Project 3 Group 18: Tornadoes

By Grant Hagen, Daniel Purrier, James Lee, and Stephen Ferrier

Data Engineering (Grant)

Thankfully our dataset was pretty clean for the most part, so there wasn't much TLC to be done on the data set. We started dropping some columns that we didn't end up needing. Such as day, time zone, date time, and a few others that did not provide value to accomplish our story. We then renamed a few of our columns to be a bit more descriptive. For instance we renamed magnitude to category. Only one data type needed to be changed. Date needed to be changed from an object to a date. We added a new column to further our analysis, and that column was seasons. We created the season column utilizing some code that I used for project one. Season was created by creating a data dictionary for each month and their corresponding season, looping through each row, and then appending each season to their appropriate month. We then exported the cleaned data into a SQLite file.

After exporting our data, we were then able to take a look at some potential ideas for our visualizations. Our two main queries that we used are those shown below. On the left is our map query, grabbing the year, state, etc based on the user selected year. The query on the right is what we utilized for our dashboard based on the user selected season.

```
# switch on user_year
if user_year != 'All':
    where_clause = f"yr = {user_year}"
else:
    where_clause = f"yr > 2000"

# build the query
query = f"""
    SELECT
        yr,
        state,
        category,
        loss,
        start_lat,
        start_longitude,
        end_latitude,
        end_longitude,
        distance_traveled,
        width
    FROM
        tornadoes
    WHERE
        {where_clause}
    ORDER BY
        date DESC;
"""
```

```
# build the query
if user_seasons != 'All':
    where_clause = f"seasons = {user_seasons}"
else:
    where_clause = "1 = 1"

query = f"""
    SELECT
        yr,
        month,
        state,
        category,
        injuries,
        fatalities,
        seasons
    FROM
        tornadoes
    WHERE
        {where_clause}
    ORDER BY
        date DESC;
"""
```

After finalizing our queries we then transferred our SQL queries to our app helper. Both shown below. On the left is our map query and on the right is our dashboard query.

```python
# Map Query
def get_map(self, year):

    # User Input
    if year != -1:
        where_clause = f"yr = {year}"
    else:
        where_clause = f"yr > 2000"

    # Query
    query = f"""
        SELECT
            yr,
            state,
            category,
            loss,
            start_lat,
            start_longitude,
            end_latitude,
            end_longitude,
            distance_traveled,
            width
        FROM
            tornadoes
        WHERE
            {where_clause};
    """

    # Convert data into dictionary
    df = pd.read_sql(text(query), con = self.engine)
    data = df.to_dict(orient="records")
    return(data)
```

