Python初級數據分析員證書

（六）數據分析 及可視化專案

13. 數據分析專案

Demo 6 – Flights Delays and Cancels

Review

- Statistics
- Hypothesis testing
- Algebra
- Linear regression
- Propositional logic
- Python
- R
- SQL

- Pandas, NumPy, SciPy
- Data Visualization, Matplotlib, Seaborn, Plotly
- Dashboard Visualization, Business Intelligence
- Storytelling

Chapter Summary

- Data source
- Import data
- Missing data summary
- Merge relational tables
- Transform weekday number and month number

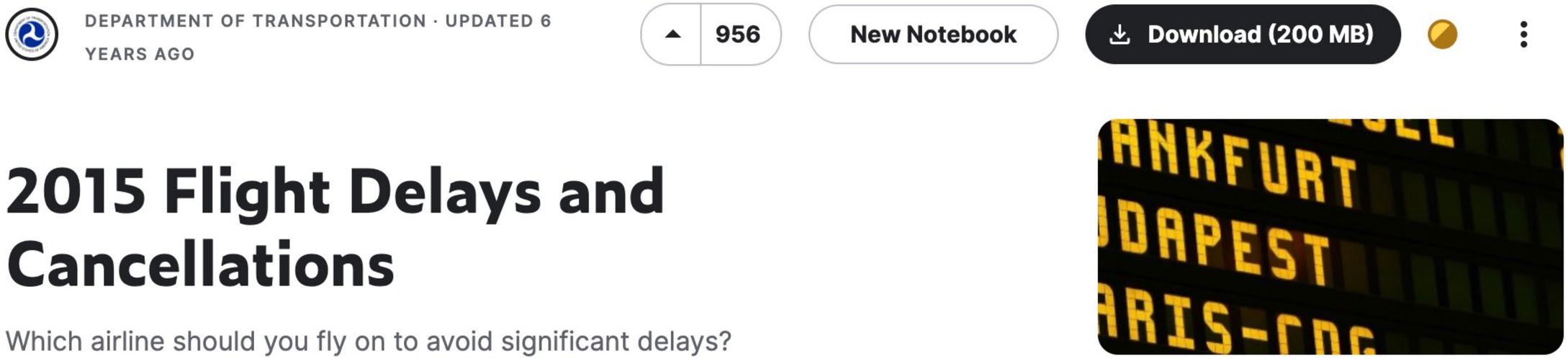
Scenario

Imagine that you are a data analysis of an airline. You are asked to analyse data from delays and cancels. Base on your analysis and visualization, conclude at least 5 key points and make suggestion to operation department.

Data Import

Dataset download link: https://www.kaggle.com/datasets/usdot/flight-delays

Or search “2015 Flight Delays and Cancellations” in Kaggle



Data source

The flight delay and cancellation data was collected and published by the The U.S. Department of Transportation’s DOT's Bureau of Transportation.

The statistics tracks the on-time performance of domestic flights operated by large air carriers and their summary information on the number of on-time, delayed, canceled, and diverted flights.

Take a glance of data

```
1  import pandas as pd
2  import numpy as np
3
4  import plotly.offline as py
5  import plotly.figure_factory as ff
6  import plotly.graph_objs as go
7  from plotly import tools
8  import matplotlib.pyplot as plt
9  import seaborn as sns
10
11 df = pd.read_csv('../flights/flights.csv', low_memory=False)
12 df
```

	YEAR	MONTH	DAY	DAY_OF_WEEK	AIRLINE	FLIGHT_NUMBER	TAIL_NUMBER	ORIGIN_AIRPORT	DESTINATION_AIRPORT	SCHEDULED_DEPARTURE
0	2015	1	1	4	AS	98	N407AS	ANC	SEA	5
1	2015	1	1	4	AA	2336	N3KUAA	LAX	PBI	1(
2	2015	1	1	4	US	840	N171US	SFO	CLT	2(
3	2015	1	1	4	AA	258	N3HYAA	LAX	MIA	2(
4	2015	1	1	4	AS	135	N527AS	SEA	ANC	2!

