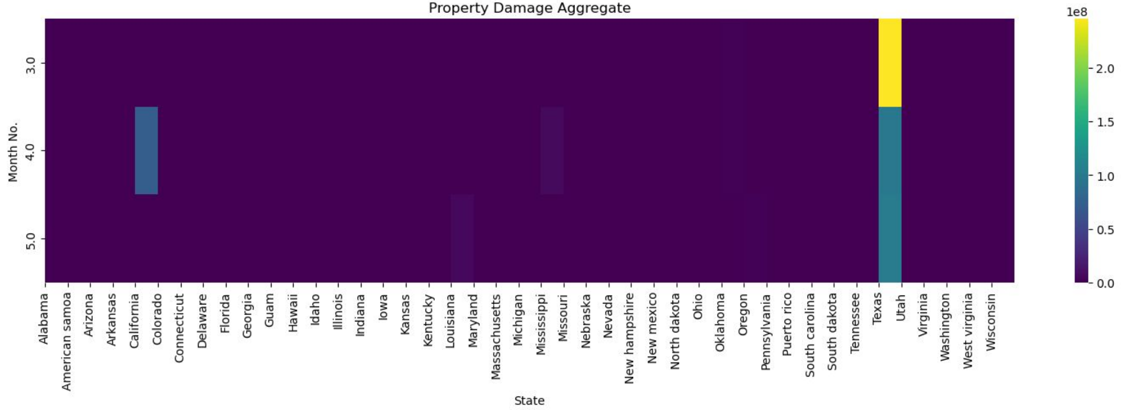




- Many injuries/deaths in the East in April.
- Most property damage was in the Central region in March.
- The Central region incurred the most property damage throughout the period.

State-wise Aggregate Heatmaps



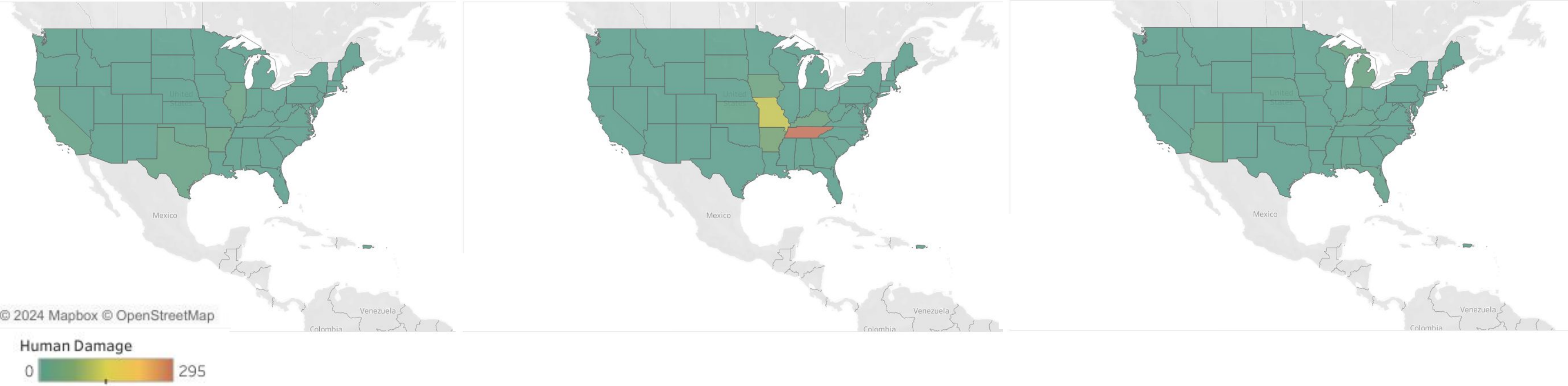


- Tennessee in April had the most injuries/deaths.
- Missouri had many injuries/deaths in March and April
- Texas had the most property damage across all months with March being the worst.
- California also had some property damage in April.

Deaths and Injuries by Natural Disasters over Time

❑ Most dangerous month: **April** with **651 injuries and deaths**

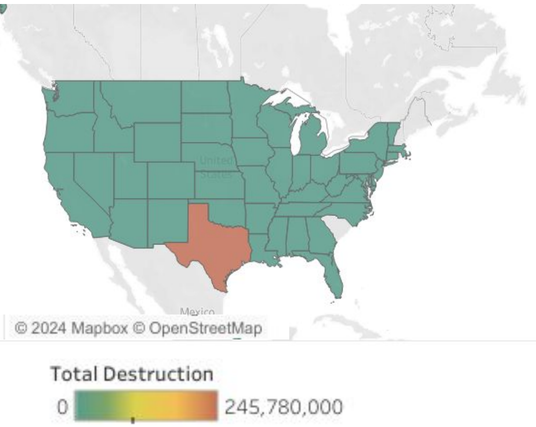
March April May



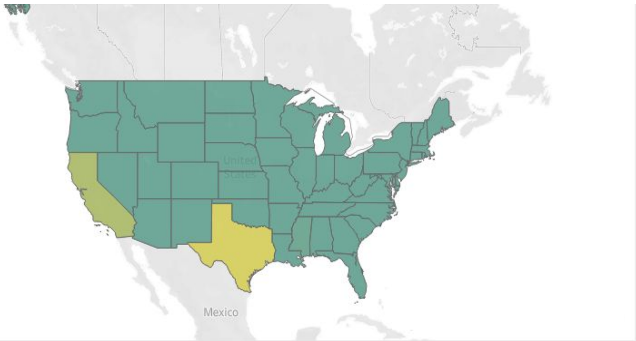
Damage from natural disasters in USD over Time

❏ Month with most damage to property: **March** with **$251 Mio**.

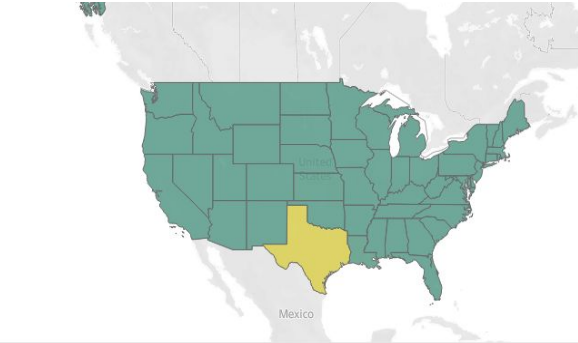
March



April



May

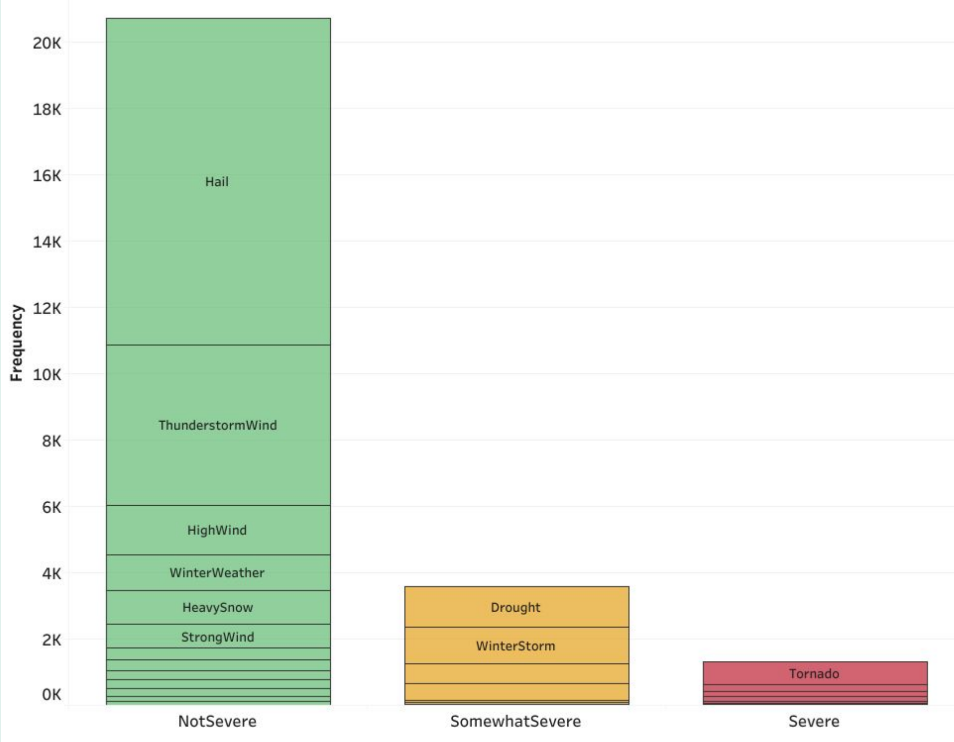


Question 2:
What are the types of events that we observe and how do they change over time?

Query (Questions 2 & 3)

```
WITH SEVERITY_TABLE AS(
    SELECT
    EVENT_ID,
    EVENT_TYPE,
    BEGIN_DATE_TIME,
    CASE
        WHEN WEATHER_EVENTS.EVENT_TYPE IN ('Blizzard', 'Tornado', 'Wildfire', 'Avalanche', 'Funnel Cloud', 'Waterspout',
'Debris Flow') THEN 'Severe'
        WHEN WEATHER_EVENTS.EVENT_TYPE IN ('Coastal Flood', 'Flash Flood', 'Flood', 'Drought', 'Dust Devil', 'Dust Storm',
'Storm Surge/Tide', 'Ice Storm', 'Winter Storm') THEN 'Somewhat Severe'
        WHEN WEATHER_EVENTS.EVENT_TYPE IN ('Cold/Wind Chill', 'Frost/Freeze', 'Heat', 'Heavy Rain', 'Heavy Snow', 'High
Wind', 'Strong Wind', 'Thunderstorm Wind', 'Winter Weather', 'Lightning', 'Marine High Wind', 'Marine Thunderstorm Wind',
'Sleet', 'WINTER WEATHER', 'High Surf', 'Marine Hail', 'Rip Current', 'Lake-Effect Snow', 'Dense Fog') THEN 'Not Severe'
        ELSE 'Unclassified'
    END AS SEVERITY
FROM AOL_SCHEMA.WEATHER_EVENTS
),
AGG_EVENTS AS(
    SELECT
        SEVERITY,
        EVENT_TYPE,
        COALESCE(COUNT(EVENT_ID),0) AS FREQ
    FROM SEVERITY_TABLE
    GROUP BY ROLLUP(SEVERITY, EVENT_TYPE))
SELECT
    SEVERITY,
    EVENT_TYPE,
    FREQ,
    RANK() OVER(PARTITION BY SEVERITY ORDER BY FREQ DESC) as RANKING
FROM AGG_EVENTS
;
```

Weather Events by Severity



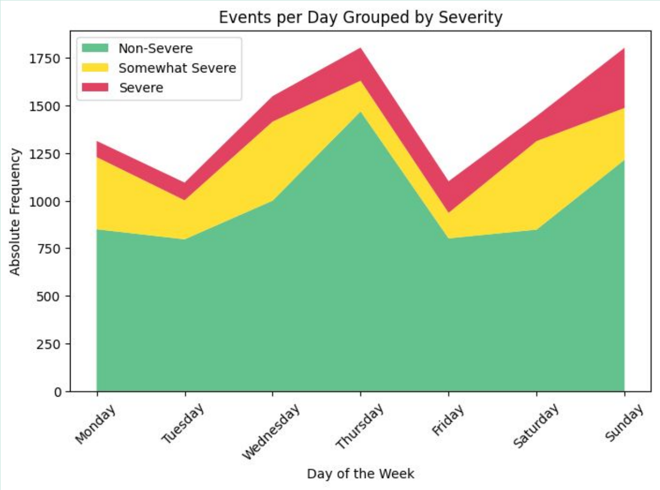
- **Cold** and **Wind** related weather events make up the majority of frequent weather events
- **Tornados** make up the majority of severe weather events
- Overall: about 26k recorded weather events in 92 days

Query

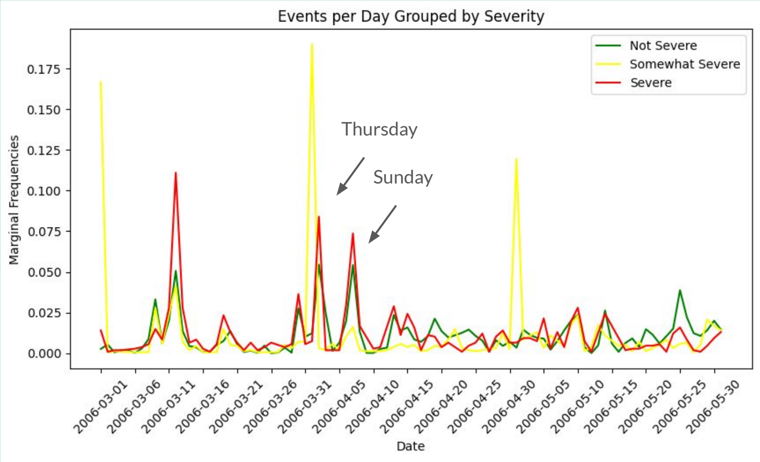
(Questions 2 & 3)

```
WITH DATE_RANGE AS (
    SELECT DATE '2006-03-01' AS EVENT_DATE
    UNION ALL SELECT DATE '2006-03-02'
    .
    .
    UNION ALL SELECT DATE '2006-05-31'
),
SEVERITY_TABLE AS(
SELECT
    EVENT_TYPE,
    EVENT_ID,
    CAST(WEATHER_EVENTS.BEGIN_DATE_TIME AS DATE) AS BEGIN_DATE,
    WEEK(WEATHER_EVENTS.BEGIN_DATE_TIME) AS BEGIN_WEEK,
    (MOD(CAST(CAST(WEATHER_EVENTS.BEGIN_DATE_TIME AS DATE) - CAST('2006-01-01' AS DATE) AS INTEGER) + 6, 7) + 1) AS WEEKDAY,
    CASE
        WHEN WEATHER_EVENTS.EVENT_TYPE IN ('Blizzard', 'Tornado', 'Wildfire', 'Avalanche', 'Funnel Cloud', 'Waterspout') THEN 'Severe'

        WHEN WEATHER_EVENTS.EVENT_TYPE IN ('Coastal Flood', 'Flash Flood', 'Flood', 'Drought', 'Dust, Devil',
                                          'Dust Storm', 'Storm Surge/Tide') THEN 'Somewhat Severe'

        WHEN WEATHER_EVENTS.EVENT_TYPE IN ('Cold/Wind Chill', 'Frost/Freeze', 'Heat', 'Heavy Rain', 'Heavy Snow', 'Hail','High Wind', 'Strong Wind', 'Thunderstorm Wind',
                                          'Winter Weather', 'Lightning', 'Marine High Wind','Marine Thunderstorm Wind', 'Sleet', 'WINTER WEATHER', '
                                          Winter Storm') THEN 'Not Severe'
        ELSE 'Unclassified'
    END AS SEVERITY
FROM AOL_SCHEMA.WEATHER_EVENTS
)

SELECT
    DATE_RANGE.EVENT_DATE,
    SEVERITY_TABLE.BEGIN_WEEK,
    SEVERITY_TABLE.WEEKDAY,
    SEVERITY_TABLE.SEVERITY,
    SEVERITY_TABLE.EVENT_TYPE,
    COALESCE(COUNT(SEVERITY_TABLE.EVENT_ID), 0) AS FREQ
FROM DATE_RANGE

LEFT JOIN SEVERITY_TABLE
ON DATE_RANGE.EVENT_DATE = SEVERITY_TABLE.BEGIN_DATE

GROUP BY ROLLUP((DATE_RANGE.EVENT_DATE, SEVERITY_TABLE.BEGIN_WEEK, SEVERITY_TABLE.WEEKDAY, SEVERITY_TABLE.SEVERITY),\
        (DATE_RANGE.EVENT_DATE, SEVERITY_TABLE.BEGIN_WEEK, SEVERITY_TABLE.WEEKDAY, SEVERITY_TABLE.EVENT_TYPE))

ORDER BY DATE_RANGE.EVENT_DATE ASC;
```

