This code is belong to Hong Phuc Pham, please do not copy and use for your university assignments.

1 Project brief

This project will observer the US domestic flight data in 2018. The data is published by US Department of Transport and can be also found by the following on Kaggle: https://www.kaggle.com/yuanyuwendymu/airline-delay-and-cancellation-data-2009-2018?select=2018.csv (https://www.kaggle.com/yuanyuwendymu/airline-delay-and-cancellation-data-2009-2018?select=2018.csv)

The report will look over the points:

1. Operation Carrier flight contribution
2. Brief view of top US domestic longest flights
3. Flight traffic
4. Delay flights
5. Flight operated of each carrier
6. Holiday destination by airway
7. Long flight connection.

2 Dataset

This dataset has 28 columns, where

- FL_DATE: Flight Date (**yyyymmdd**)
- OP_CARRIER: Operation Carrier - Supported lookup table: https://www.transtats.bts.gov/FieldInfo.asp?Svryq_Qr5p=h0v37r%FDPn44vr4%FDP1qr.%FDjur0%FD6ur%FD5nzr%FDp1qr%FDun5%FDorr0%FD75rq% (https://www.transtats.bts.gov/FieldInfo.asp?Svryq_Qr5p=h0v37r%FDPn44vr4%FDP1qr.%FDjur0%FD6ur%FD5nzr%FDp1qr%FDun5%FDorr0%FD75rq%
- OP_CARRIER_FL_NUM: Operation Carrier Flight Number
- ORIGIN: Origin Airport - Supported lookup table: https://www.transtats.bts.gov/FieldInfo.asp?Svryq_Qr5p=b4vtv0%FDNv42146&Svryq_gB2r=Pun4&Y11x72_gnoyr=Y_NVecbeg&gnoyr_VQ=FGK&flf_gnc (https://www.transtats.bts.gov/FieldInfo.asp?Svryq_Qr5p=b4vtv0%FDNv42146&Svryq_gB2r=Pun4&Y11x72_gnoyr=Y_NVecbeg&gnoyr_VQ=FGK&flf_gnc
- DEST: Destination Airport - Supported lookup table: https://www.transtats.bts.gov/FieldInfo.asp?Svryq_Qr5p=Qr56v0n6v10%FDNv42146&Svryq_gB2r=Pun4&Y11x72_gnoyr=Y_NVecbeg&gnoyr_VQ=FGK8 (https://www.transtats.bts.gov/FieldInfo.asp?Svryq_Qr5p=Qr56v0n6v10%FDNv42146&Svryq_gB2r=Pun4&Y11x72_gnoyr=Y_NVecbeg&gnoyr_VQ=FGK8
- CRS_DEP_TIME: CRS Departure Time (local time: **hhmm**)
- DEP_TIME: Actual Departure Time (local time: **hhmm**)
- DEP_DELAY: Difference in minutes between scheduled and actual departure time. *Early departures show **negative** numbers*.
- TAXI_OUT: Taxi Out Time, in **Minutes**
- WHEELS_OFF: Wheels Off Time (local time: **hhmm**)
- WHEELS_ON: Wheels On Time (local time: **hhmm**)
- TAXI_IN: Taxi In Time, in **Minutes**
- CRS_ARR_TIME: CRS Arrival Time (local time: **hhmm**)
- ARR_TIME: Actual Arrival Time (local time: **hhmm**)

- ARR_DELAY: Difference in minutes between scheduled and actual arrival time. *Early departures show **negative** numbers*.
- CANCELED: Canceled Flight Indicator (**1=Yes**)
- CANCELLATION_CODE: Specifies The Reason For Cancellation - Supported lookup table: https://www.transtats.bts.gov/FieldInfo.asp?Svryq_Qr5p=f2rpvsvr5%FDgur%FDern510%FDS14%FDPn0pryyn6v10&Svryq_gB2r=Pun4&Y11x72_gnoyr=(https://www.transtats.bts.gov/FieldInfo.asp?Svryq_Qr5p=f2rpvsvr5%FDgur%FDern510%FDS14%FDPn0pryyn6v10&Svryq_gB2r=Pun4&Y11x72_gnoyr=
- DIVERTED: Diverted Flight Indicator (**1=Yes**)
- CRS_ELAPSED_TIME: CRS Elapsed Time of Flight, in **Minutes**
- ACTUAL_ELAPSED_TIME: Elapsed Time of Flight, in **Minutes**
- AIR_TIME: Flight Time,in **Minutes**
- DISTANCE: Distance between airports **(miles)**
- CARRIER_DELAY: Carrier Delay, in **Minutes**
- WEATHER_DELAY: Weather Delay, in **Minutes**
- NAS_DELAY: National Air System Delay, in **Minutes**
- SECURITY_DELAY: Security Delay, in **Minutes**
- LATE_AIRCRAFT_DELAY: Late Aircraft Delay, in **Minutes**
- Unnammed: 27 (not included in US Department of Transport document)

Dataset contains about 7 million rows. However, only the first quarter of 2018 data will be observe in this project. Therefore, data from 1st of January, 2018 to 31st of March will be used to analyse and plot generating.

Different data types are in delay/ canceled dataset include: int, float, string, datetime.

3 Challenges

Flight dataset is not a stand-alone sheet, it need to merge and connect with other dataset provided by US Department of Transport to make the data set more meaning. Likes: CANCELLATION_CODE, ORIGIN, DES

Some data is display as the wrong datatype, such as time instead of float.

Some cleanup might need to be done as few cells has missmatched data or no data.

4 Technical Summary

Libraries use:

```
- Pandas
- Plotly
- Datetime
- Time
- Numpy
```

5 Solution

5.1 Cleaning data

As the main data set has unclear data need to take care.

Many columns containing float data types where they should be integer.

Some integer columns are used to represent the time and need to formated to datetime object.

Some empty cells need to fill up.

Some columns has the abbreviations data and need to translate.

Because, this dataset is big and only the 1st quarter of 2018 will be used, so the origin data set will be slice into smaller piece. Then some columns values will be interpreting by using other support lookup tables. Some of them will be extract into smaller components for further analysing purposes.

5.1.1 Pseudo Code

1. Getting the the data frame then slice out the last columns as it does not contain any data also not have any meaning related to others.
2. Take out any rows that exceed 31/03/2018 by using Pandas data frame drop() function.
3. Convert 2 columns CRS_DEP_TIME and CRS_ARR_TIME to string for later manipulating.
   A. Using Pandas data frame apply() function to change the string into right format.
      a. 24 hours format is used.
      b. Data will be split out to hours part and minutes part seperated by colon. Any first 2 character is 24 or empty will be converted to 0.
      c. Those data will be convert into datetime by function to_datetime() function.
   B. Using Pandas data frame apply() function to fill up the empty cells and also convert target columns have float data type into integer datatype.
4. Import support lookup tables
5. Using Pandas dataframe map() function to update abbreviations. This will be link the data with dictionary base on the lookup tables.
6. Data in ORIGIN, DEST columns will be extracted and cut into small part then stored into new columns- ORIGIN_CITY, ORIGIN_STATE, ORIGIN_AIRPORT, DEST_CITY, DEST_STATE , DEST_AIRPORT respectively. In this data manipulating, split() function will be used with the sparators are ',' and ':'.
7. ORIGIN_STATE and DEST_STATE abbreviations will be interpret with other support data set.

5.1.2 Solution Code

In [1]:

```
# Import library
import numpy as np
import pandas as pd
import datetime as dt
from datetime import date

df = pd.read_csv('2018_flight_data.csv')
# Slicing data frame to not take the redundant column 'Unnamed: 27'
df = df.iloc[::,:-1:]

# In this practice we just work on the first quarter of 2018
# Therefore anyday that over 31/3/2018 will be drop
df.drop(df[df.FL_DATE > '2018-03-31' ].index, inplace = True)

df.head(5)
```

executed in 1m 17.5s, finished 10:50:17 2021-11-15

Out[1]:

	FL_DATE	OP_CARRIER	OP_CARRIER_FL_NUM	ORIGIN	DEST	CRS_DEP_TIME	DEP_TIME
0	2018-01-01	UA	2429	EWR	DEN	1517	1512.0
1	2018-01-01	UA	2427	LAS	SFO	1115	1107.0
2	2018-01-01	UA	2426	SNA	DEN	1335	1330.0
3	2018-01-01	UA	2425	RSW	ORD	1546	1552.0
4	2018-01-01	UA	2424	ORD	ALB	630	650.0

