Data Visualization and Analytics of US Accidents from 2016 to 2021

Brief Introduction

A data analysis of all the accidents in the United States of America from 2016 to December of 2021

Table of Contents

1. Overview of the Data Analytics and Visualization

This will cover cover the following visualizations of the dataset:

- Total accidents per year
- Top 15 states and their total accidents by severity
- Total accidents by weather conditions
- Total Accidents by temperature
- Distrubution of total accidents per state
- Heatmaps of the accidents of the entire US and per state
- Correlation between the factors in the analysis

2. Factors in the Analysis

- Weather conditions
- Wind speed
- Visiblity
- Precense of precipiation
- Temperature
- Severity

3. Data Processing

Below are the pre-requisites for processing the dataset:

IMPORTS

In [339…
```
# Packages that needs to be installed

#pip install pandas
#pip install pyarrow
#pip install datashader
#pip install "holoviews[recommended]"
#pip install hvplot
#pip install seaborn
#pip install bokeh
```

In [340…
```
# Imports for data processing
import pandas as pd
import numpy as np
from pyarrow.feather import _feather
```

In [341…
```
# Imports for descriptive analytics and for correlational analytics

# Matplotlib imports
import matplotlib.pyplot as plt

# Seaborn imports
import seaborn as sns

sns.set_theme(style="whitegrid")
```

In [342…

```
# Imports for map analyatics

# Datashader imports
import datashader as ds
import datashader.transfer_functions as tf
from datashader.utils import export_image
from datashader.utils import lnglat_to_meters as webm
from datashader.colors import colormap_select

# Holoviews imports
import holoviews as hv

# Bokeh imports
from bokeh.plotting import figure
from bokeh.io import show

# Utility imports
from functools import partial
from colorcet import fire
from IPython.display import HTML, display

hv.extension('bokeh', 'matplotlib')

%matplotlib inline
```

3.1 Reading the Dataset

Link for all dataset files

> Google Drive Link: https://drive.google.com/drive/folders/1EeLJvzltBig7DdLTiZ2_2siO1N2mifQ7?usp=share_link

In [343…
```
#us_accidents_dataset = pd.read_csv("US_Accidents_Dec21_updated.csv")
```

3.2 Converting the Dataset to a Faster Format

In [344…
```
#us_accidents_dataset.to_parquet("us_accidents_parquet.parq")
```

3.3 Cleaning the Dataset

In [345…
```
#us_accidents_dataset = pd.read_parquet("us_accidents_parquet.parq")
```

In [346…
```
# Removing columns that has more than 20% null values then removing all rows that has a null value
def clean_data_set(dataset):
    """Data cleaning of the dataset by removing columns that has more than 20% null values

    Args:
        dataset (dataframe): the dataset to be passed

    Returns:
        dataset: returns the clean dataset
    """

    dataset = dataset.dropna(axis='columns', thresh=int(0.80 * len(dataset)))
    #dataset = dataset.dropna(axis='rows')

    return dataset
```

In [347…
```
# The clean data set

""" us_accidents_dataset_clean = clean_data_set(us_accidents_dataset)
us_accidents_dataset_clean.head() """
```

Out[347]:
```
' us_accidents_dataset_clean = clean_data_set(us_accidents_dataset)\nus_accidents_dataset_clean.head() '
```

In [348…
```
# Printing the columns in the dataset and cherry-picking

#print(list(us_accidents_dataset_clean.columns.values))
```

We only need the following columns for the data analysis of this dataset:

Details from kaggle: https://www.kaggle.com/datasets/sobhanmoosavi/us-accidents

Severity - how severe the accident where in it ranges from 1 to 4 where 1 indicates the least impact on traffic
Start_Time - time when the accident initially occured
End_Lat - latitude in the gps
End_Lng - longitude in the gps
City - address of city
County - address of county
State - address of the state
Country - address country
Temperature - temperature that was present (in Fahrenheit)
Visibility - visibility at that time (in miles)
Wind_Speed - the wind speed in mph
Precipitation - amount of rainfall in inches
Weather_Condition - condition of wether at that time e.g. rain, thunderstorm, blizzard

In [349…
```
# Selecting the columns that we only need

""" us_accidents_dataset_clean = us_accidents_dataset_clean[['Severity', 'Start_Time', 'End_Lat', 'End_Lr
us_accidents_dataset_clean.head() """
```

Out[349]:
```
" us_accidents_dataset_clean = us_accidents_dataset_clean[['Severity', 'Start_Time', 'End_Lat', 'End_Lng
', 'City', 'County', 'State', 'Country', 'Temperature(F)', 'Visibility(mi)', 'Wind_Speed(mph)', 'Precipi
tation(in)', 'Weather_Condition']]\nus_accidents_dataset_clean.head() "
```

3.4 Optimizing the Dataset

In [350…
```
#us_accidents_dataset_clean.info()
```

In [351…
```
# Down casting int values

""" us_accidents_dataset_clean['Severity'] = us_accidents_dataset_clean['Severity'].astype('int8')
us_accidents_dataset_clean['End_Lat'] = us_accidents_dataset_clean['End_Lat'].astype('float32')
us_accidents_dataset_clean['End_Lng'] = us_accidents_dataset_clean['End_Lng'].astype('float32')
us_accidents_dataset_clean['Temperature(F)'] = us_accidents_dataset_clean['Temperature(F)'].astype('float
us_accidents_dataset_clean['Visibility(mi)'] = us_accidents_dataset_clean['Visibility(mi)'].astype('float
us_accidents_dataset_clean['Wind_Speed(mph)'] = us_accidents_dataset_clean['Wind_Speed(mph)'].astype('flo
us_accidents_dataset_clean['Precipitation(in)'] = us_accidents_dataset_clean['Precipitation(in)'].astype(
```

Out[351]:
```
" us_accidents_dataset_clean['Severity'] = us_accidents_dataset_clean['Severity'].astype('int8')\nus_acc
idents_dataset_clean['End_Lat'] = us_accidents_dataset_clean['End_Lat'].astype('float32')\nus_accidents_
dataset_clean['End_Lng'] = us_accidents_dataset_clean['End_Lng'].astype('float32')\nus_accidents_dataset
_clean['Temperature(F)'] = us_accidents_dataset_clean['Temperature(F)'].astype('float32')\nus_accidents_
dataset_clean['Visibility(mi)'] = us_accidents_dataset_clean['Visibility(mi)'].astype('float32')\nus_acc
idents_dataset_clean['Wind_Speed(mph)'] = us_accidents_dataset_clean['Wind_Speed(mph)'].astype('float32'
)\nus_accidents_dataset_clean['Precipitation(in)'] = us_accidents_dataset_clean['Precipitation(in)'].ast
ype('float32') "
```

In [352…
```
# Rounding off latitude and longitude values

""" us_accidents_dataset_clean['End_Lat'].apply(lambda x: '%.2f' % x)
us_accidents_dataset_clean['End_Lng'].apply(lambda x: '%.2f' % x) """
```

Out[352]:
```
" us_accidents_dataset_clean['End_Lat'].apply(lambda x: '%.2f' % x)\nus_accidents_dataset_clean['End_Lng
'].apply(lambda x: '%.2f' % x) "
```

In [353…
```
# Converting latitude and longtitude values to the web mercator fomrmat

#us_accidents_dataset_clean.loc[:, 'Easting'], us_accidents_dataset_clean.loc[:, 'Northing'] = webm(us_ac

# Rounding off to decimal places to remove scientific notation

""" us_accidents_dataset_clean['Easting'].apply(lambda x: '%.2f' % x)
us_accidents_dataset_clean['Northing'].apply(lambda x: '%.2f' % x) """

# Converting back to float

""" us_accidents_dataset_clean['Easting'] = us_accidents_dataset_clean['Easting'].astype('float32')
us_accidents_dataset_clean['Northing'] = us_accidents_dataset_clean['Northing'].astype('float32') """
```

Out[353]:
```
" us_accidents_dataset_clean['Easting'] = us_accidents_dataset_clean['Easting'].astype('float32')\nus_ac
cidents_dataset_clean['Northing'] = us_accidents_dataset_clean['Northing'].astype('float32') "
```

In [354…
```
#us_accidents_dataset_clean.head()
```

In [355…
```
# Optimizing the date and time

""" us_accidents_dataset_clean['Start_Time'] = pd.to_datetime(us_accidents_dataset_clean['Start_Time'])
us_accidents_dataset_clean['Date'] = us_accidents_dataset_clean['Start_Time'].dt.normalize() """
```

Out[355]:
```
" us_accidents_dataset_clean['Start_Time'] = pd.to_datetime(us_accidents_dataset_clean['Start_Time'])\nu
s_accidents_dataset_clean['Date'] = us_accidents_dataset_clean['Start_Time'].dt.normalize() "
```

In [356…
```
# Dropping start time since we no longer need it

#us_accidents_dataset_clean.drop(['Start_Time'], axis=1, inplace=True)
```

In [357…
```
#us_accidents_dataset_clean.head()
```

In [358…
```
# Write again to parquet

#us_accidents_dataset_clean.to_parquet("us_accidents_final.parq")
```

4. Data Visualization

In [359…
```python
# Reads the final processed version

us_accidents_dataset_final = pd.read_parquet("us_accidents_final.parq")
```

Important Note:

> When you run the plots and all of them are in stacked with each other, just run the idividual plot again