Take a glance of data

For many columns that could not be shown horizontally, you may transpose a few rows to view sample data vertically.

```
1 df.head(2).T
```

	0	1
YEAR	2015	2015
MONTH	1	1
DAY	1	1
DAY_OF_WEEK	4	4
AIRLINE	AS	AA
FLIGHT_NUMBER	98	2336
TAIL_NUMBER	N407AS	N3KUAA
ORIGIN_AIRPORT	ANC	LAX
DESTINATION_AIRPORT	SEA	PBI
SCHEDULED_DEPARTURE	5	10
DEPARTURE_TIME	2354.0	2.0

Assumption in variables calculation

- taxi_out = wheels_off - departure_time
- taxi_in = arrival_time - wheels_on
- air_time = wheels_on - wheels_off
- elapsed_time =air_time + taxi_in + taxi_out

Missing data summary

```
pd.concat([df.isnull().sum(), 100 * df.isnull().sum()/len(df)],
                axis=1).rename(columns={0:'Missing Records', 1:'Percentage (%)'})
```

	Missing Records	Percentage (%)
YEAR	0	0.000000
MONTH	0	0.000000
DAY	0	0.000000
DAY_OF_WEEK	0	0.000000
AIRLINE	0	0.000000
FLIGHT_NUMBER	0	0.000000
TAIL_NUMBER	14721	0.252978
ORIGIN_AIRPORT	0	0.000000
DESTINATION_AIRPORT	0	0.000000
SCHEDULED_DEPARTURE	0	0.000000

Import airlines

```
1 airlines = pd.read_csv('../flights/airlines.csv')
2 airlines
```

	IATA_CODE	AIRLINE
0	UA	United Air Lines Inc.
1	AA	American Airlines Inc.
2	US	US Airways Inc.
3	F9	Frontier Airlines Inc.
4	B6	JetBlue Airways
5	OO	Skywest Airlines Inc.
6	AS	Alaska Airlines Inc.
7	NK	Spirit Air Lines
8	WN	Southwest Airlines Co.
9	DL	Delta Air Lines Inc.
10	EV	Atlantic Southeast Airlines
11	HA	Hawaiian Airlines Inc.
12	MQ	American Eagle Airlines Inc.
13	VX	Virgin America

Merge df and airlines' name

```
1 df = pd.merge(df,airlines, left_on='AIRLINE', right_on = 'IATA_CODE')
2 df.insert(loc=5, column='AIRLINE', value=df.AIRLINE_y)
3 df = df.drop(['AIRLINE_y','IATA_CODE'], axis=1)
```

```
1 df
```

	YEAR	MONTH	DAY	DAY_OF_WEEK	AIRLINE_x	AIRLINE	FLIGHT_NUMBER	TAIL_NUMBER	ORIGIN_AIRPORT	D
0	2015	1	1	4	AS	Alaska Airlines Inc.	98	N407AS	ANC	
1	2015	1	1	4	AS	Alaska Airlines Inc.	135	N527AS	SEA	
2	2015	1	1	4	AS	Alaska Airlines Inc.	108	N309AS	ANC	
3	2015	1	1	4	AS	Alaska Airlines Inc.	122	N413AS	ANC	

Merge airport data into df

```
airport = pd.read_csv('../flights/airports.csv')
df = pd.merge(df,airport[['IATA_CODE','AIRPORT','CITY']], left_on='ORIGIN_AIRPORT', right_on = 'IATA_CODE')
df = df.drop(['IATA_CODE'], axis=1)
df = pd.merge(df,airport[['IATA_CODE','AIRPORT','CITY']], left_on='DESTINATION_AIRPORT', right_on = 'IATA_CODE')
df = df.drop(['IATA_CODE'], axis=1)
```