5 rows × 27 columns

In [2]:

```
# Checking shape
print(df.shape)
```

executed in 13ms, finished 10:50:17 2021-11-15

```
(1702836, 27)
```

In [3]:

```
import time
# # Covert CRS_DEP_TIME to string data type to easier to convert to datetime object lat
df['CRS_DEP_TIME'] = df['CRS_DEP_TIME'].apply(str)
df['CRS_ARR_TIME'] = df['CRS_DEP_TIME'].apply(str)

# Convert to datetime object datatype
df['CRS_DEP_TIME'] = pd.to_datetime(df['CRS_DEP_TIME'].apply(
    lambda x: str(f'{x[0:-2] if (x[0:-2] != "" and x[0:-2] != "24") else "0"}:{x[-2:]}'
                                  format='%H:%M').dt.time

df['CRS_ARR_TIME'] = pd.to_datetime(df['CRS_ARR_TIME'].apply(
    lambda x: str(f'{x[0:-2] if (x[0:-2] != "" and x[0:-2] != "24") else "0"}:{x[-2:]}'
                                  format='%H:%M').dt.time
```

executed in 11.2s, finished 10:50:29 2021-11-15

In [4]:

```
# Fill most column with na by 0
# As most values in float but actually not go in detail to decimal point.
# Those data is minute counting, therefore it is better to work with them as integer
intData_colList= ['OP_CARRIER_FL_NUM','DEP_TIME','DEP_DELAY','TAXI_OUT','WHEELS_OFF','W
   'ARR_TIME','ARR_DELAY','CANCELLED','DIVERTED','CRS_ELAPSED_TIME','ACTUAL_ELAPSED_TIM
   'AIR_TIME','DISTANCE','CARRIER_DELAY','WEATHER_DELAY','NAS_DELAY','SECURITY_DELAY',

df[intData_colList] = df[intData_colList].apply(
    lambda x: x.fillna(0)).apply(lambda x: x.astype('int64'))

# Re-check with 5 first rows print out
df.head(5)
```

executed in 3.69s, finished 10:50:32 2021-11-15

Out[4]:

	FL_DATE	OP_CARRIER	OP_CARRIER_FL_NUM	ORIGIN	DEST	CRS_DEP_TIME	DEP_TIME
0	2018-01-01	UA	2429	EWR	DEN	15:17:00	1512
1	2018-01-01	UA	2427	LAS	SFO	11:15:00	1107
2	2018-01-01	UA	2426	SNA	DEN	13:35:00	1330
3	2018-01-01	UA	2425	RSW	ORD	15:46:00	1552
4	2018-01-01	UA	2424	ORD	ALB	06:30:00	650

5 rows × 27 columns

In [5]:

```
# Import OP CARRIER lookup table
op_carrier_df = pd.read_csv('op_carrier.csv')
op_carrier_df.head(5)
```

executed in 107ms, finished 10:50:32 2021-11-15

Out[5]:

	Code	Description
0	02Q	Titan Airways
1	04Q	Tradewind Aviation
2	05Q	Comlux Aviation, AG
3	06Q	Master Top Linhas Aereas Ltd.
4	07Q	Flair Airlines Ltd.

In [6]:

```
# Import Origin lookup table
origin_df = pd.read_csv('origin.csv')
origin_df.head(5)
```

executed in 605ms, finished 10:50:33 2021-11-15

Out[6]:

	Code	Description
0	01A	Afognak Lake, AK: Afognak Lake Airport
1	03A	Granite Mountain, AK: Bear Creek Mining Strip
2	04A	Lik, AK: Lik Mining Camp
3	05A	Little Squaw, AK: Little Squaw Airport
4	06A	Kizhuyak, AK: Kizhuyak Bay

In [7]:

```
# Import Dest lookup table
destination_df = pd.read_csv('destination.csv')
destination_df.head(5)
```

executed in 605ms, finished 10:50:34 2021-11-15

Out[7]:

	Code	Description
0	01A	Afognak Lake, AK: Afognak Lake Airport
1	03A	Granite Mountain, AK: Bear Creek Mining Strip
2	04A	Lik, AK: Lik Mining Camp
3	05A	Little Squaw, AK: Little Squaw Airport
4	06A	Kizhuyak, AK: Kizhuyak Bay

In [8]:

```
# Import Cancel code lookup table
cancel_code_df = pd.read_csv('cancel_code.csv')
cancel_code_df
```

executed in 399ms, finished 10:50:34 2021-11-15

Out[8]:

	Code	Description
0	A	Carrier
1	B	Weather
2	C	National Air System
3	D	Security

In [9]:

```
# Update OP Carrier
df['OP_CARRIER'] = df['OP_CARRIER'].map(op_carrier_df.set_index('Code')['Description'].

# Update Origin
df['ORIGIN_DESC'] = df['ORIGIN'].map(origin_df.set_index('Code')['Description'].to_dict

# Update Destination
df['DEST_DESC'] = df['DEST'].map(destination_df.set_index('Code')['Description'].to_dic

# Update Cancel code
df['CANCELLATION_CODE'] = df['CANCELLATION_CODE'].map(cancel_code_df.set_index('Code')[
df['CANCELLATION_CODE'].fillna('N/A')
```

executed in 1.92s, finished 10:50:36 2021-11-15

Out[9]:

```
0          N/A
1          N/A
2          N/A
3          N/A
4          N/A
          ...
1702831    N/A
1702832    N/A
1702833    N/A
1702834    N/A
1702835    N/A
Name: CANCELLATION_CODE, Length: 1702836, dtype: object
```

In [10]:

```
# Flight info
df[['ORIGIN_CITY','ORIGIN_STATE', 'ORIGIN_AIRPORT']] = df.ORIGIN_DESC.str.split('[,|:]'
df[['DEST_CITY','DEST_STATE' , 'DEST_AIRPORT']] = df.DEST_DESC.str.split('[,|:]', expar
df['FLIGHT'] = df['ORIGIN_CITY'] + ' - ' + df['DEST_CITY']

# Strip the space in front
df['ORIGIN_STATE'] = df['ORIGIN_STATE'].str.strip()
df['DEST_STATE'] = df['DEST_STATE'].str.strip()

# Import State abbreviation table
states_df = pd.read_csv('America_States.csv')

# Update States name
df['ORIGIN_STATE'] = df['ORIGIN_STATE'].map(states_df.set_index('Code')['State'].to_dic
df['DEST_STATE'] = df['DEST_STATE'].map(states_df.set_index('Code')['State'].to_dict())

df.drop(columns=['ORIGIN_DESC', 'DEST_DESC'])

df.head(5)
```

executed in 28.0s, finished 10:51:04 2021-11-15

Out[10]:

	FL_DATE	OP_CARRIER	OP_CARRIER_FL_NUM	ORIGIN	DEST	CRS_DEP_TIME	DEP_TIME	DEP_DELAY
0	2018-01-01	United Air Lines Inc.	2429	EWR	DEN	15:17:00	1512	-5
1	2018-01-01	United Air Lines Inc.	2427	LAS	SFO	11:15:00	1107	-8
2	2018-01-01	United Air Lines Inc.	2426	SNA	DEN	13:35:00	1330	-5
3	2018-01-01	United Air Lines Inc.	2425	RSW	ORD	15:46:00	1552	6

Note: As the state is recorded as the abbreviation - the post code. To make the data more meaningful, the America States description table from this link: https://data.humdata.org/dataset/ourairports-usa (https://data.humdata.org/dataset/ourairports-usa)

5.2 Analysing

5.2.1 Operation Carrier Proportion in the First Quarter of 2018

This section will consider the proportion of Operation Carrier operating through 1/1/2018 to 31/3/2018. The result will be shown as pie chart to support in understanding US Carrier operating attributes to overall US domestic flight in the first quarter of 2018.

5.2.1.1 Pseudo Code

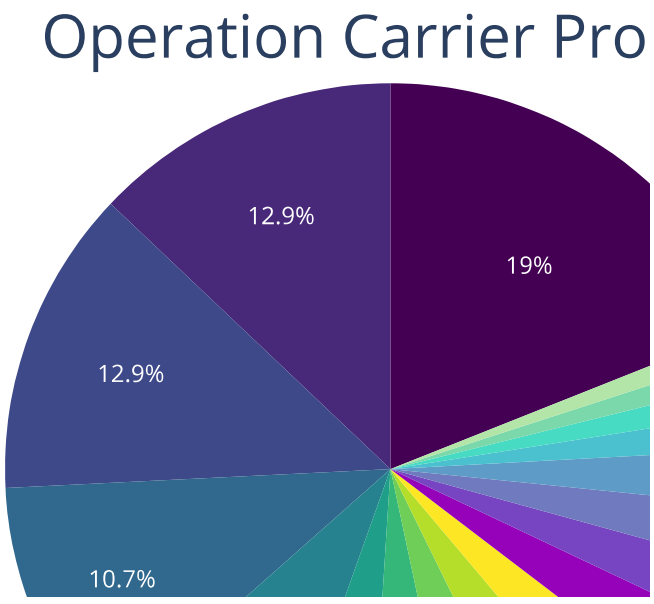
1. Getting unique value and and number of time that value appears in dataset by using numpy unique() function with argument return_counts set to True.
2. Plotting the pie graph by using Plotly library.

5.2.1.2 Solution Code & Visualised Result

In [11]:

```
# Getting data
unique_op_carrier, op_counts = np.unique(df['OP_CARRIER'], return_counts=True)

# Import library
import plotly.express as px

# Ploting
fig = px.pie(df, values=op_counts, names=unique_op_carrier, color_discrete_sequence=px.

# Configuring display
fig.update_layout( legend_bgcolor='#eeeeee',
                  legend_title="Operation Carriers:",
                   title={
                       'text': 'Operation Carrier Proportion',
                       'y': 0.95,
                       'x': 0.5,
                       'xanchor': 'center',
                       'yanchor': 'top',
                       'font_size': 30})

# Configure info display and hover text (label + value)
fig.update_traces(textinfo="percent", hovertext="all")

# Showing plot
fig.show()
```

executed in 27.4s, finished 10:51:31 2021-11-15



*Hover on the chart to get further information

In [12]:

```
# Double check with function
df['OP_CARRIER'].describe()
```

executed in 646ms, finished 10:51:32 2021-11-15

Out[12]:

```
count                    1702836
unique                        18
top       Southwest Airlines Co.
freq                      323113
Name: OP_CARRIER, dtype: object
```

In [13]:

```
print('Mean of operated flight count: ', op_counts.mean())
```

executed in 42ms, finished 10:51:32 2021-11-15

```
Mean of operated flight count:  94602.0
```

5.2.1.3 Discussion

Info returned from describe() function had confirmed the chart display. Southwest Airline Co. had operated most US domestic flights (with 323,113 in total). The number is nearly 1/5 of total flights count. Then it followed by American Airline Inc, Delta Air Lines Inc, SkyWest Airlines Incs. Together, they are top 4 companies take more than 50 percent of overall flights. In this first quarter of 2018, Virgin America seem to be less operated US domestic flights, which is only 17,670 which is less than 76932 flights than the mean of operated flight counts.