4.1 Descriptive Analytics

> ### HELPER FUNCTIONS

In [360…
```python
# HELPER FUNCTIONS

def group_aggregate(dataset, group_by_column, agg_columns: dict):
    """This is used to groupby a dataset then also aggregated the given columns

    Args:
        dataset (dataframe): the dataset to be passed
        group_by_column (column): the column to be group by
        agg_columns (dict): the column(s) to be aggegrated

    Returns:
        dataset: returns the grouped and aggregated dataset
    """

    dataset = dataset.copy()
    aggregated_dataset = dataset.groupby([group_by_column]).agg(agg_columns).reset_index()
```

```
    return aggregated_dataset


def get_n_rows_sorted(dataset, n_rows, column):
    """Gets the largest values in the dataset then sorts it in a descending order

    Args:
        dataset (dataframe): the dataset to be passed
        n_rows (int): the number of rows to be sorted
        column (int): the column to be sorted

    Returns:
        dataset: returns a dataset that is sorted in a descending order with the largest values
    """

    return dataset.nlargest(n_rows, column, keep='last')

def round_off(dataset, column, sig_figures):
    """This rounds of the values of a column that contains float type then returns a float value of it

    Args:
        dataset (dataframe): the dataset to be passed
        column (float): the column to be rounded off (must be a float)
        sig_figures (int): the number of significant figures for rounding off

    Returns:
        column: returns the rounded off column
    """

    return dataset.round({column: sig_figures})

def plot_barplot(dataset, x_range, y_range, order, color, orient):
    """Plots a bar plot

    Args:
        dataset (dataframe): dataset to be oassed
        x_range (any): the x_range of the plot
        y_range (any): the y_range of the plot
        order (any): how the values will be arranged
```

```
        color (str): pallete of the plot
        orient (str): vertical or horizontal (v or h)
    """

    sns.barplot(data=dataset, x=x_range, y=y_range, order=dataset[order], palette=color, orient=orient)
```

In [361…

```
# Addes the accidents for each state then puts on a new column called Total_Accidents
us_accidents_total = us_accidents_dataset_final.copy()
us_accidents_total["Total_Accidents"] = us_accidents_total["State"].map(us_accidents_total["State"].value
us_accidents_total.info()
```

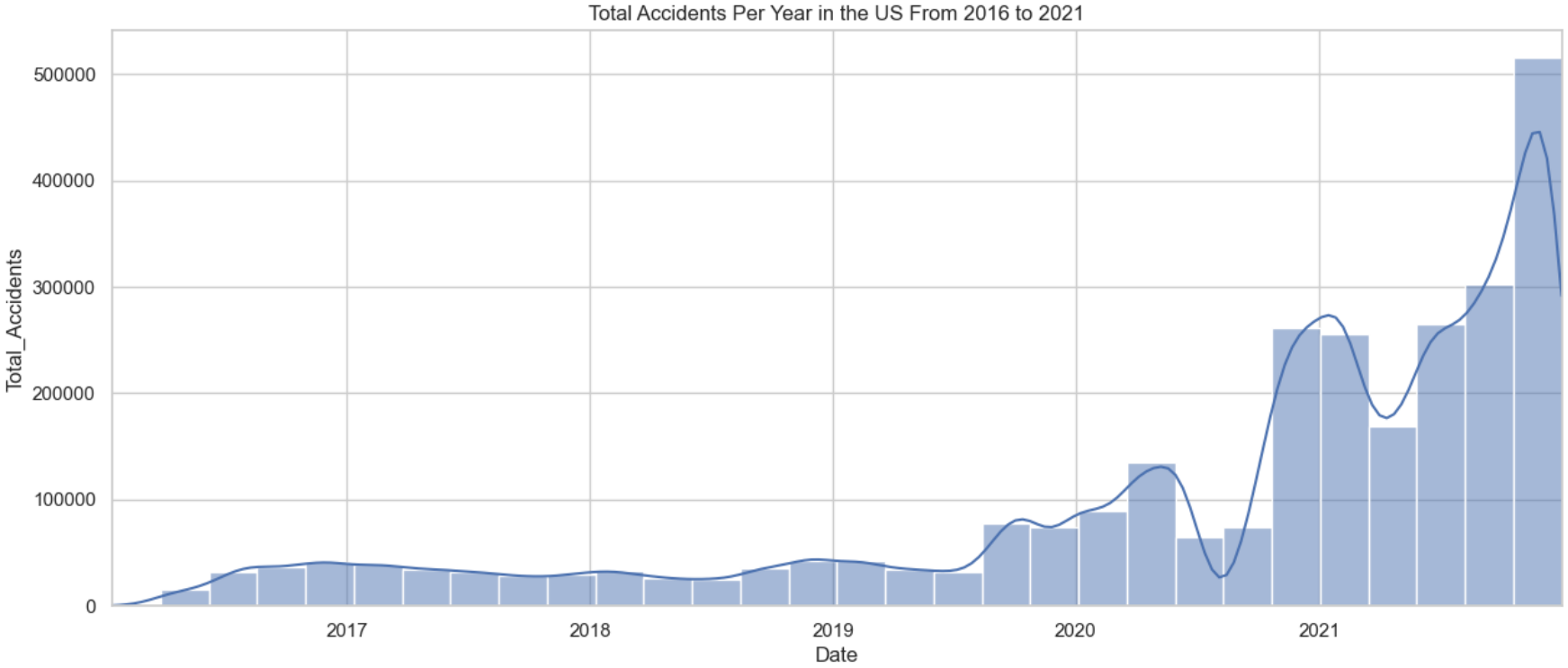
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2845342 entries, 0 to 2845341
Data columns (total 16 columns):
 #   Column             Dtype
---  ------             -----
 0   Severity           int8
 1   End_Lat            float32
 2   End_Lng            float32
 3   City               object
 4   County             object
 5   State              object
 6   Country            object
 7   Temperature(F)     float32
 8   Visibility(mi)     float32
 9   Wind_Speed(mph)    float32
 10  Precipitation(in)  float32
 11  Weather_Condition  object
 12  Easting            float32
 13  Northing           float32
 14  Date               datetime64[ns]
 15  Total_Accidents    int64
dtypes: datetime64[ns](1), float32(8), int64(1), int8(1), object(5)
memory usage: 241.5+ MB
```

4.1a All 50 states and Their Total Accidents From 2016 to 2021

In [362…
```
# Copies the us_accidents_total data set into a new variable called us_accidents_timespan
us_accidents_timespan = us_accidents_total.copy()
```

In [363…
```
# Plotting histogram of the accidents per year from 2016 to 2021 in the US

f, ax = plt.subplots(figsize=(15, 6))
plt.ticklabel_format(style='plain', axis='y')

fig = sns.histplot(data=us_accidents_timespan, x='Date', bins=30, kde=True)
fig.set_xlim(us_accidents_timespan['Date'].min(), us_accidents_timespan['Date'].max())
fig.set_ylabel("Total_Accidents")

plt.title("Total Accidents Per Year in the US From 2016 to 2021")
```

Out[363]: Text(0.5, 1.0, 'Total Accidents Per Year in the US From 2016 to 2021')



> Given above, we can say that 2021 was a spike in the total accidents per year. But upon observing we can see that the around the middle of 2020, accidents dropped by a quiet a lot actually as 2020 was the time of lockdowns and quaratines due to the Covid-19 pandemic. 2016 to 2019 was ranging betwen a few thousands to tens of thounsands but 2021 was a huge spike as that is the time that US started to loosen the lockdowns in their country.

4.1b Top 15 States and Their Total Accidents by Severity

In [364…

```
# The aggregrated top 15 states and their total accidents and the severity for each state