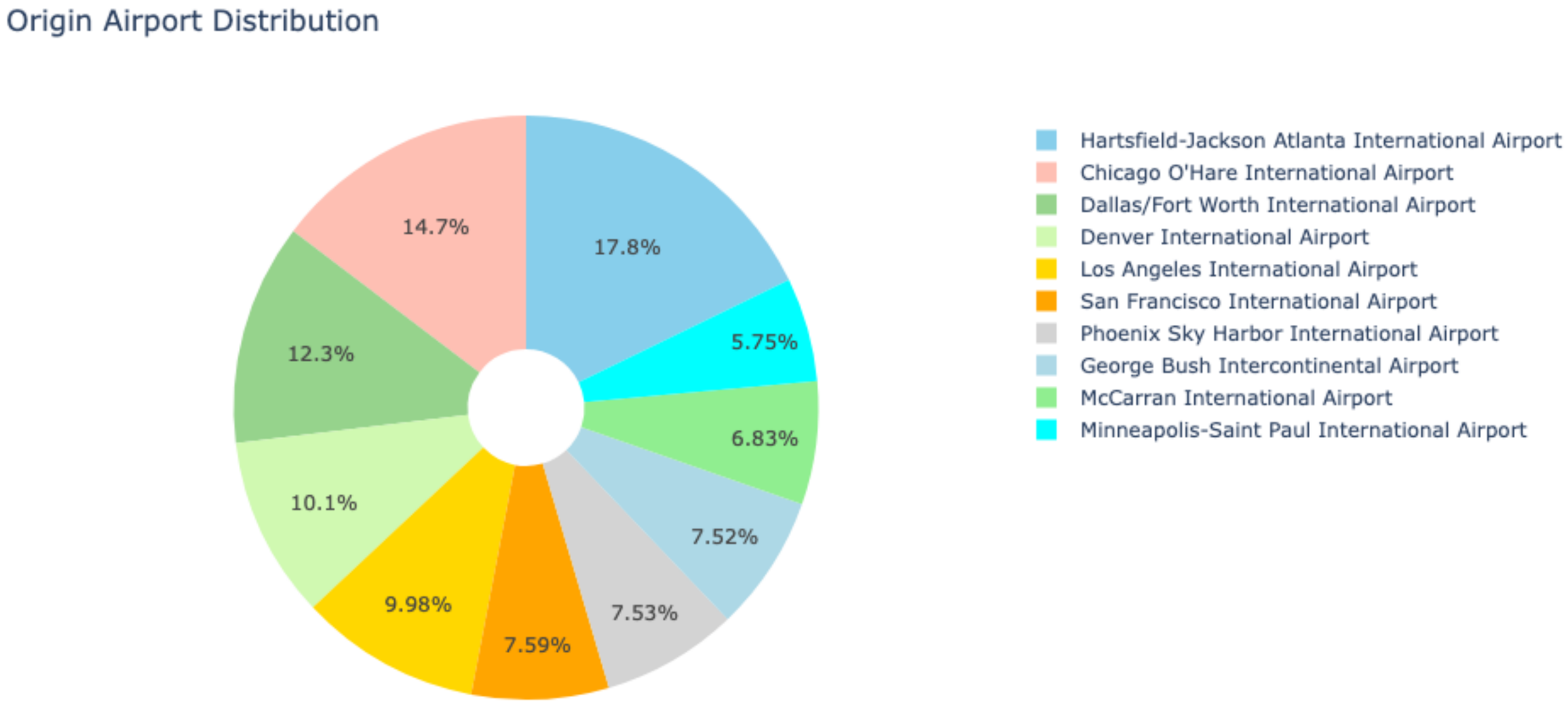
```
df
```

	YEAR	MONTH	DAY	DAY_OF_WEEK	AIRLINE_x	AIRLINE	FLIGHT_NUMBER	TAIL_NUMBER	ORIGIN_AIRPORT	DESTINATION_AIRPORT	...	CANCEL
0	2015	1	1	4	AS	Alaska Airlines Inc.	98	N407AS	ANC	SEA	...	
1	2015	1	1	4	AS	Alaska Airlines Inc.	108	N309AS	ANC	SEA	...	
2	2015	1	1	4	AS	Alaska Airlines Inc.	136	N431AS	ANC	SEA	...	

Departure Airports Counts

```
dff = df['AIRPORT_x'].value_counts()[:10]
label = dff.index
size = dff.values

colors = ['skyblue', '#FEBFB3', '#96D38C', '#D0F9B1', 'gold', 'orange', 'lightgrey',
          'lightblue','lightgreen','aqua']
trace = go.Pie(labels=label, values=size, marker=dict(colors=colors),hole = .2)

data = [trace]
layout = go.Layout( title='Origin Airport Distribution' )

fig = go.Figure(data=data, layout=layout)

fig.show()
```