Severity of Weather Events

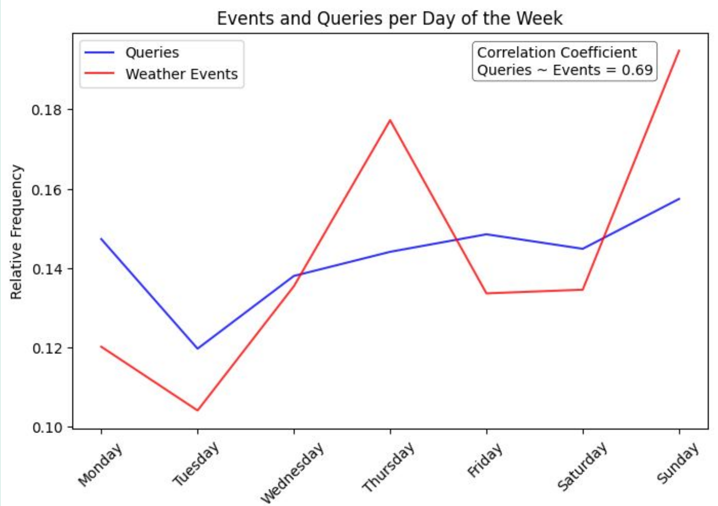


- ❑ The number of events **fluctuates heavily** over the week
- ❑ Cause: Might be due to **few extreme days**



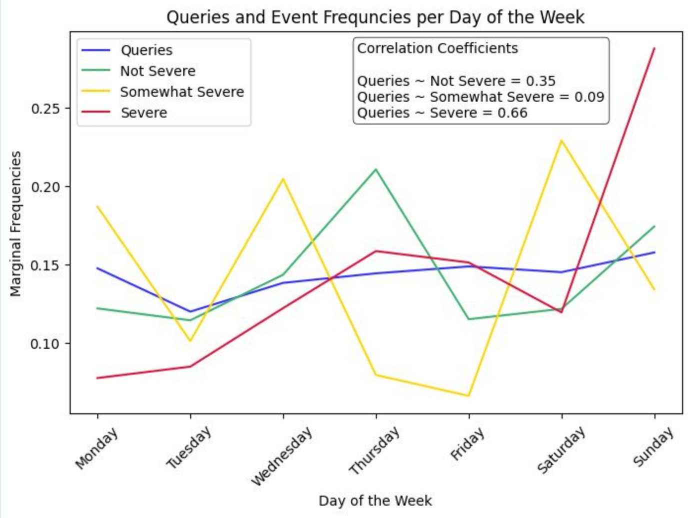
- ❑ **Extreme spikes** in weather events: Most days have few events, but occasional days experience a sharp surge

Searches and Severity



- ❏ Searches for weather events roughly follow the number of events

- ❏ Strongest correlation: Severe weather events

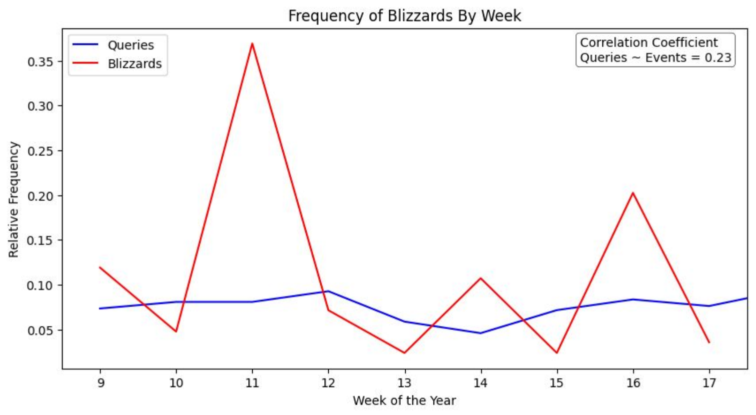


Question 3:
How does the frequency of searches change during different weather events?

Query

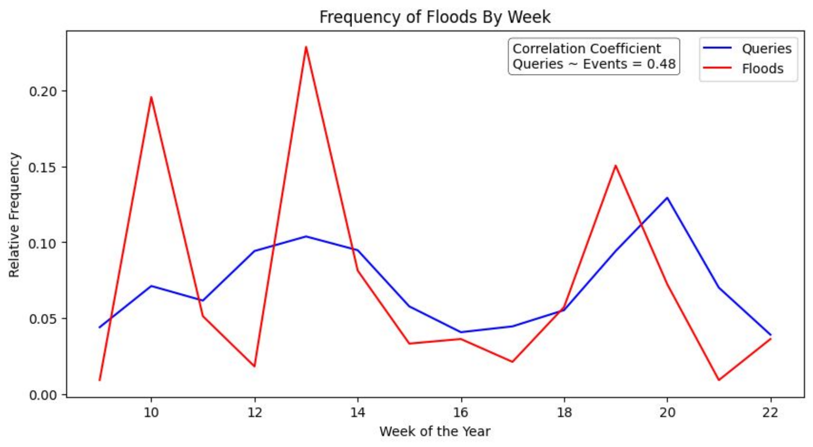
- See query before

Blizzards and Floods

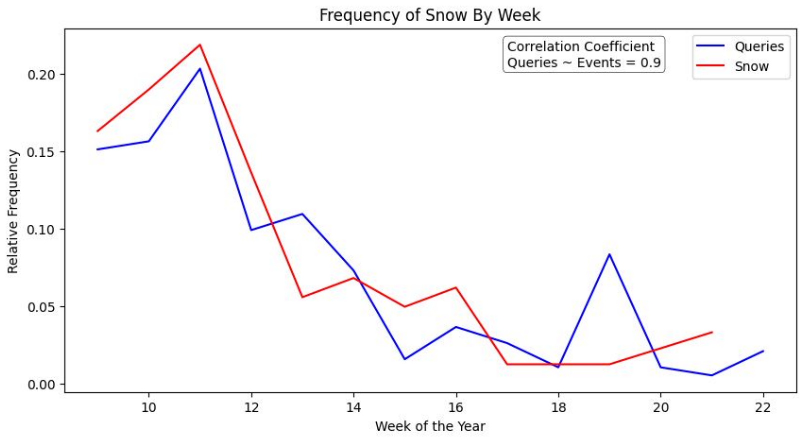


- Low correlation
- Dairy Queen had a Blizzard promotion
- Blizzard Entertainment games were very popular at the time (World of Warcraft)

- Clear correlation between Floods and searches for flooding

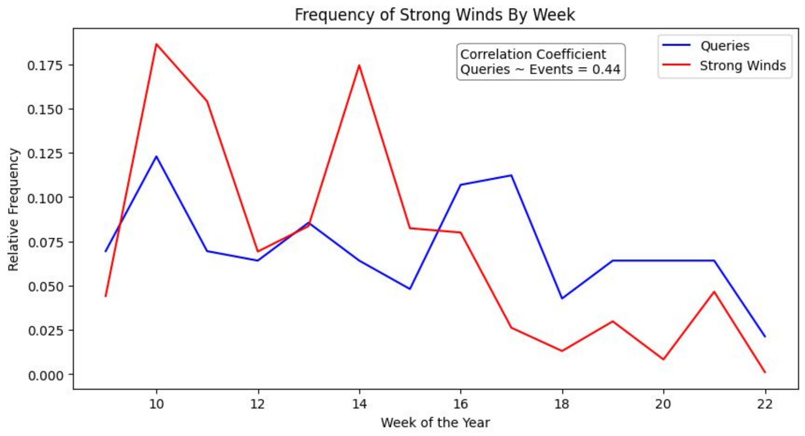


Snowfall and Strong Wind



❑ Strong correlation between queries for snow and actual snowy days

❑ Only a weak to medium relationship between strong wind and searches for "strong wind"



Digging Deeper

- ❑ We saw that tornadoes made up the majority of the severe events
- ❑ Does this correspond with weather related searches? If so, how?
- ❑ There was a series of devastating, newsworthy tornadoes during this period in the USA. (According to Wikipedia)

Queries

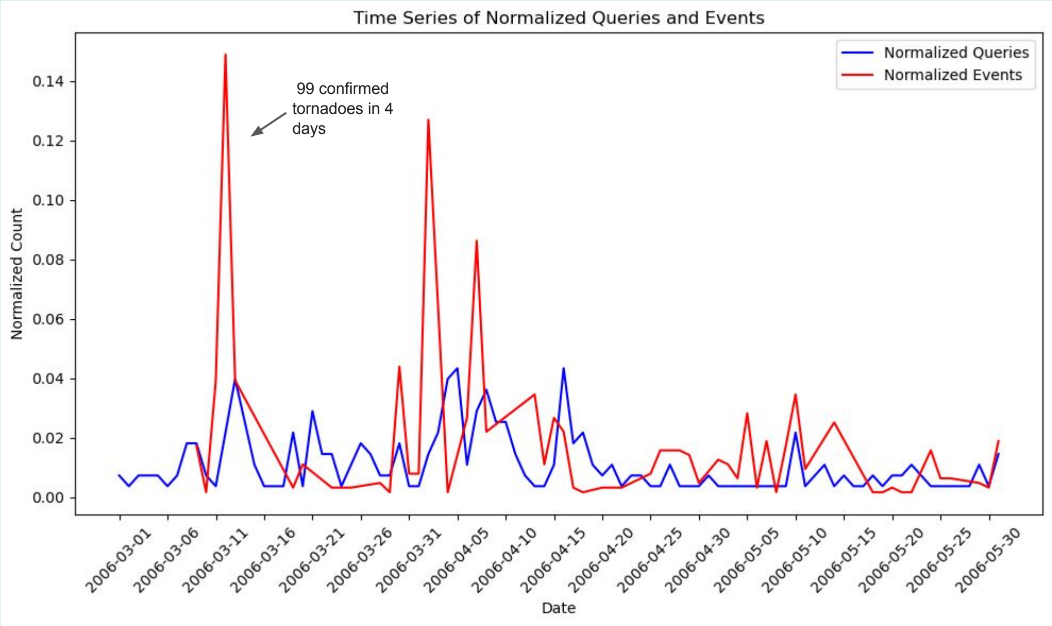
Query Frequency

```sql
WITH TopURLs AS (
    SELECT URLDIM.URL
    FROM AOL_SCHEMA.FACTS
    JOIN AOL_SCHEMA.URLDIM ON AOL_SCHEMA.FACTS.URLID = AOL_SCHEMA.URLDIM.ID
    WHERE AOL_SCHEMA.FACTS.CLICK = 1
    GROUP BY URLDIM.URL
    ORDER BY COUNT(AOL_SCHEMA.FACTS.CLICK) DESC
    LIMIT 20
),
FREQCOMP AS (
    SELECT
        FACTS.ANONID,
        QUERYDIM.QUERY,
        CAST(
            CONCAT(
                '2006-',
                LPAD(CASE
                    WHEN TIMEDIM."day of the year" BETWEEN 60 AND 90 THEN '03'
                    WHEN TIMEDIM."day of the year" BETWEEN 91 AND 120 THEN '04'
                    WHEN TIMEDIM."day of the year" BETWEEN 121 AND 151 THEN '05'
                    ELSE '01'
                END, 2, '0'), '-',
                LPAD(TIMEDIM."day of the month", 2, '0'), ' ',
                LPAD(TIMEDIM."hour", 2, '0'), ':',
                LPAD(TIMEDIM."minute", 2, '0'), ':',
                LPAD(TIMEDIM."second", 2, '0')
            ) AS TIMESTAMP
        ) AS time_as_datetime
    FROM
        AOL_SCHEMA.FACTS
        LEFT JOIN AOL_SCHEMA.TIMEDIM ON FACTS.TIMEID = TIMEDIM.ID
        LEFT JOIN AOL_SCHEMA.URLDIM ON FACTS.URLID = URLDIM.ID
        LEFT JOIN AOL_SCHEMA.QUERYDIM ON FACTS.QUERYID = QUERYDIM.ID
    WHERE FACTS.CLICK = 1
        AND (
            URLDIM.URL IN (SELECT URL FROM TopURLs)
            OR LOWER(URLDIM.URL) LIKE '%weather%'
        )
        AND FACTS.ANONID IS NOT NULL
        AND TIMEDIM."hour" IS NOT NULL
        AND TIMEDIM."minute" IS NOT NULL
        AND TIMEDIM."second" IS NOT NULL
        AND TIMEDIM."day of the year" IS NOT NULL
),
DateRange AS (
    SELECT DATE '2006-03-01' AS EVENT_DATE
    UNION ALL SELECT DATE '2006-03-02'
    UNION ALL SELECT DATE '2006-03-03'
    ...
    UNION ALL SELECT DATE '2006-05-31'
)
SELECT
    DateRange.EVENT_DATE AS query_date,
    COALESCE(COUNT(*), 0) AS number_of_queries
FROM
    DateRange
LEFT JOIN
    FREQCOMP E
ON
    CAST(E.time_as_datetime AS DATE) = DateRange.EVENT_DATE
AND LOWER(E.QUERY) LIKE '%tornado%'
GROUP BY
    DateRange.EVENT_DATE
ORDER BY
    query_date;
```

Tornado Frequency

```sql
WITH DateRange AS (
    SELECT DATE '2006-03-01' AS EVENT_DATE
    UNION ALL SELECT DATE '2006-03-02'
    ...
    UNION ALL SELECT DATE '2006-05-31'
)
SELECT
    DateRange.EVENT_DATE,
    COALESCE(COUNT(E.EPISODE_ID), 0) AS EVENT_COUNT
FROM
    DateRange
LEFT JOIN
    AOL_SCHEMA.WEATHER_EVENTS E
ON
    CAST(E.BEGIN_DATE_TIME AS DATE) = DateRange.EVENT_DATE
    AND E.EVENT_TYPE = 'Tornado'
GROUP BY
    DateRange.EVENT_DATE
ORDER BY
    DateRange.EVENT_DATE;
```

Searches for Tornados vs. actual Tornados



- There is at least a slight relationship between the occurrence of a tornado and the frequency of queries
- Queries counted from clicks on top 20 urls or weather sites, and containing *'tornado'* in query text
- Pearson's Correlation: ~0.29

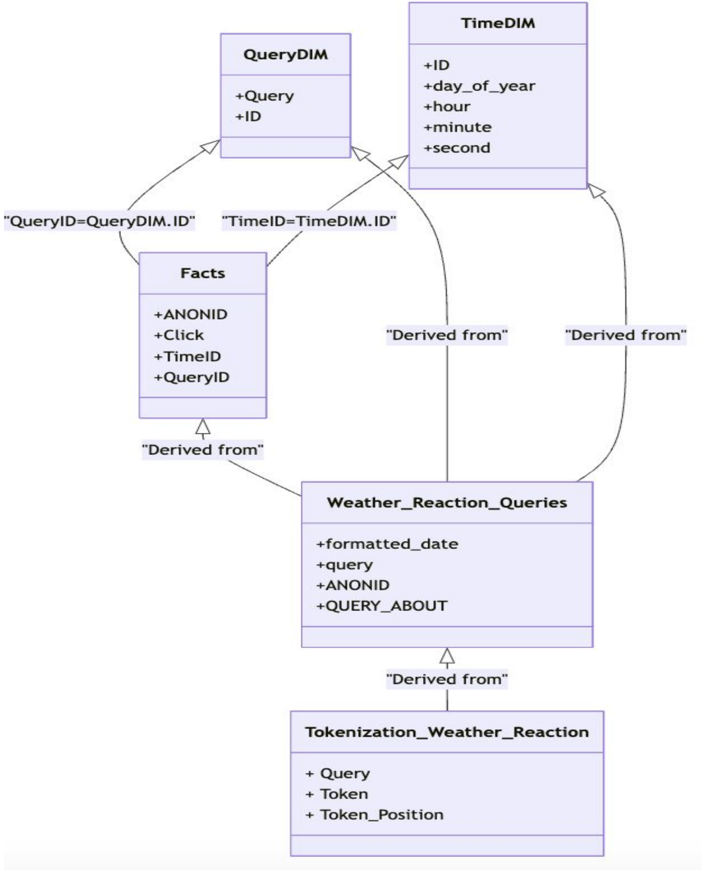
Conclusion

- Severity of weather events seems to be a driver of search engine activity
- Lagged correlation measures could help this analysis

Question 4:
What are people most concerned about during and after tornadoes, as shown in keywords?