us_accidents_top15 = group_aggregate(dataset=us_accidents_total, group_by_column='State', agg_columns={'T
us_accidents_top15 = get_n_rows_sorted(dataset=us_accidents_top15, n_rows=15, column='Total_Accidents')
us_accidents_top15 = round_off(dataset=us_accidents_top15, column='Severity', sig_figures=1)
us_accidents_top15.head(15)
```

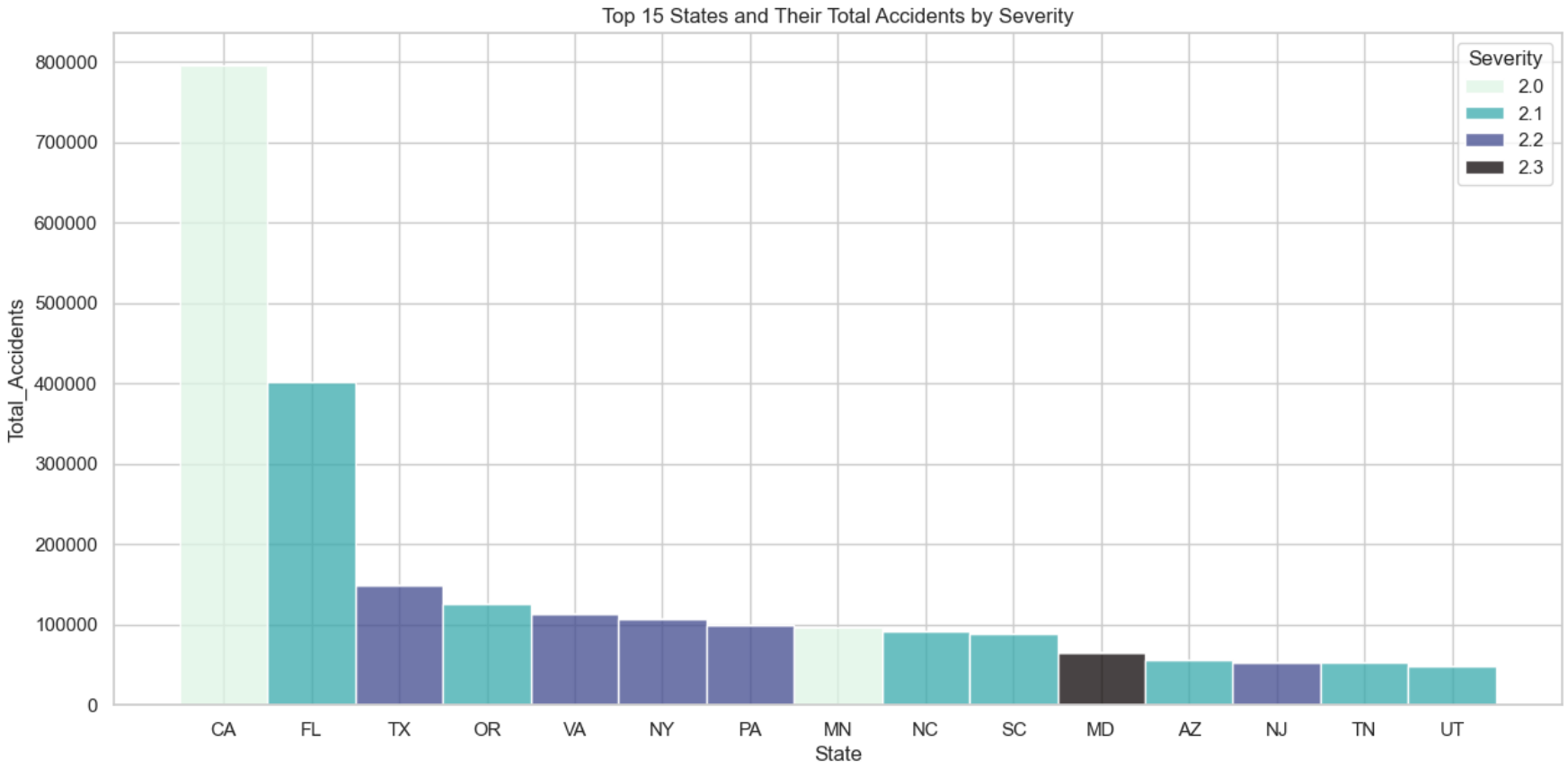
Out[364]:

	State	Total_Accidents	Severity
3	CA	795868	2.0
8	FL	401388	2.1
41	TX	149037	2.2
35	OR	126341	2.1
43	VA	113535	2.2
32	NY	108049	2.2
36	PA	99975	2.2
21	MN	97185	2.0
25	NC	91362	2.1
38	SC	89216	2.1
18	MD	65085	2.3
2	AZ	56504	2.1
29	NJ	52902	2.2
40	TN	52613	2.1
42	UT	49193	2.1

In [365…

```
# Plot the top 15 states and their total accidents by severity
f, ax = plt.subplots(figsize=(15, 7))

fig = sns.histplot(data=us_accidents_top15, x='State', hue='Severity', weights='Total_Accidents', multipl
fig.set_ylabel('Total_Accidents')
legend = fig.get_legend()
legend.set_bbox_to_anchor((1, 1))

plt.title("Top 15 States and Their Total Accidents by Severity")
```

Out[365]: Text(0.5, 1.0, 'Top 15 States and Their Total Accidents by Severity')



In [366…

```
# Shows the top 15 states with the most accidents
us_accidents_top15.head(15)
```

Out[366]:

	State	Total_Accidents	Severity
3	CA	795868	2.0
8	FL	401388	2.1
41	TX	149037	2.2
35	OR	126341	2.1
43	VA	113535	2.2
32	NY	108049	2.2
36	PA	99975	2.2
21	MN	97185	2.0
25	NC	91362	2.1
38	SC	89216	2.1
18	MD	65085	2.3
2	AZ	56504	2.1
29	NJ	52902	2.2
40	TN	52613	2.1
42	UT	49193	2.1

> We can see here the state with the most accidents is California while New Jersey being with the least amount. Although the accidents of California may be expontial but its severity is actually reasonable at 2.0 while the least state with accidents actually has the most severe severity of 2.2 while other states average from 2.0 to 2.1.

4.1c Total Accidents by Weather Conditions

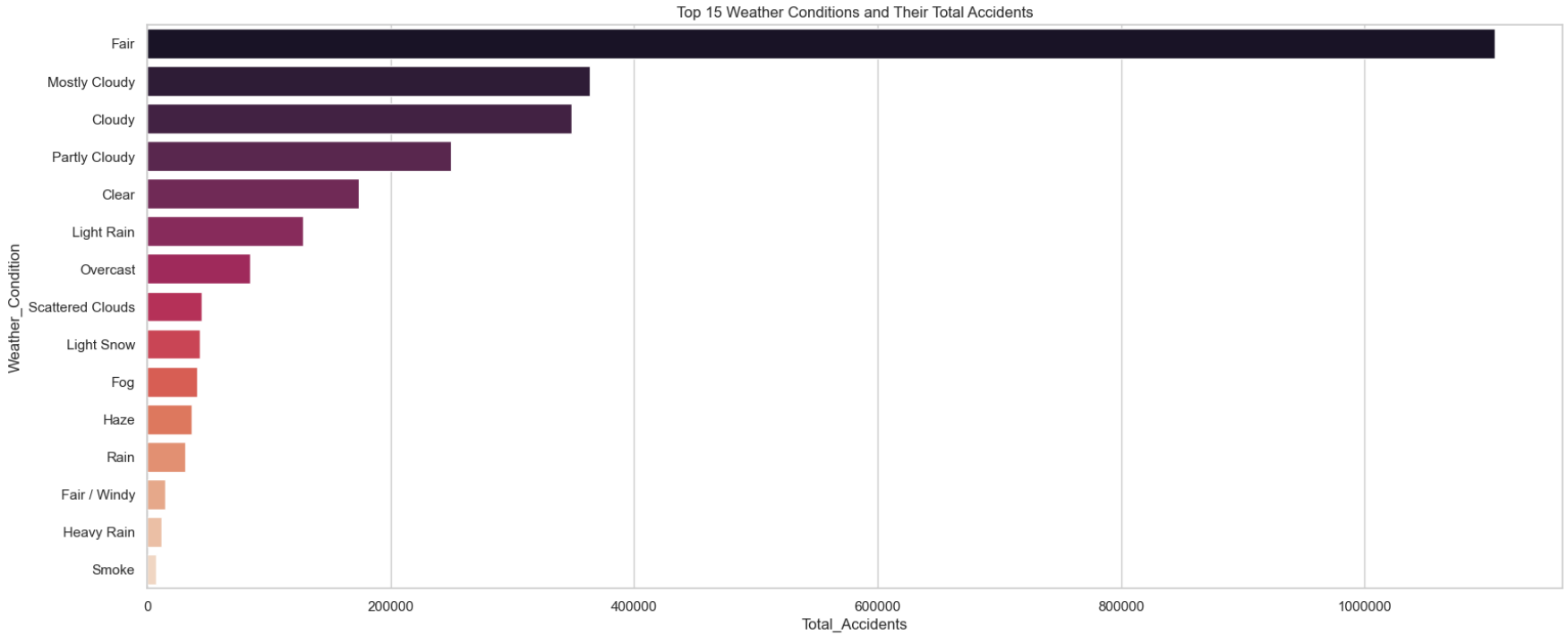
In [367…
```
# Grouping the accidents by weather then aggregrating the total accidents and the mean of the total sever
us_accidents_top15_weather = us_accidents_total.copy()
us_accidents_top15_weather = group_aggregate(dataset=us_accidents_top15_weather, group_by_column='Weather
us_accidents_top15_weather = get_n_rows_sorted(dataset=us_accidents_top15_weather, n_rows=15, column='Tot
us_accidents_top15_weather = round_off(dataset=us_accidents_top15_weather, column='Severity', sig_figures
us_accidents_top15_weather.head()
```

Out[367]:

	Weather_Condition	Total_Accidents	Severity
15	Fair	1107194	2.1
76	Mostly Cloudy	363959	2.1
7	Cloudy	348767	2.1
81	Partly Cloudy	249939	2.1
6	Clear	173823	2.5

In [368…
```
# Creating the top 15 weather conditions and ther total accidents by severity

f, ax = plt.subplots(figsize=(20, 8))

plt.ticklabel_format(style='plain', axis='x')

plot_barplot(dataset=us_accidents_top15_weather, x_range='Total_Accidents', y_range='Weather_Condition',