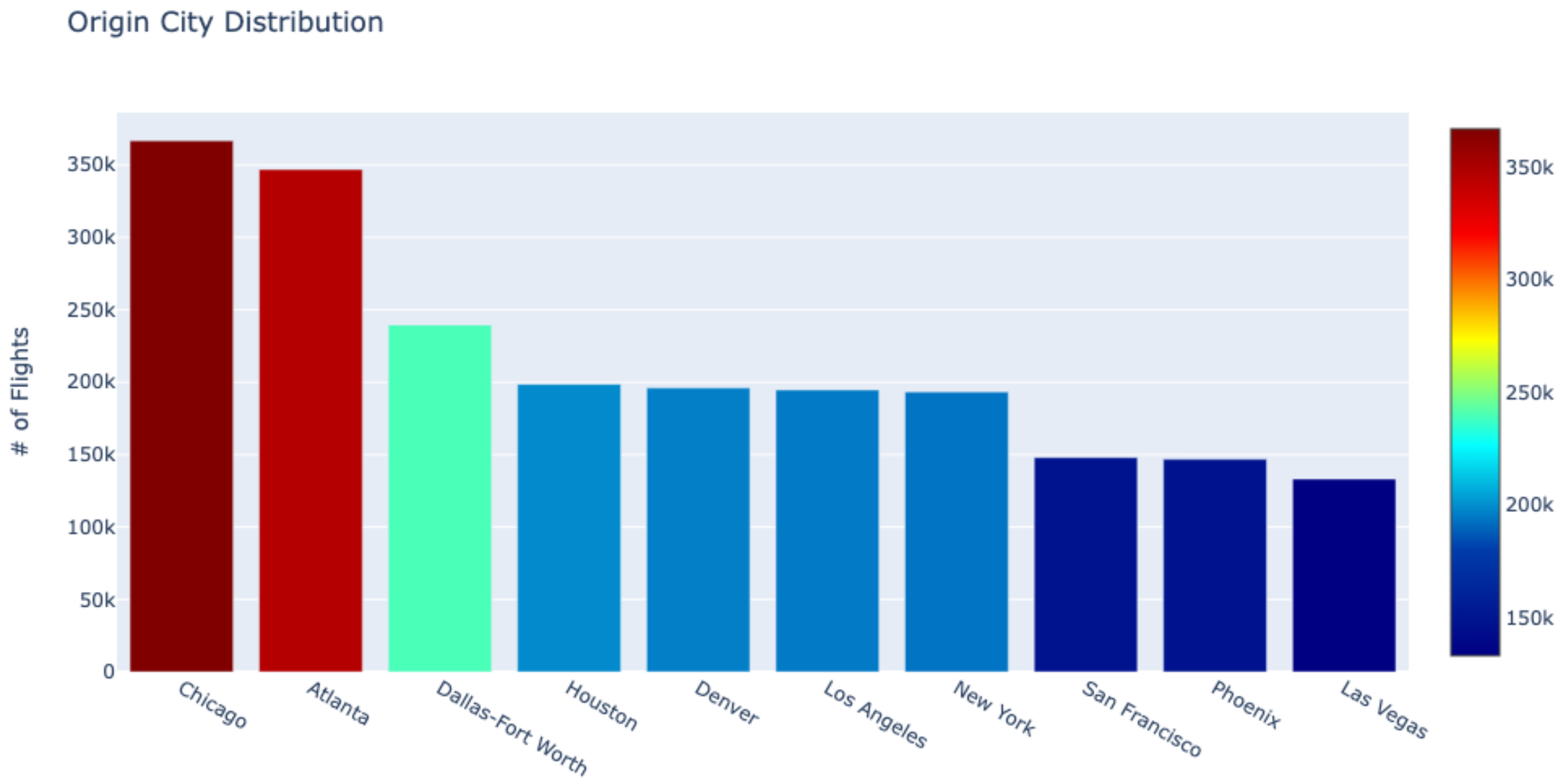
Departure Airports Counts



Departure City Counts

```
dff = df.CITY_x.value_counts()[:10]

trace = go.Bar(
    x=dff.index,
    y=dff.values,
    marker=dict(
        color = dff.values,
        colorscale='Jet',
        showscale=True
    )
)

data = [trace]
layout = go.Layout(
    title='Origin City Distribution',
    yaxis = dict(title = '# of Flights')
)

fig = go.Figure(data=data, layout=layout)
fig.show()
```