Schema and Info

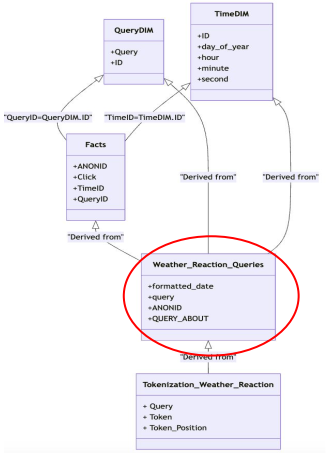
- ❑ Weather Tracking and Alerts - 39,091 queries
- ❑ Damage Assessment and Recovery - 15,631 queries
- ❑ Emergency Preparedness and Safety - 7,330 queries
- ❑ Relief and Support - 7,299 queries
- ❑ Shelter and Immediate Protection - 1,980 queries



Weather Queries Tables

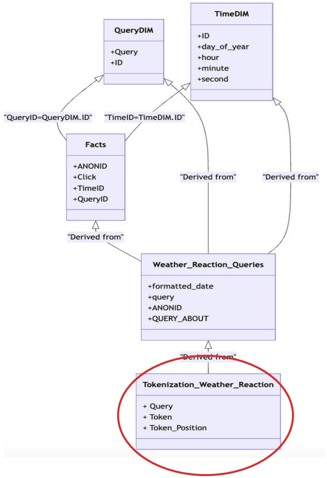
```sql
CREATE OR REPLACE TABLE AOL_SCHEMA.Weather_Reaction_Queries AS
WITH MonthLookup AS (
    SELECT 'january' AS month_name, '01' AS month_number
    UNION ALL
    SELECT 'february', '02'
    UNION ALL
    SELECT 'march', '03'
    UNION ALL
    SELECT 'april', '04'
    ...
    SELECT 'december', '12'
),
TrimmedQuery AS (
    SELECT
        TRIM(LOWER(QUERYDIM."QUERY")) AS query_lower
    FROM AOL_SCHEMA."QUERYDIM"
),
PreparedData AS (
    SELECT
        TIMEDIM."month",
        TIMEDIM."day of the month",
        TIMEDIM."hour",
        TIMEDIM."minute",
        QUERYDIM."QUERY",
        FACTS."ANONID",
        TO_TIMESTAMP(
            '2006-' ||
            COALESCE(MonthLookup.month_number, '00') ||
            '-' ||
            LPAD(CAST(TIMEDIM."day of the month" AS VARCHAR(10)), 2, '0') ||
            ' ' ||
            LPAD(CAST(TIMEDIM."hour" AS VARCHAR(2)), 2, '0') || ':' ||
            LPAD(CAST(TIMEDIM."minute" AS VARCHAR(2)), 2, '0')
        ) AS formatted_date,
        TRIM(LOWER(QUERYDIM."QUERY")) AS query_lower_trimmed
    FROM AOL_SCHEMA."TIMEDIM"
    INNER JOIN AOL_SCHEMA."QUERYDIM" ON QUERYDIM."ID" = TIMEDIM."ID"
    INNER JOIN AOL_SCHEMA."FACTS" ON FACTS."QUERYID" = QUERYDIM."ID"
    LEFT JOIN MonthLookup ON LOWER(TRIM(TIMEDIM."month")) = MonthLookup.month_name
    WHERE FACTS."CLICK" = 1
)
```

```sql
SELECT DISTINCT
    formatted_date,
    QUERY AS query_lower_trimmed,
    ANONID,
    CASE
        WHEN query_lower_trimmed LIKE '%shelter%'
            OR query_lower_trimmed LIKE '%safe%room%'
            OR query_lower_trimmed LIKE '%temporary%housing%'
            OR query_lower_trimmed LIKE '%housing%assistance%' THEN 'Shelter and Immediate Protection'

        WHEN query_lower_trimmed LIKE '%safety%'
            OR query_lower_trimmed LIKE '%safe%'
            ...
            OR query_lower_trimmed LIKE '%tracker%' THEN 'Emergency Preparedness and Safety'

        WHEN query_lower_trimmed LIKE '%damage%'
            OR query_lower_trimmed LIKE '%repair%'
            OR query_lower_trimmed LIKE '%insurance%' THEN 'Damage Assessment and Recovery'

        WHEN query_lower_trimmed LIKE '%relief%'
            OR query_lower_trimmed LIKE '%volunteer%'
            ...
            OR query_lower_trimmed LIKE '%charity%'  THEN 'Relief and Support'

        WHEN query_lower_trimmed LIKE '%weather%' OR
            query_lower_trimmed LIKE '%climate%' OR
            ...
            query_lower_trimmed LIKE '%weather%radar%' THEN 'Weather Tracking and Alerts'
        ELSE 'Other'
    END AS QUERY_ABOUT
FROM PreparedData
WHERE (
        query_lower_trimmed LIKE '%shelter%'
        OR query_lower_trimmed LIKE '%safety%'
        OR query_lower_trimmed LIKE '%safe%'
        OR query_lower_trimmed LIKE '%emergency%'
        OR query_lower_trimmed LIKE '%tornado%'
        OR query_lower_trimmed LIKE '%storm%'
        OR query_lower_trimmed LIKE '%damage%'
        OR query_lower_trimmed LIKE '%evacuation%'
        OR query_lower_trimmed LIKE '%survival%'
        OR query_lower_trimmed LIKE '%insurance%'
        OR query_lower_trimmed LIKE '%repair%'
        OR query_lower_trimmed LIKE '%fund%'
        OR query_lower_trimmed LIKE '%relief%'
        OR query_lower_trimmed LIKE '%warning%'
        OR query_lower_trimmed LIKE '%volunteer%'
        OR query_lower_trimmed LIKE '%donate%'
        OR query_lower_trimmed LIKE '%charity%'
        OR query_lower_trimmed LIKE '%forecast%'
        ...
        OR query_lower_trimmed LIKE '%frost%'
    )
```



Creating Tokenization Table

```sql
CREATE OR REPLACE TABLE AOL_SCHEMA.tokenization_weather_reaction AS
WITH limited_querydim AS (
    SELECT DISTINCT QUERY_LOWER_TRIMMED
    FROM AOL_SCHEMA.Weather_Reaction_Queries
    WHERE QUERY_LOWER_TRIMMED IS NOT NULL
      AND NOT (QUERY_LOWER_TRIMMED LIKE '%com' OR  QUERY_LOWER_TRIMMED LIKE '%net' OR  QUERY_LOWER_TRIMMED LIKE '%org' OR  QUERY_LOWER_TRIMMED  LIKE 'http%' OR
QUERY_LOWER_TRIMMED LIKE 'www%')
),
tokenized_query AS (
    SELECT
        QUERY_LOWER_TRIMMED AS QUERY,  -- Fix: Use QUERY_LOWER_TRIMMED consistently
        REGEXP_SUBSTR(
            REGEXP_REPLACE(QUERY_LOWER_TRIMMED, '[^[:alnum:] ]', ''),  -- Clean the query
            '[^0-9[:space:]]+',  -- Extract tokens
            1,
            LEVEL
        ) AS TOKEN,
        LEVEL AS TOKEN_POSITION
    FROM limited_querydim
    CONNECT BY PRIOR QUERY_LOWER_TRIMMED = QUERY_LOWER_TRIMMED
           AND LEVEL <= LENGTH(REGEXP_REPLACE(QUERY_LOWER_TRIMMED, '[^ ]+', '')) + 1  -- Number of words
           AND QUERY_LOWER_TRIMMED IS NOT NULL
           AND REGEXP_SUBSTR(
               REGEXP_REPLACE(QUERY_LOWER_TRIMMED, '[^[:alnum:] ]', ''),  -- Clean query string
               '[^0-9[:space:]]+',  -- Extract tokens
               1,
               LEVEL
           ) IS NOT NULL
)
SELECT
    QUERY,
    TOKEN,
    TOKEN_POSITION
FROM tokenized_query
WHERE TOKEN IS NOT NULL  -- Exclude NULL tokens
  AND TOKEN NOT IN ('the', 'and', 'are', 'is', 'in', 'to', 'for', 'on', 'of', 'or', 'no', 'what', 'with', 'http', 'com', 'how', 'www', 'you', 'our', 'from',
'las', 'all', 'new', 'who', 'where', 'when', 'why', 'whom', 'whose', 'which') -- Exclude common stopwords
  AND LENGTH(TOKEN) > 2  -- Exclude tokens shorter than 3 characters
ORDER BY QUERY, TOKEN_POSITION;
```



During and After Tornado Queries

During Tornado

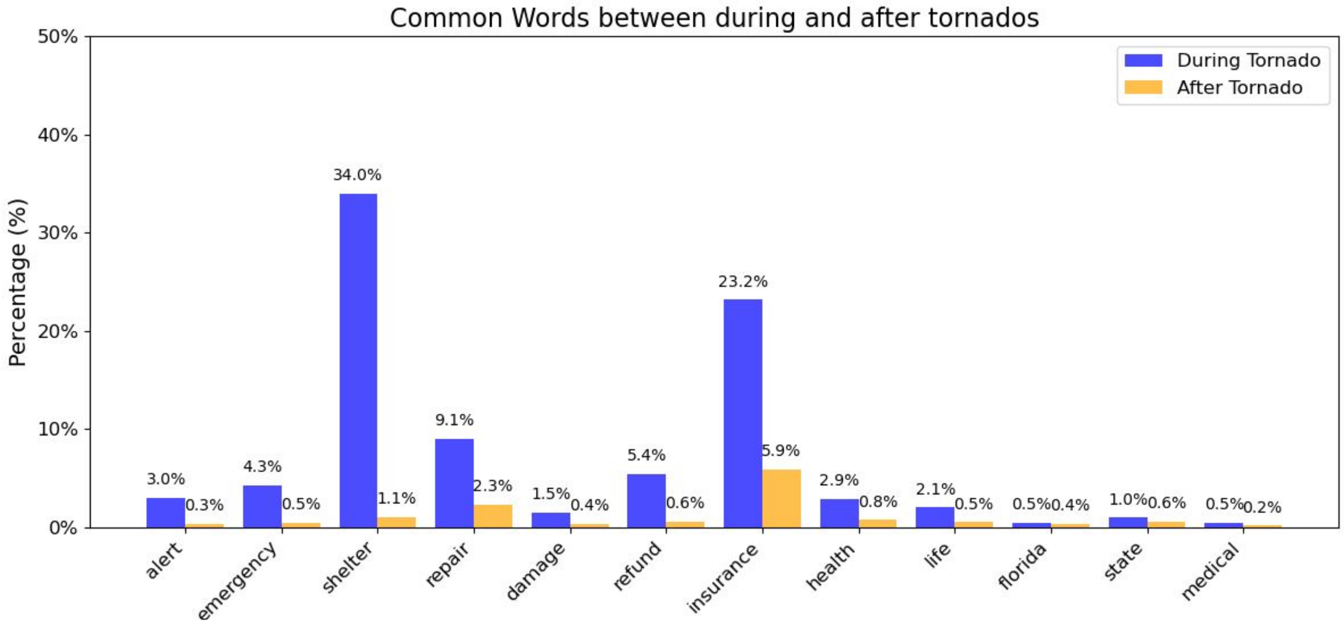
```
WITH Events_DATA AS (
    SELECT
        we.EVENT_TYPE,
        we.BEGIN_DATE_TIME,
        we.END_DATE_TIME
    FROM AOL_SCHEMA.WEATHER_EVENTS we
    WHERE we.BEGIN_DATE_TIME >= '2006-03-01 00:01:00.000000'
      AND LOWER(we.EVENT_TYPE) LIKE 'tornado'
),
Relevant_Queries AS (
    SELECT
        QUERIES_WEATHER.QUERY_LOWER_TRIMMED AS QUERY,
        QUERIES_WEATHER.FORMATTED_DATE,
        QUERIES_WEATHER.QUERY_ABOUT,
        we.EVENT_TYPE
    FROM Events_DATA we
    INNER JOIN AOL_SCHEMA.Weather_Reaction_Queries AS QUERIES_WEATHER
        ON QUERIES_WEATHER.FORMATTED_DATE BETWEEN we.BEGIN_DATE_TIME AND we.END_DATE_TIME
),
Grouped_Results AS (
    SELECT
        QUERY,
        QUERY_ABOUT,
        EVENT_TYPE,
        COUNT(*) AS query_count
    FROM Relevant_Queries
    GROUP BY GROUPING SETS (
        (QUERY, QUERY_ABOUT, EVENT_TYPE),
        (QUERY,EVENT_TYPE)
    )
),
Tokens_Queries AS (
    SELECT
        token.TOKEN,
        ex.QUERY,
        ex.QUERY_ABOUT,
        ex.EVENT_TYPE,
        ex.query_count
    FROM Grouped_Results ex
    INNER JOIN AOL_SCHEMA.tokenization_weather_reaction token
        ON ex.QUERY = token.QUERY
),
```

```
TOKEN_COUNT AS (
    SELECT
        TOKEN,
        COALESCE(QUERY_ABOUT, 'UNKNOWN') AS QUERY_ABOUT,
        SUM(query_count) AS token_count
    FROM Tokens_Queries
    GROUP BY TOKEN, QUERY_ABOUT
    HAVING SUM(query_count)>2
    ORDER BY token_count DESC
),
TOTAL_QUERY_COUNT AS (
    SELECT
        QUERY_ABOUT,
        SUM(token_count) AS total_query_count
    FROM TOKEN_COUNT
    GROUP BY QUERY_ABOUT
),
NORMALIZED_TOKEN_COUNT AS (
    SELECT
        tc.TOKEN,
        tc.QUERY_ABOUT,
        tc.token_count,
        tqc.total_query_count,
        CASE
            WHEN tc.QUERY_ABOUT = 'UNKNOWN' THEN
                (tc.token_count * 1.0 /
                 (SELECT SUM(token_count) FROM TOKEN_COUNT WHERE QUERY_ABOUT = 'UNKNOWN'))
            ELSE
                (tc.token_count * 1.0 / tqc.total_query_count)
        END AS Group_Specific_Token_Probability
    FROM TOKEN_COUNT tc
    JOIN TOTAL_QUERY_COUNT tqc
        ON tc.QUERY_ABOUT = tqc.QUERY_ABOUT
)
SELECT
    TOKEN,
    QUERY_ABOUT,
    token_count,
    total_query_count,
    Group_Specific_Token_Probability,
    Group_Specific_Token_Probability*100 AS Group_specific_Token_Percentage
FROM NORMALIZED_TOKEN_COUNT
ORDER BY QUERY_ABOUT,Group_Specific_Token_Probability DESC;
```

