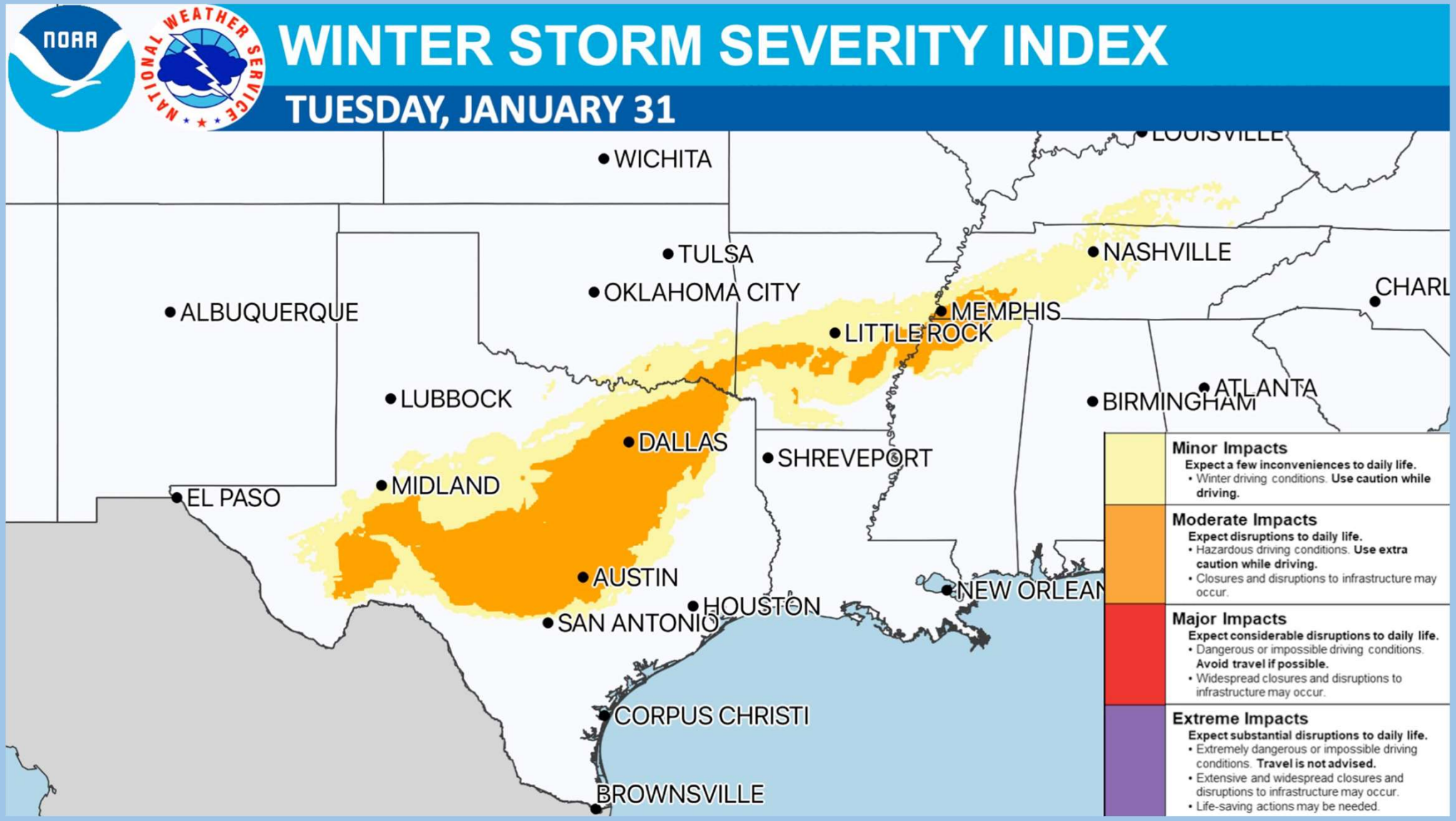
Freezing Star

Presented by Tavleen, Jörn and Louiza

31 January - 2 February 2023
North American Ice Storm

An Arctic cold passage met warm and moist air from the Gulf of Mexico, creating an ice storm over southern United States.

This weather event was marked by sleet and freezing rain in numerous localities.


NOAA
NATIONAL WEATHER SERVICE
WINTER STORM SEVERITY INDEX
TUESDAY, JANUARY 31
WICHITA
TULSA
OKLAHOMA CITY
ALBUQUERQUE
LUBBOCK
DALLAS
MIDLAND
EL PASO
AUSTIN
SAN ANTONIO
HOUSTON
CORPUS CHRISTI
BROWNSVILLE
LITTLE ROCK
MEMPHIS
NASHVILLE
SHREVEPORT
BIRMINGHAM
ATLANTA
NEW ORLEAN
Minor Impacts
Expect a few inconveniences to daily life.
• Winter driving conditions. Use caution while driving.
Moderate Impacts
Expect disruptions to daily life.
• Hazardous driving conditions. Use extra caution while driving.
• Closures and disruptions to infrastructure may occur.
Major Impacts
Expect considerable disruptions to daily life.
• Dangerous or impossible driving conditions. Avoid travel if possible.
• Widespread closures and disruptions to infrastructure may occur.
Extreme Impacts
Expect substantial disruptions to daily life.
• Extremely dangerous or impossible driving conditions. Travel is not advised.
• Extensive and widespread closures and disruptions to infrastructure may occur.
• Life-saving actions may be needed.