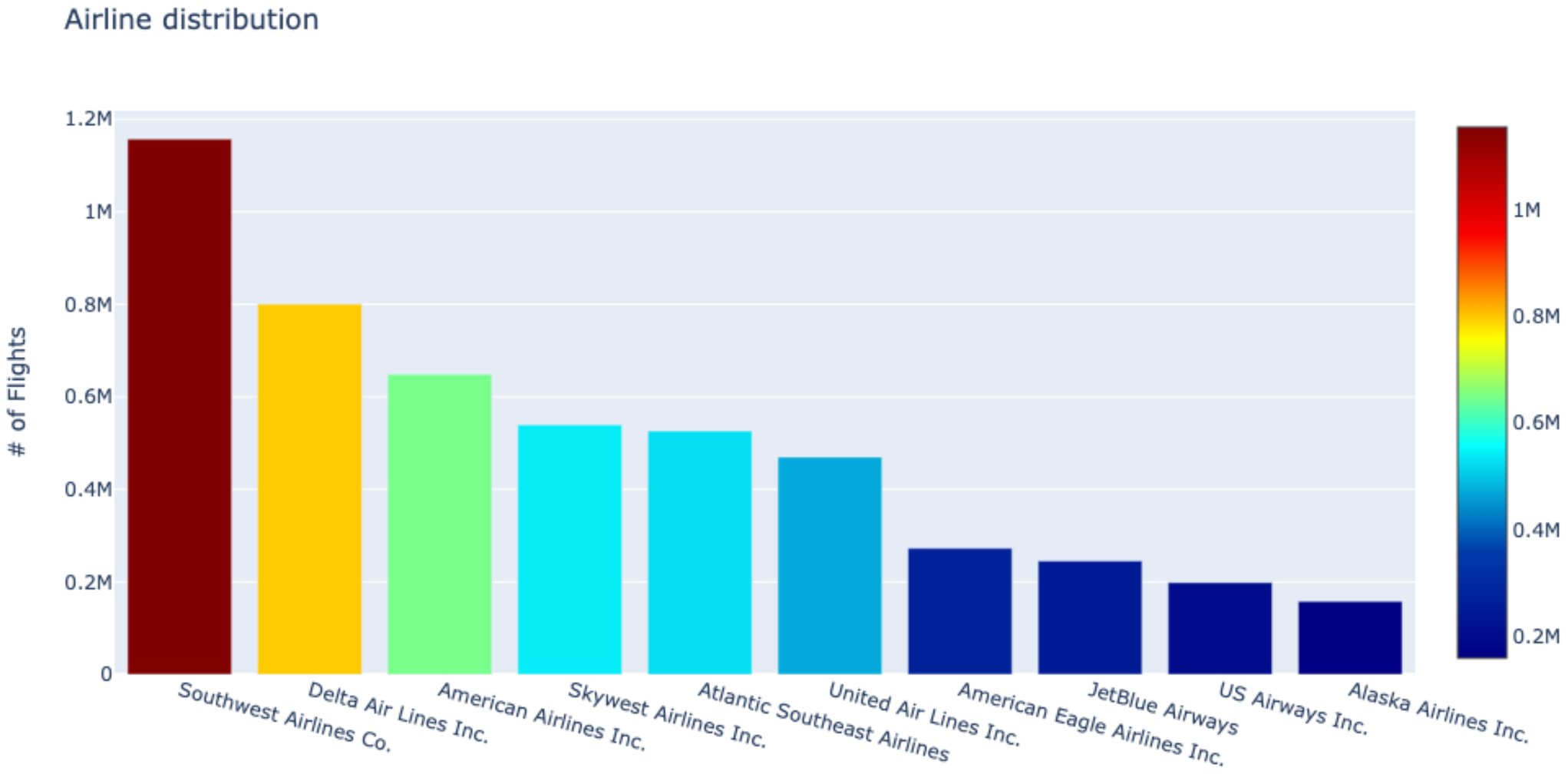
Departure city flights count



Airlines flights counts

```
dff = df.AIRLINE.value_counts()[:10]

trace = go.Bar(
    x=dff.index,
    y=dff.values,
    marker=dict(
        color = dff.values,
        colorscale='Jet',
        showscale=True)
)

data = [trace]
layout = go.Layout(xaxis=dict(tickangle=15),
    title='Airline distribution', yaxis = dict(title = '# of Flights'))

fig = go.Figure(data=data, layout=layout)
fig.show()
```