After Tornado

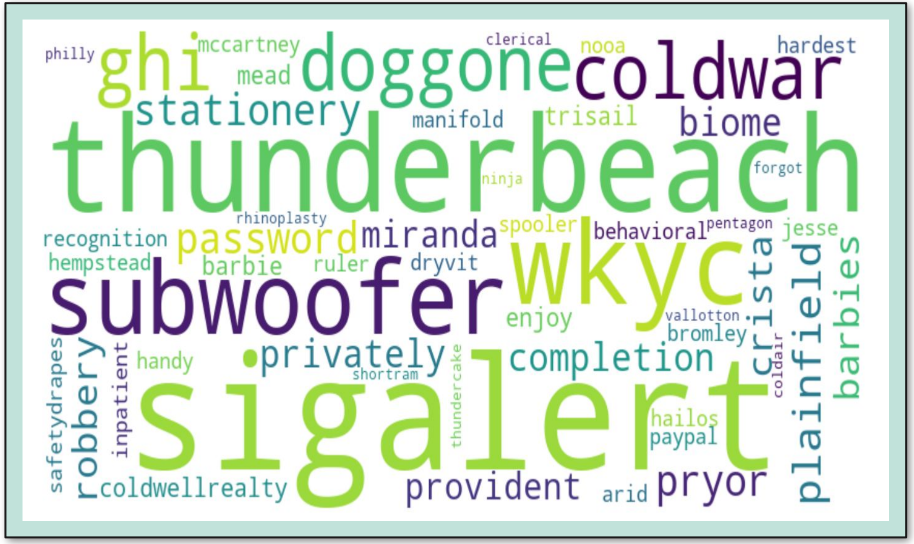
```
WITH Ordered_Events AS (
    SELECT
        EVENT_TYPE,
        REGION,
        BEGIN_DATE_TIME,
        END_DATE_TIME,
        LEAD(BEGIN_DATE_TIME) OVER (ORDER BY BEGIN_DATE_TIME) AS NEXT_END_DATE_TIME
    FROM AOL_SCHEMA.WEATHER_EVENTS
    WHERE EVENT_TYPE = 'Tornado'
    ORDER BY BEGIN_DATE_TIME
),
Calculated_Events AS (
    SELECT
        EVENT_TYPE,
        BEGIN_DATE_TIME,
        END_DATE_TIME,
        ADD_MINUTES(TO_TIMESTAMP(SUBSTR(END_DATE_TIME, 1, 19)), 1) AS WINDOW_START,
        ADD_MINUTES(TO_TIMESTAMP(SUBSTR(NEXT_END_DATE_TIME, 1, 19)), -1) AS WINDOW_END
    FROM Ordered_Events
),
Window_Events AS (
    SELECT
        EVENT_TYPE,
        BEGIN_DATE_TIME,
        END_DATE_TIME,
        WINDOW_START,
        WINDOW_END
    FROM Calculated_Events
    WHERE SECONDS_BETWEEN(WINDOW_END, WINDOW_START) >= 0
),
Relevant_Queries AS (
    SELECT
        qt.QUERY_LOWER_TRIMMED AS QUERY,
        qt.QUERY_ABOUT,
        qt.FORMATTED_DATE,
        tq.EVENT_TYPE,
        tq.BEGIN_DATE_TIME,
        tq.END_DATE_TIME,
        tq.WINDOW_START,
        tq.WINDOW_END
    FROM Window_Events tq
    INNER JOIN AOL_SCHEMA.Weather_Reaction_Queries qt
        ON qt.FORMATTED_DATE BETWEEN tq.WINDOW_START AND tq.WINDOW_END
),
```

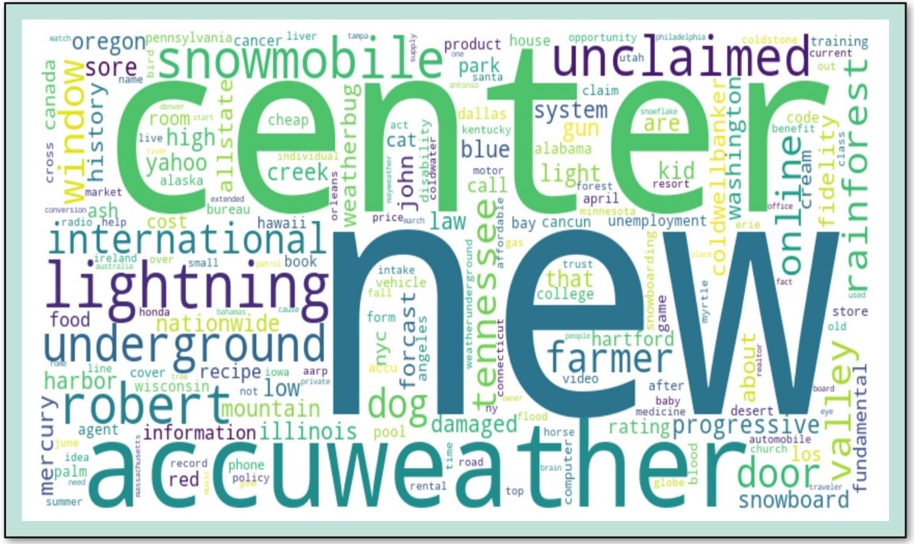
Commonly Searched Keywords during and after Tornadoes



- **During Tornado,** the focus is on immediate response, emergency services, and damage repair, with terms like **shelter, repair, insurance, and alert.**

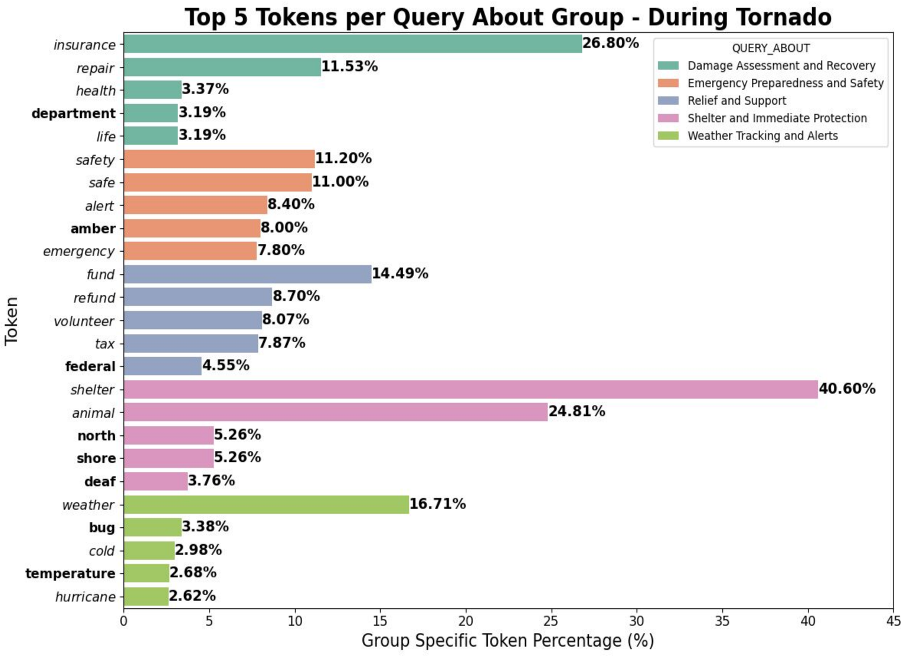
Distinct Keywords during and after Tornadoes

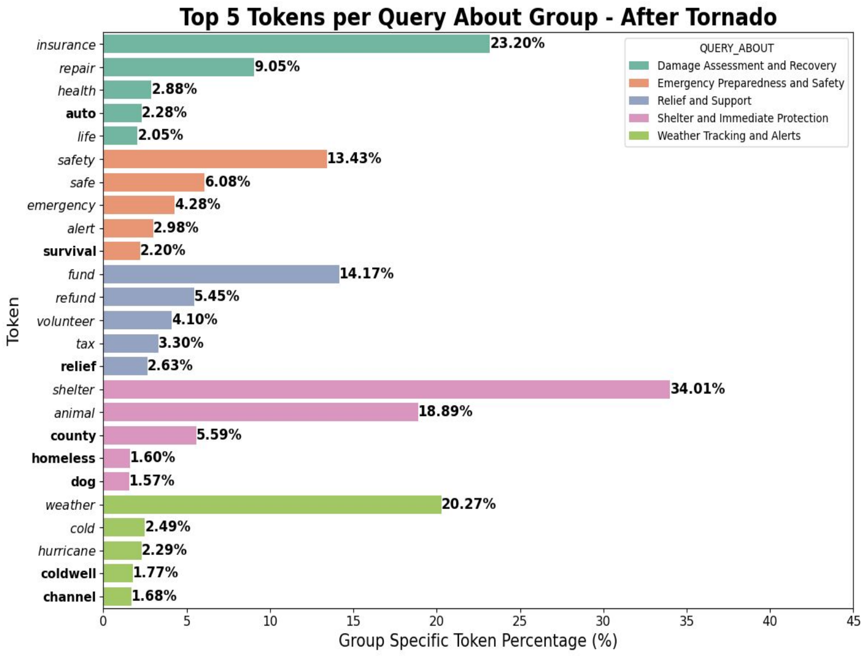




- **During Tornado**: Focuses on immediate responses with terms like "sigalert" and "wkyc" highlighting alerts and local events.
- **After Tornado**: Focuses on recovery with terms like "accuweather" for weather monitoring

Relevant Keywords before and after Tornadoes





Conclusion

- Terms like **insurance, safety, shelter** are searched for more during tornadoes which might indicate an enhanced focus on **security**
- Users search for **alert related sites** more during tornadoes than after tornadoes

Question 5:
Do users behave more anxiously (click faster) during disasters*?

*Specifically during tornadoes

Interclick Times

Theory: People will click on the next link faster during Tornadoes

Approach: Create a new measure as the time between clicking on links for each user

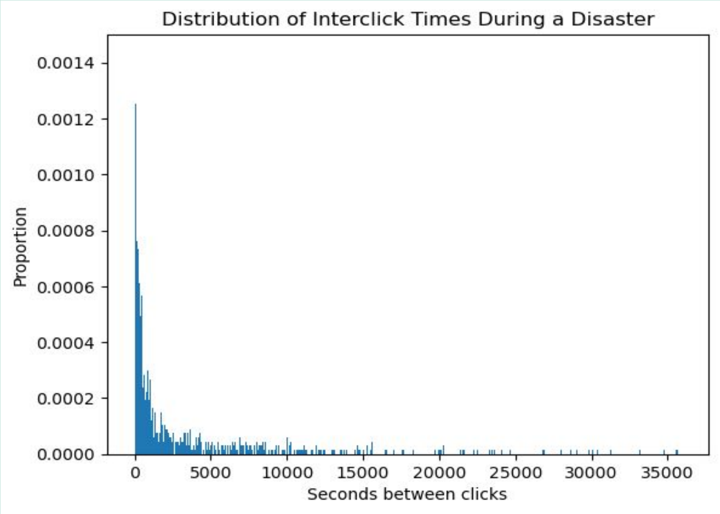
Creating a Table & Base Query

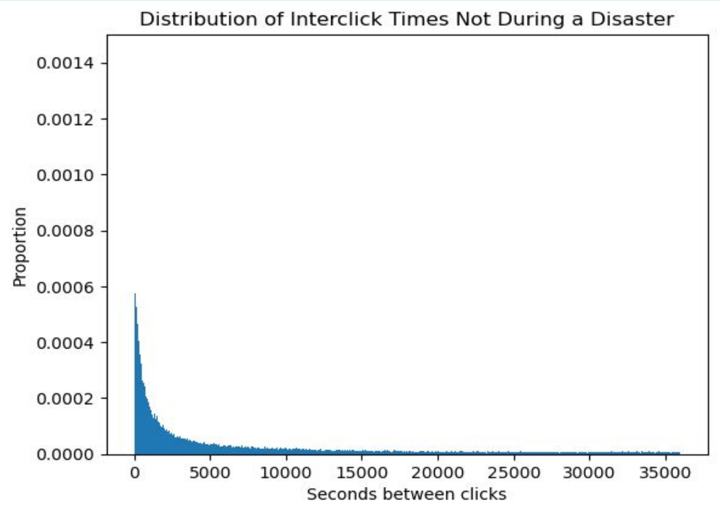
```
CREATE TABLE AOL_SCHEMA.INTERARRIVAL TIMES AS
    SELECT
        FACTS.ANONID,
        CAST(
        CONCAT(
            '2006-'
            LPAD (CASE
                WHEN TIMEDIM.[day of the year] BETWEEN 60 AND 90 THEN '03'
                WHEN TIMEDIM.[day of the year] BETWEEN 91 AND 120 THEN '04'
                WHEN TIMEDIM.[day of the year] BETWEEN 121 AND 151 THEN '05'
                ELSE '01'
            END, 2, '0'), '-'
            LPAD(TIMEDIM.[day of the month], 2, '0'), ' ',
            LPAD(TIMEDIM.[hour], 2, '0'), ':',
            LPAD(TIMEDIM.[minute], 2, '0'), ':',
            LPAD(TIMEDIM.[second], 2, '0')
        ) AS TIMESTAMP) AS time_as_datetime
    FROM
        AOL_SCHEMA.FACTS LEFT JOIN AOL_SCHEMA.TIMEDIM ON FACTS.TIMEID = TIMEDIM.ID
        LEFT JOIN AOL_SCHEMA.URLDIM ON FACTS.URLID = URLDIM.ID
    WHERE FACTS.CLICK = 1
        AND URLDIM.URL IN (
            SELECT URLDIM.URL
            FROM AOL SCHEMA.FACTS
            JOIN AOL SCHEMA.URLDIM ON AOL_SCHEMA.FACTS.URLID = AOL_SCHEMA.URLDIM. ID
            WHERE AOL_SCHEMA.FACTS.CLICK = 1
            GROUP BY URLDIM.URL
            ORDER BY COUNT(AOL_SCHEMA.FACTS.CLICK) DESC
            LIMIT 20
        )
    AND FACTS.ANONID IS NOT NULL
    AND TIMEDIM.[hour] IS NOT NULL
    AND TIMEDIM.[minute] IS NOT NULL
    AND TIMEDIM.[second] IS NOT NULL
    AND TIMEDIM.[day of the year] IS NOT NULL
```

```
SELECT
    T1.ANONID,
    T1.TIME_AS_DATETIME,
    COALESCE(LAG(TIME_AS_DATETIME) OVER (PARTITION BY ANONID ORDER BY TIME_AS_DATETIME), TIME_AS_DATETIME) AS LaggedDateTime,
    SECONDS BETWEEN(T1.TIME_AS_DATETIME, COALESCE(LAG(TIME_AS_DATETIME) OVER (PARTITION BY ANONID ORDER BY TIME_AS_DATETIME), TIME_AS_DATETIME)) as
Seconds_Difference,
    MINUTES BETWEEN(T1.TIME_AS_DATETIME, COALESCE(LAG(TIME_AS_DATETIME) OVER (PARTITION BY ANONID ORDER BY TIME_AS_DATETIME), TIME_AS_DATETIME)) as
Minutes_Difference
FROM
    AOL_SCHEMA.INTERARRIVAL_TIMES as T1
WHERE
    T1.ANONID IN (
        SELECT T2.ANONID
        FROM AOL_SCHEMA. INTERARRIVAL_TIMES as T2
        GROUP BY T2.ANONID
        HAVING COUNT(T2.ANONID) >- 10
    )
    AND EXISTS(
        SELECT 1
        FROM AOL_SCHEMA.WEATHER_EVENTS as T3
        WHERE
        (T1.TIME_AS_DATETIME BETWEEN T3.BEGIN_DATE_TIME AND T3.END_DATE_TIME)
        AND (T3.EVENT_TYPE = 'Tornado')
    )
ORDER BY
    T1.ANONID,
    T1.TIME_AS_DATETIME
```

- The query to obtain the interclick times not occurring during a tornado is only modified in the exists statement where clause.

Times between Clicks





- ❏ Higher spike for short interclick times during tornadoes
- ❏ Visual comparison is insufficient ⇒ Statistical test

Statistical Test

We use a **T-Test** assuming unequal variances with a **significance level** of **0.05**

H_0: There is **no difference in mean interclick time** between those occurring during and those not occurring during a natural disaster.