plt.title("Top 15 Weather Conditions and Their Total Accidents")
```

Out[368]:
```
Text(0.5, 1.0, 'Top 15 Weather Conditions and Their Total Accidents')
```



> Upon observing, we can state that weather condition 'fair' is the condition where most of the accidents occured. While the weather condition of smoke or smokey, the accidents that occured is rare and marginal. So this means that these accidents occured mostly on the 'average' or 'good' weather condition.

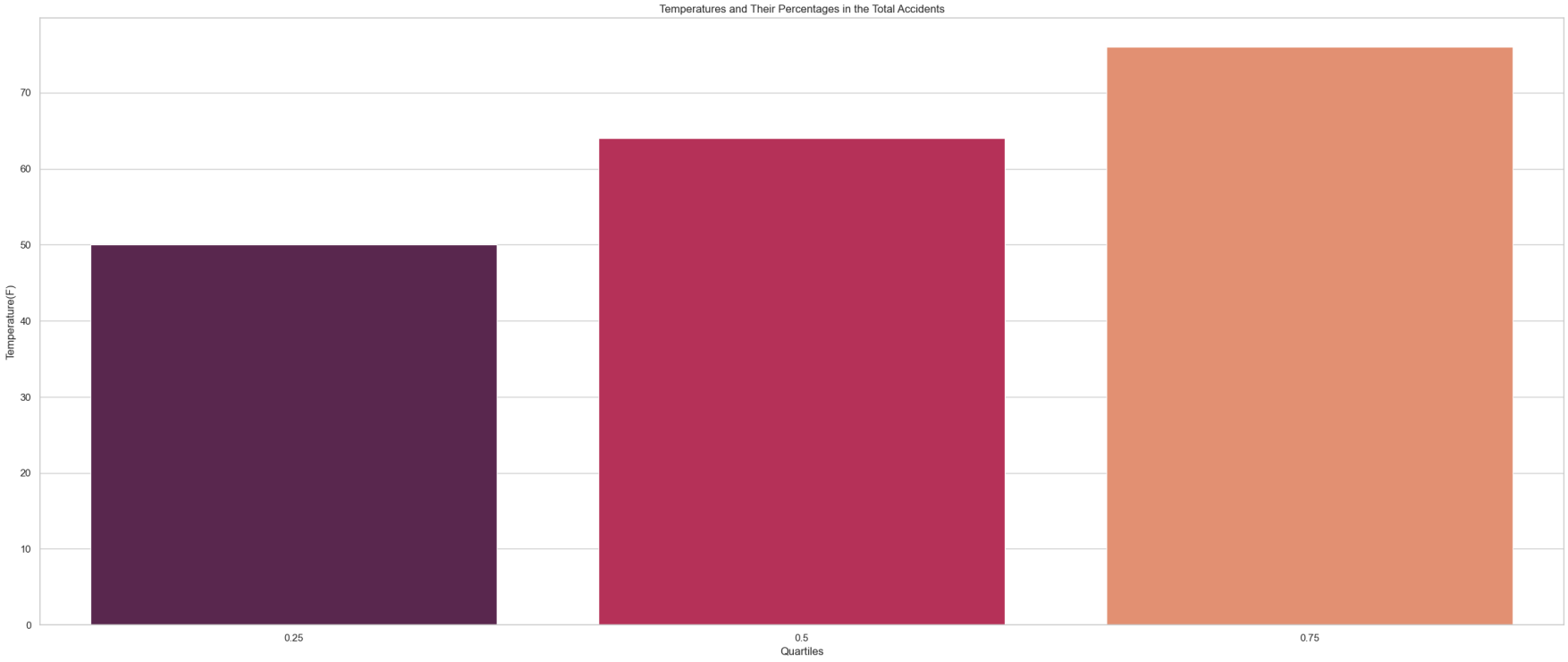
4.1d Total Accidents by Temperature

In [369…

```
# Gets the quartiles of the temperatures
us_accidents_temps = us_accidents_total.copy()
us_accidents_temps = us_accidents_temps['Temperature(F)'].quantile([.25, .5, .75]).rename_axis('Quartiles

# 0.25 = 25%
# 0.50 = 50%
# 0.75 = 75%
```

In [370…
```
# Creating the top 15 temperature levels and their total accidents

f, ax = plt.subplots(figsize=(30, 12))
plt.ticklabel_format(style='plain', axis='y')

plot_barplot(dataset=us_accidents_temps, x_range='Quartiles', y_range='Temperature(F)', order='Quartiles'

plt.title("Temperatures and Their Percentages in the Total Accidents")
```

Out[370]:
```
Text(0.5, 1.0, 'Temperatures and Their Percentages in the Total Accidents')
```



In [371…
```
# Gets the quartiles and values of the temperatures
us_accidents_temps.head()
```

Out[371]:

	Quartiles	Temperature(F)
0	0.25	50.0
1	0.50	64.0
2	0.75	76.0

> Given the bar plot above, we can dictate that 75% of the total accidents occured on a temperature of 76 farenheit while 50% occur 64 farehenheit and with most least occuring on is 50 farehneit. 76 farehnheit can be obsevered as 'fair' weather which we can cross-check from the bar plot from 4.1c which they have the same theory that most of the accidents occur on a fair weather. While a quarter of the accidents occur on rather 'cold' conditions as which can be felt at in the range of 50 farenheit.

4.2 Map Analytics

This section will focus on the aggregated accidents account of all states and heatmaps

4.2a Choropleth Map of the US Total Accidents by State

In [372…

```
# Imports for the choropleth map specifically

from bokeh.models import LogColorMapper, ColorBar
from bokeh.palettes import Oranges256 as oranges
from bokeh.sampledata.us_states import data as us_states
```

In [373…
```
# Setting contstants for the plots

PLOT_WIDTH = 900
PLOT_HEIGHT = 600
LINE_COLOR = "white"
LINE_WIDTH = 0.5
```

In [374…
```
# Prepation of the aggregated dataset

us_accidents_per_state = us_accidents_total.copy()
us_accidents_per_state = group_aggregate(dataset=us_accidents_per_state, group_by_column='State', agg_col
```

In [375…
```
# Converting the us_states to a dataframe as and getting the longtitude and latitude values

us_states_total = pd.DataFrame(us_states).T
us_states_total = us_states_total[~us_states_total["name"].isin(['Alaska', "Hawaii"])]
us_states_total["lons"] = us_states_total.lons.values.tolist()
us_states_total["lats"] = us_states_total.lats.values.tolist()
us_states_total = us_states_total.reset_index()
us_states_total.head()
```

Out[375]:

	index	name	region	lats	lons
0	NV	Nevada	Southwest	[40.68928, 40.4958, 40.30302, 40.09896, 39.999...	[-114.04392, -114.04558, -114.04619, -114.0464...
1	AZ	Arizona	Southwest	[34.87057, 35.00186, 35.00332, 35.07971, 35.11...	[-114.63332, -114.63349, -114.63423, -114.6089...
2	WI	Wisconsin	Central	[42.49273, 42.49433, 42.49562, 42.49561, 42.49...	[-87.8156, -87.93137, -88.10268, -88.20645, -8...
3	GA	Georgia	Southeast	[32.29667, 32.24425, 32.09197, 32.03256, 32.02...	[-81.12387, -81.15654, -81.02071, -80.75203, -...
4	KS	Kansas	Central	[36.99927, 36.99879, 36.99914, 36.99903, 36.99...	[-96.28415, -96.55381, -96.91244, -97.1197, -9...

In [376…
```
# Merging the US boundary dataframe and the aggregrated US accidents per state

us_states_total = us_states_total.merge(us_accidents_per_state[['State','Total_Accidents']], how='left',
us_states_total.head()
```

Out[376]:

	index	name	region	lats	lons	State	Total_Accidents
0	NV	Nevada	Southwest	[40.68928, 40.4958, 40.30302, 40.09896, 39.999...	[-114.04392, -114.04558, -114.04619, -114.0464...	NV	6197
1	AZ	Arizona	Southwest	[34.87057, 35.00186, 35.00332, 35.07971, 35.11...	[-114.63332, -114.63349, -114.63423, -114.6089...	AZ	56504
2	WI	Wisconsin	Central	[42.49273, 42.49433, 42.49562, 42.49561, 42.49...	[-87.8156, -87.93137, -88.10268, -88.20645, -8...	WI	7896
3	GA	Georgia	Southeast	[32.29667, 32.24425, 32.09197, 32.03256, 32.02...	[-81.12387, -81.15654, -81.02071, -80.75203, -...	GA	40086
4	KS	Kansas	Central	[36.99927, 36.99879, 36.99914, 36.99903, 36.99...	[-96.28415, -96.55381, -96.91244, -97.1197, -9...	KS	9033

In [377…
```
# Puts the list values of the dataset into a dictionary

us_accidents_choropleth = {}
us_accidents_choropleth['lons'] = us_states_total.lons.values.tolist()
us_accidents_choropleth['lats'] = us_states_total.lats.values.tolist()
us_accidents_choropleth["name"] = us_states_total.name.values.tolist()
us_accidents_choropleth["StateCodes"] = us_states_total.index.values.tolist()
us_accidents_choropleth['TotalAccidents'] = us_states_total.Total_Accidents.values.tolist()
```

In [378…
```
# Plotting the choropleth map
fig = figure(plot_width=PLOT_WIDTH, plot_height=PLOT_HEIGHT, title="US Accidents Per State Choropleth Map
           x_axis_location=None, y_axis_location=None,
           tooltips=[("State","@name"), ("TotalAccidents", "@TotalAccidents"), ("(Long, Lat)", "($x, $y)
           )

fig.grid.grid_line_color = None

color_mapper = LogColorMapper(palette=oranges[::-1], low=us_accidents_total['Total_Accidents'].min(), hig

color_bar = ColorBar(color_mapper=color_mapper)

fig.patches("lons", "lats", source=us_accidents_choropleth,
           fill_color={'field': 'TotalAccidents', 'transform': color_mapper},
           fill_alpha=0.7, line_color=LINE_COLOR, line_width=LINE_WIDTH)

fig.add_layout(color_bar, "right")