5.2.2 Top 15 Longest Distance Flights in US

This section will look into the flight distance data, then visualise them into bar graph to have better evaluate flight distance difference. Even the data set had been scale down, however, it still contains many flights as US is a broad country with 50 states, each state alone also have many airports. Mapping them the number can go over at least 50*(50+1)*0.5 = 1275 connected air ways (in the case, every states only have one airport). Therefore, in this section, only top 15 longest flights will be looked into.

5.2.2.1 Pseudo Code

1. Getting the data frame also dropping any duplicated recored with Pandas DataFrame drop_duplicates() function.
   A. Only the first record in the duplicated group will be keep as we only work on connected way not the flight, the distance between two location are the same, therefore no point to get both ways (or return flight data).
   B. They will be group based on their distance.
2. Filtering dataset to get top 15 records with Pandas DataFrame nlargest() function.

5.2.2.2 Solution Code & Visualised Result

In [14]:

```
# Getting data frame
# In this data frame will contain duplicated data as most flight has 2 ways travels the
fl_distance_data = df[['FLIGHT','DISTANCE']].drop_duplicates(subset=['DISTANCE'], keep=
print('Dataframe size before filtering: ', fl_distance_data.shape)

# As data frame is big therefore it need to narrow down to get 15 longest distance flig
fl_distance_data = fl_distance_data.nlargest(15,'DISTANCE')
```

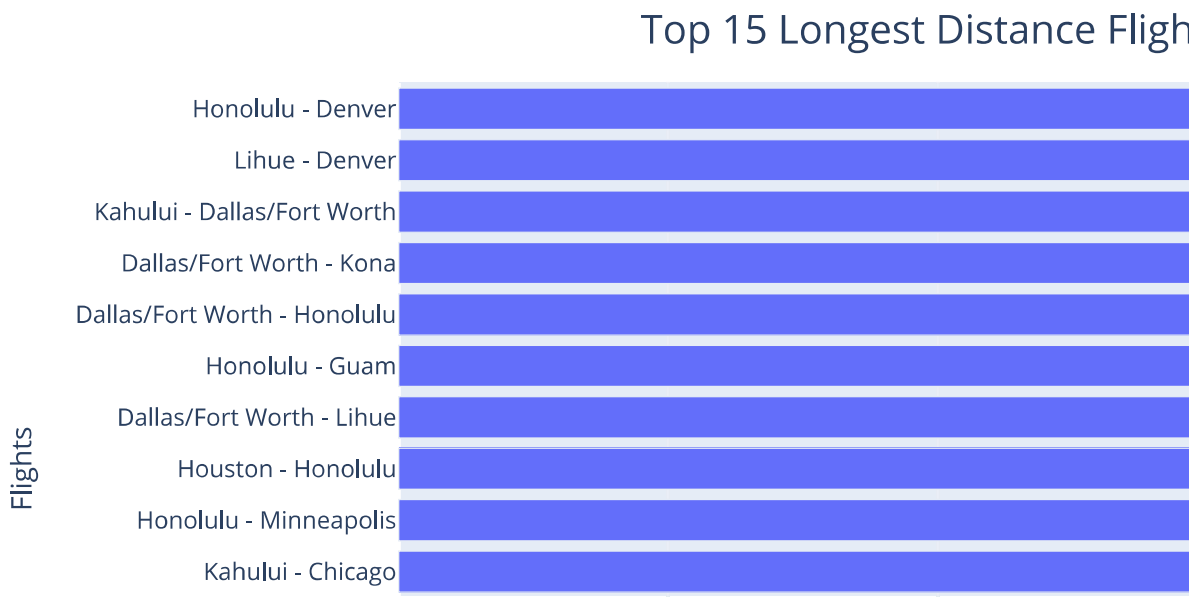
executed in 608ms, finished 10:51:33 2021-11-15

```
Dataframe size before filtering:  (1413, 2)
```

In [15]:

```
# Ploting the bar chart
top_distance_fig = px.bar(fl_distance_data,
             x = fl_distance_data['DISTANCE'],
             y = fl_distance_data['FLIGHT'],
             orientation ='h',
             text = fl_distance_data['DISTANCE'],
             labels = { 'FLIGHT': 'Flights', 'DISTANCE': 'Distance in Miles'})

# Configure layout (adding title)
top_distance_fig.update_layout(
    title_font_size = 20,
    title = {
      'text': 'Top 15 Longest Distance Flights in US',
      'y': 0.95,
      'x': 0.5}
)

# Adding distance show at the end of each bars.
top_distance_fig.update_traces(texttemplate = '%{text}', textposition = 'outside')

# Showing the plot
top_distance_fig.show()
```

executed in 987ms, finished 10:51:34 2021-11-15



*Hover on the chart to get further information

5.2.2.3 Dicussion

Base on the graph, it is can be noticed that in top 15 there are many flights connect to Honolulu, Hawaii. Also in this top rank, other connected points of Hawaii State (Lihue, Kahului). Other connect points laid around from the center US to the East coast. This is understandable as their natural geography distance.

5.2.3 US Domestic Flight Traffic in the 1st Quarter of 2018

If we try to look at on the airport traffic, it seem like the chart has too many factors to look at.Therefore, an zoom out view will be easier to observe. In this case, aviator traffic between state will be looked into. In the previous mention for the connecting flights, it will be there an extreme large number of connecting ways when they are looked as cities level. Thus, a zoom out view will giving better understand and less distracted concepts like which state this city is belongs to or where are these cities. As working one the connecting way, the data frame will be focus on the origin and destination locations.

The objective of this activities is observing the number of flights had been made throughout 1/1/2018 to 31/3/2018. Also the flights operation will be investigate to understand common traffic of each particular carriers.

5.2.3.1 Pseudo Code

1. State by state heat map.
    A. Getting dataframe by slicing it, take ORIGIN_STATE and DEST_STATE columns only.
    B. Adding new column to store counting later by using Pandas dataframe.insert() function.
    C. Counting the the connecting flights by count() function after applying groupby() function to gather the similar ways into group for counting.
    D. Generate lookup table for plotting with Pandas data frame pivot_table() function.
    E. Plotting.

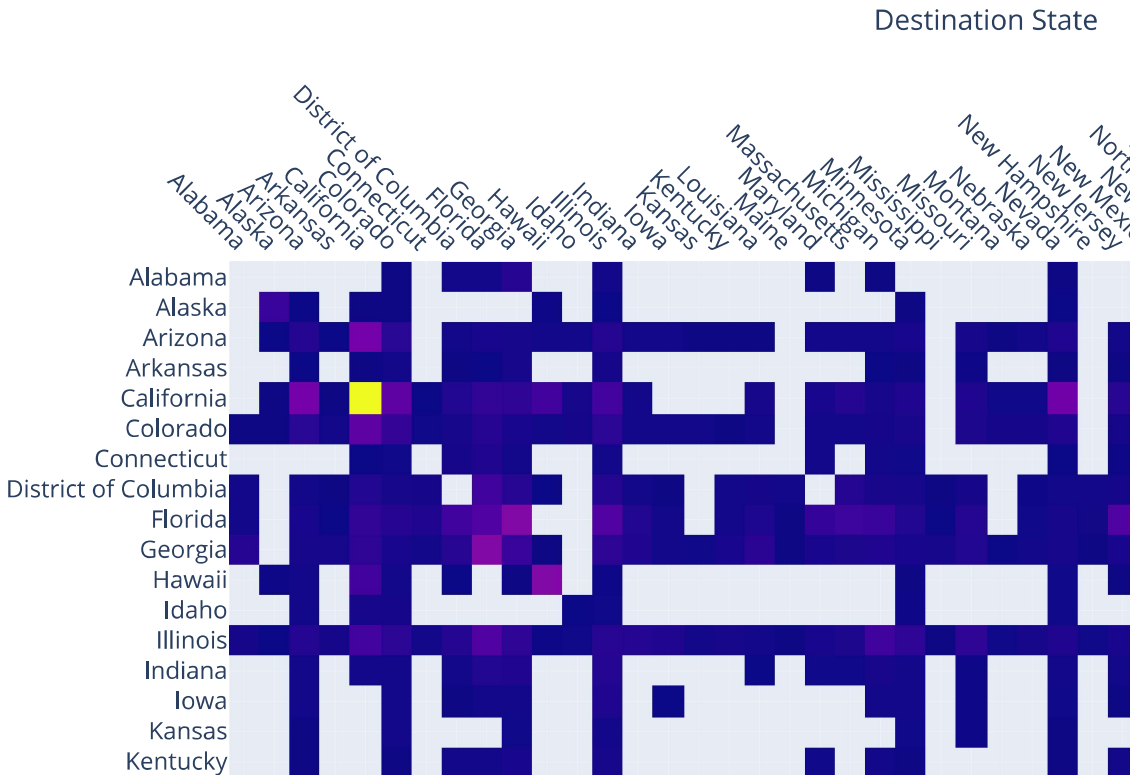
2. Operation Carrier and go-to states flight
    A. Getting dataframe by slicing it, take OP_CARRIER and DEST_STATE columns only.
    B. Adding new column to store counting later by using Pandas dataframe.insert() function.
    C. Counting the the connecting flights by count() function after applying groupby() function to gather the similar ways into group for counting.
    D. Generate lookup table for plotting with Pandas data frame pivot_table() function.
    E. Plotting.