Hypothesis Weather

We expect to see:

1. a significant increase in precipitation (snow, sleet, freezing rain)
1. a decrease in temperature

from January 31st - February 2nd

Hypothesis Flights

We expect to see:

1. higher frequency of flight delays
1. higher frequency of cancelled flights

from January 31st - February 2nd

Data Wrangling & Data Cleaning

defining a function for API call

```
def api_call(station_no, tz="America/Chicago"):
    load_dotenv()
    url = "https://meteostat.p.rapidapi.com/stations/hourly"
    querystring = {"station":station_no,"start":"2023-01-17","end":"2023-02-
15","tz":tz}
    headers = {
    "X-RapidAPI-Key": os.getenv('rapidapi_key'),
    "X-RapidAPI-Host": "meteostat.p.rapidapi.com"}
    r = requests.get(url, headers=headers, params=querystring)
    json_object = json.loads(r.content)
    return r.json()
```

call the function & create a dataframe

```
weather_json = api_call("74745")
```

```
weather_xxx = pd.DataFrame(weather_json['data'])
```

defining a function in order to export the table

```
schema = 'cgn_analytics_24_1'
engine = get_engine()
def export_table(df, airport_code):
    # creating table_name with jlt_airport and the airport_code
    global engine
    table_name = f'jlt_airport_{airport_code}'
    if engine!=None:
        try:
            df.to_sql(name=table_name, # Name of SQL table
                      con=engine, # Engine or connection
                      if_exists='replace', # Drop the table before inserting new values
                  schema=schema, # Use schmea that was defined earlier
                      index=False, # Write DataFrame index as a column
                      chunksize=5000, # Specify the number of rows in each batch to be written at a time
                        method='multi') # Pass multiple values in a single INSERT clause
            print(f"The {table_name} table was imported successfully.")
    # Error handling
        except (Exception, psycopg2.DatabaseError) as error:
            print(error)
            engine = None
```

export the corresponding flights data

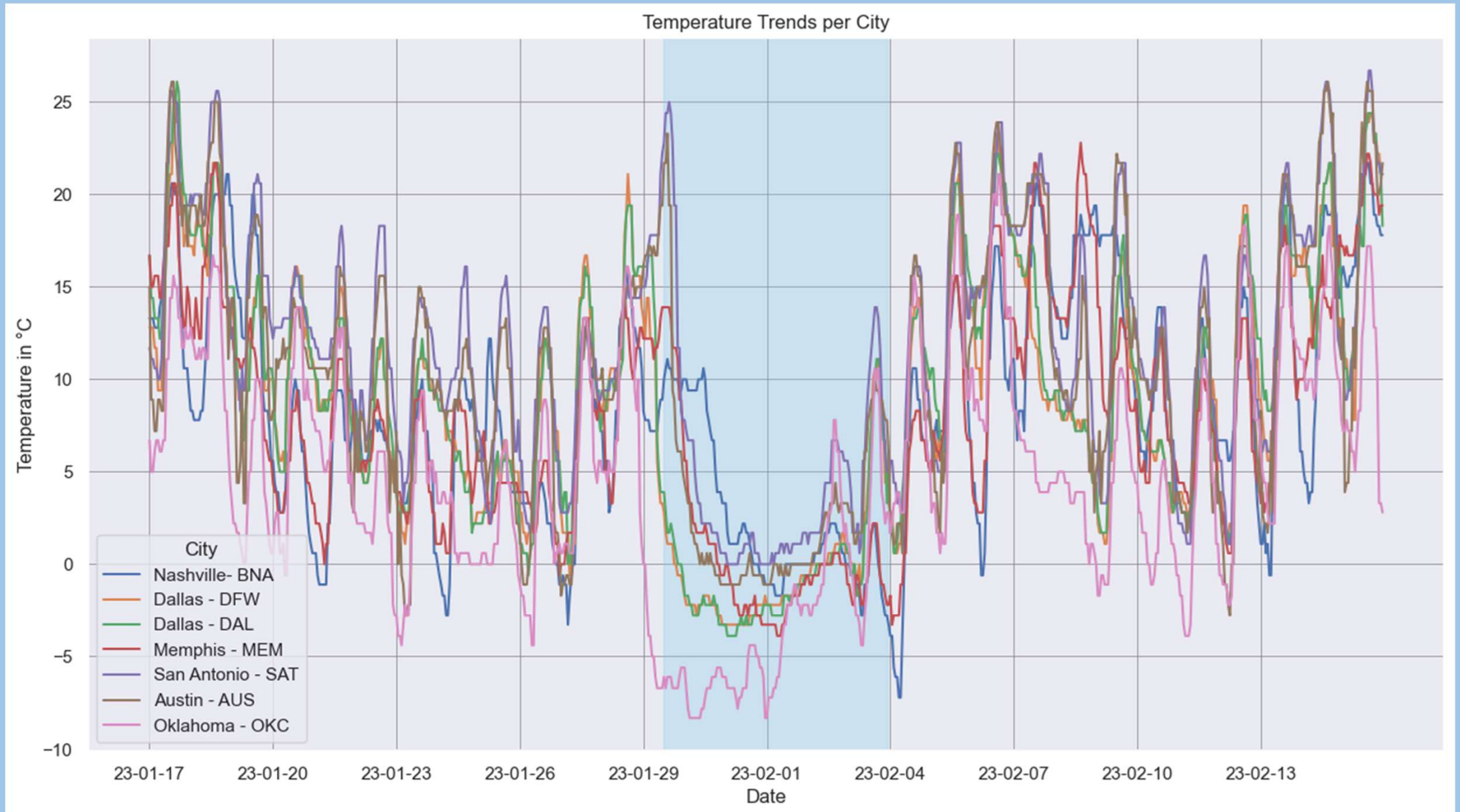
```
flights_df = get_dataframe(f"Select * from {schema}.jlt_flights WHERE flight_date < '2023-02-16 00:00:00.000' AND flight_date > '2023-01-16 00:00:00.000' ORDER BY flight_date ASC;")


# make a new column with timestamps for dep, sched_dep, arr and sched_arr and then merge with corresponding weather data for each airport


def convert_timestamp(df, column):
    df['hours'] = df[f'{column}'] // 100
    df['minutes'] = df[f'{column}'] % 100
    df[f'{column}stamp'] = df['flight_date'] + pd.to_timedelta(flights_df['hours'], unit='h') + \
                           pd.to_timedelta(flights_df['minutes'], unit='m')
    df[f'{column}stamp_merge'] = flights_df['flight_date'] + pd.to_timedelta(flights_df['hours'], unit='h')
    df.drop('hours', axis=1, inplace=True)
    df.drop('minutes', axis=1, inplace=True)
```