show(fig)
```

4.2b The Heatmap of US Accidents From 2016 to 2021

In [379…
```
map_total_accidents = pd.read_parquet('us_accidents_final.parq', columns=['Easting', 'Northing'])
len(map_total_accidents)
map_total_accidents.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2845342 entries, 0 to 2845341
Data columns (total 2 columns):
 #   Column    Dtype
---  ------    -----
 0   Easting   float32
 1   Northing  float32
dtypes: float32(2)
memory usage: 21.7 MB
```

In [380…
```
# Setting the width and height of the heatmap
plot_width  = int(3840)
plot_height = int(2160)
```

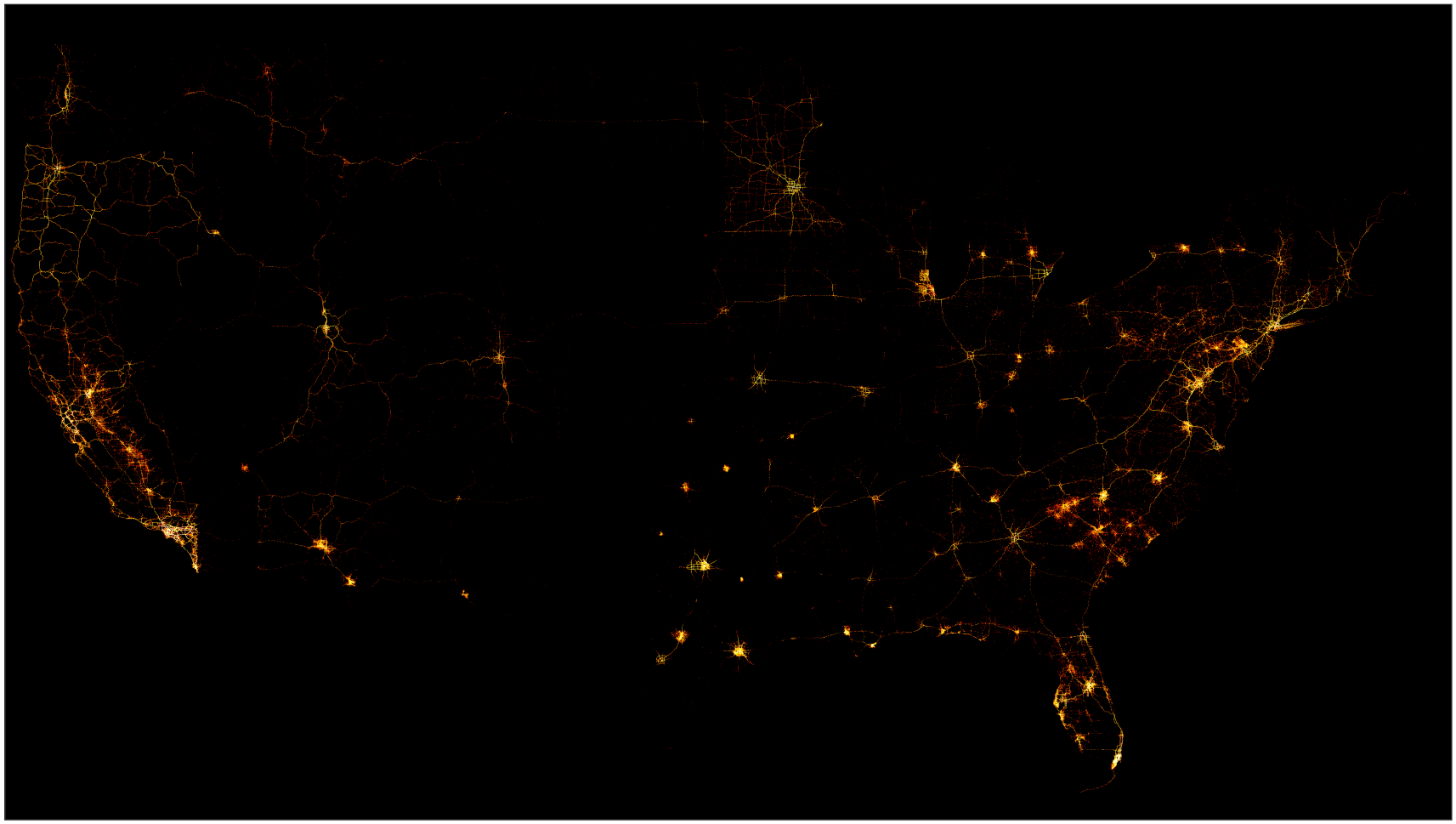
In [381…
```
# Setting the boundaries of the of the heatmap which is the entire US

USA = ((-124.72,  -66.95), (23.55, 50.06))
```

In [382…
```
# Setting the heatmap colormaps and background

background = "black"

export = partial(export_image, background=background, export_path="export")
cm = partial(colormap_select, reverse=(background!="black"))

display(HTML("<style>.container  { width:100% !important; }</style>"))
```

In [383…
```
def render_heatmap(dataset, x_range, y_range, file_name, location = None):
    """This function is created to reduce code duplication and to also take in
       arguments in one line. The main goal of this function to render a heatmap
       the chosen dataset with the following paramters:

    Args:
        dataset (dataframe): dataset or dataframe
        x_range (float): x_axis in the mercator format
        y_range (float): y_axis in the mercator format
        location (tuple): the boundaries of the location

    Returns:
        image: returns a render of the heatmap and exports it locally in a 'export' folder
    """

    # Checks if a location parameter is passed if there is
    # then will return it with converted web mercator format,
    # if none then it will not pass in the location into the canvas
    #
    # note: the location must be tuples of tuples

    if location is not None:
        cvs = ds.Canvas(plot_width, plot_height, *webm(*location))
    elif location is None:
        cvs = ds.Canvas(plot_width, plot_height)

    agg = cvs.points(dataset, x_range, y_range)

    return export(tf.shade(agg, cmap=cm(fire), how='eq_hist'), file_name)
```

In [384…
```
# Rendering and exporting the heatmap into 4k image and render
render_heatmap(dataset=map_total_accidents, x_range='Easting', y_range='Northing', location=USA, file_nam
```

Out[384]:



> As seen above, a pixel represents an accident and we can see that the east side of the US alone is scattered with points and specifically the state of California. We can support this statement as we plotted above at the '4.2a Choropleth Map' section, we can say that California has the most accidents accumulated from 2016 to 2021. If we look closer, we can observe that majority of these accidents are in cities or largely condensed areas that are urbanized as we can see in the heatmap above.

4.2c Top 5 States and Their Heatmaps

In [385…
```
# Geting the top 5 cities first and their total accidents

us_accidents_top5_states = us_states_total.copy()
us_accidents_top5_states = get_n_rows_sorted(dataset=us_accidents_top5_states, n_rows=5, column='Total_Ac
us_accidents_top5_states
```

Out[385]:

	index	name	region	lats	lons	State	Total_Accidents
46	CA	California	Southwest	[37.77205, 37.77078, 37.76913, 37.76387, 37.75...	[-123.00111, -122.99754, -122.99509, -122.9874...	CA	795868
33	FL	Florida	Southeast	[24.72148, 24.72333, 24.72623, 24.72628, 24.70...	[-82.88318, -82.87484, -82.86562, -82.80018, -...	FL	401388
48	TX	Texas	Southwest	[33.56679, 33.56763, 33.55209, 33.57438, 33.59...	[-94.26958, -94.26926, -94.23197, -94.19515, -...	TX	149037
43	OR	Oregon	Northwest	[46.29443, 46.29684, 46.2584, 46.14706, 46.145...	[-124.03622, -124.0356, -123.55518, -123.37257...	OR	126341
13	VA	Virginia	Mid-Atlantic	[38.90237, 38.90084, 38.8951, 38.88678, 38.880...	[-77.07827, -77.06992, -77.06611, -77.06283, -...	VA	113535

In [386…
```
# Helper function for finding the state boundaries and the coordinates of their accidents

def get_per_state_accidents(dataset, state_id):