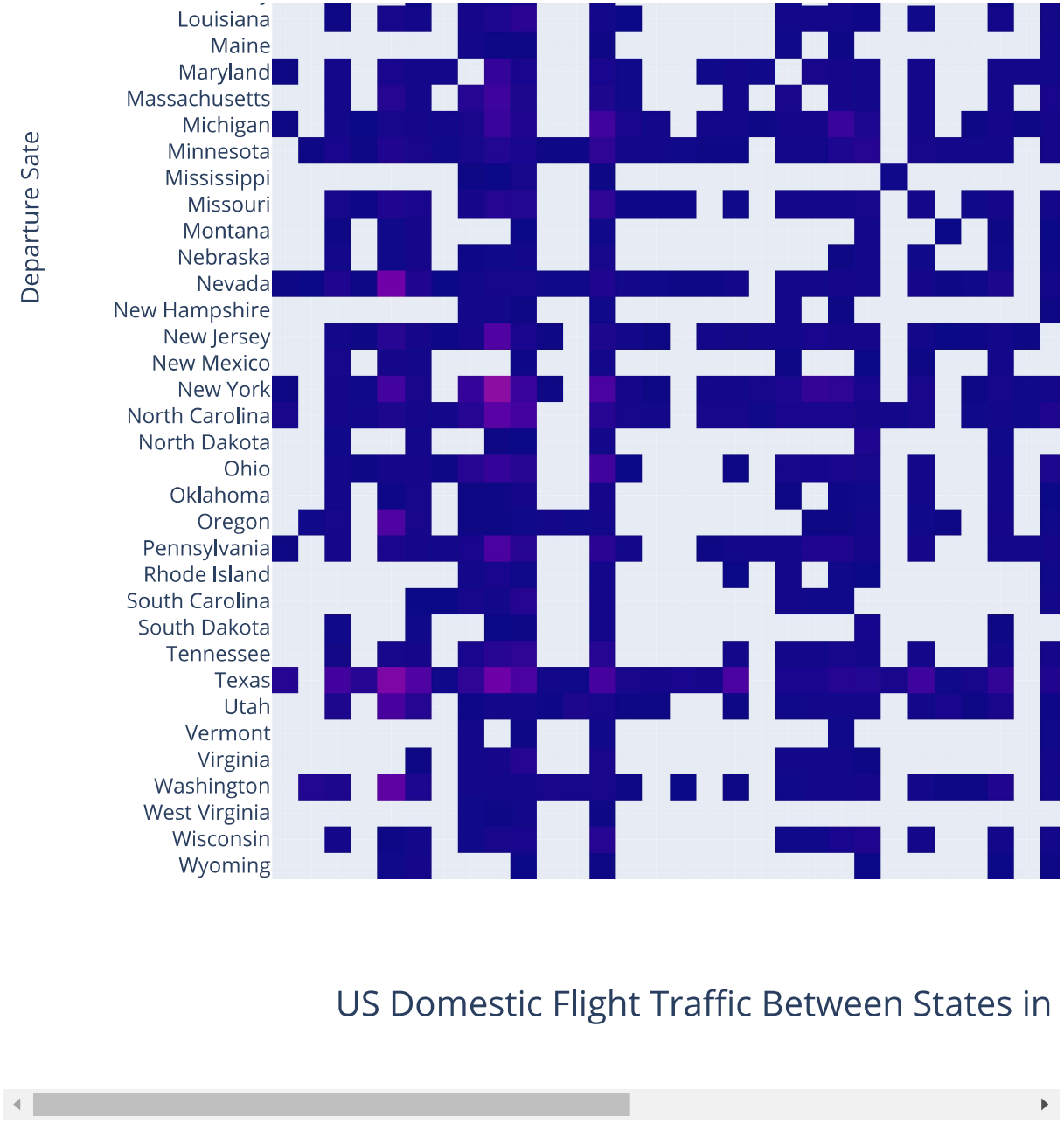
5.2.3.2 Solution Code & Visualised Result

In [16]:

```
# Getting dataset with states
state_traffic_df = df[['ORIGIN_STATE','DEST_STATE']]
state_traffic_df.insert(2,'COUNT','')
state_traffic_df = state_traffic_df.groupby(['ORIGIN_STATE','DEST_STATE']).count().rese

# Create pivot table of flight traffic
states_traffic_tbl =  pd.pivot_table(state_traffic_df, 'COUNT', 'ORIGIN_STATE', 'DEST_S

# Plotting
state_fig = px.imshow(states_traffic_tbl,
                labels = dict(x = 'Destination State', y = 'Departure Sate', color = "F
                x = states_traffic_tbl.columns.tolist(),
                y = states_traffic_tbl.index.tolist())

# Configure layout - figure size, x-axis label's position and map's title
state_fig.update_layout(
    autosize = False,
    width = 1000,
    height = 1000,
    xaxis = {'side':"top"},
    title_font_size = 20,
    title = {
      'text': 'US Domestic Flight Traffic Between States in the 1st Quarter of 2018',
      'y': 0.05,
      'x': 0.5,
      'xanchor': 'center',
      'yanchor': 'top'}
)

# Modify tick - rotate the text to make them easier to read.
state_fig.update_xaxes(categoryorder = 'category ascending',tickangle = 45)

# Display heat map.
state_fig.show()
```

executed in 1.41s, finished 10:51:35 2021-11-15



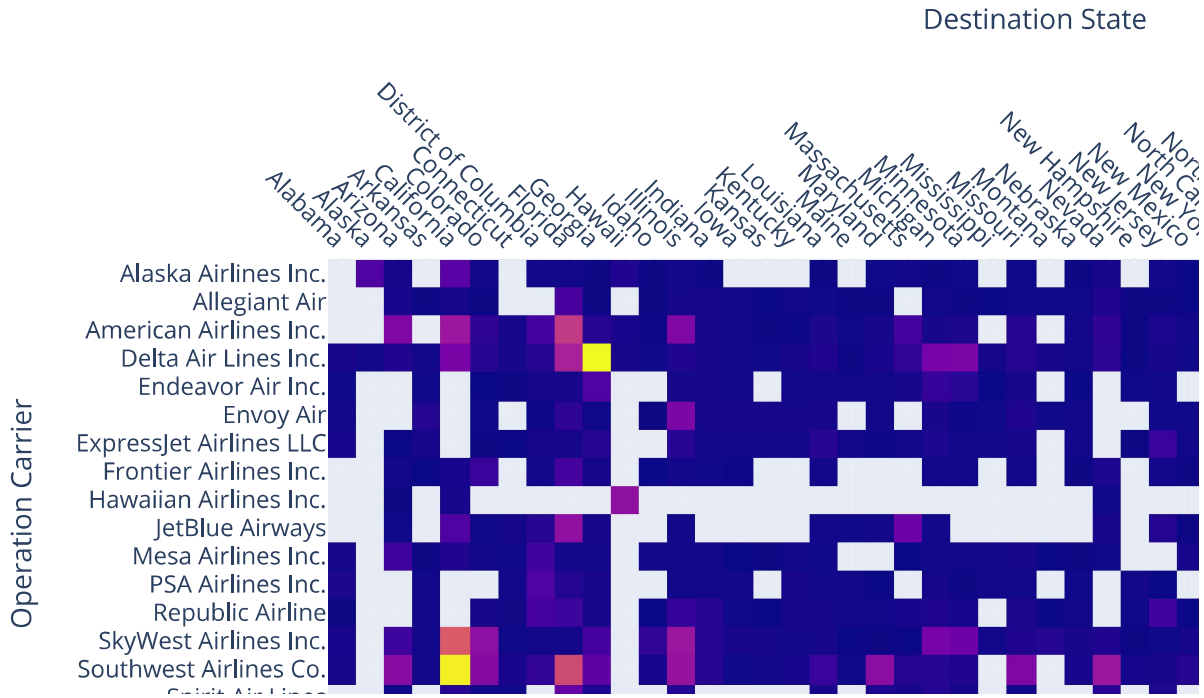


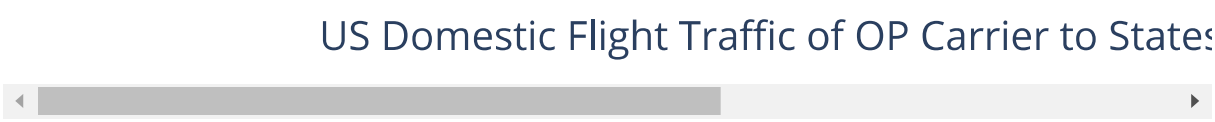
*Hover on the chart to get further information