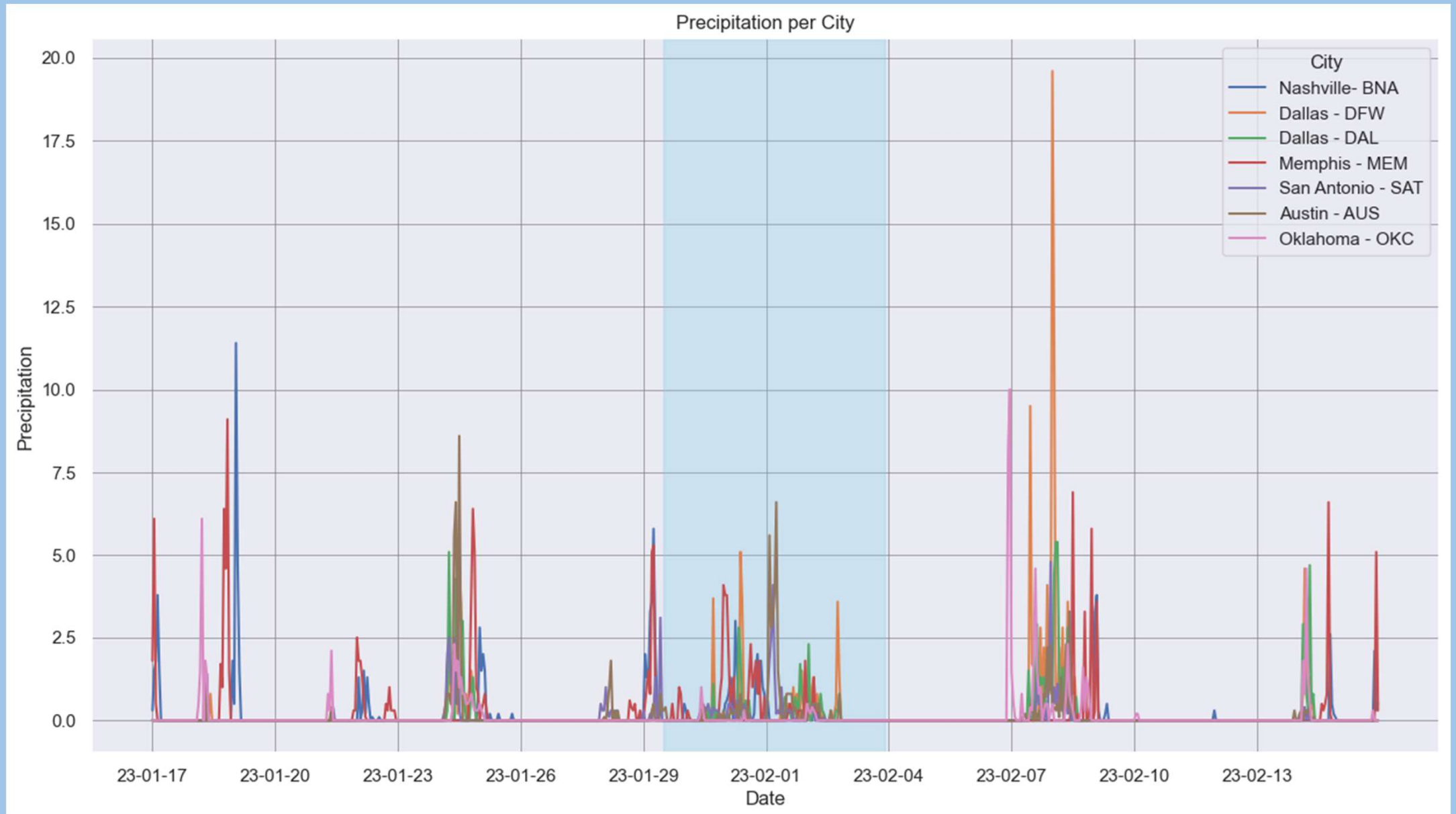
function to merge weather & flights data

```
def airport_df(df, airport_code):
    # Merge on 'sched_dep_timestamp_merge' if origin=airport_code
    dep = df.copy()
    dep = dep.loc[df['origin'] == f'{airport_code}']
    merged_dep = pd.merge(dep,
                          eval(f'weather_{airport_code.lower()}'),
                          left_on='sched_dep_timestamp_merge',
                          right_on='time',
                          how='left')
    # Merge on 'sched_arr_timestamp_merge' if dest=airport_code
    arr = df.copy()
    arr = arr.loc[df['dest'] == f'{airport_code}']
    merged_arr = pd.merge(arr,
                          eval(f'weather_{airport_code.lower()}'),
                          left_on='sched_arr_timestamp_merge',
                          right_on='time',
                          how='left')
    return pd.concat([merged_dep, merged_arr], ignore_index=True)
```