H_1: There **is a difference in mean interclick time** between those occurring during and those not occurring during a natural disaster.

T-Test Results:
Test Statistic: -44.5
P-Value: 0.0000*

*Below Numerical Precision

Conclusion

- P-value < 0.05 ⇒ **reject H_0**
- The data suggests there is a **statistically significant difference** in mean time between clicks when there is a tornado compared to when there is none

Additional Insight:

- The number of clicks in a period of time could be modeled as a Poisson process
- This would allow you to construct data-driven query engine traffic simulations

Summary

Summary of Findings

1. Damages were concentrated in single states
2. The majority of weather events: Not severe and cold or wind related
3. Search trends for weather events followed actual events, especially severe ones
4. Searches for keywords such as *'Shelter'* or *'Insurance'* see a significant increase during tornadoes.
5. Users clicked on links faster on average during tornadoes

Thank you for your attention!

Question (Bonus):
Which domains were most clicked during weather events that occurred in different regions?

Query

Most clicked domains or URLs and matched by time to specific weather events and regions where they occurred

```
WITH ClickedDomains AS (
    SELECT
        AOL_SCHEMA.WEATHER_EVENTS.REGION AS REGION,
        AOL_SCHEMA.WEATHER_EVENTS.EVENT_TYPE AS EVENT_TYPE,
        AOL_SCHEMA.WEATHER_EVENTS.BEGIN_DATE_TIME AS BEGIN_DATE_TIME,
        AOL_SCHEMA.URLDIM.THISDOMAIN AS THISDOMAIN,
        AOL_SCHEMA.URLDIM.URL AS URL,
        COUNT(AOL_SCHEMA.FACTS.URLID) AS CLICK_COUNT
    FROM
        AOL_SCHEMA.WEATHER_EVENTS
    JOIN
        AOL_SCHEMA.FACTS
        ON AOL_SCHEMA.WEATHER_EVENTS.BEGIN_DAY = AOL_SCHEMA.FACTS.TIMEID
    JOIN
        AOL_SCHEMA.URLDIM
        ON AOL_SCHEMA.FACTS.URLID = AOL_SCHEMA.URLDIM.ID
    WHERE
        AOL_SCHEMA.FACTS.CLICK = TRUE
        AND (AOL_SCHEMA.URLDIM.THISDOMAIN IS NOT NULL OR AOL_SCHEMA.URLDIM.URL IS NOT NULL)
        AND AOL_SCHEMA.WEATHER_EVENTS.BEGIN_DATE_TIME BETWEEN '2006-03-01 00:00:00' AND '2006-05-31 23:59:59'
    GROUP BY ROLLUP(
        AOL_SCHEMA.WEATHER_EVENTS.REGION,
        AOL_SCHEMA.WEATHER_EVENTS.EVENT_TYPE,
        AOL_SCHEMA.WEATHER_EVENTS.BEGIN_DATE_TIME,
        AOL_SCHEMA.URLDIM.THISDOMAIN,
        AOL_SCHEMA.URLDIM.URL
    )
),
RankedDomains AS (
    SELECT
        REGION,
        EVENT_TYPE,
        BEGIN_DATE_TIME,
        THISDOMAIN,
        URL,
        CLICK_COUNT,
        ROW_NUMBER() OVER (
            PARTITION BY REGION, EVENT_TYPE
            ORDER BY CLICK_COUNT DESC
        ) AS RANK
    FROM
        ClickedDomains
    WHERE
        THISDOMAIN IS NOT NULL OR URL IS NOT NULL
)
SELECT
    REGION,
    EVENT_TYPE,
    BEGIN_DATE_TIME,
    THISDOMAIN,
    URL,
    CLICK_COUNT
FROM
    RankedDomains
WHERE
    RANK = 1
ORDER BY
    REGION,
    EVENT_TYPE,
    BEGIN_DATE_TIME;
```

Result

	REGION	EVENT_TYPE	BEGIN_DATE_TIME	THISDOMAIN	URL	CLICK_COUNT
0	Alabama	Flash Flood	2006-03-20 18:45:00.000000	nau	http://www.nau.edu	3
1	Alabama	Funnel Cloud	2006-03-20 17:55:00.000000	nau	http://www.nau.edu	3
2	Alabama	Hail	2006-04-20 17:08:00.000000	nau	http://www.nau.edu	6
3	Alabama	Lightning	2006-04-18 18:10:00.000000	citysearch	http://pittsburgh.citysearch.com	1
4	Alabama	Strong Wind	2006-03-09 14:15:00.000000	ca	http://gocalif.ca.gov	1
...	...	...	...	...	...	...
487	Wyoming	Heavy Snow	2006-05-09 04:00:00.000000	ca	http://gocalif.ca.gov	2
488	Wyoming	Lightning	2006-05-08 14:10:00.000000	ebay.co	http://www.ebay.co.uk	1
489	Wyoming	Thunderstorm Wind	2006-05-26 16:39:00.000000	yahoo	http://mail.yahoo.com	1
490	Wyoming	Wildfire	2006-04-10 09:00:00.000000	bilkent.edu	http://web.bilkent.edu.tr	1
491	Wyoming	Winter Storm	2006-04-24 01:00:00.000000	sharesong	http://www.sharesong.org	4

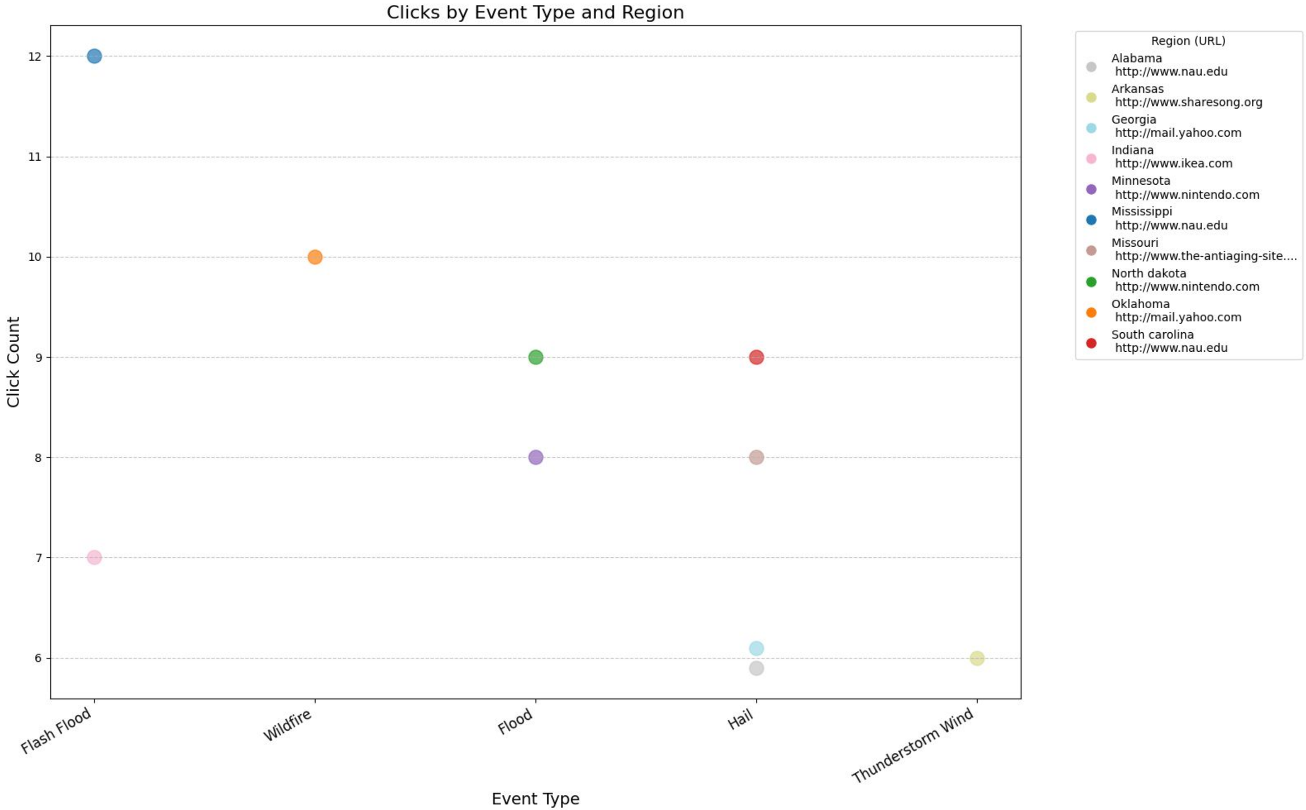
492 rows × 6 columns

Query

Top 10 most clicked URLs during severe weather events

```
WITH ClickedDomains AS (
    SELECT
        AOL_SCHEMA.WEATHER_EVENTS.REGION AS REGION,
        AOL_SCHEMA.WEATHER_EVENTS.EVENT_TYPE AS EVENT_TYPE,
        AOL_SCHEMA.WEATHER_EVENTS.BEGIN_DATE_TIME AS BEGIN_DATE_TIME,
        AOL_SCHEMA.URLDIM.THISDOMAIN AS THISDOMAIN,
        AOL_SCHEMA.URLDIM.URL AS URL,
        COUNT(AOL_SCHEMA.FACTS.URLID) AS CLICK_COUNT
    FROM
        AOL_SCHEMA.WEATHER_EVENTS
    JOIN
        AOL_SCHEMA.FACTS
        ON AOL_SCHEMA.WEATHER_EVENTS.BEGIN_DAY = AOL_SCHEMA.FACTS.TIMEID
    JOIN
        AOL_SCHEMA.URLDIM
        ON AOL_SCHEMA.FACTS.URLID = AOL_SCHEMA.URLDIM.ID
    WHERE
        AOL_SCHEMA.FACTS.CLICK = TRUE
        AND (AOL_SCHEMA.URLDIM.THISDOMAIN IS NOT NULL OR AOL_SCHEMA.URLDIM.URL IS NOT NULL)
        AND AOL_SCHEMA.WEATHER_EVENTS.BEGIN_DATE_TIME BETWEEN '2006-03-01 00:00:00' AND '2006-05-31 23:59:59'
    GROUP BY
        AOL_SCHEMA.WEATHER_EVENTS.REGION,
        AOL_SCHEMA.WEATHER_EVENTS.EVENT_TYPE,
        AOL_SCHEMA.WEATHER_EVENTS.BEGIN_DATE_TIME,
        AOL_SCHEMA.URLDIM.THISDOMAIN,
        AOL_SCHEMA.URLDIM.URL
),
RankedDomains AS (
    SELECT
        REGION,
        EVENT_TYPE,
        BEGIN_DATE_TIME,
        THISDOMAIN,
        URL,
        CLICK_COUNT,
        ROW_NUMBER() OVER (
            PARTITION BY REGION
            ORDER BY CLICK_COUNT DESC, BEGIN_DATE_TIME ASC
        ) AS RANK
    FROM
        ClickedDomains
)
SELECT
    REGION,
    EVENT_TYPE,
    BEGIN_DATE_TIME,
    THISDOMAIN,
    URL,
    CLICK_COUNT
FROM
    RankedDomains
WHERE
    RANK = 1
ORDER BY CLICK_COUNT DESC
LIMIT 10;
```

Visualization



Conclusion

During severe weather events, people are staying at home and use mostly entertainment or education websites

Specific weather events (like Wildfire or Flash Flood) are correlated with spikes in clicks

Appendix

Diving into the queries: A Closer Look

This section provides an in-depth analysis of how SQL queries were crafted to answer key questions using ROLAP techniques.

"Ever wondered how SQL queries bring multidimensional data to life?"

Query 1

```sql
WITH NewWeatherData AS (
    SELECT
        MONTH(BEGIN_DATE_TIME) AS BEGIN_MON,
        REGION,
        CASE
            -- West Region
            WHEN STATE_FIPS IN ('02', '04', '06', '08', '15', '16', '30', '32', '35', '41', '49', '53', '56') THEN 'West'
            -- Central Region
            WHEN STATE_FIPS IN ('05', '17', '18', '19', '20', '26', '27', '29', '31', '38', '39', '46', '48', '55') THEN 'Central'
            -- East Region
            WHEN STATE_FIPS IN ('01', '09', '10', '11', '12', '13', '21', '22', '23', '24', '25', '28', '33', '34', '36', '37', '40', '42', '44', '45', '47', '50', '51', '54') THEN 'East'
            ELSE 'Unknown'
        END AS TRISECTION,
        COALESCE(INJURIES_DIRECT, 0) + COALESCE(INJURIES_INDIRECT, 0) + COALESCE(DEATHS_DIRECT, 0) + COALESCE(DEATHS_INDIRECT, 0) AS HUMAN_DAMAGE,
        COALESCE(DAMAGE_PROPERTY, 0) + COALESCE(DAMAGE_CROPS, 0) AS NON_HUMAN_DAMAGE
    FROM
        AOL_SCHEMA.WEATHER_EVENTS
    WHERE
        MONTH(BEGIN_DATE_TIME) >= 3.0
)
SELECT
    BEGIN_MON,
    REGION,
    TRISECTION,
    SUM(HUMAN_DAMAGE) AS TOTAL_HUMAN_DAMAGE,
    SUM(NON_HUMAN_DAMAGE) AS TOTAL_NON_HUMAN_DAMAGE
FROM
    NewWeatherData
GROUP BY
    CUBE(BEGIN_MON, REGION, TRISECTION)
HAVING
    SUM(HUMAN_DAMAGE) > 0
    AND SUM(NON_HUMAN_DAMAGE) > 0
ORDER BY
    BEGIN_MON,
    TRISECTION,
    REGION;
```

Key operations utilized: Cube

Working of the query: Uses the CUBE operation to aggregate property and personnel damage by states, regions, and the starting months of natural disasters.

Output:

	BEGIN_MON	REGION	TRISECTION	TOTAL_HUMAN_DAMAGE	TOTAL_NON_HUMAN_DAMAGE
0	3.0	Arkansas	Central	19	175000
1	3.0	Illinois	Central	39	300000
2	3.0	Indiana	Central	3	163000
3	3.0	Iowa	Central	3	220000
4	3.0	Kansas	Central	5	296000
...	...	...	...	...	...
257	NaN	Virginia	NaN	3	215000
258	NaN	Washington	NaN	3	49000
259	NaN	West virginia	NaN	3	104000
260	NaN	Wisconsin	NaN	7	403000
261	NaN	NaN	NaN	1173	549989300

Questions (⅔)$_a$

Key Operators:

- ROLLUP
- RANK
- PARTITION BY

```
WITH SEVERITY_TABLE AS(
    SELECT
    EVENT_ID,
    EVENT_TYPE,
    BEGIN_DATE_TIME,
    CASE
        WHEN WEATHER_EVENTS.EVENT_TYPE IN ('Blizzard', 'Tornado', 'Wildfire', 'Avalanche', 'Funnel Cloud', 'Waterspout',
'Debris Flow') THEN 'Severe'
        WHEN WEATHER_EVENTS.EVENT_TYPE IN ('Coastal Flood', 'Flash Flood', 'Flood', 'Drought', 'Dust Devil', 'Dust Storm',
'Storm Surge/Tide', 'Ice Storm', 'Winter Storm') THEN 'Somewhat Severe'
        WHEN WEATHER_EVENTS.EVENT_TYPE IN ('Cold/Wind Chill', 'Frost/Freeze', 'Heat', 'Heavy Rain', 'Heavy Snow', 'High
Wind', 'Strong Wind', 'Thunderstorm Wind', 'Winter Weather', 'Lightning', 'Marine High Wind', 'Marine Thunderstorm Wind',
'Sleet', 'WINTER WEATHER', 'High Surf', 'Marine Hail', 'Rip Current', 'Lake-Effect Snow', 'Dense Fog') THEN 'Not Severe'
        ELSE 'Unclassified'
    END AS SEVERITY
FROM AOL_SCHEMA.WEATHER_EVENTS
),
AGG_EVENTS AS(
    SELECT
        SEVERITY,
        EVENT_TYPE,
        COALESCE(COUNT(EVENT_ID),0) AS FREQ
    FROM SEVERITY_TABLE
    GROUP BY ROLLUP(SEVERITY, EVENT_TYPE)
SELECT
    SEVERITY,
    EVENT_TYPE,
    FREQ,
    RANK() OVER(PARTITION BY SEVERITY ORDER BY FREQ DESC) as RANKING
FROM AGG_EVENTS
;
```

Output

The total amount for each EVENT_TYPE is aggregated. Then hierarchically grouped by SEVERITY. The EVENT_TYPEs in each SEVERITY grouping are then ranked from the most to least frequent events.