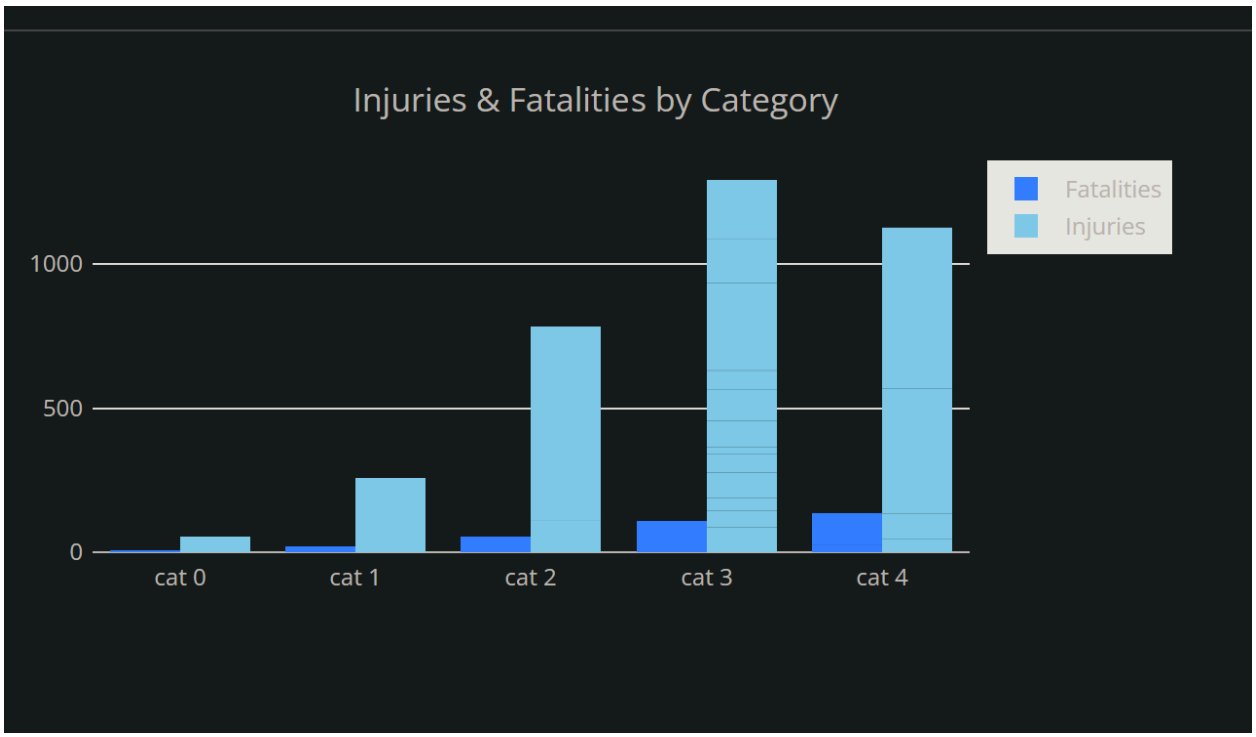
```python
# Bar Graph Query
def get_bar(self, user_seasons):

    # User Input
    if user_seasons != 'All':
        where_clause = f"seasons LIKE '{user_seasons}'"
    else:
        where_clause = "1 = 1"

    # Query
    query = f"""
        SELECT
            yr,
            category,
            injuries,
            fatalities
        FROM
            tornadoes
        WHERE
            {where_clause}
        ORDER BY
            category ASC;
    """

    # Convert data into dictionary
    df = pd.read_sql(text(query), con = self.engine)
    data = df.to_dict(orient="records")
    return(data)
```

And lastly our corresponding app routes, shown below.

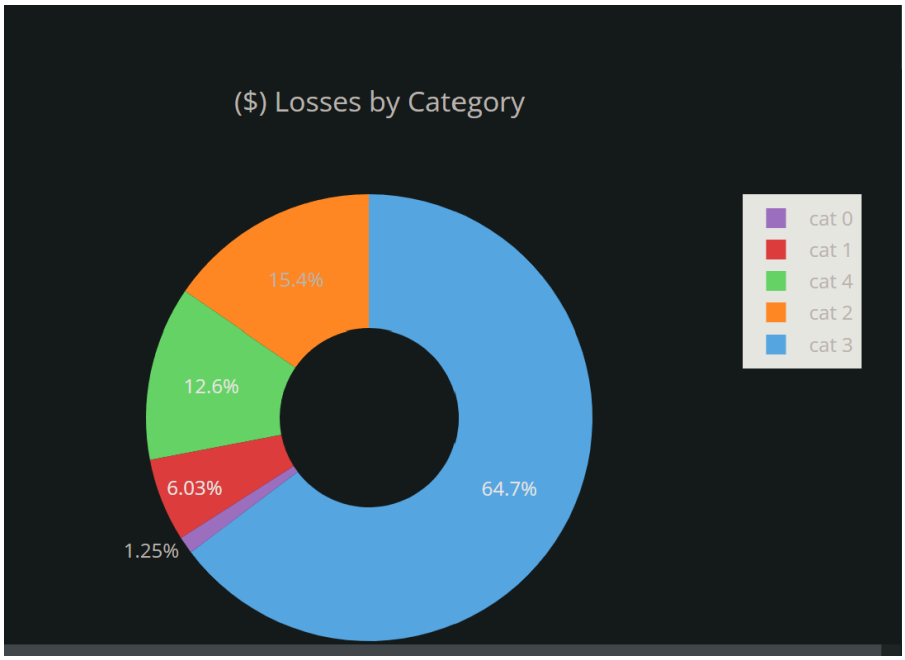
```python
# HTML ROUTES
@app.route("/")
def index():
    return render_template("home.html")

@app.route("/dashboard")
def dashboard():
    return render_template("dashboard.html")

@app.route("/map")
def map():
    return render_template("map.html")

@app.route("/about_us")
def about_us():
    return render_template("about_us.html")

@app.route("/work_cited")
def work_cited():
    return render_template("workcited.html")

# Dashboard SQL Query
@app.route("/api/v1.0/get_dashboard/<user_seasons>")
def get_dashboard(user_seasons):
    bar_data = sql.get_bar(user_seasons)
    pie_data = sql.get_pie(user_seasons)
    table_data = sql.get_table(user_seasons)
    data = {
        "bar_data": bar_data,
        "pie_data": pie_data,
        "table_data": table_data,
    }
    return(jsonify(data))

# Map SQL Query
@app.route("/api/map/<year>")
def get_map(year):
    year = int(year)
    map_data = sql.get_map(year)
    return(jsonify(map_data))

# Run the App
if __name__ == '__main__':
    app.run(debug=True)
```