EDA

Can we see the weather event in the weather data?


Temperature Trends per City
Temperature in °C
25
20
15
10
5
0
−5
−10
23-01-17
23-01-20
23-01-23
23-01-26
23-01-29
23-02-01
23-02-04
23-02-07
23-02-10
23-02-13
Date
City
Nashville- BNA
Dallas - DFW
Dallas - DAL
Memphis - MEM
San Antonio - SAT
Austin - AUS
Oklahoma - OKC


Precipitation per City
Precipitation
20.0
17.5
15.0
12.5
10.0
7.5
5.0
2.5
0.0
23-01-17
23-01-20
23-01-23
23-01-26
23-01-29
23-02-01
23-02-04
23-02-07
23-02-10
23-02-13
Date
City
Nashville- BNA
Dallas - DFW
Dallas - DAL
Memphis - MEM
San Antonio - SAT
Austin - AUS
Oklahoma - OKC

Code snippets

```
# create a seperate dataframe with temp. per day for each city

weather_l = [weather_bna,  weather_dfw, weather_dal, weather_mem, weather_sat,
weather_aus, weather_okc]
temp_columns = []
for df in weather_l:
    temp_columns.append(df['temp'])

merged_df_temp = pd.DataFrame(temp_columns).T
merged_df_temp.columns = ['Nashville- BNA', 'Dallas - DFW', 'Dallas - DAL','Memphis
- MEM', 'San Antonio - SAT',  'Austin - AUS', 'Oklahoma - OKC']
```

Code snippets

```
# Plot merged_df_daily
plt.figure(figsize=(15, 8))
for city in merged_df_temp.columns:
    plt.plot(merged_df_temp.index, merged_df_temp[city], label=city)
plt.title('Temperature Trends per City')
plt.xlabel('Date')
plt.ylabel('Temperature in °C')
plt.legend(title='City')
start_date = '2023-01-17'
end_date = '2023-02-15'
date_range = pd.date_range(start='2023-01-17', end='2023-02-15', freq='3D')
date_strings = [date.strftime('%y-%m-%d') for date in date_range]
plt.xticks(ticks=(np.arange(0, 700, step=72)),  labels=date_strings)
highlight_start = 300
highlight_end = 430
plt.axvspan(highlight_start, highlight_end, color='skyblue', alpha=0.3)
plt.grid(True, color='gray', linestyle='-', linewidth=0.5)
```

We confirm our Hypothesis Weather

We see

1. a significant increase in precipitation (snow,± sleet, freezing rain)
1. a decrease in temperature

from January 31st - February 2nd

Can we see the weather event in the flights data?