	SEVERITY	EVENT_TYPE	FREQ	RANKING
0	Not Severe	NaN	20832	1
1	Not Severe	Hail	9843	2
2	Not Severe	Thunderstorm Wind	4844	3
3	Not Severe	High Wind	1498	4
4	Not Severe	Heavy Snow	999	5
5	Not Severe	Strong Wind	738	6
6	Not Severe	Winter Weather	647	7
7	Not Severe	WINTER WEATHER	421	8
8	Not Severe	Marine Thunderstorm Wind	344	9
9	Not Severe	Cold/Wind Chill	339	10
10	Not Severe	Dense Fog	265	11
11	Not Severe	Heavy Rain	258	12
12	Not Severe	Lightning	235	13
13	Not Severe	High Surf	157	14
14	Not Severe	Frost/Freeze	125	15
15	Not Severe	Heat	66	16
16	Not Severe	Lake-Effect Snow	21	17
17	Not Severe	Marine Hail	20	18
18	Not Severe	Marine High Wind	6	19
19	Not Severe	Rip Current	5	20
20	Not Severe	Sleet	1	21

21	Severe	NaN	1326	1
22	Severe	Tornado	705	2
23	Severe	Wildfire	191	3
24	Severe	Funnel Cloud	166	4
25	Severe	Blizzard	139	5
26	Severe	Waterspout	52	6
27	Severe	Avalanche	41	7
28	Severe	Debris Flow	32	8
29	Somewhat Severe	NaN	3576	1
30	Somewhat Severe	Drought	1213	2
31	Somewhat Severe	Winter Storm	1097	3
32	Somewhat Severe	Flash Flood	599	4
33	Somewhat Severe	Flood	516	5
34	Somewhat Severe	Coastal Flood	67	6
35	Somewhat Severe	Ice Storm	48	7
36	Somewhat Severe	Dust Storm	19	8
37	Somewhat Severe	Storm Surge/Tide	15	9
38	Somewhat Severe	Dust Devil	2	10
39	NaN	NaN	25734	1

Questions (⅔)$_b$

Key Operators:

- **ROLLUP**

```sql
WITH DATE_RANGE AS (
    SELECT DATE '2006-03-01' AS EVENT_DATE
    UNION ALL SELECT DATE '2006-03-02'
    .
    .
    UNION ALL SELECT DATE '2006-05-31'
),
SEVERITY_TABLE AS(
SELECT
    EVENT_TYPE,
    EVENT_ID,
    CAST(WEATHER_EVENTS.BEGIN_DATE_TIME AS DATE) AS BEGIN_DATE,
    WEEK(WEATHER_EVENTS.BEGIN_DATE_TIME) AS BEGIN_WEEK,
    (MOD(CAST(CAST(WEATHER_EVENTS.BEGIN_DATE_TIME AS DATE) - CAST('2006-01-01' AS DATE) AS INTEGER) + 6, 7) + 1) AS WEEKDAY,
    CASE
        WHEN WEATHER_EVENTS.EVENT_TYPE IN ('Blizzard', 'Tornado', 'Wildfire', 'Avalanche', 'Funnel Cloud', 'Waterspout') THEN 'Severe'

        WHEN WEATHER_EVENTS.EVENT_TYPE IN ('Coastal Flood', 'Flash Flood', 'Flood', 'Drought', 'Dust, Devil',
                                           'Dust Storm', 'Storm Surge/Tide') THEN 'Somewhat Severe'

        WHEN WEATHER_EVENTS.EVENT_TYPE IN ('Cold/Wind Chill', 'Frost/Freeze', 'Heat', 'Heavy Rain', 'Heavy Snow', 'Hail','High Wind', 'Strong Wind', 'Thunderstorm Wind',
                                           'Winter Weather', 'Lightning', 'Marine High Wind','Marine Thunderstorm Wind', 'Sleet', 'WINTER WEATHER', '
                                           Winter Storm') THEN 'Not Severe'
        ELSE 'Unclassified'
    END AS SEVERITY
FROM AOL_SCHEMA.WEATHER_EVENTS
)

SELECT
    DATE_RANGE.EVENT_DATE,
    SEVERITY_TABLE.BEGIN_WEEK,
    SEVERITY_TABLE.WEEKDAY,
    SEVERITY_TABLE.SEVERITY,
    SEVERITY_TABLE.EVENT_TYPE,
    COALESCE(COUNT(SEVERITY_TABLE.EVENT_ID), 0) AS FREQ
FROM DATE_RANGE

LEFT JOIN SEVERITY_TABLE
ON DATE_RANGE.EVENT_DATE = SEVERITY_TABLE.BEGIN_DATE

GROUP BY ROLLUP((DATE_RANGE.EVENT_DATE, SEVERITY_TABLE.BEGIN_WEEK, SEVERITY_TABLE.WEEKDAY, SEVERITY_TABLE.SEVERITY),\
                (DATE_RANGE.EVENT_DATE, SEVERITY_TABLE.BEGIN_WEEK, SEVERITY_TABLE.WEEKDAY, SEVERITY_TABLE.EVENT_TYPE))

ORDER BY DATE_RANGE.EVENT_DATE ASC;
```

Output

Again, the total occurrences of each EVENT_TYPE is aggregated then hierarchically grouped by severity. Additionally, each data point has the date, week, and day of the week that the event occurred on. This allowed us to analyze all EVENT_TYPEs and SEVERITY groups by any relevant time dimension.

	EVENT_DATE	BEGIN_WEEK	WEEKDAY	SEVERITY	EVENT_TYPE	FREQ
0	2006-03-01	9.0	3.0	Not Severe	Heavy Snow	20
1	2006-03-01	9.0	3.0	Not Severe	Winter Weather	5
2	2006-03-01	9.0	3.0	Not Severe	High Wind	16
3	2006-03-01	9.0	3.0	Not Severe	Strong Wind	1
4	2006-03-01	9.0	3.0	Not Severe	Cold/Wind Chill	3
...	...	...	...	...	...	...
1262	2006-05-31	22.0	3.0	Not Severe	NaN	149
1263	2006-05-31	22.0	3.0	Somewhat Severe	NaN	22
1264	2006-05-31	22.0	3.0	Severe	NaN	14
1265	2006-05-31	22.0	3.0	Unclassified	NaN	95
1266	NaN	NaN	NaN	NaN	NaN	19545

Questions (⅔)$_b$

This SQL query was used to help generate the graphs on slides **23, 26,** and **27**. Here, we are joining only rows where the AOL query contains the keyword ‘weather’. For other graphs, the keyword was changed to ‘blizzard’, ‘snow’, ‘wind’, and ‘flood’. In the query for blizzards, we had to drop the condition that the URL also be weather related. Otherwise the dataset was too small to use. However, this also introduced many non-relevant AOL queries to the data, resulting in low-quality data.

```sql
WITH DATE_RANGE AS (
    SELECT DATE '2006-03-01' AS EVENT_DATE
    UNION ALL SELECT DATE '2006-03-02'
                .
                .
                .
    UNION ALL SELECT DATE '2006-05-31'
),
QUERY_FREQ AS (
        SELECT
            QUERYDIM.QUERY,
            TIMEDIM.[calender week],
            TIMEDIM.[day of the week],
            CAST (TO_TIMESTAMP(
                CONCAT(
                    TIMEDIM.[year], '-',
                    CASE
                        WHEN CAST(TIMEDIM.[month] AS CHAR(5)) = 'march' THEN '03'
                        WHEN CAST(TIMEDIM.[month] AS CHAR(5)) = 'april' THEN '04'
                        WHEN CAST(TIMEDIM.[month] AS CHAR(5)) = 'may' THEN '05'
                        ELSE 'NULL'
                    END
                    , '-', LPAD(TIMEDIM.[day of the month], 2, '0')),
                'YYYY-MM-DD'
            ) AS DATE) AS YMD_TIMESTAMP

        FROM AOL_SCHEMA.FACTS
        LEFT JOIN AOL_SCHEMA.TIMEDIM ON FACTS.TIMEID = TIMEDIM.ID
        LEFT JOIN AOL_SCHEMA.QUERYDIM ON FACTS.QUERYID = QUERYDIM.ID
        LEFT JOIN AOL_SCHEMA.URLDIM ON FACTS.URLID = URLDIM.ID
        WHERE LOWER(QUERYDIM.QUERY) LIKE '%weather%'
            AND (LOWER(URLDIM.URL) LIKE '%weather%'
                OR LOWER(URLDIM.DESCRIPTION) LIKE '%weather%')
        AND TIMEDIM.[year] IS NOT NULL
        AND TIMEDIM.[month] IS NOT NULL
        AND TIMEDIM.[day of the month] IS NOT NULL
        ORDER BY YMD_TIMESTAMP ASC
)
SELECT
    DATE_RANGE.EVENT_DATE,
    QUERY_FREQ.[calender week],
    QUERY_FREQ.[day of the week],
    COALESCE(COUNT(QUERY_FREQ.QUERY), 0) AS FREQ
FROM DATE_RANGE
LEFT JOIN QUERY_FREQ
ON DATE_RANGE.EVENT_DATE = QUERY_FREQ.YMD_TIMESTAMP
GROUP BY (DATE_RANGE.EVENT_DATE, QUERY_FREQ.[calender week], QUERY_FREQ.[day of the week])
ORDER BY DATE_RANGE.EVENT_DATE;'''
```

Output

Here, we can see the frequency of all weather related queries grouped by date.

	EVENT_DATE	calender week	day of the week	FREQ
0	2006-03-01	9	3	401
1	2006-03-02	9	4	314
2	2006-03-03	9	5	248
3	2006-03-04	9	6	244
4	2006-03-05	9	7	309
...	...	...	...	...
87	2006-05-27	21	6	279
88	2006-05-28	21	7	157
89	2006-05-29	22	1	334
90	2006-05-30	22	2	328
91	2006-05-31	22	3	267

Question 3

```
WITH TopURLs AS (
    SELECT URLDIM.URL
    FROM AOL_SCHEMA.FACTS
    JOIN AOL_SCHEMA.URLDIM ON AOL_SCHEMA.FACTS.URLID = AOL_SCHEMA.URLDIM.ID
    WHERE AOL_SCHEMA.FACTS.CLICK = 1
    GROUP BY URLDIM.URL
    ORDER BY COUNT(AOL_SCHEMA.FACTS.CLICK) DESC
    LIMIT 20
),
FREQCOMP AS (
    SELECT
        FACTS.ANONID,
        QUERYDIM.QUERY,
        CAST(
            CONCAT(
                '2006-',
                LPAD(CASE
                    WHEN TIMEDIM."day of the year" BETWEEN 60 AND 90 THEN '03'
                    WHEN TIMEDIM."day of the year" BETWEEN 91 AND 120 THEN '04'
                    WHEN TIMEDIM."day of the year" BETWEEN 121 AND 151 THEN '05'
                    ELSE '01'
                END, 2, '0'), '-',
                LPAD(TIMEDIM."day of the month", 2, '0'), ' ',
                LPAD(TIMEDIM."hour", 2, '0'), ':',
                LPAD(TIMEDIM."minute", 2, '0'), ':',
                LPAD(TIMEDIM."second", 2, '0')
            ) AS TIMESTAMP
        ) AS time_as_datetime
    FROM
        AOL_SCHEMA.FACTS
        LEFT JOIN AOL_SCHEMA.TIMEDIM ON FACTS.TIMEID = TIMEDIM.ID
        LEFT JOIN AOL_SCHEMA.URLDIM ON FACTS.URLID = URLDIM.ID
        LEFT JOIN AOL_SCHEMA.QUERYDIM ON FACTS.QUERYID = QUERYDIM.ID
    WHERE FACTS.CLICK = 1
        AND (
            URLDIM.URL IN (SELECT URL FROM TopURLs)
            OR LOWER(URLDIM.URL) LIKE '%weather%'
        )
        AND FACTS.ANONID IS NOT NULL
        AND TIMEDIM."hour" IS NOT NULL
        AND TIMEDIM."minute" IS NOT NULL
        AND TIMEDIM."second" IS NOT NULL
        AND TIMEDIM."day of the year" IS NOT NULL
),
DateRange AS (
    SELECT DATE '2006-03-01' AS EVENT_DATE
    UNION ALL SELECT DATE '2006-03-02'
    UNION ALL SELECT DATE '2006-03-03'
    ...
    UNION ALL SELECT DATE '2006-05-31'
)
SELECT
    DateRange.EVENT_DATE AS query_date,
    COALESCE(COUNT(*), 0) AS number_of_queries
FROM
    DateRange
LEFT JOIN
    FREQCOMP E
ON
    CAST(E.time_as_datetime AS DATE) = DateRange.EVENT_DATE
AND LOWER(E.QUERY) LIKE '%tornado%'
GROUP BY
    DateRange.EVENT_DATE
ORDER BY
    query_date;
```

```
WITH DateRange AS (
    SELECT DATE '2006-03-01' AS EVENT_DATE
    UNION ALL SELECT DATE '2006-03-02'
    ...
    UNION ALL SELECT DATE '2006-05-31'
)
SELECT
    DateRange.EVENT_DATE,
    COALESCE(COUNT(E.EPISODE_ID), 0) AS EVENT_COUNT
FROM
    DateRange
LEFT JOIN
    AOL_SCHEMA.WEATHER_EVENTS E
ON
    CAST(E.BEGIN_DATE_TIME AS DATE) = DateRange.EVENT_DATE
    AND E.EVENT_TYPE = 'Tornado'
GROUP BY
    DateRange.EVENT_DATE
ORDER BY
    DateRange.EVENT_DATE;
```

Working of the query: These queries count AOL queries and events for each day, utilizing basic aggregate operations and SLICE/DICE.