Top 10 Airlines flights counts

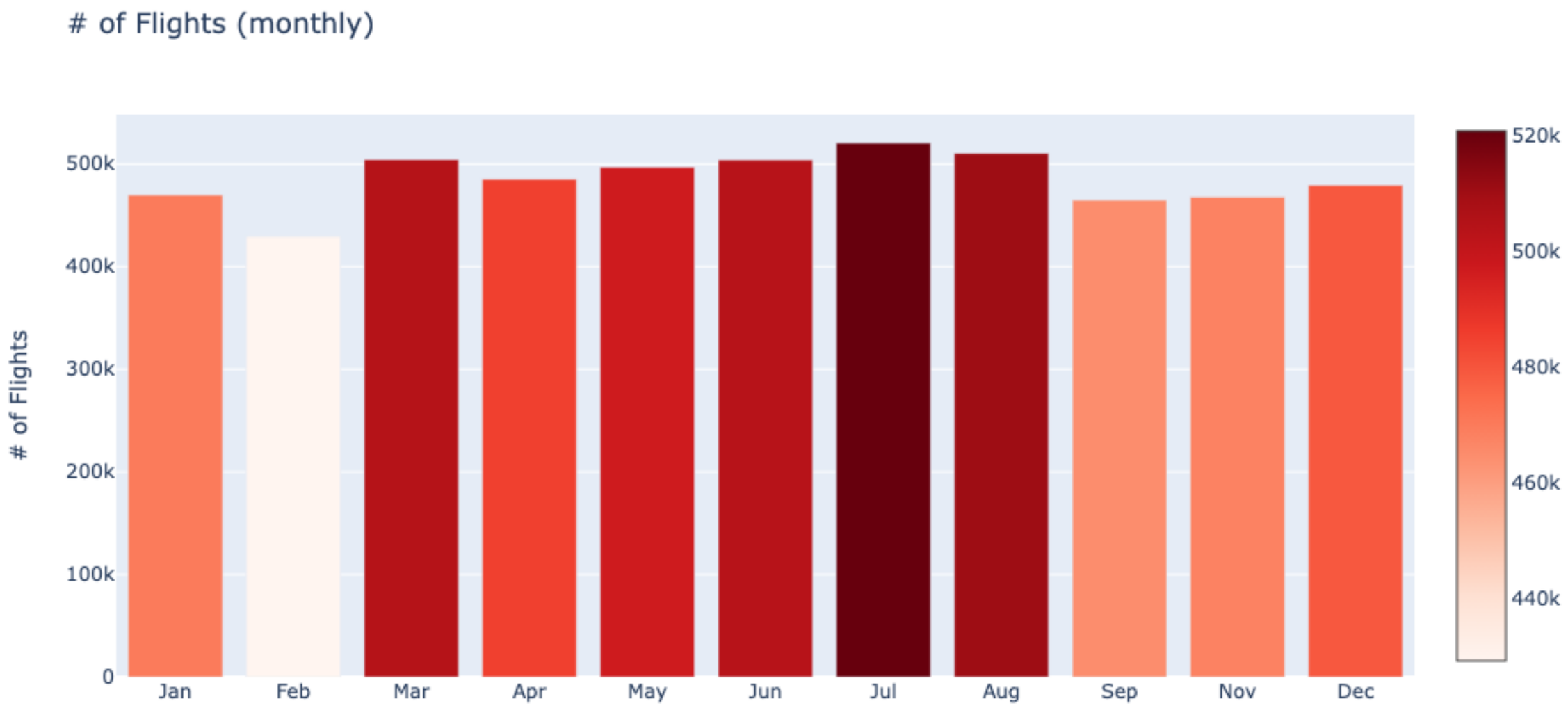


Transform month number

```
dff = df.MONTH.value_counts().to_frame().reset_index().sort_values(by='index')
dff.columns = ['month', 'flight_num']
month = {1: 'Jan', 2: 'Feb', 3: 'Mar', 4: 'Apr', 5: 'May',
            6: 'Jun', 7: 'Jul', 8: 'Aug', 9: 'Sep', 10: 'Oct', 11: 'Nov', 12: 'Dec'}
dff.month = dff.month.map(month)

trace = go.Bar(
    x=dff.month,
    y=dff.flight_num,
    marker=dict(
        color = dff.flight_num,
        colorscale='Reds',
        showscale=True)
)

data = [trace]
layout = go.Layout(
    title='# of Flights (monthly)',
    yaxis = dict(title = '# of Flights' ) )

fig = go.Figure(data=data, layout=layout)
fig.show()
```