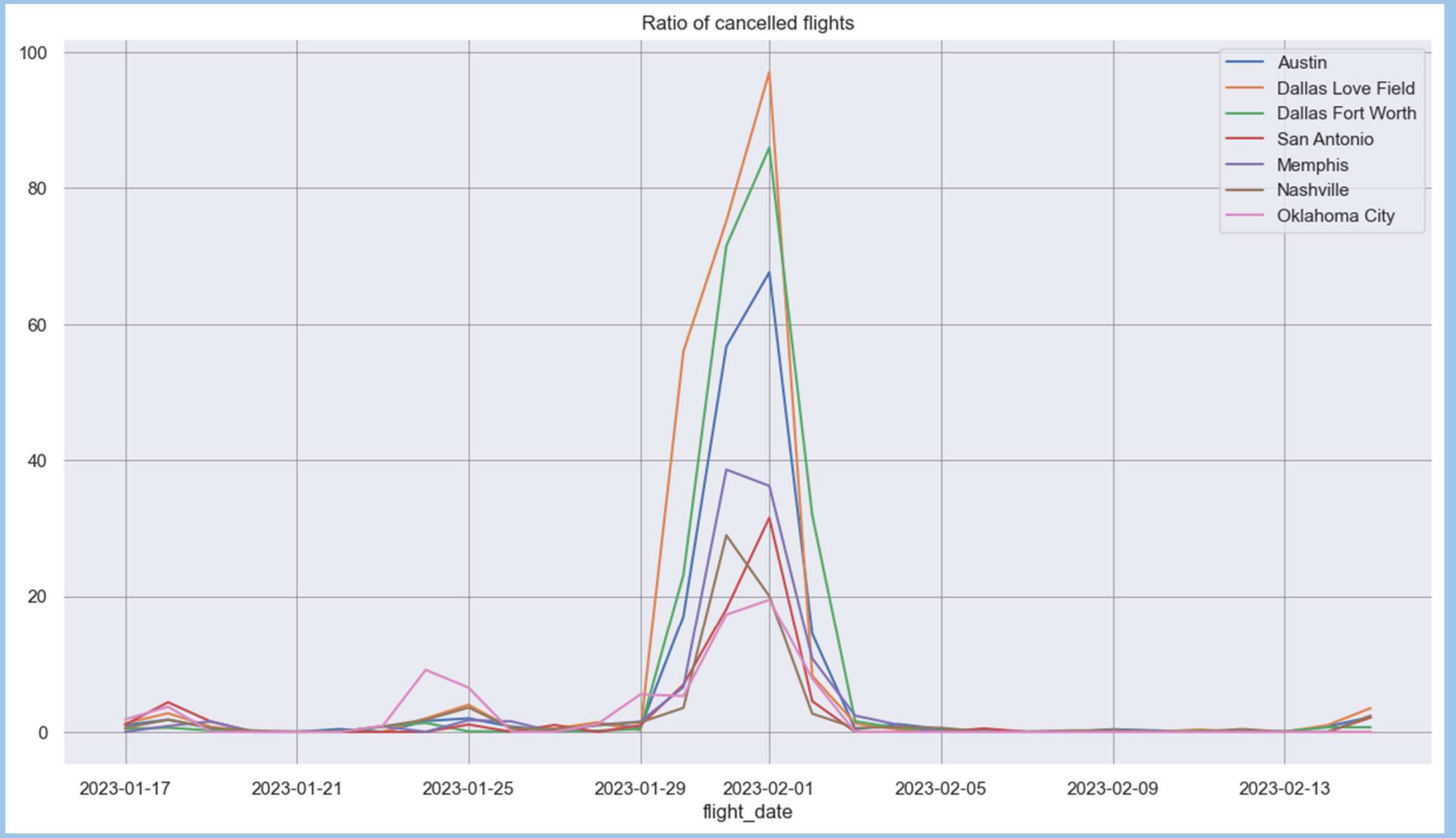
Frequency of cancellations & observed weather condition

```
cancelled_all=all_airports_concat.groupby(['flight_date','weather_condition'])['cancelled'].sum().sort_values()
cancelled_all.tail(25)
```

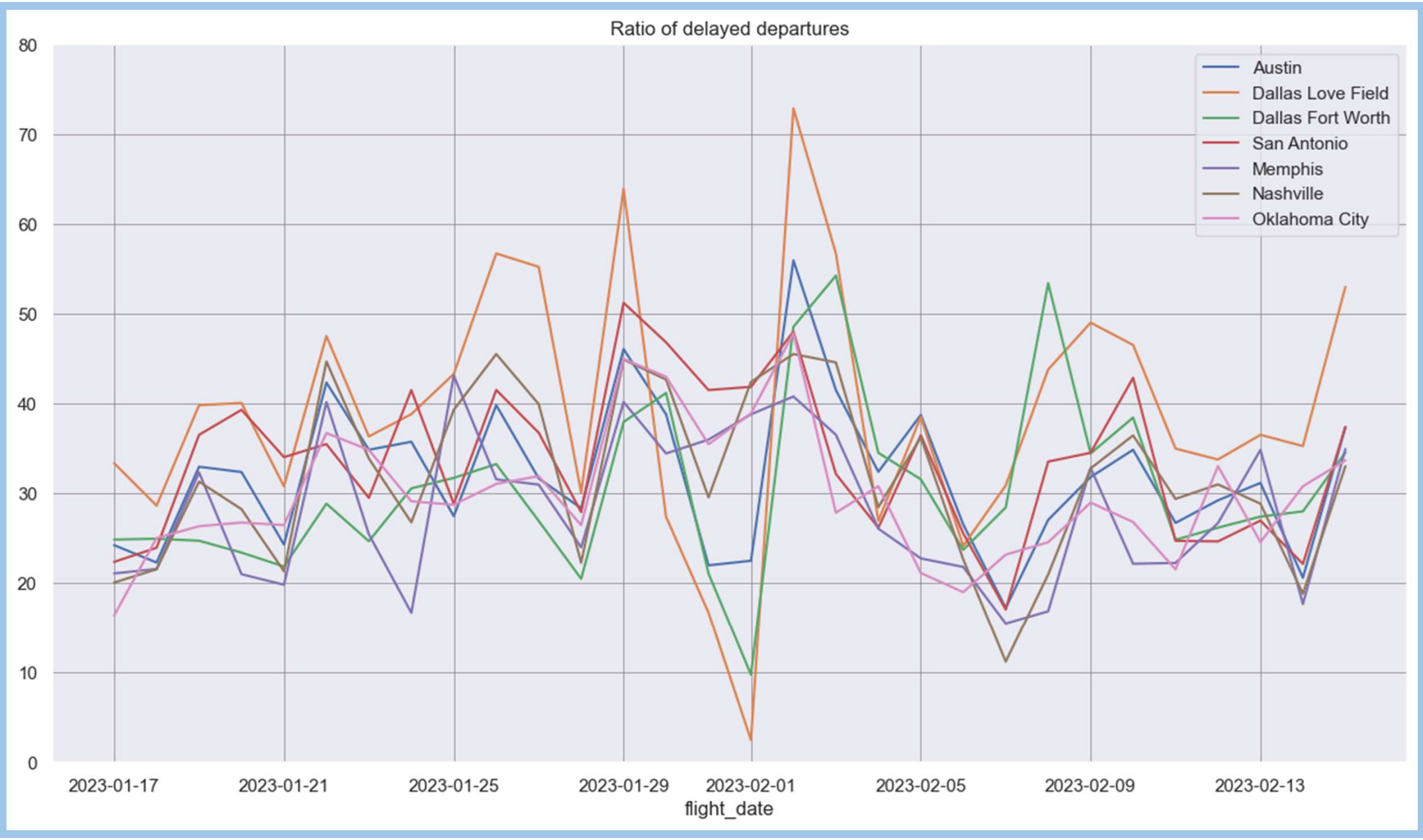
flight_date	weather_condition	cancellations
2023-01-31	Overcast	36
2023-02-02	Sleet	39
2023-02-01	Rain	40
	Light Snowfall	47
	Heavy Snowfall	48
	Overcast	54
2023-02-02	Light Rain	63
2023-01-30	Rain	74
2023-01-31	Light Rain	86
	Snowfall	109
2023-01-30	Fog	132
2023-01-31	Sleet	151