Output

	QUERY_DATE	NUMBER_OF_QUERIES
0	2006-03-01	18158
1	2006-03-02	19156
2	2006-03-03	17264
3	2006-03-04	19478
4	2006-03-05	21853
...	...	...
87	2006-05-27	17136
88	2006-05-28	12652
89	2006-05-29	19669
90	2006-05-30	18852
91	2006-05-31	18833

	EVENT_DATE	EVENT_COUNT
0	2006-03-01	318
1	2006-03-02	92
2	2006-03-03	14
3	2006-03-04	35
4	2006-03-05	30
...	...	...
87	2006-05-27	210
88	2006-05-28	191
89	2006-05-29	277
90	2006-05-30	374
91	2006-05-31	280

Question 4a

```sql
WITH Events_DATA AS (
    SELECT
        we.EVENT_TYPE,
        we.BEGIN_DATE_TIME,
        we.END_DATE_TIME
    FROM AOL_SCHEMA.WEATHER_EVENTS we
    WHERE we.BEGIN_DATE_TIME >= '2006-03-01 00:01:00.000000'
      AND LOWER(we.EVENT_TYPE) LIKE 'tornado'
),
Relevant_Queries AS (
    SELECT
        QUERIES_WEATHER.QUERY_LOWER_TRIMMED AS QUERY,
        QUERIES_WEATHER.FORMATTED_DATE,
        QUERIES_WEATHER.QUERY_ABOUT,
        we.EVENT_TYPE
    FROM Events_DATA we
    INNER JOIN AOL_SCHEMA.Weather_Reaction_Queries AS QUERIES_WEATHER
        ON QUERIES_WEATHER.FORMATTED_DATE BETWEEN we.BEGIN_DATE_TIME AND we.END_DATE_TIME
),
Grouped_Results AS (
    SELECT
        QUERY,
        QUERY_ABOUT,
        EVENT_TYPE,
        COUNT(*) AS query_count
    FROM Relevant_Queries
    GROUP BY GROUPING SETS (
        (QUERY, QUERY_ABOUT, EVENT_TYPE),
        (QUERY,EVENT_TYPE)
    )
),
```

```sql
TOKEN_COUNT AS (
    SELECT
        TOKEN,
        COALESCE(QUERY_ABOUT, 'UNKNOWN') AS QUERY_ABOUT, -- Treat NULLs as 'UNKNOWN' for group by
        SUM(query_count) AS token_count
    FROM Tokens_Queries
    GROUP BY TOKEN, QUERY_ABOUT
    HAVING SUM(query_count)>2
    ORDER BY token_count DESC
),
TOTAL_QUERY_COUNT AS (
    SELECT
        QUERY_ABOUT,
        SUM(token_count) AS total_query_count
    FROM TOKEN_COUNT
    GROUP BY QUERY_ABOUT
),
NORMALIZED_TOKEN_COUNT AS (
    SELECT
        tc.TOKEN,
        tc.QUERY_ABOUT,
        tc.token_count,
        tqc.total_query_count,
        CASE
            WHEN tc.QUERY_ABOUT = 'UNKNOWN' THEN
                (tc.token_count * 1.0 /
                 (SELECT SUM(token_count) FROM TOKEN_COUNT WHERE QUERY_ABOUT = 'UNKNOWN'))
            ELSE
                (tc.token_count * 1.0 / tqc.total_query_count)
        END AS Group_Specific_Token_Probability
    FROM TOKEN_COUNT tc
    JOIN TOTAL_QUERY_COUNT tqc
        ON tc.QUERY_ABOUT = tqc.QUERY_ABOUT
)
SELECT
    TOKEN,
    QUERY_ABOUT,
    token_count,
    total_query_count,
    Group_Specific_Token_Probability,
    Group_Specific_Token_Probability*100 AS Group_specific_Token_Percentage
FROM NORMALIZED_TOKEN_COUNT
ORDER BY QUERY_ABOUT,Group_Specific_Token_Probability DESC;
```

Key operations utilized: INNER JOIN, GROUPING SETS.

Working of the query: This Query obtain the weather queries for each query about (Weather and Tracking Alerts, Damage Assessment and Recovery etc...) during tornado. It also obtains the group specific token probability and percentage.

Output:

	TOKEN	QUERY_ABOUT	TOKEN_COUNT	TOTAL_QUERY_COUNT	GROUP_SPECIFIC_TOKEN_PROBABILITY	GROUP_SPECIFIC_TOKEN_PERCENTAGE
739	weather	Weather Tracking and Alerts	331	1981	0.167087	16.708733
740	bug	Weather Tracking and Alerts	67	1981	0.033821	3.382130
741	cold	Weather Tracking and Alerts	56	1981	0.028269	2.826855
742	temperature	Weather Tracking and Alerts	46	1981	0.023221	2.322060
743	hurricane	Weather Tracking and Alerts	43	1981	0.021706	2.170621
...	...	...	...	...	...	...
968	cheese	Weather Tracking and Alerts	3	1981	0.001514	0.151439
969	east	Weather Tracking and Alerts	3	1981	0.001514	0.151439
970	changes	Weather Tracking and Alerts	3	1981	0.001514	0.151439
971	handy	Weather Tracking and Alerts	3	1981	0.001514	0.151439
972	knife	Weather Tracking and Alerts	3	1981	0.001514	0.151439

Question 4b

```sql
WITH Ordered_Events AS (
    SELECT
        EVENT_TYPE,
        REGION,
        BEGIN_DATE_TIME,
        END_DATE_TIME,
        LEAD(BEGIN_DATE_TIME) OVER (ORDER BY BEGIN_DATE_TIME) AS NEXT_END_DATE_TIME
    FROM AOL_SCHEMA.WEATHER_EVENTS
    WHERE EVENT_TYPE = 'Tornado'
    ORDER BY BEGIN_DATE_TIME
),
Calculated_Events AS (
    SELECT
        EVENT_TYPE,
        BEGIN_DATE_TIME,
        END_DATE_TIME,
        ADD_MINUTES(TO_TIMESTAMP(SUBSTR(END_DATE_TIME, 1, 19)), 1) AS WINDOW_START,
        ADD_MINUTES(TO_TIMESTAMP(SUBSTR(NEXT_END_DATE_TIME, 1, 19)), -1) AS WINDOW_END
    FROM Ordered_Events
),
Window_Events AS (
    SELECT
        EVENT_TYPE,
        BEGIN_DATE_TIME,
        END_DATE_TIME,
        WINDOW_START,
        WINDOW_END
    FROM Calculated_Events
    WHERE SECONDS_BETWEEN(WINDOW_END, WINDOW_START) >= 0
),
Relevant_Queries AS (
    SELECT
        qt.QUERY_LOWER_TRIMMED AS QUERY,
        qt.QUERY_ABOUT,
        qt.FORMATTED_DATE,
        tq.EVENT_TYPE,
        tq.BEGIN_DATE_TIME,
        tq.END_DATE_TIME,
        tq.WINDOW_START,
        tq.WINDOW_END
    FROM Window_Events tq
    INNER JOIN AOL_SCHEMA.Weather_Reaction_Queries qt
        ON qt.FORMATTED_DATE BETWEEN tq.WINDOW_START AND tq.WINDOW_END
),
Grouped_Results AS (
    SELECT
        QUERY,
        EVENT_TYPE,
        QUERY_ABOUT,
        COUNT(*) AS query_count
    FROM Relevant_Queries
    GROUP BY GROUPING SETS (
        (QUERY, EVENT_TYPE, QUERY_ABOUT),
        (QUERY, EVENT_TYPE)
    )
),
```

```sql
Tokens_Queries AS (
    SELECT
        token.TOKEN,
        ex.EVENT_TYPE,
        ex.QUERY_ABOUT,
        ex.query_count
    FROM Grouped_Results ex
    INNER JOIN AOL_SCHEMA.tokenization_new_queries token
        ON ex.QUERY = token.QUERY
),
TOKEN_COUNT AS (
    SELECT
        TOKEN,
        COALESCE(QUERY_ABOUT, 'UNKNOWN') AS QUERY_ABOUT, -- Treat NULLs as 'UNKNOWN' for group by
        SUM(query_count) AS token_count
    FROM Tokens_Queries
    GROUP BY TOKEN, QUERY_ABOUT
    HAVING SUM(query_count)>2
    ORDER BY token_count DESC
),
TOTAL_QUERY_COUNT AS (
    SELECT
        QUERY_ABOUT,
        SUM(token_count) AS total_token_count
    FROM TOKEN_COUNT
    GROUP BY QUERY_ABOUT
),
-- Normalize token probabilities for 'UNKNOWN' and other QUERY_ABOUT
NORMALIZED_TOKEN_COUNT AS (
    SELECT
        tc.TOKEN,
        tc.QUERY_ABOUT,
        tc.token_count,
        tqc.total_token_count,  -- Correct column name
        CASE
            WHEN tc.QUERY_ABOUT = 'UNKNOWN' THEN
                (tc.token_count * 1.0 /
                 (SELECT SUM(token_count) FROM TOKEN_COUNT WHERE QUERY_ABOUT = 'UNKNOWN'))
            ELSE
                (tc.token_count * 1.0 / tqc.total_token_count)  -- Update here as well
        END AS Group_Specific_Token_Probability
    FROM TOKEN_COUNT tc
    JOIN TOTAL_QUERY_COUNT tqc
        ON tc.QUERY_ABOUT = tqc.QUERY_ABOUT
)
-- Final selection with normalized probabilities
SELECT
    TOKEN,
    QUERY_ABOUT,
    token_count,
    total_token_count,
    Group_Specific_Token_Probability,
    Group_Specific_Token_Probability*100 AS Group_Specific_Token_Percentage
FROM NORMALIZED_TOKEN_COUNT
ORDER BY QUERY_ABOUT,Group_Specific_Token_Probability DESC;
```

Key operations utilized: LEAD, ADD_MINUTES, SECONDS_BETWEEN, INNER JOIN, GROUPING SETS..

Working of the query: This Query obtain the weather queries for each query about (Weather and Tracking Alerts, Damage Assessment and Recovery etc…) after Tornado.. It Also obtain the group specific token probability and percentage.

Output:

	TOKEN	QUERY_ABOUT	TOKEN_COUNT	TOTAL_TOKEN_COUNT	GROUP_SPECIFIC_TOKEN_PROBABILITY	GROUP_SPECIFIC_TOKEN_PERCENTAGE
0	insurance	Damage Assessment and Recovery	9625	41534	0.231738	23.173785
1	repair	Damage Assessment and Recovery	3398	41534	0.081812	8.181249
2	health	Damage Assessment and Recovery	1198	41534	0.028844	2.884384
3	auto	Damage Assessment and Recovery	943	41534	0.022704	2.270429
4	life	Damage Assessment and Recovery	852	41534	0.020513	2.051331
...	...	...	...	...	...	...
13971	marco	Weather Tracking and Alerts	3	74301	0.000040	0.004038
13972	cooler	Weather Tracking and Alerts	3	74301	0.000040	0.004038
13973	straw	Weather Tracking and Alerts	3	74301	0.000040	0.004038
13974	brunswick	Weather Tracking and Alerts	3	74301	0.000040	0.004038
13975	killed	Weather Tracking and Alerts	3	74301	0.000040	0.004038

Code 7:

```
import pandas as pd
import nltk
from nltk.stem import WordNetLemmatizer

# Download required NLTK resources (if you haven't already)
nltk.download('wordnet')
nltk.download('omw-1.4')

# Initialize the lemmatizer
lemmatizer = WordNetLemmatizer()

# Function to lemmatize tokens
def lemmatize_token(token):
    return lemmatizer.lemmatize(token)

# Apply lemmatization to the 'TOKEN' column
query_events['LEMMATIZED_TOKEN'] = query_events['TOKEN'].apply(lemmatize_token)

query_result = query_events.groupby(
    ['QUERY_ABOUT', 'LEMMATIZED_TOKEN'], as_index=False
).agg({
    'GROUP_SPECIFIC_TOKEN_PROBABILITY': 'sum',
    'GROUP_SPECIFIC_TOKEN_PERCENTAGE': 'sum'
})
# Print the result
print(query_result)
```

This code performs word lemmatization, converting words to their base form, and appends their associated group-specific token probability and percentage.

Libraries used: nltk

Required nltk resources: wordnet, omw-1.4

Code 8:

```
during_tokens = query_during_tornado[['LEMMATIZED_TOKEN', 'GROUP_SPECIFIC_TOKEN_PROBABILITY']]
after_tokens = query_after_tornado[['LEMMATIZED_TOKEN', 'GROUP_SPECIFIC_TOKEN_PROBABILITY']]

# Create weighted frequency dictionaries for both during and after tornado
def create_weighted_dict(tokens_df):
    """
    Creates a dictionary of tokens with their respective probability values.
    """
    return dict(zip(tokens_df['LEMMATIZED_TOKEN'], tokens_df['GROUP_SPECIFIC_TOKEN_PROBABILITY']))

# Create the weighted dictionaries for both during and after tornado groups
during_weighted_dict = create_weighted_dict(during_tokens)
after_weighted_dict = create_weighted_dict(after_tokens)

# Find common tokens between the two dataframes
common_tokens = set(during_weighted_dict.keys()) & set(after_weighted_dict.keys())

## Filter the weighted dictionaries to include only common tokens
common_during_dict = {token: during_weighted_dict[token] for token in common_tokens}
common_after_dict = {token: after_weighted_dict[token] for token in common_tokens}

unique_during_tokens = set(during_weighted_dict.keys()) - set(after_weighted_dict.keys())
unique_after_tokens = set(after_weighted_dict.keys()) - set(during_weighted_dict.keys())

# Filter the weighted dictionaries to include only common tokens
unique_during_dict = {token: during_weighted_dict[token] for token in unique_during_tokens}
unique_after_dict = {token: after_weighted_dict[token] for token in unique_after_tokens}


# Word Cloud for During Tornado
unique_during_wordcloud = WordCloud(width=800, height=400,background_color='white').generate_from_frequencies(unique_during_dict)

# Word Cloud for After Tornado
unique_after_wordcloud = WordCloud(width=800, height=400,background_color='white').generate_from_frequencies(unique_after_dict)
```

This code filters the common and distinct words between during and after tornado.

Query 5a

```
SELECT
    T1.ANONID,
    T1.TIME_AS_DATETIME,
    COALESCE(LAG(TIME_AS_DATETIME) OVER (PARTITION BY ANONID ORDER BY TIME_AS_DATETIME), TIME_AS_DATETIME) AS LaggedDateTime,
    SECONDS_BETWEEN(T1.TIME_AS_DATETIME, COALESCE(LAG(TIME_AS_DATETIME) OVER (PARTITION BY ANONID ORDER BY TIME_AS_DATETIME), TIME_AS_DATETIME)) as Seconds_Difference,
    MINUTES_BETWEEN(T1.TIME_AS_DATETIME, COALESCE(LAG(TIME_AS_DATETIME) OVER (PARTITION BY ANONID ORDER BY TIME_AS_DATETIME), TIME_AS_DATETIME)) as Minutes_Difference
FROM
    AOL_SCHEMA.INTERARRIVAL_TIMES as T1
WHERE
    T1.ANONID IN (
        SELECT T2.ANONID
        FROM AOL_SCHEMA.INTERARRIVAL_TIMES as T2
        GROUP BY T2.ANONID
        HAVING COUNT(T2.ANONID) >= 10
    )
    AND NOT EXISTS(
        SELECT 1
        FROM AOL_SCHEMA.WEATHER_EVENTS as T3
        WHERE
            (T1.TIME_AS_DATETIME BETWEEN T3.BEGIN_DATE_TIME AND T3.END_DATE_TIME)
            AND (T3.EVENT_TYPE = 'Tornado')
    )
ORDER BY
    T1.ANONID,
    T1.TIME_AS_DATETIME
```

Key operations utilized: Coalesce, Partition By ,Seconds/Minutes Between, Slice/Dice

Working of the query:
Retrieves the UserID, time, lagged time, and interclick times (in seconds and minutes) for users who clicked on at least 10 of the top 20 most popular links, excluding clicks during disasters.