    return dataset.loc[dataset['State'] == state_id]
```

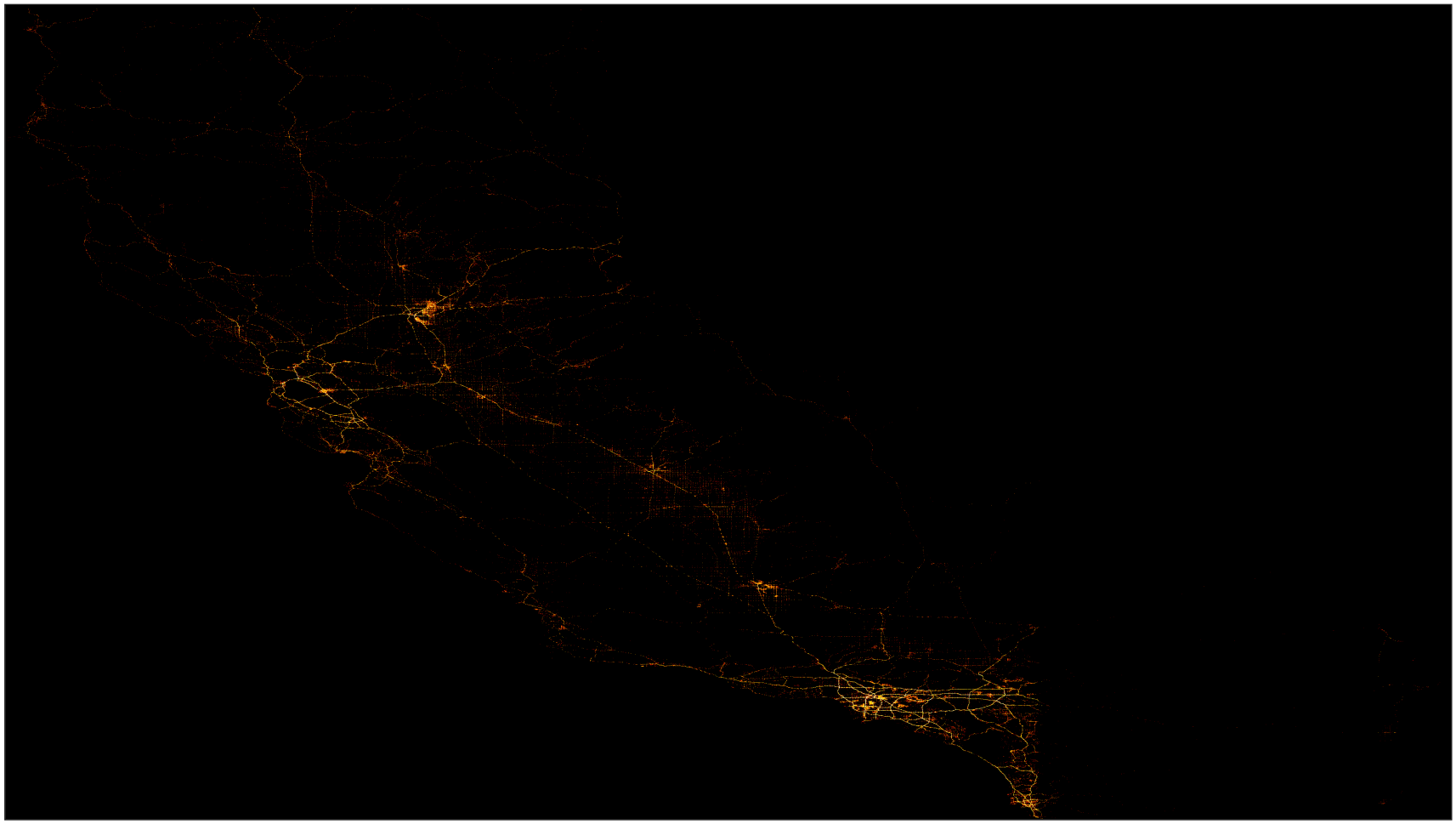
In [387…
```
# Getting the boundaries of the cities

# STATE ID
# CA: California
# FL: Florida
# TX: Texas
# OR: Oregon
# VA: Virginia

us_states_heatmap = us_accidents_total.copy()
us_states_heatmap = us_states_heatmap[['State', 'Easting', 'Northing']]

# Accident coordinates of the top 5 statess

CALIFORNIA = get_per_state_accidents(us_states_heatmap, state_id='CA')
FLORIDA    = get_per_state_accidents(us_states_heatmap, state_id='FL')
TEXAS      = get_per_state_accidents(us_states_heatmap, state_id='TX')
OREGON     = get_per_state_accidents(us_states_heatmap, state_id='OR')
VIRGINIA   = get_per_state_accidents(us_states_heatmap, state_id='VA')
```

In [388…
```
# Create the heatmaps

california_heatmap = render_heatmap(dataset=CALIFORNIA, x_range='Easting', y_range='Northing', file_name=
florida_heatmap    = render_heatmap(dataset=FLORIDA, x_range='Easting', y_range='Northing', file_name="Fl
texas_heatmap      = render_heatmap(dataset=TEXAS, x_range='Easting', y_range='Northing', file_name="Texa
oregon_heatmap     = render_heatmap(dataset=OREGON, x_range='Easting', y_range='Northing', file_name="Ore
virginia_heatmap   = render_heatmap(dataset=VIRGINIA, x_range='Easting', y_range='Northing', file_name="V
```

California Heatmap

In [389…
```
california_heatmap
```

Out[389]:

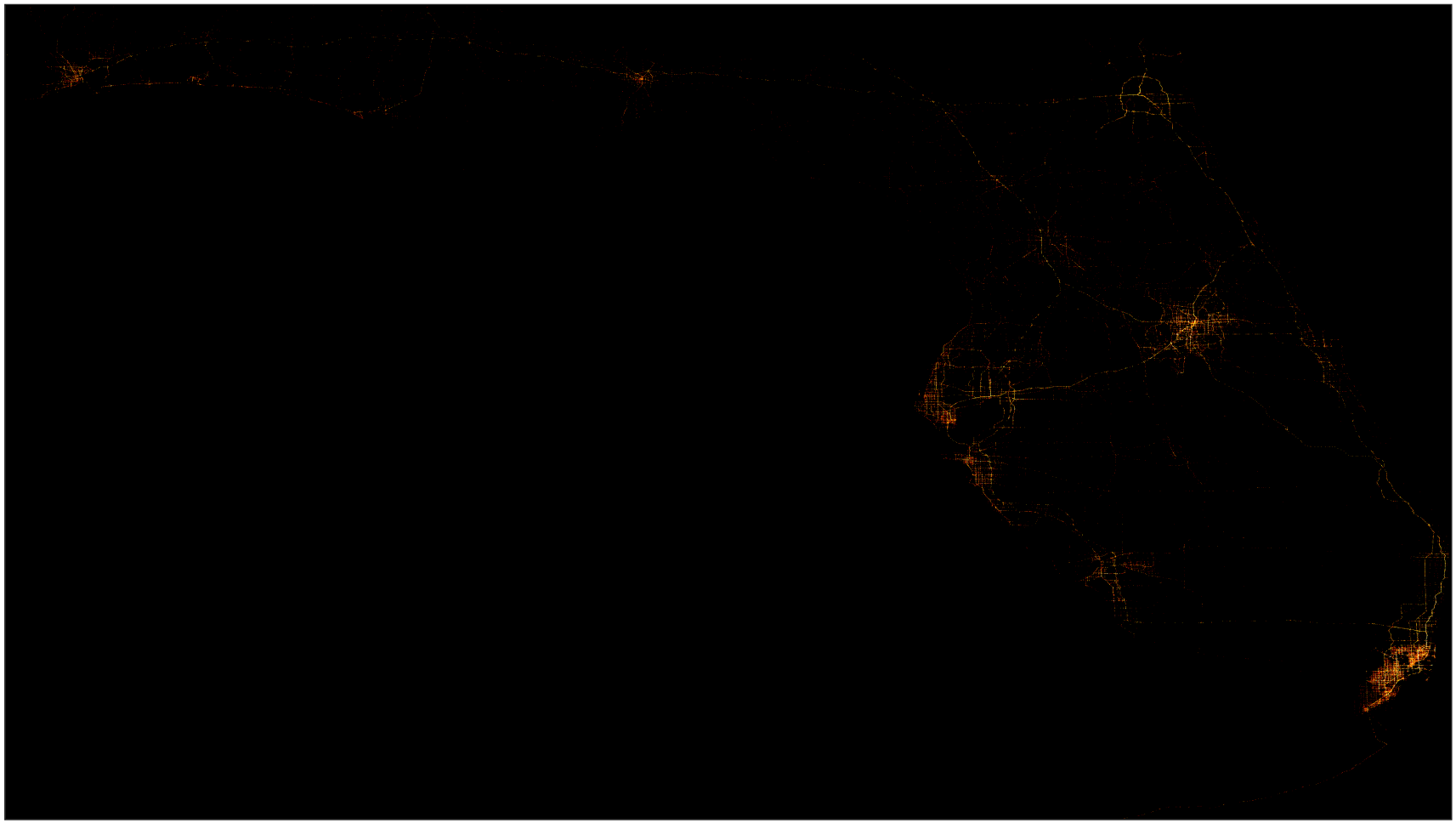


> Looking at this heatmap, we can definitely say that California has the most accidents from 2016 to 2021. As stated from *("'True Crime: On the Run' Gives an Inside Look at Southern California's High-speed Police Chases," 2022)* of ABC news, California in 2020 alone had 2,200 police pursuits and that was the year when the COVID-19 Pandemic was on the rise. It was also stated that the top reason for all these chases are stealing vehicles or car jacking which explains why California is the state with the most accidents.

Florida Heatmap

In [390…
```
florida_heatmap
```

Out[390]:



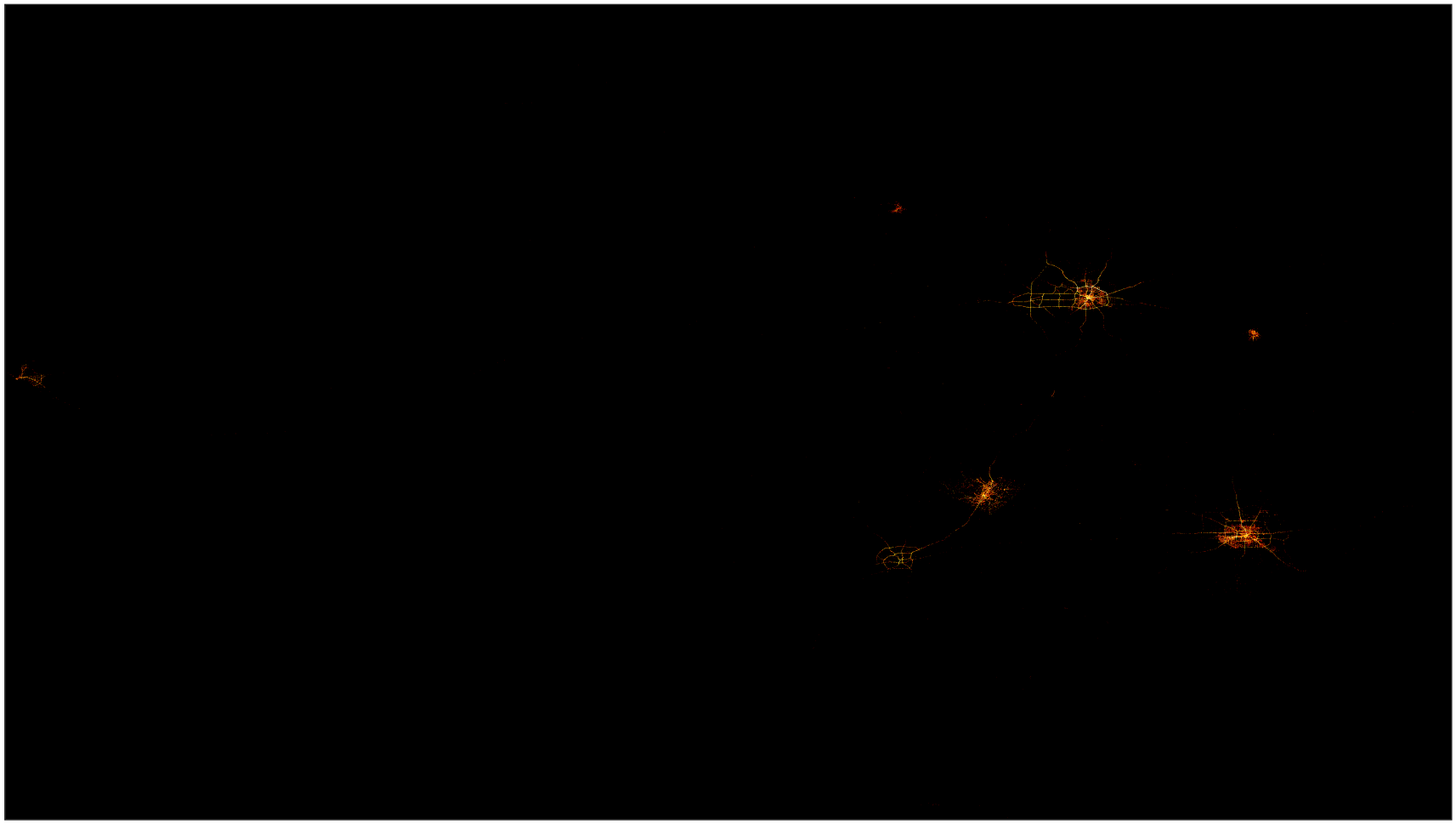
> As we look at the heatmap above, the accidents are scattered and dense than California but it is the second state with the most accidents as according to this article *Hit-And-Run Crashes On Florida Roads Average Over 103,000 Per Year; Drivers Urged To Stay At The Scene* (Chern, 2023) , it stated that in 2021, there were 109,624 Hit-And-Run crashes. And that was the year when America started to loosen up their restrictions even the Pandemic is making itself more contagious and dangerous.

Texas Heatmap

In [391…