	0	1
YEAR	2015	2015
MONTH	1	1
DAY	1	1
DAY_OF_WEEK	4	4

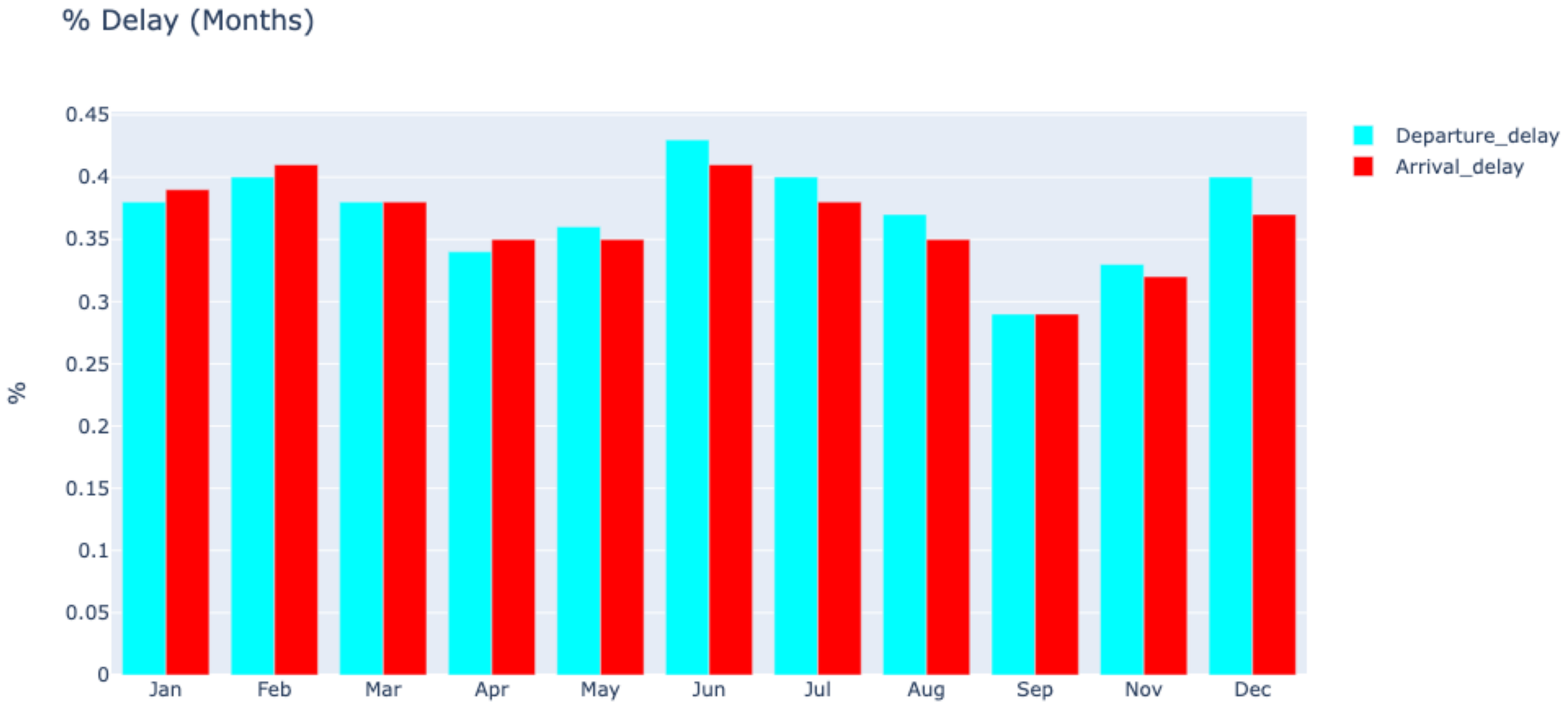
Total flights per month



Delay ratio per month

```
df['dep_delay'] = np.where(df.DEPARTURE_DELAY>0,1,0)
df['arr_delay'] = np.where(df.ARRIVAL_DELAY>0,1,0)
dff = df.groupby('MONTH').dep_delay.mean().round(2)

dff.index = dff.index.map(month)
trace1 = go.Bar(
    x=dff.index, y=dff.values,
    name = 'Departure_delay',
    marker = dict( color = 'aqua' ) )

dff = df.groupby('MONTH').arr_delay.mean().round(2)
dff.index = dff.index.map(month)

trace2 = go.Bar(
    x=dff.index, y=dff.values,
    name='Arrival_delay',
    marker=dict( color = 'red'  ) )

data = [trace1,trace2]
layout = go.Layout(
    title='% Delay (Months)',
    yaxis = dict(title = '%')
)

fig = go.Figure(data=data, layout=layout)
fig.show()
```