In [17]:

```
# Getting dataset with states
carrier_traffic_df = df[['OP_CARRIER','DEST_STATE']]
carrier_traffic_df.insert(2,'COUNT','')
carrier_traffic_df = carrier_traffic_df.groupby(['OP_CARRIER','DEST_STATE']).count()

# Create pivot table of flight traffic
carrier_traffic_tbl =  pd.pivot_table(carrier_traffic_df, 'COUNT', 'OP_CARRIER', 'DEST_

carrier_traffic_fig = px.imshow(carrier_traffic_tbl,
                labels = dict(x = 'Destination State', y = 'Operation Carrier', color =
                x = carrier_traffic_tbl.columns.tolist(),
                y = carrier_traffic_tbl.index.tolist())

# Configure layout - figure size, x-axis label's position and map's title
carrier_traffic_fig.update_layout(
    autosize= False,
    width = 1000,
    height = 500,
    xaxis = {'side':"top"},
    title_font_size = 20,
    title = {
        'text': 'US Domestic Flight Traffic of OP Carrier to States in the 1st Quarter
        'y': 0.05,
        'x': 0.5,
        'xanchor': 'center',
        'yanchor': 'top'},
    coloraxis_colorbar = {
        'len': 1,
    }
)

# Modify tick - rotate the text to make them easier to read. Also sorting the states di
carrier_traffic_fig.update_xaxes(categoryorder = 'category ascending',tickangle = 45)

# Display heat map.
carrier_traffic_fig.show()
```

executed in 765ms, finished 10:51:36 2021-11-15



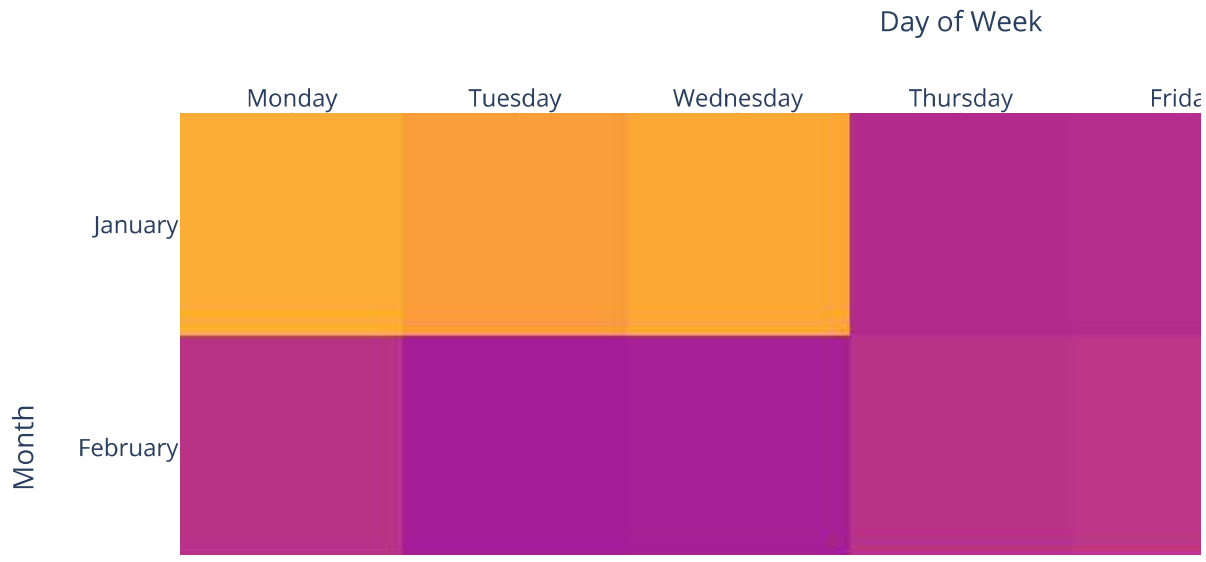


*Hover on the chart to get further information

In [18]:

```
# Generate working dataset
date_df = pd.DataFrame()
date_df['DAY'] = pd.to_datetime(df['FL_DATE']).dt.day_name()
date_df['MONTH'] = pd.to_datetime(df['FL_DATE']).dt.month_name()
date_df.insert(2,'COUNT','')

date_df = date_df.groupby(['DAY','MONTH']).count().reset_index()

# Create pivot table of flight traffic
date_tbl =  pd.pivot_table(date_df, 'COUNT', 'MONTH', 'DAY')

date_tbl = date_tbl.reindex(['January', 'February', 'March'])
date_tbl = date_tbl[['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday'
date_fig = px.imshow(date_tbl,
                labels = dict(x = 'Day of Week', y = 'Month', color = "Flights Count"),
                x = date_tbl.columns.tolist(),
                y = date_tbl.index.tolist())

# Configure layout - figure size, x-axis label's position and map's title
date_fig.update_layout(
    autosize= False,
    width = 1000,
    height = 500,
    xaxis = {'side':"top"},
    title_font_size = 20,
    title = {
        'text': 'US Domestic Flight Traffic by Date of Week throughout the First Quarte
        'y': 0.05,
        'x': 0.5,
        'xanchor': 'center',
        'yanchor': 'top'},
    coloraxis_colorbar = {
        'len': 1,
    }
)

# date_fig.update_yaxes(category_orders = {'Month': ['January', 'February', 'March']})

# Display heat map.
date_fig.show()
```

executed in 5.92s, finished 10:51:42 2021-11-15



*Hover on the chart to get further information

5.2.3.3 Discussion

From the first graph, the info that can be extract that internal state flight from Texas and California. Also they are two states thats have most visit from other states and also in return. Florida and Georgia are the next two states has high traffic even more than New York.

In the next heat maps, it again confirm the fact from the previous charts that California, Texas, Floria, Georgia have busy traffic. Those flights also operated by top 4 Carriers: Delta Air Lines Inc., America Airlines Inc., SkyWest Airlines Inc. and Southwest Airlines Co.; United Air Lines Inc. also had connected some high traffic locations.

- **America Airlines Inc.** had operated flights to: Arizona, California, Florida, Georgia, Michigan, Minnesota, New York, North Carolina, Pennsylvania, Texas
- **Delta Airline Inc.** arranged many flights to: California, Florida, Georgia, New York, Utah
- **SkyWest Airlines Inc.** had flights connect to: California, Colorado, Illinois, Michigan, Minnesota, Vermont, Utah
- **Southwest Airlines Co.** dominated for flights to: California, Colorado, Illinois, Maryland, Missouri, Nevada, New York, Texas

Travelers behaviour changed a bit through months. However, most of the flights were made in the week days rather than weekends. Little shift happened in March, customers start to have more trip in weekends. More intensive on Thursday and Friday in March, while its was Monday to Wednesday in January. February is slightly even out for every day of week, expect Saturday which has lowed number of flight. It can be said that, February was an transit period.

5.2.4 Daily Delay Counting

In this section, the delay time will be observed to understand their number of delay flights everyday.

5.2.4.1 Pseudo Code

1. Getting dataframe by slicing it, take FL_DATE, OP_CARRIER columns only.
   A. Calculate the total delay times by summing DEP_DELAY and ARR_DELAY column, the value will be put into new column names TOTAL_DELAY.
   B. Filter to get the delay flights, whose records had positive value of TOTAL_DELAY.

C. Couting the delayed flights by count() function after applying groupby() function to gather the similar ways into group for counting.

2. Calculate the total delayed time.

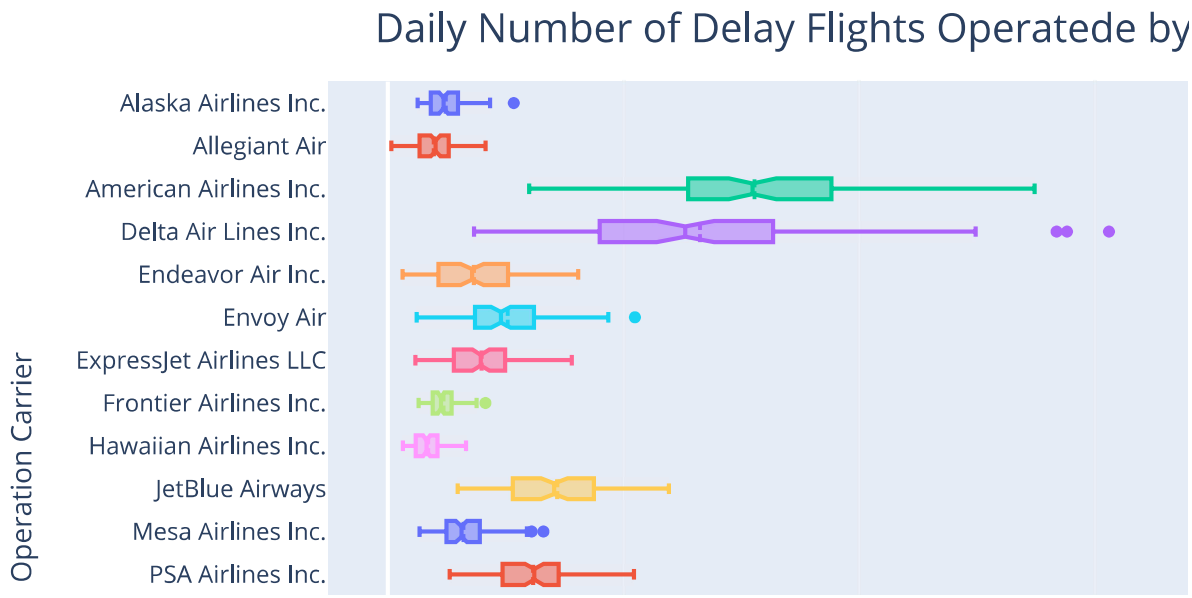
3. Sketching the box plot.