Dashboard

Throughout this project we focused our dashboard to better visualize key information related to tornado occurrences in the United States. The dashboard aimed to provide insights into tornado seasons, severity, and associated impacts such as injuries, casualties, and damage costs. For our design we opted for a simple yet informative style. This style incorporated a bar chart, pie chart, and detailed table to present data effectively. The information for this page was categorized by seasons to get a deeper understanding of the concept of tornado seasonality and identify any trends in tornado occurrence throughout the year.
The bar chart was aimed at showcasing the number of injuries and casualties caused by tornadoes in different categories, emphasizing the severity of tornadoes in each season as shown below.



The pie chart aimed to visualize the total damage costs incurred by tornadoes in each category. Here we found that even though the category 4 tornadoes are individually more dangerous than category 3 tornadoes, the category 3 tornadoes caused the most monetary damage and human damage. We found this was because of the number of individual category 3 versus category 4 tornadoes. We used the pie chart to compare with the bar graph and come up with an estimate of the dangers of the tornado categories.



The last item in the dashboard was our table. Our table provides a comprehensive overview of the various statistics available to us in our data set. The table reiterates some points already shown by the bar and pie charts, but shows more detail. For ease of the user, we added filters to the table to get the information needed in a faster way.

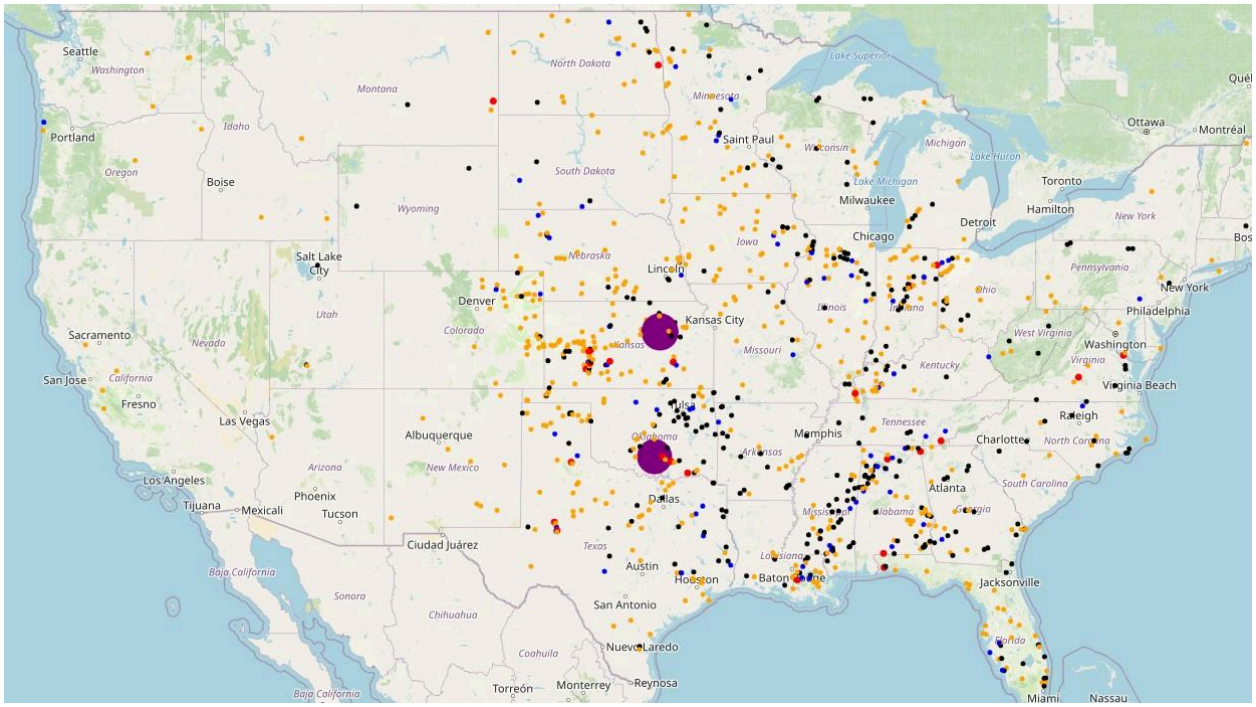
10 entries per page Search:

Year	Season	State	Category	Injuries	Fatalities	Losses
2016	spring	KS	4	8	0	0
2016	spring	OK	4	0	1	1000000
2016	fall	KS	3	0	0	0
2016	fall	AL	3	0	0	0
2016	fall	AL	3	9	4	0
2016	fall	TN	3	0	2	0
2016	winter	FL	3	3	0	5750000
2016	winter	LA	3	0	0	0
2016	winter	FL	3	3	0	22000000
2016	winter	VA	3	7	1	11200000

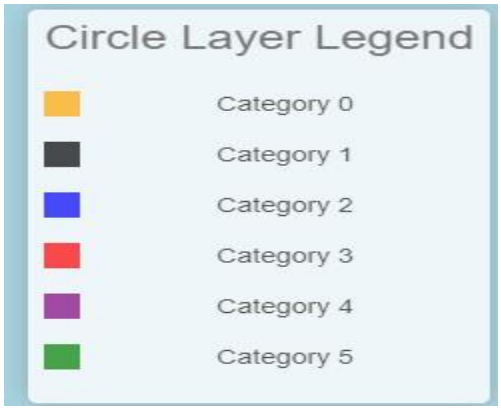
During the creation of the dashboard, we encountered a few challenges, notably with the bar chart displaying incorrect data due to miscommunication on the backend of the code. We were able to get the dashboard working properly during an impromptu session outside of our structured working hours. After ensuring that all aspects of the dashboard were working we switched our focus to optimizing performance for the user. Specifically we limited the table to only load a handful of items at a time and added search capabilities for easier navigation.