flight_date	weather_condition	cancellations
2023-02-01	Fog	158
2023-01-31	Fog	160
2023-01-30	Cloudy	161
	Overcast	196
2023-02-01	Freezing Rain	227
2023-01-31	Cloudy	246
2023-02-01	Snowfall	309
	Cloudy	361
2023-02-02	Freezing Rain	373
2023-02-01	Light Rain	376
2023-01-31	Freezing Rain	412
2023-02-01	Heavy Freezing Rain	451
2023-01-31	Light Snowfall	490

Can we see a higher frequency of cancelled flights?

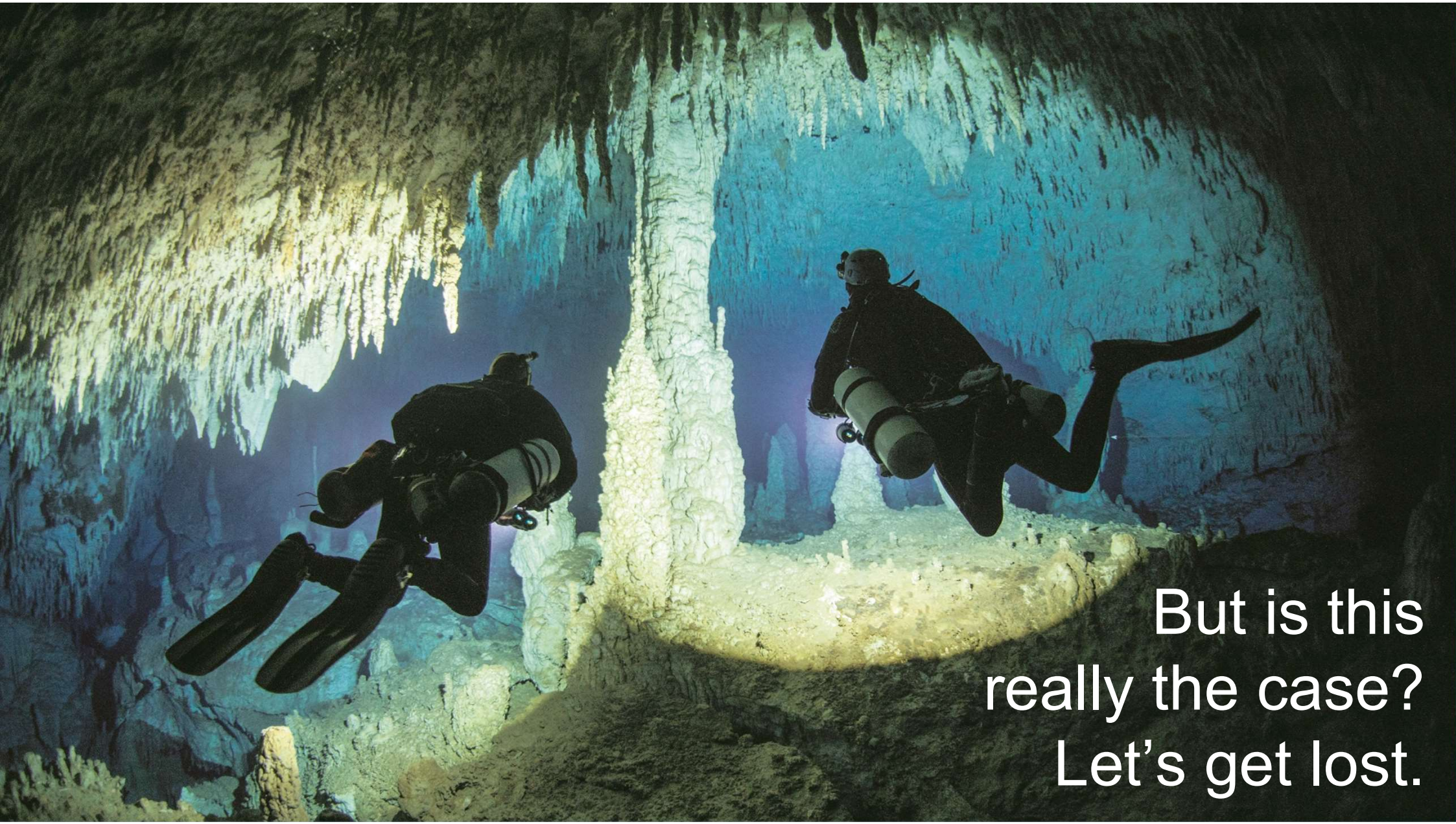

Ratio of cancelled flights
100
80
60
40
20
0
Austin
Dallas Love Field
Dallas Fort Worth
San Antonio
Memphis
Nashville
Oklahoma City
2023-01-17
2023-01-21
2023-01-25
2023-01-29
2023-02-01
2023-02-05
2023-02-09
2023-02-13
flight_date

Can we see a higher frequency of delayed departures?