5.2.4.2 Solution Code & Visualised Result

In [19]:

```
# Suppress unnecessary messages
pd.options.mode.chained_assignment = None

# Extract and get the work data set
delay_df = df[['FL_DATE','OP_CARRIER']]
delay_df['TOTAL_DELAY'] = df['DEP_DELAY'] + df['ARR_DELAY']
daily_delay_df = delay_df[delay_df['TOTAL_DELAY'] > 0]
daily_delay_df = daily_delay_df.groupby(['FL_DATE','OP_CARRIER']).count()
daily_delay_df = daily_delay_df.reset_index()

# Sketching box plot
daily_delay_fig = px.box(daily_delay_df, x = daily_delay_df['TOTAL_DELAY'], y = daily_
                         notched = True,

                         color = daily_delay_df['OP_CARRIER'],
                         labels = { 'OP_CARRIER': 'Operation Carrier', 'TOTAL_DELAY':

# Display mean
daily_delay_fig.update_traces(boxmean = True)

# Add the title
daily_delay_fig.update_layout(
    title_font_size = 20,
    title = {
      'text': 'Daily Number of Delay Flights Operatede by Aviation Carrier in US',
      'y': 0.95,
      'x': 0.5}
)

# Display boxplot
daily_delay_fig.show()
```

executed in 2.00s, finished 10:51:44 2021-11-15



*Hover on the chart to get further information

5.2.4.3 Discussion

In this plot, the significant info that can be extracted is top 4 airline companies (Delta Air Lines Inc., America Airlines Inc., SkyWest Airlines Inc. and Southwest Airlines Co.) had most delay flights. One of them, Southwest Airlines Co. has highest numbers reach, also the average figure. Not only about the qunatity but also they have big gap of difference in this collection, it could be the variation value of daily delayed days.

5.2.5 Flights Operated by Carrier

In this section, daily flights operated by each operaters will be look into, also the growth of the flights througout the first quarter of 2018.

5.2.5.1 Pseudo Code

1. Daily fight counting
   A. Garther data set by slicing and taking data from two columns: FL_DATE, OP_CARRIER.
   B. Adding new column for later store calculated flight counting.
   C. Couting the delayed flights by count() function after applying groupby() function to gather the similar ways into group for counting.
   D. Ploting chart.

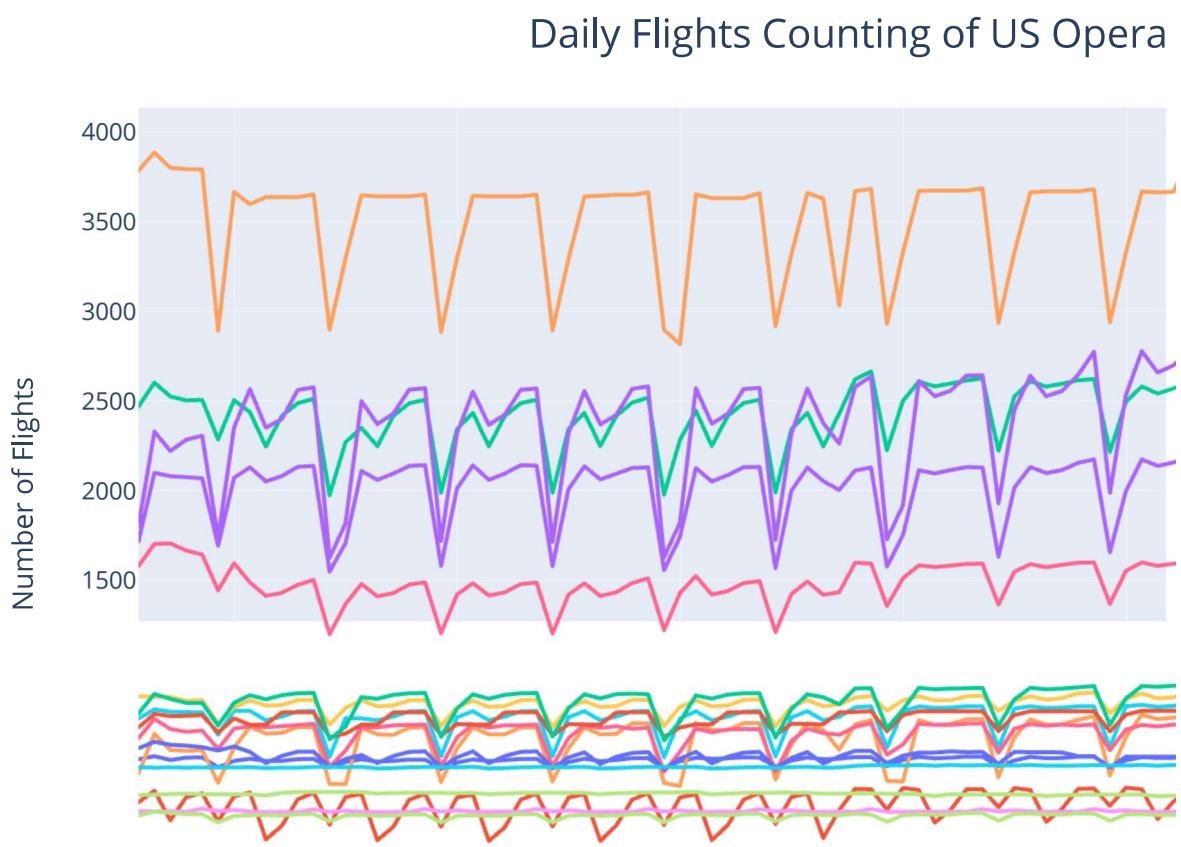
2. Flights operation growth
   A. Garther data set by slicing and taking data from two columns: FL_DATE, OP_CARRIER.
   B. Adding new column for later store calculated flight counting.
   C. Couting the delayed flights by count() function after applying groupby() function to gather the similar ways into group for counting.
   D. Modify the dataframe to get the right format to calculate the accumulate sum.
   E. Calculate accumulate sum with cumsum() function.
   F. Ploting chart.

5.2.5.2 Solution Code & Visualised Result

In [20]:

```
# Creating dataframe
daily_flight_df = df[['FL_DATE','OP_CARRIER']]
daily_flight_df.insert(2,'COUNT','')
daily_flight_df = daily_flight_df.groupby(['FL_DATE','OP_CARRIER']).count().reset_index

# Plotting chart
daily_flight_fig = px.line(daily_flight_df, x = "FL_DATE", y = "COUNT", color = 'OP_CAR
                           labels = { 'OP_CARRIER': 'Operation Carrier', 'FL_DATE': 'Fli
                                      'COUNT': 'Number of Flights'})

# Adding title and modify the position of title
daily_flight_fig.update_layout(
    title_font_size = 20,
    title = {
      'text': 'Daily Flights Counting of US Operation Carriers',
      'y': 0.97,
      'x': 0.5,
      'xanchor': 'center',
      'yanchor': 'top'}
)

# Display chart
daily_flight_fig.show()
```