```
texas_heatmap
```

Out[391]:

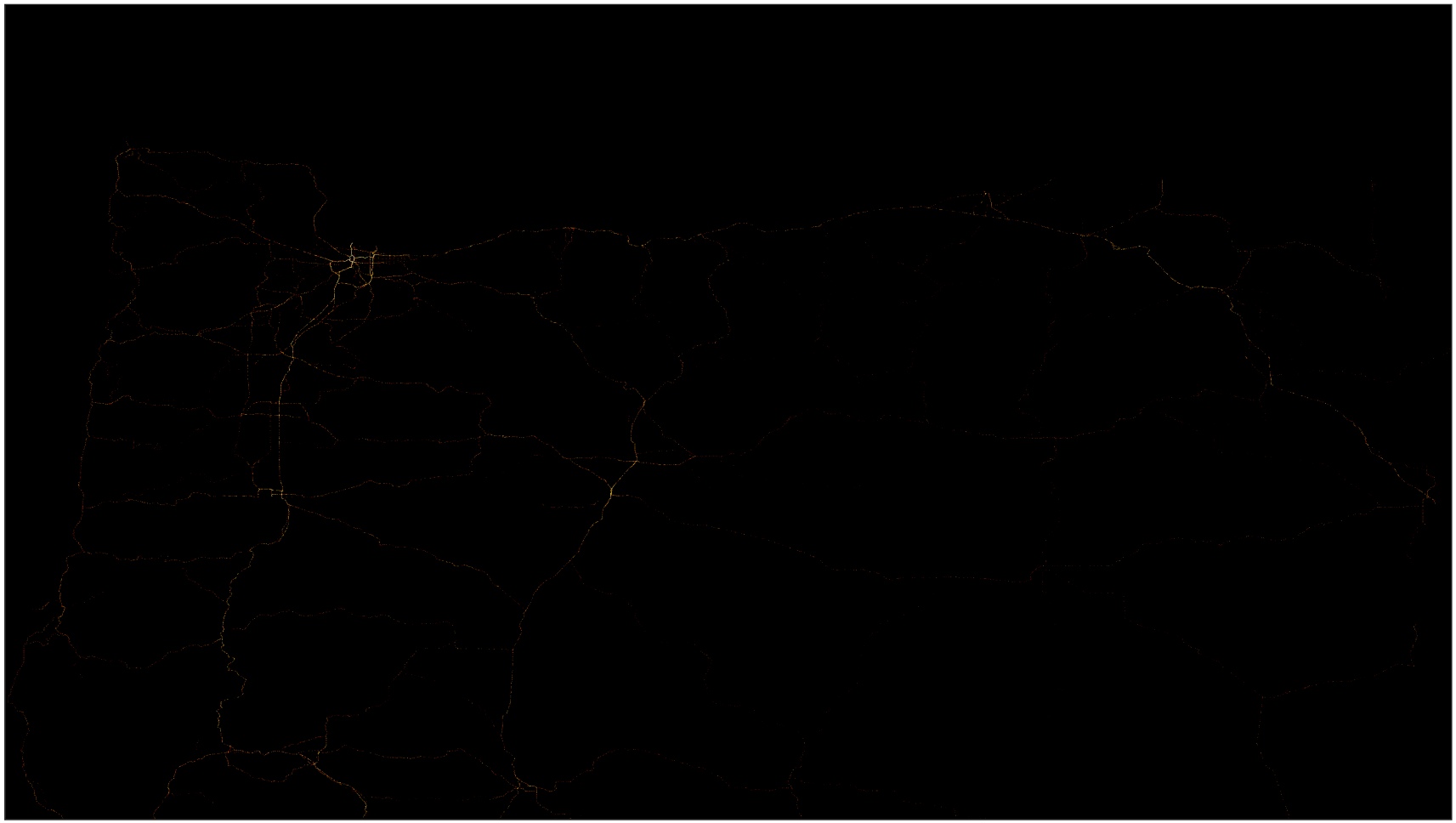


> Texas is the second largest state in the US and geographically speaking, it is the farm capital of US. And due to that factor, cities are more apart and not condensed as the land area is huge. We can see that here that most of the hot spots are from the major cities such as Houston, Austin, Dallas, And San Antonio. And since most of the land they have are farms, highways are much less and are only in the cities which means the crash rate they have seven times less than California

Oregon Heatmap

In [392…
```
oregon_heatmap
```

Out[392]:



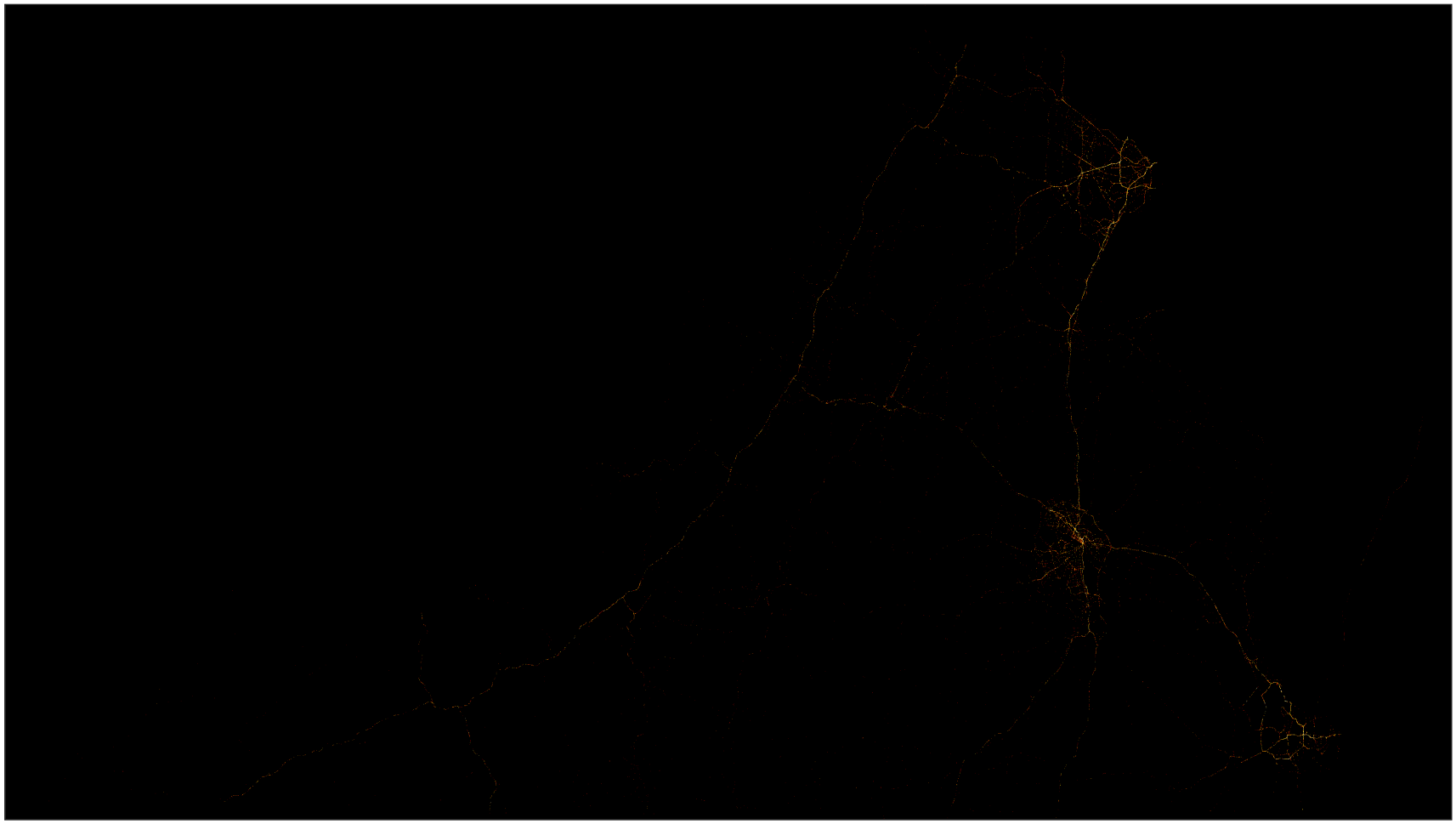
> Oregon is state that lives by the mountains and due to that, the accidents we can see in the heatmap are more on the state roads rather than city highways or urban streets. One hot spot we can see is on the upper side and that is the largest City of Oregon which is Portland. But despite being a big city most of the accidents are still on the roads leading up or down to the mountains due to factors such as speeding up in high sections, low light in the roads as most of it are in the less urban areas, and environmental reasons.

Virginia Heatmap

In [393…