Map

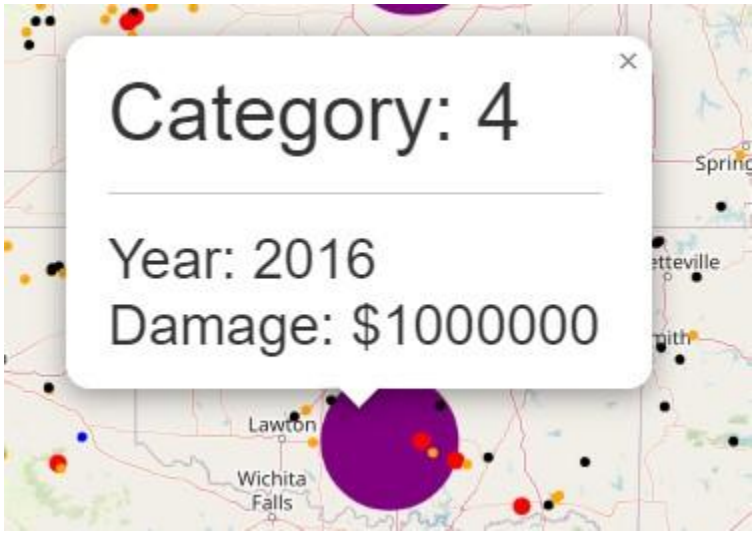
For our project a circle map was created to show tornado information for our dataset based on the year selected. Our in class activities were very helpful in assisting with this. The main issue that presented itself while getting this map to display was the syntax. However once corrected the circle map did a really great job of showing the precise locations of the tornadoes compared to the Marker or Heatmap. See below for an example.



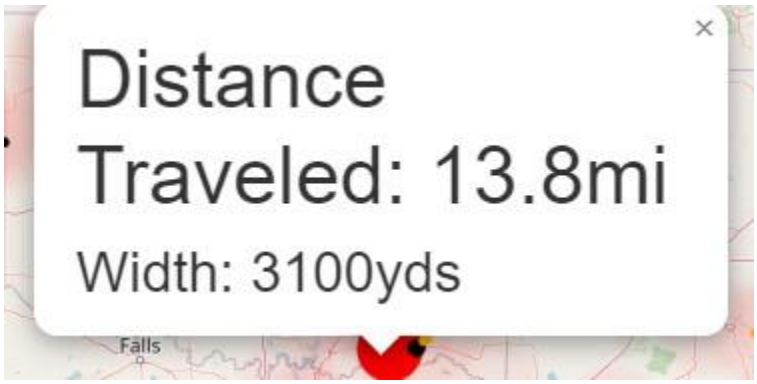
The circles were coded to be larger based on the category size of the tornadoes with category 0 being the smallest and category 5 the largest. It was a bit difficult to get the dot size of the category 0, 1, and 2 tornadoes to show as separate sizes. As a result the dots were color coordinated so categories could be easily distinguished. A legend was also created to identify the different colors.



Originally the use of different colors was desired to make visualizations more attractive. Unfortunately it was difficult to find ones that could be easily seen. For this reason the colors you see were selected instead. Finally, popups were coded into the map. For each dot on the map you can click on it and the category, year, and damage that tornado did would be displayed.



It is worth noting that what is stated above is the original way this map was coded. Changes however were made for the final project with the dots representing the location of the tornado, distance traveled being displayed, and dot size indicating the size of the tornado by width. The popup was also changed to show the distance the tornado traveled, as well as the width of the storm in yards.



Findings & Conclusions

Based on the dataset we compiled we were able to solve a few questions that we had. The first of which was which state had the most tornadoes. The answer is Texas. The next question that we wanted to answer was which year had the most tornadoes. 2019 had the most tornadoes, 1517 to be exact. Lastly, we wanted to learn what were the most destructive tornadoes. Since there were so many category 3 tornadoes compared to category 4, category 3 tornadoes accounted for the majority of the property losses and the most amount of injuries. However, category 4 tornadoes had the most fatalities.

Limitations

While we did choose a pretty encompassing dataset, that doesn't mean our data did not have any biases. The first bias of our data was it only contained data from 2016-2022. Therefore we cannot cast judgements on tornado data from this limited sample size. Another bias we had is that the tornado data is only from the United States. There are tornadoes in other countries besides the United State and could offer some interesting perspectives. The last bias we wanted to note was accounting. Tornadoes are recorded at the county level. All tornadoes regardless of rating are counted but some weaker storms with minimal losses might be missed or under reported.

Future Work

Future work could include climate change models to measure the impact of tornadoes. We could also create a map that showcases the paths of all the tornadoes. Lastly, we could incorporate AI to help build a predictive model for tornadoes.