executed in 1.76s, finished 10:51:46 2021-11-15

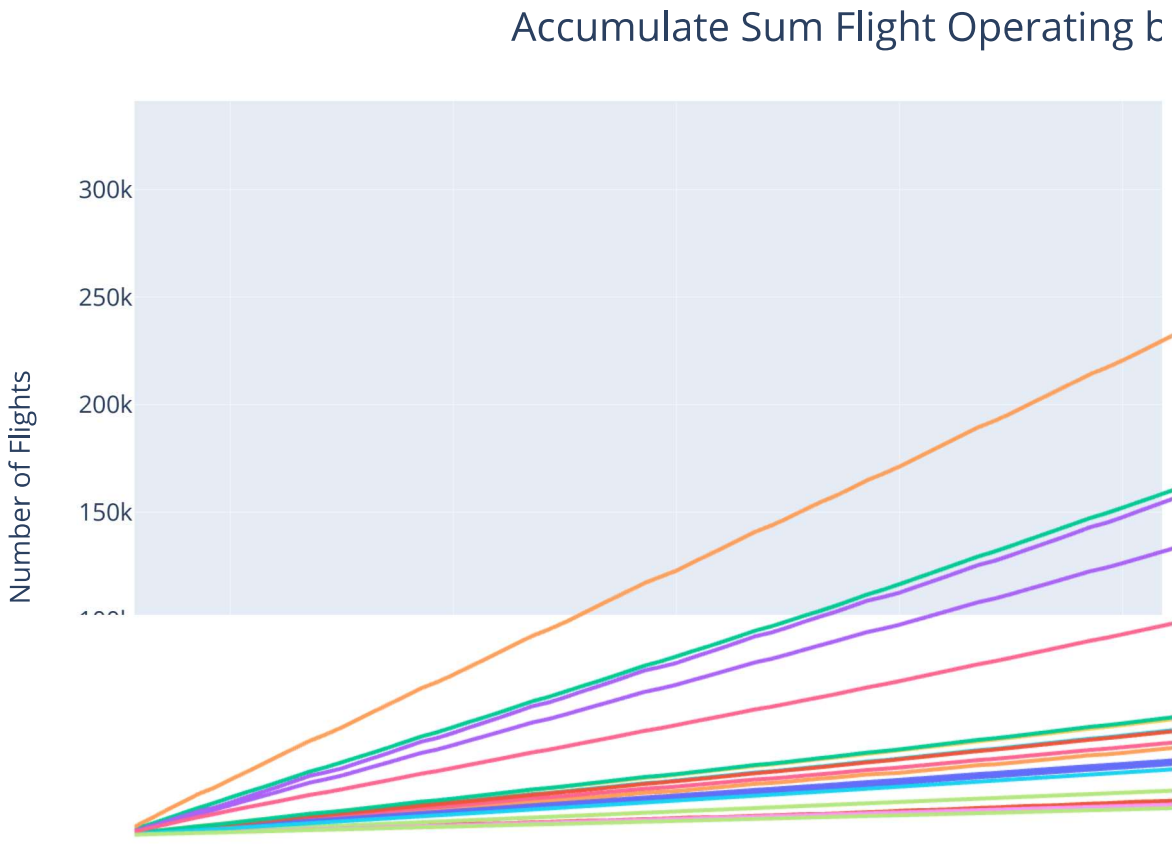


*Hover on the chart to get further information

In [21]:

```
# Collecting data and put into new data frame
flight_df = df[['FL_DATE','OP_CARRIER']]
flight_df.insert(2,'COUNT','')
flight_df = flight_df.groupby(['FL_DATE','OP_CARRIER']).count().reset_index()

# Create pivot table of flight traffic
traffic_pivot_tbl =  pd.pivot_table(flight_df, 'COUNT', 'FL_DATE', 'OP_CARRIER')

# Calculate accumulate sum
cumsum_flight_df = traffic_pivot_tbl.cumsum()

# Plotting chart
cumsum_flight_fig = px.line(cumsum_flight_df, x = cumsum_flight_df.index, y =  cumsum_f

# Adding title and modify the position of title
cumsum_flight_fig.update_layout(
    title_font_size = 20,
    title = {
      'text': 'Accumulate Sum Flight Operating by US Carriers',
      'y': 0.97,
      'x': 0.5,
      'xanchor': 'center',
      'yanchor': 'top'},
    xaxis_title = "Flight Date",
    yaxis_title = "Number of Flights",
)

# Display the chart
cumsum_flight_fig.show()
```

executed in 1.20s, finished 10:51:47 2021-11-15



*Hover on the chart to get further information

5.2.5.3 Discussion

In the daily flight counting line chart, Delta Air Lines Inc., America Airlines Inc., SkyWest Airlines Inc. and Southwest Airlines Co. has most flights operated everyday. Southwest Airlines Co. led in this area. Weekly, there is one day, every carriers has the number of flight drops. This event happens which brings the counting of Skywest and Delta come closely in the first half of January.

Spirit Air Lines, Frontiers Airlines Inc., United Air Lines Inc. and Virgin America have most stable count with less than 500 flights everyday.

Again, top 4 in the previous chart also take the lead in the growth of accumulate count. In contract, the top 4 stable fights number has slow growth.

5.2.6 First of January 2018 Travel Destinations

This section will study the flight destination of everyone on the New Year Eve. The destination will be popup on the US map. As the plotting function need the latitude and longitude to locate the position on the map, therefor, the support dataset will be import to support this activity.

Location support dataset link:
https://github.com/plotly/datasets/blob/master/2011_february_us_airport_traffic.csv
(https://github.com/plotly/datasets/blob/master/2011_february_us_airport_traffic.csv)

5.2.6.1 Pseudo Code

1. Slice the original dataframe to get the approriate working data.
2. Import support dataset.
3. Adding new column to store counting later by using Pandas dataframe.insert() function.
4. Update/ Adding new columns for latitude and longitude by using Pandas dataframe map() function to update abbreviations. This will be link the data with dictionary base on the lookup tables.
5. Adding explanation text to support the plot hover info display later.
6. Plotting

5.2.6.2 Solution Code & Visualised Result

In [22]:

```
# Getting data frame
newyear_df = df.loc[df.FL_DATE == '2018-01-01', ['DEST']]
# Import airport location to support the graph plotting
airport_location_df = pd.read_csv('https://raw.githubusercontent.com/plotly/datasets/ma
```

executed in 795ms, finished 10:51:48 2021-11-15

In [23]:

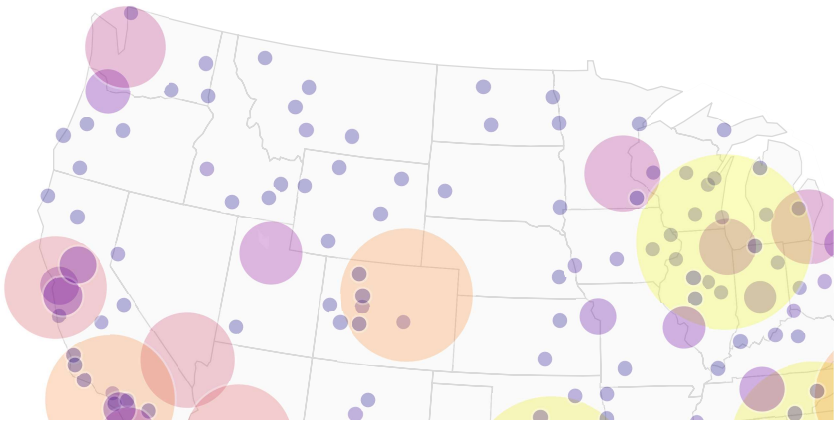
```
# Adding new column for inserting counting number later
newyear_df.insert(1,'COUNT','')
newyear_df = newyear_df.groupby('DEST').count().reset_index()
airport_location_df = airport_location_df[['iata', 'lat', 'long']]

# Adding new columns, values is intepreted from DEST column by using dictionary formed f
newyear_df['LATITUDE'] = newyear_df['DEST'].map(airport_location_df.set_index('iata')[
newyear_df['LONGTITUDE'] = newyear_df['DEST'].map(airport_location_df.set_index('iata')

# Update Destination and adding new DESC column to support hover info display
newyear_df['DESC'] = newyear_df['DEST'].map(destination_df.set_index('Code')['Descripti

newyear_df.insert(5,'MRK_SIZE', 0)
```

executed in 41ms, finished 10:51:48 2021-11-15

In [24]:

```
# Import library
import plotly.graph_objects as go

# Update the marker size
newyear_df.loc[((newyear_df['COUNT']/10) <= 10),'MRK_SIZE'] =  8
newyear_df.loc[((newyear_df['COUNT']/10) > 10),'MRK_SIZE'] =  ((newyear_df['COUNT']/10)

# Configure graph display
newyear_fig = go.Figure(data=go.Scattergeo(
    locationmode = 'USA-states',
    lon = newyear_df['LONGTITUDE'],
    lat = newyear_df['LATITUDE'],
    text = newyear_df['DESC'],
    mode = 'markers',
    marker = dict(
        size = newyear_df['MRK_SIZE'],
        reversescale  = False,
        opacity = 0.3,
        symbol = 'circle',
        line = dict(
            width=1,
            color='rgba(102, 102, 102)'
        ),
        cmin = 0,
        color = newyear_df['COUNT'],
        cmax = newyear_df['COUNT'].max(),
        colorbar_title="Incoming flights<br>1/1/2018"
    )))

# Add Title and chosing the country for map plot
newyear_fig.update_layout(
    title={
        'text': '2018 New Year Travel Destinations',
        'y':0.95,
        'x':0.5,
        'xanchor': 'center',
        'yanchor': 'top',
        'font_size': 30},
    geo = dict(
        scope ='usa',
        projection_type='albers usa',
        showland = True,
        landcolor = "rgb(250, 250, 250)",
        subunitcolor = "rgb(217, 217, 217)",
        countrycolor = "rgb(217, 217, 217)",
        countrywidth = 0.5,
        subunitwidth = 0.5
        ),
    )

# Display plot
newyear_fig.show()
```

executed in 2.63s, finished 10:51:50 2021-11-15



*Hover on the chart to get further information

5.2.6.3 Discussion

Base one the visual result of the map, on 1/1/2018 most flight direct to either West or East coast. However, the half East side gain more arrivals especially: Texas, Illinois, Georgia, North Carolina. West side, there are two high traffic states, California, Arizona, Washington, Nevada. In the center of US, Colorado is the one has most flights came to.Travelers spreed out throughout East side, when on West side, they more focus on West coast or up north.

5.2.7 General Flight Distance Operated by Aviator Carriers for US Domestic Flights

This part will study about the flight destinations and which state has long distance connections and how many flights did reach them.