Output

	ANONID	TIME_AS_DATETIME	LAGGEDDATETIME	SECONDS_DIFFERENCE	MINUTES_DIFFERENCE
0	15	2006-03-11 09:55:17.000000	2006-03-11 09:55:17.000000	0	0.000000
1	15	2006-03-18 21:02:45.000000	2006-03-11 09:55:17.000000	644848	10747.466667
2	15	2006-03-18 21:06:01.000000	2006-03-18 21:02:45.000000	196	3.266667
3	15	2006-03-21 19:55:51.000000	2006-03-18 21:06:01.000000	254990	4249.833333
4	15	2006-03-21 20:09:22.000000	2006-03-21 19:55:51.000000	811	13.516667
...	...	...	...	...	...
999853	657399	2006-04-11 09:28:37.000000	2006-04-11 09:22:01.000000	396	6.600000
999854	657399	2006-04-11 09:33:27.000000	2006-04-11 09:28:37.000000	290	4.833333
999855	657399	2006-04-11 09:36:22.000000	2006-04-11 09:33:27.000000	175	2.916667
999856	657399	2006-04-24 19:48:08.000000	2006-04-11 09:36:22.000000	1159906	19331.766667
999857	657399	2006-04-29 19:11:47.000000	2006-04-24 19:48:08.000000	429819	7163.650000

Query 5b

```sql
SELECT
    T1.ANONID,
    T1.TIME_AS_DATETIME,
    COALESCE(LAG(TIME_AS_DATETIME) OVER (PARTITION BY ANONID ORDER BY TIME_AS_DATETIME), TIME_AS_DATETIME) AS LaggedDateTime,
    SECONDS_BETWEEN(T1.TIME_AS_DATETIME, COALESCE(LAG(TIME_AS_DATETIME) OVER (PARTITION BY ANONID ORDER BY TIME_AS_DATETIME), TIME_AS_DATETIME)) as Seconds_Difference,
    MINUTES_BETWEEN(T1.TIME_AS_DATETIME, COALESCE(LAG(TIME_AS_DATETIME) OVER (PARTITION BY ANONID ORDER BY TIME_AS_DATETIME), TIME_AS_DATETIME)) as Minutes_Difference
FROM
    AOL_SCHEMA.INTERARRIVAL_TIMES as T1
WHERE
    T1.ANONID IN (
        SELECT T2.ANONID
        FROM AOL_SCHEMA.INTERARRIVAL_TIMES as T2
        GROUP BY T2.ANONID
        HAVING COUNT(T2.ANONID) >= 10
    )
    AND EXISTS(
        SELECT 1
        FROM AOL_SCHEMA.WEATHER_EVENTS as T3
        WHERE
            (T1.TIME_AS_DATETIME BETWEEN T3.BEGIN_DATE_TIME AND T3.END_DATE_TIME)
            AND (T3.EVENT_TYPE = 'Tornado')
    )
ORDER BY
    T1.ANONID,
    T1.TIME_AS_DATETIME
```

Key operations utilized: Coalesce, Partition By, Seconds/Minutes Difference, Exists

Working of the query: Retrieves the UserID, time, lagged time, and interclick times (in seconds and minutes) for users who clicked on at least 10 of the top 20 most popular links during a tornado

Output

	ANONID	TIME_AS_DATETIME	LAGGEDDATETIME	SECONDS_DIFFERENCE	MINUTES_DIFFERENCE
0	25	2006-04-02 20:15:56.000000	2006-04-02 20:15:56.000000	0	0.000000
1	25	2006-04-15 15:55:54.000000	2006-04-02 20:15:56.000000	1107598	18459.966667
2	25	2006-04-15 17:30:44.000000	2006-04-15 15:55:54.000000	5690	94.833333
3	29	2006-03-12 19:47:08.000000	2006-03-12 19:47:08.000000	0	0.000000
4	29	2006-03-12 21:54:36.000000	2006-03-12 19:47:08.000000	7648	127.466667
...	...	...	...	...	...
28512	657282	2006-04-16 13:35:13.000000	2006-04-16 13:35:13.000000	0	0.000000
28513	657283	2006-04-02 16:37:24.000000	2006-04-02 16:37:24.000000	0	0.000000
28514	657283	2006-04-02 19:10:12.000000	2006-04-02 16:37:24.000000	9168	152.800000
28515	657283	2006-04-02 19:10:12.000000	2006-04-02 19:10:12.000000	0	0.000000
28516	657307	2006-04-07 15:49:40.000000	2006-04-07 15:49:40.000000	0	0.000000

Bonus Question - a

This query identifies the most clicked domains or URLs and matches them by time to specific weather events and regions where they occurred, ranking them based on the number of clicks

```sql
WITH ClickedDomains AS (
    SELECT
        AOL_SCHEMA.WEATHER_EVENTS.REGION AS REGION,
        AOL_SCHEMA.WEATHER_EVENTS.EVENT_TYPE AS EVENT_TYPE,
        AOL_SCHEMA.WEATHER_EVENTS.BEGIN_DATE_TIME AS BEGIN_DATE_TIME,
        AOL_SCHEMA.URLDIM.THISDOMAIN AS THISDOMAIN,
        AOL_SCHEMA.URLDIM.URL AS URL,
        COUNT(AOL_SCHEMA.FACTS.URLID) AS CLICK_COUNT
    FROM
        AOL_SCHEMA.WEATHER_EVENTS
    JOIN
        AOL_SCHEMA.FACTS
        ON AOL_SCHEMA.WEATHER_EVENTS.BEGIN_DAY = AOL_SCHEMA.FACTS.TIMEID
    JOIN
        AOL_SCHEMA.URLDIM
        ON AOL_SCHEMA.FACTS.URLID = AOL_SCHEMA.URLDIM.ID
    WHERE
        AOL_SCHEMA.FACTS.CLICK = TRUE
        AND (AOL_SCHEMA.URLDIM.THISDOMAIN IS NOT NULL OR AOL_SCHEMA.URLDIM.URL IS NOT NULL)
        AND AOL_SCHEMA.WEATHER_EVENTS.BEGIN_DATE_TIME BETWEEN '2006-03-01 00:00:00' AND '2006-05-31 23:59:59'
    GROUP BY ROLLUP(
        AOL_SCHEMA.WEATHER_EVENTS.REGION,
        AOL_SCHEMA.WEATHER_EVENTS.EVENT_TYPE,
        AOL_SCHEMA.WEATHER_EVENTS.BEGIN_DATE_TIME,
        AOL_SCHEMA.URLDIM.THISDOMAIN,
        AOL_SCHEMA.URLDIM.URL
    )
),
RankedDomains AS (
    SELECT
        REGION,
        EVENT_TYPE,
        BEGIN_DATE_TIME,
        THISDOMAIN,
        URL,
        CLICK_COUNT,
        ROW_NUMBER() OVER (
            PARTITION BY REGION, EVENT_TYPE
            ORDER BY CLICK_COUNT DESC
        ) AS RANK
    FROM
        ClickedDomains
    WHERE
        THISDOMAIN IS NOT NULL OR URL IS NOT NULL
)
SELECT
    REGION,
    EVENT_TYPE,
    BEGIN_DATE_TIME,
    THISDOMAIN,
    URL,
    CLICK_COUNT
FROM
    RankedDomains
WHERE
    RANK = 1
ORDER BY
    REGION,
    EVENT_TYPE,
    BEGIN_DATE_TIME;
```

Output

	REGION	EVENT_TYPE	BEGIN_DATE_TIME	THISDOMAIN	URL	CLICK_COUNT
0	Alabama	Flash Flood	2006-03-20 18:45:00.000000	nau	http://www.nau.edu	3
1	Alabama	Funnel Cloud	2006-03-20 17:55:00.000000	nau	http://www.nau.edu	3
2	Alabama	Hail	2006-04-20 17:08:00.000000	nau	http://www.nau.edu	6
3	Alabama	Lightning	2006-04-18 18:10:00.000000	citysearch	http://pittsburgh.citysearch.com	1
4	Alabama	Strong Wind	2006-03-09 14:15:00.000000	ca	http://gocalif.ca.gov	1
...	...	...	...	...	...	...
487	Wyoming	Heavy Snow	2006-05-09 04:00:00.000000	ca	http://gocalif.ca.gov	2
488	Wyoming	Lightning	2006-05-08 14:10:00.000000	ebay.co	http://www.ebay.co.uk	1
489	Wyoming	Thunderstorm Wind	2006-05-26 16:39:00.000000	yahoo	http://mail.yahoo.com	1
490	Wyoming	Wildfire	2006-04-10 09:00:00.000000	bilkent.edu	http://web.bilkent.edu.tr	1
491	Wyoming	Winter Storm	2006-04-24 01:00:00.000000	sharesong	http://www.sharesong.org	4

492 rows × 6 columns

Bonus Question - b

This query returns the top 10 most clicked URLs / domains, that correspond to the certain weather events by time

```
WITH ClickedDomains AS (
    SELECT
        AOL_SCHEMA.WEATHER_EVENTS.REGION AS REGION,
        AOL_SCHEMA.WEATHER_EVENTS.EVENT_TYPE AS EVENT_TYPE,
        AOL_SCHEMA.WEATHER_EVENTS.BEGIN_DATE_TIME AS BEGIN_DATE_TIME,
        AOL_SCHEMA.URLDIM.THISDOMAIN AS THISDOMAIN,
        AOL_SCHEMA.URLDIM.URL AS URL,
        COUNT(AOL_SCHEMA.FACTS.URLID) AS CLICK_COUNT
    FROM
        AOL_SCHEMA.WEATHER_EVENTS
    JOIN
        AOL_SCHEMA.FACTS
        ON AOL_SCHEMA.WEATHER_EVENTS.BEGIN_DAY = AOL_SCHEMA.FACTS.TIMEID
    JOIN
        AOL_SCHEMA.URLDIM
        ON AOL_SCHEMA.FACTS.URLID = AOL_SCHEMA.URLDIM.ID
    WHERE
        AOL_SCHEMA.FACTS.CLICK = TRUE
        AND (AOL_SCHEMA.URLDIM.THISDOMAIN IS NOT NULL OR AOL_SCHEMA.URLDIM.URL IS NOT NULL)
        AND AOL_SCHEMA.WEATHER_EVENTS.BEGIN_DATE_TIME BETWEEN '2006-03-01 00:00:00' AND '2006-05-31 23:59:59'
    GROUP BY
        AOL_SCHEMA.WEATHER_EVENTS.REGION,
        AOL_SCHEMA.WEATHER_EVENTS.EVENT_TYPE,
        AOL_SCHEMA.WEATHER_EVENTS.BEGIN_DATE_TIME,
        AOL_SCHEMA.URLDIM.THISDOMAIN,
        AOL_SCHEMA.URLDIM.URL
),
RankedDomains AS (
    SELECT
        REGION,
        EVENT_TYPE,
        BEGIN_DATE_TIME,
        THISDOMAIN,
        URL,
        CLICK_COUNT,
        ROW_NUMBER() OVER (
            PARTITION BY REGION
            ORDER BY CLICK_COUNT DESC, BEGIN_DATE_TIME ASC
        ) AS RANK
    FROM
        ClickedDomains
)
SELECT
    REGION,
    EVENT_TYPE,
    BEGIN_DATE_TIME,
    THISDOMAIN,
    URL,
    CLICK_COUNT
FROM
    RankedDomains
WHERE
    RANK = 1
ORDER BY CLICK_COUNT DESC
LIMIT 10;
```

Output

	REGION	EVENT_TYPE	BEGIN_DATE_TIME	THISDOMAIN	URL	CLICK_COUNT
0	Mississippi	Flash Flood	2006-03-20 15:00:00.000000	nau	http://www.nau.edu	12
1	Oklahoma	Wildfire	2006-03-15 12:00:00.000000	yahoo	http://mail.yahoo.com	10
2	North dakota	Flood	2006-04-01 00:00:00.000000	lib.rochester	http://www.lib.rochester.edu	9
3	South carolina	Hail	2006-05-20 13:50:00.000000	nau	http://www.nau.edu	9
4	Minnesota	Flood	2006-04-01 00:00:00.000000	nintendo	http://www.nintendo.com	8
5	Missouri	Hail	2006-04-02 14:30:00.000000	the-antiaging-site	http://www.the-antiaging-site.com	8
6	Indiana	Flash Flood	2006-03-12 05:30:00.000000	ikea	http://www.ikea.com	7
7	Texas	High Wind	2006-03-20 12:05:00.000000	nau	http://www.nau.edu	6
8	Washington	Thunderstorm Wind	2006-04-15 15:15:00.000000	yahoo	http://mail.yahoo.com	6
9	North carolina	Winter Weather	2006-03-20 12:00:00.000000	nau	http://www.nau.edu	6

Bonus Question diagram creation code

The output is on the slide 61

```
import matplotlib.pyplot as plt
import matplotlib.cm as cm
import numpy as np

# Generate a list of unique colors for each region
regions = top_domains['REGION'].unique()
colors = cm.tab20(np.linspace(0, 1, len(regions)))  # Use colormap to generate colors
region_color_map = {region: colors[i] for i, region in enumerate(regions)}  # Map each region to a unique color

# Add a small reduced offset to CLICK_COUNT for overlapping points
jittered_clicks = top_domains.copy()
jittered_clicks['JITTERED_CLICK_COUNT'] = jittered_clicks['CLICK_COUNT']
jittered_clicks.loc[jittered_clicks.duplicated(subset=['EVENT_TYPE', 'CLICK_COUNT'], keep=False), 'JITTERED_CLICK_COUNT'] += np.linspace(-0.1, 0.1, sum(jittered_clicks.duplicated(subset=['EVENT_TYPE', 'CLICK_COUNT'], keep=False)))

# Create the scatter plot
plt.figure(figsize=(16, 10))
for i, row in jittered_clicks.iterrows():
    plt.scatter(row['EVENT_TYPE'], row['JITTERED_CLICK_COUNT'], color=region_color_map[row['REGION']], s=150, alpha=0.7)

# Create the legend
legend_elements = [
    plt.Line2D([0], [0], marker='o', color='w', markerfacecolor=region_color_map[region],
               markersize=10, label=f"{region} \n {row['URL'][:30]}{'...' if len(row['URL']) > 30 else ''}")
    for region, row in jittered_clicks.groupby('REGION').first().iterrows()
]
plt.legend(handles=legend_elements, title="Region (URL)", bbox_to_anchor=(1.05, 1), loc='upper left', fontsize=10)

# Add titles and labels
plt.title('Clicks by Event Type and Region', fontsize=16)
plt.xlabel('Event Type', fontsize=14)
plt.ylabel('Click Count', fontsize=14)
plt.xticks(rotation=30, ha='right', fontsize=12)  # Rotate x-axis labels for readability
plt.grid(axis='y', linestyle='--', alpha=0.7)  # Add a light grid for clarity
plt.tight_layout()

# Show the plot
plt.show()
```

References

- https://www.ncei.noaa.gov/pub/data/swdi/stormevents/csvfiles/legacy/
- https://www.ncdc.noaa.gov/stormevents/ftp.jsp
- A Picture of Search
- GitHub: https://github.com/chandlerNick/BHT_BI_WiSe2425.git

Bonus Slide - How to deal with the old DB

Tips on getting the DB running (in the event our presentation is posted to moodle):

1. Install virtual box
2. Load your database image
3. Ensure you use both the NAT and Host only adapter in network settings
4. Install ODBC Data Sources and test a system DSN with the proper credentials
5. Use the given fingerprint in between the ip address and port
6. If using pyexasol, ensure you use the fingerprint again and that the protocol is the oldest one.

I hope this helps.

Problem Requirements

Subtasks

1. **Define five interesting analysis question on this data set**. You might pick up more, since not all chosen questions are answerable with the existing data and your additional data sources.
2. **Import missing data from one additional data source of your choice** for resolving your queries into the database. Use your knowledge on JDBC and Extract-Transform-Load. Please check legal issues when importing data from "The Web".
3. **Formalize at least five of your queries with ROLAP Statements on EXASOL**. Utilize operators such as SLICE, DICE, CUBE, ROLLUP, PARTITION BY, GROUPING SETS and other standard SQL statements such as joins, unions or intersections etc. (see the EXASOL manual for details on the syntax). You might also use PANDAS or Python functions to predict from the data.
4. **Display your results as charts**, for example with http://d3js.org or JFreeChart
5. **Create a presentation for about 15 minutes** and explain your analysis goal, your data sets, showcase selected "cool/surprising" queries and results/insights, explain why this is an important valuable finding, show your schema and explain your workload.
6. **Create an appendix in your presentation**, where you **show the ROLAP queries and results** as screenshots. Name on each slide, what this query should have done. Add to the appendix screenshots of the tables you created, including schema information.
7. **Upload this presentation to the Moodle-system** with a filename <your name> (PDF/PPT) and present it in front of your peers. Check if your peers liked it and considered it insightful. ☺