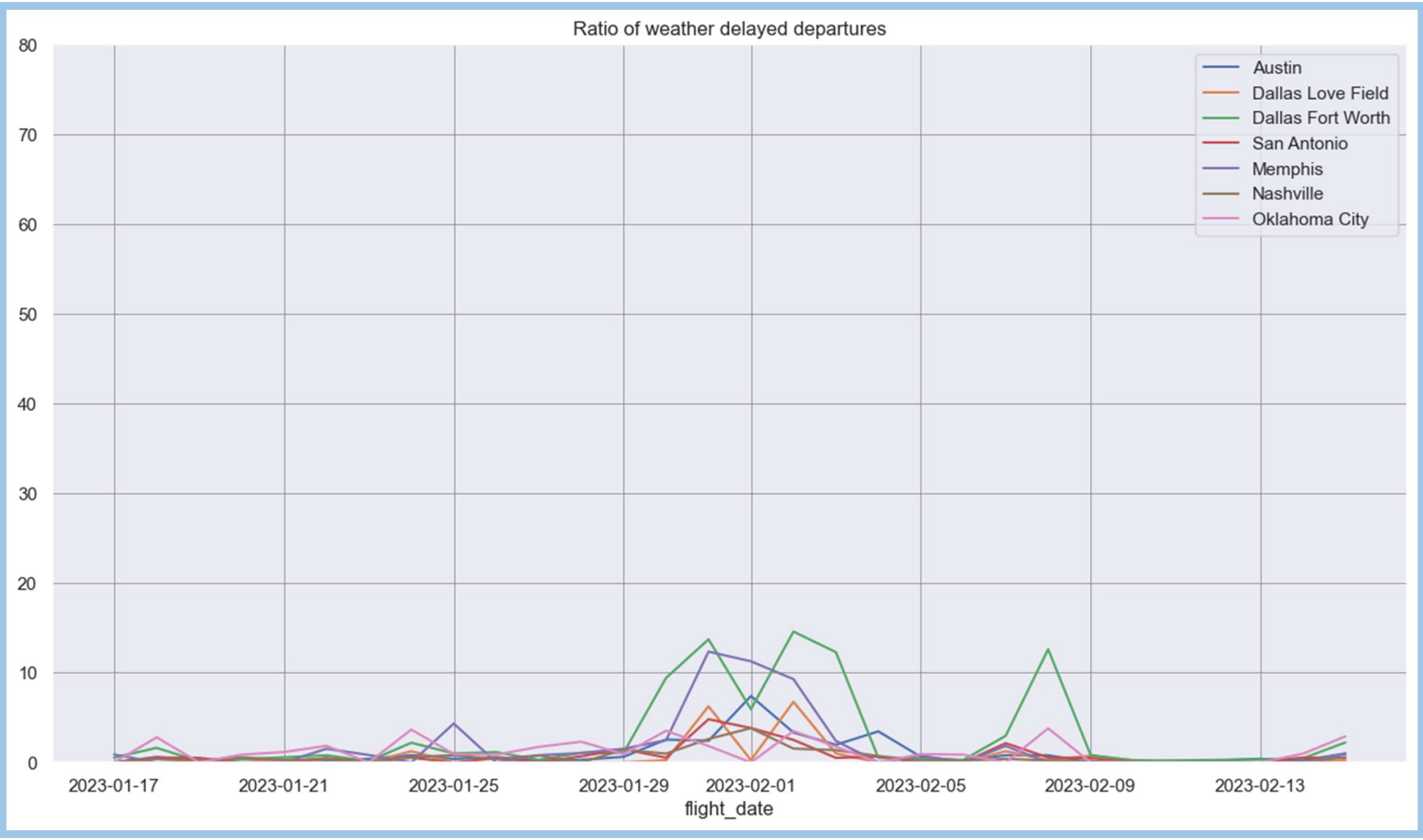

Ratio of delayed departures
80
70
60
50
40
30
20
10
0
2023-01-17
2023-01-21
2023-01-25
2023-01-29
2023-02-01
2023-02-05
2023-02-09
2023-02-13
flight_date
Austin
Dallas Love Field
Dallas Fort Worth
San Antonio
Memphis
Nashville
Oklahoma City

So why does the number of delayed flights drop dramatically?