```
virginia_heatmap
```

Out[393]:



> Virginia is one of those states like Texas that majority of the land is nature and urbanized sections present are not much. If we observe the heat map above, the hot spots are in the big cities such as the Richmond, the capital of Virginia. but we can also see hot spots above and below Richmond these cities are Washington DC that is miles away from the border of Virginia and Norfolk, located in the southeastern side of Virginia.

4.3 Correlational Analytics

4.3a Correlations The Variables

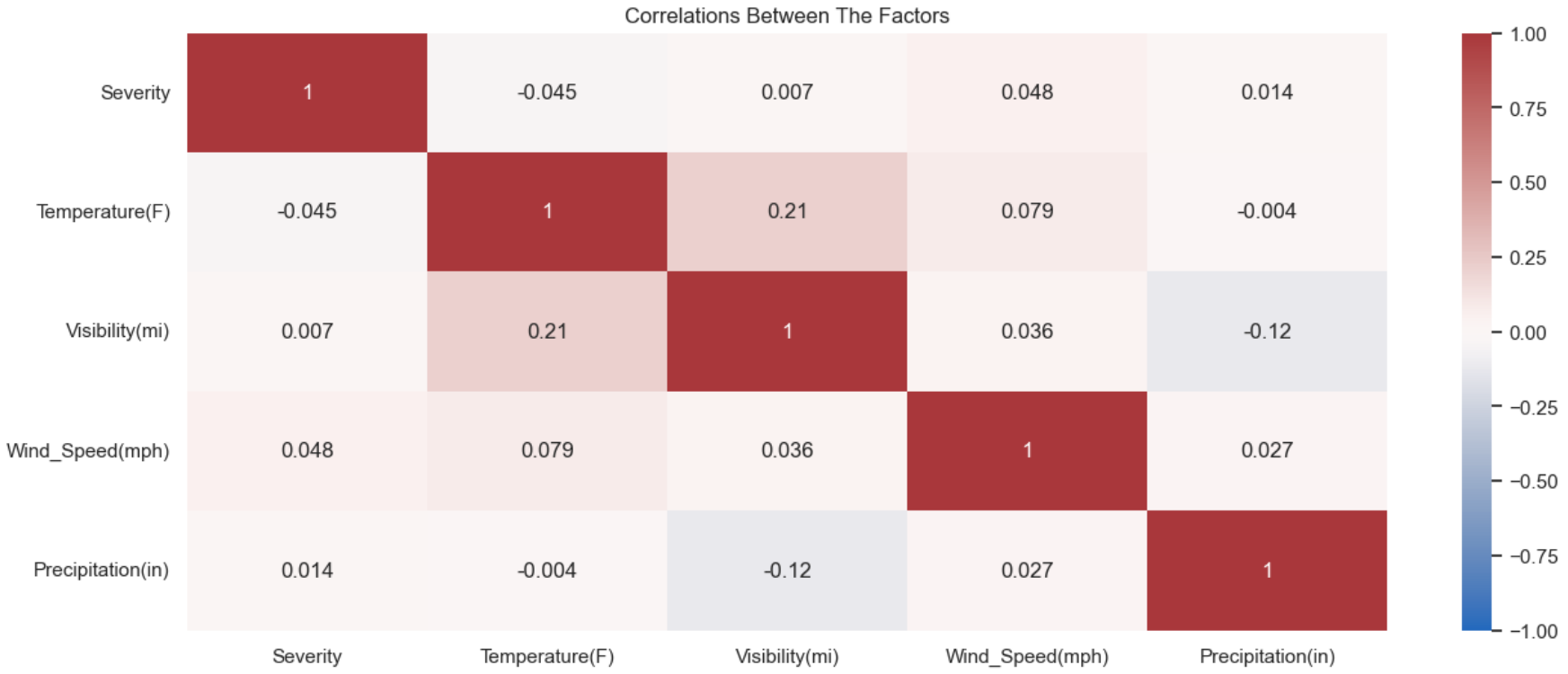
In [394…
```
#%matplotlib inline

# Getting the correlation of the variables that we are going to analyze
temps_correlation = us_accidents_total[['Severity','Temperature(F)','Visibility(mi)','Wind_Speed(mph)','P
temps_correlation = temps_correlation.corr().round(3)
temps_correlation
```

Out[394]:

	Severity	Temperature(F)	Visibility(mi)	Wind_Speed(mph)	Precipitation(in)
Severity	1.000	-0.045	0.007	0.048	0.014
Temperature(F)	-0.045	1.000	0.211	0.079	-0.004
Visibility(mi)	0.007	0.211	1.000	0.036	-0.122
Wind_Speed(mph)	0.048	0.079	0.036	1.000	0.027
Precipitation(in)	0.014	-0.004	-0.122	0.027	1.000

In [395…
```
# Plotting the correlation matrix (I am going to use a correlation matrix so that I can see the correlati

f, ax = plt.subplots(figsize=(15, 6))

s = sns.heatmap(temps_correlation, annot=True, vmax=1, vmin=-1, center=0, cmap='vlag')
plt.title("Correlations Between The Factors")
```

Out[395]:
```
Text(0.5, 1.0, 'Correlations Between The Factors')
```



Some observations that we can see:

- Severity has little to nothing when it comes to correlation with the variables.
- Interestingly visiblity and temperature has a slight positive correlation
- Severity has a slight negative correlation with visibility
- Visibility's correlation is equl with severity's correlation with visibility

What can we conclude?

We can say that external factors of the accident such as the temperature, visiblity, wind speed, and precipitation, has no correlation at all with the severity level of the accident. This means these external factors has almost to no connection with the level of severity. So we may see a fast wind speed but its severity is low. So, to conclude severity is not connected with the external factors but we can make a hypthosis that severity may be hugely affected by factors such as speed of the vehicle, was it a huge accident, and, the geographical location.

5. Summary

- California is the state with the most accidents with it being 65.90% greater from the second state .
- Most of the accidents occured on a 'fair' weather condition.
- From 2016 to 2021, the year 2020 had a huge drop due to the pandemic Covid-19 but then it soon spiked in 2021 as US started to loosen restrictions.
- We can assert that the temperature of 76 farenheit and the weather condition 'fair' is directy connected as both of them has the most accidents that occured to.
- In most states that have many cities, accidents are more frequent in the urban areas such as streets, highways, and, avenues.
- Severity has no correlation with the external factors of the accident that are temperature, visiblity, wind speed, and, precipitation.

6. Biliography

- *Chern, A. (2023, February 10). Hit-And-Run Crashes On Florida Roads Average Over 103,000 Per Year; Drivers Urged To Stay At The Scene. Florida Department of Highway Safety and Motor Vehicles. https://www.flhsmv.gov/2023/02/01/hit-and-run-crashes-on-florida-roads-average-over-103000-per-year-drivers-urged-to-stay-at-the-scene/*
- *Statista. (2022, June 21). Top U.S. states based on number of farms 2021. https://www.statista.com/statistics/196114/top-10-us-states-by-number-of-farms/*
- *"True Crime: On the Run" gives an inside look at Southern California's high-speed police chases. (2022, September 3). ABC7 Los Angeles. https://abc7.com/true-crime-on-the-run-police-chases-los-angeles-southern-california-how-many-in/12181278/*
- *N. (2022, August 13). Calculate and Plot a Correlation Matrix in Python and Pandas. Datagy. https://datagy.io/python-correlation-matrix/*
- *Developers, P. (n.d.). Census - Data Shader Example. https://datashader.org/topics/index.html. https://examples.pyviz.org/census/census.html*
- *Solanki, S. (2021, October 10). Geoviews - Choropleth Maps using Bokeh and Matplotlib [Python]. https://coderzcolumn.com/tutorials/data-science/geoviews-choropleth-maps*