5.2.7.1 Pseudo Code

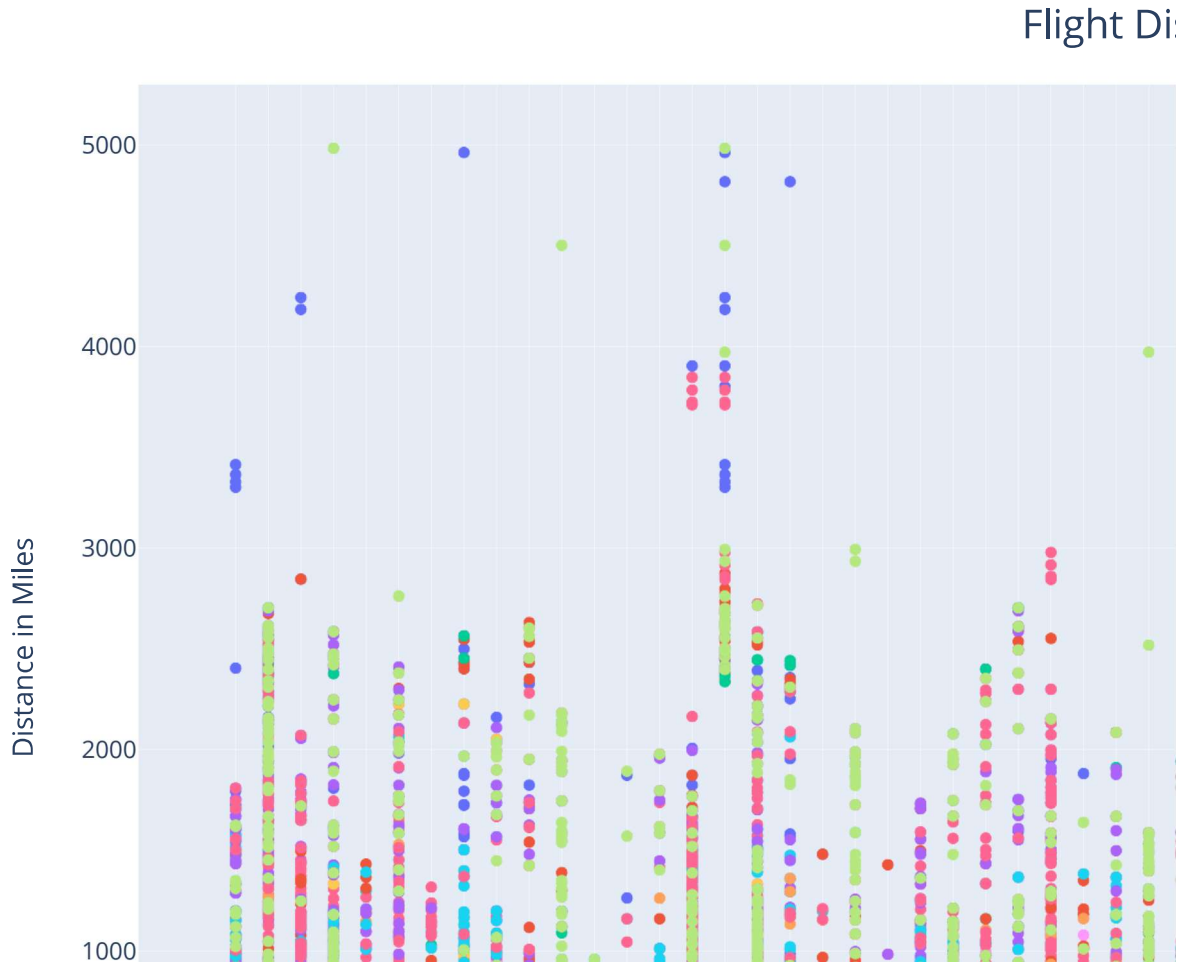
1. Getting the working dataset with data from OP_CARRIER, DEST_STATE, DISTANCE
2. Plotting chart

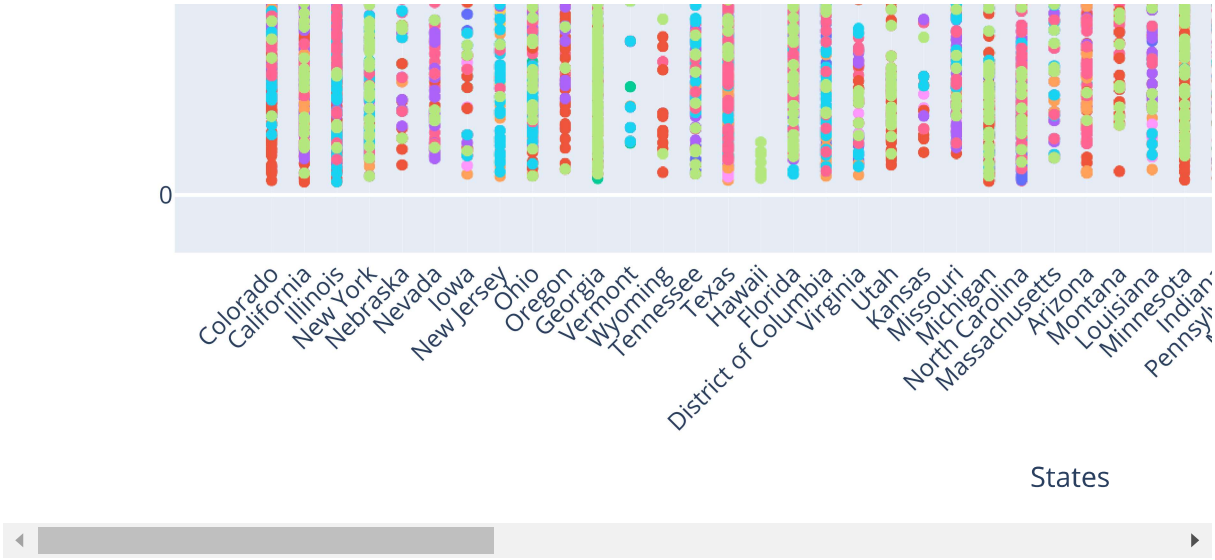
5.2.7.2 Solution Code & Visualised Result

In [25]:

```
# Getting data set
dist_carrier_df = df[['OP_CARRIER', 'DEST_STATE','DISTANCE']].drop_duplicates( keep='fi

# Plotting
dist_carrier_fig = px.scatter(dist_carrier_df, x="DEST_STATE", y="DISTANCE",
                              color="OP_CARRIER", marginal_y="rug")

# Configure layout, at graph title, x and y axises title also with figure size
dist_carrier_fig.update_layout(
    title_font_size = 20,
    title = {
      'text': 'Flight Distance to States Operating by US Carriers',
      'y': 0.97,
      'x': 0.5,
      'xanchor': 'center',
      'yanchor': 'top'},
    xaxis_title = 'States',
    yaxis_title = 'Distance in Miles',
    legend_title = 'Operation Carriers',
    height = 750,
    width = 1500,
)

# Modify the tick lable display angle
dist_carrier_fig.update_xaxes(tickangle = -45)

# Show plot.
dist_carrier_fig.show()
```

executed in 2.18s, finished 10:51:52 2021-11-15





*Hover on the chart to get further information

5.2.7.3 Discussion

The chart marked that these states - New York, New Jersey, Hawaii, District of Columbia, Georgia, Illinois have some long flight distance (over 4000 miles). Few of them have flight operated with the distance from 3000 to 4000 miles, only Colorado, Texas, Hawaii, Minnesota. Every states has short distance flight below 1000 miles.

United Air Lines Inc., Hawaii Airlines Inc., Delta Air Lines Inc. operate most long flights. Beside, United Air Lines Inc., Delta Air Lines Inc.; America Airlines Inc. is another option for the flight between 3000 to 4000 miles. In the range below 3000 miles, most carrier companies has connecting flights. However, Endeavor Air Inc, Envoy Air, PSA Airlines Inc., ExpressJet Airlines LLC, Skywest Airlines Inc, Mesa Airlines Incs and republic Airline only operate flight below about 1500 miles. Hawaii Airlines Inc. does not have any flight in the range of 300 miles to 2300 miles.

6 Limitations

This report does not reflect whole view about US domestic flights traffic or behaviour. As the main working dataset is taken records in the first quarter of 2018.

Some of the factors are not fully preset or will be miss out in this report likes: holidays, business factors, weather factors, customer's reference, deal or promotion offer from the carriers,...

In the map, as the feature is not support to sketching Hawaii state, therefore it is miss out on the display for the point 5.2.5

This report had a general counting of the delay flight however, it does not focus on the reason why they got delay.

7 Conclusion

There are some point that can be concluded from the data and visualised graphs:

1. Top 4 Operational Aviation Carriers contribute to the mass number of flights, also number of delayed flights:
    - Southwest Airline Co.

- American Airline Inc
- Delta Air Lines Inc
- SkyWest Airlines Incs.

2. Most longest flights are:
    - Connecting between main continent and Hawaii state
    - From East coast to West coast and vice versa

3. California, Texas, Floria, Georgia most have busy traffic.

In the part 5.2.6, even on the holiday, the traffic does not change. 1/1/2018 is on Monday, there are some factors that need to be consider about the quantity of flights. On this day, travelers might back from holiday or fly to other state as they took first week of January for holiday.

Top 4 Carriers also the one has most delayed flight, but base on number of flight they had operated, which could be that, the customers still have a trust in these company. The number of they delay with great number of flight provided does not mean they have long delayed (which this report did not investigate). As this report only focus on whether is delay or not, therefor the enormous count in delayed flights is obviously understandable.

Interestingly, only Southwest Airline in the top 4 originally founded on the West sides while other 3 is from East parts. Together, they are connected East point and West points.

In those high aviator traffics states, only Texas has high ranking in the economic growth rate (top 3) of 2018 base on Business Insider Australia news; following by Florida (rank 13), California (rank 16) and Georgia (rank 17). As the limited of the scope, and the investigate time frame, this is why considering the aviation traffic cannot fully reflect the economics. Every states have their firm strength and industry. In addition, there are other option for transportation. Airlines is mostly beneficial for hospitality, transportation, tourism industry, which are not the leading industry or just a partial contribute to the overall growth of those states.

People seem to travel in the week days more than weekend. This is could base on the working time of most economic activity. Also, some operation carrier could have some promotion programs to encourage their customer travel in this time. On the other hand, work-from-home or distanced/ off-site working model had been might had applied before the Covid. It also could be most companies have many branches that allow their employees to work most of the place in US. It's understandable for the common sense that people will travel on Friday for weekend holiday to save time rather than make a flight on the following day, which explain the large number of flights on Friday. However, as the dataset did not observer the public holidays of US, therefor the change in the flight number might affect as well as which day will they take flights.