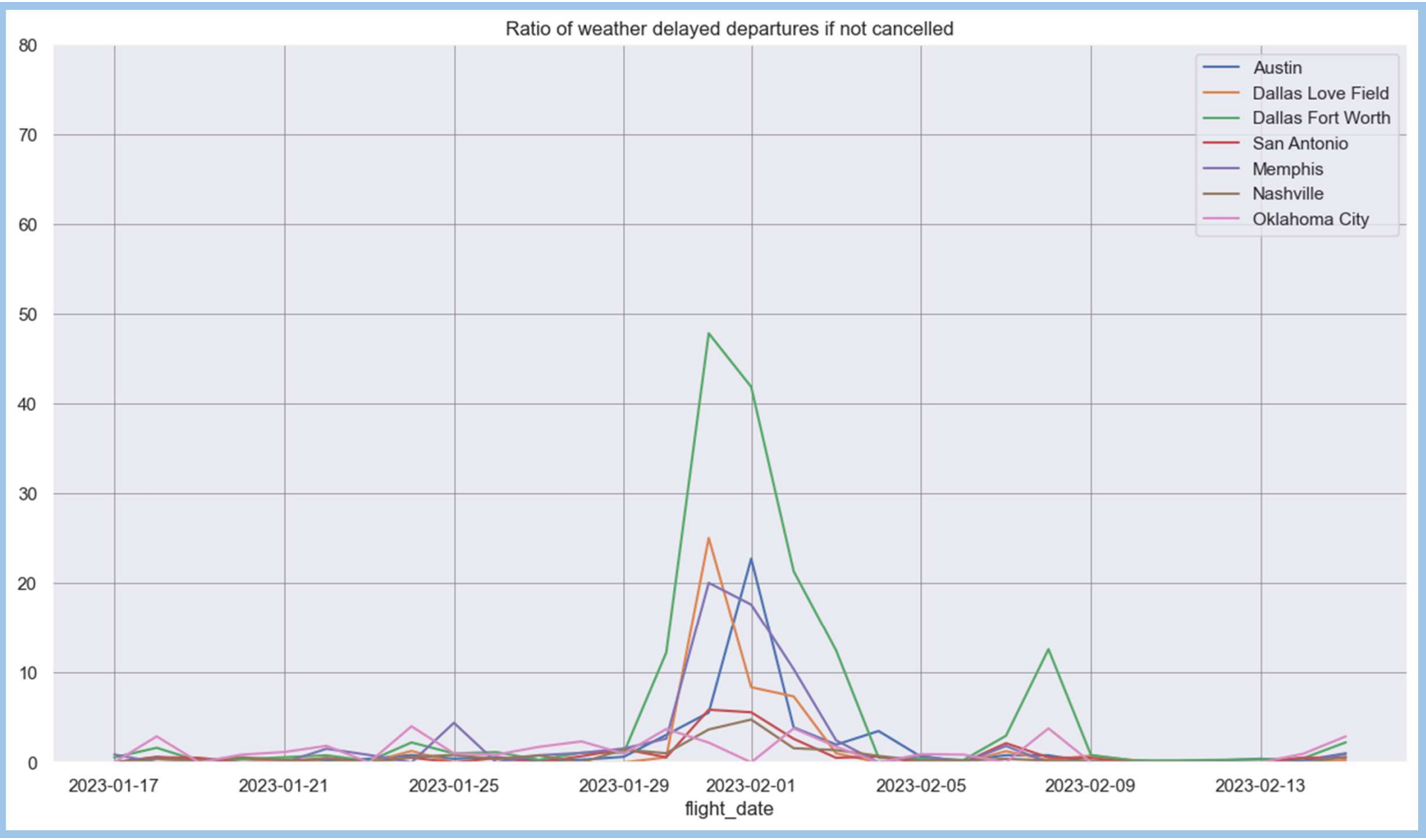
Simply put all flights were cancelled!


But is this
really the case?
Let’s get lost.

So what about delays caused by weather?


Ratio of weather delayed departures
80
70
60
50
40
30
20
10
0
2023-01-17
2023-01-21
2023-01-25
2023-01-29
2023-02-01
2023-02-05
2023-02-09
2023-02-13
flight_date
Austin
Dallas Love Field
Dallas Fort Worth
San Antonio
Memphis
Nashville
Oklahoma City

Let’s remove the cancelled flights.


Ratio of weather delayed departures if not cancelled
80
70
60
50
40
30
20
10
0
2023-01-17
2023-01-21
2023-01-25
2023-01-29
2023-02-01
2023-02-05
2023-02-09
2023-02-13
flight_date
Austin
Dallas Love Field
Dallas Fort Worth
San Antonio
Memphis
Nashville
Oklahoma City

If we disregard cancelled flights, number of weather delays rises considerably

Code snippets

```
# Defining a function to create a weather delay column


def wdel_col(df):
    wdelayed_col=[]
    for i in df['weather_delay']:
        if i >0:
            wdelayed_col.append(1)
        else:
            wdelayed_col.append(0)
    df['weather_delayed']=wdelayed_col
    return(df)
```

```
# Thanks Param
```

Code snippets

```
# Defining a function to calculate the ratio of weather delayed flights


def wdelayed_func(df):
    number_flights=df.groupby('flight_date')['flight_date'].count()
    wdelayed=df.groupby('flight_date')['weather_delayed'].sum()
    ratio_wdelayed=(wdelayed/number_flights)*100
    return(ratio_wdelayed)
```

We confirm our Hypothesis Flights

We see

1. higher frequency of flight delays (but only if we explore the data properly)
1. higher frequency of cancelled flights

from January 31st - February 2nd