Delay ratio per month

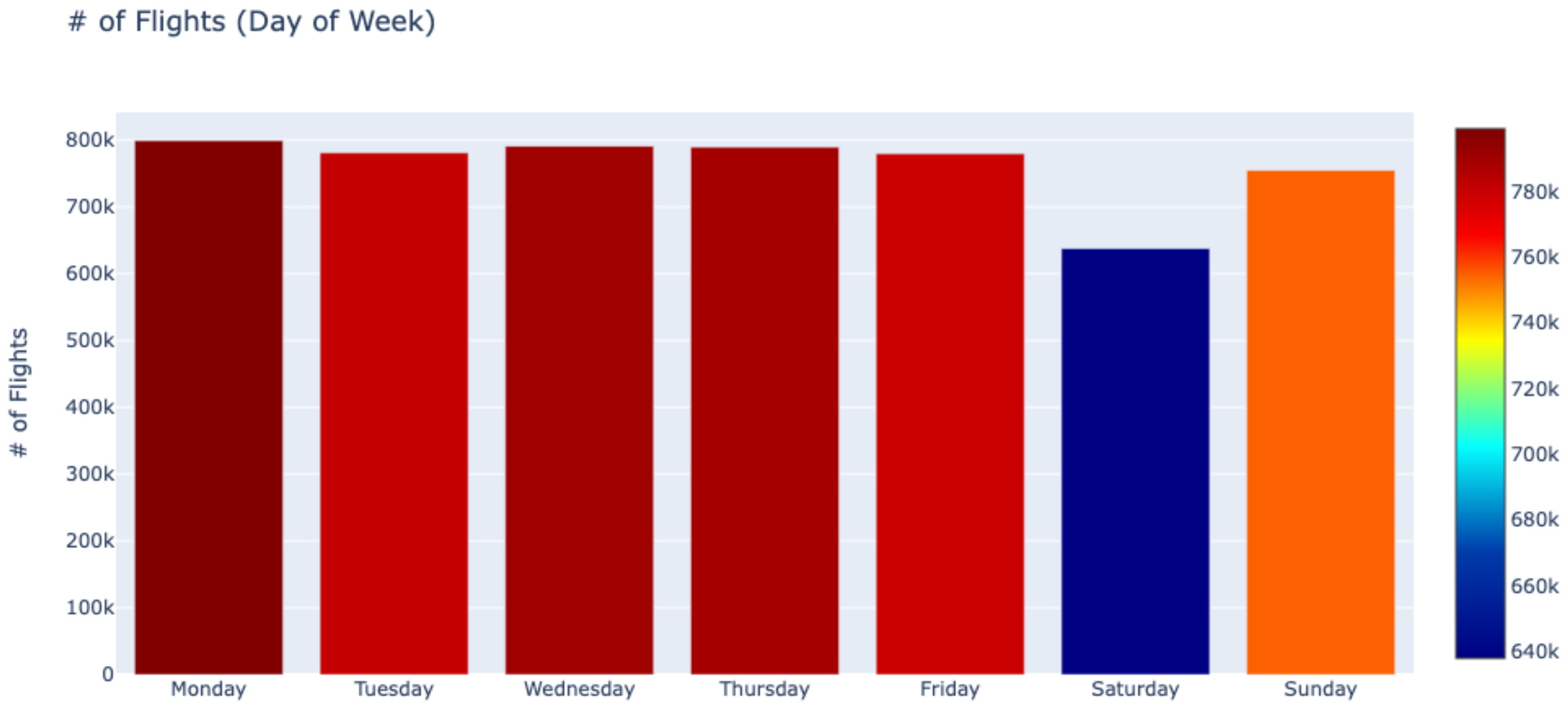


Number of flights per week

```
dayOfWeek={1:'Monday', 2:'Tuesday', 3:'Wednesday', 4:'Thursday', 5:'Friday',
                                                   6:'Saturday', 7:'Sunday'}

dff = df.DAY_OF_WEEK.value_counts()
dff = dff.to_frame().sort_index()
dff.index = dff.index.map(dayOfWeek)

trace1 = go.Bar(
    x=dff.index,
    y=dff.DAY_OF_WEEK,
    name = 'Weather',
    marker=dict(
        color = dff.DAY_OF_WEEK,
        colorscale='Jet',
        showscale=True ) )

data = [trace1]
layout = go.Layout(
    title='# of Flights (Day of Week)',
    yaxis = dict(title = '# of Flights'  ) )

fig = go.Figure(data=data, layout=layout)
fig.show()
```

Transform week day number

Number of flights per week



Air traffic by cities

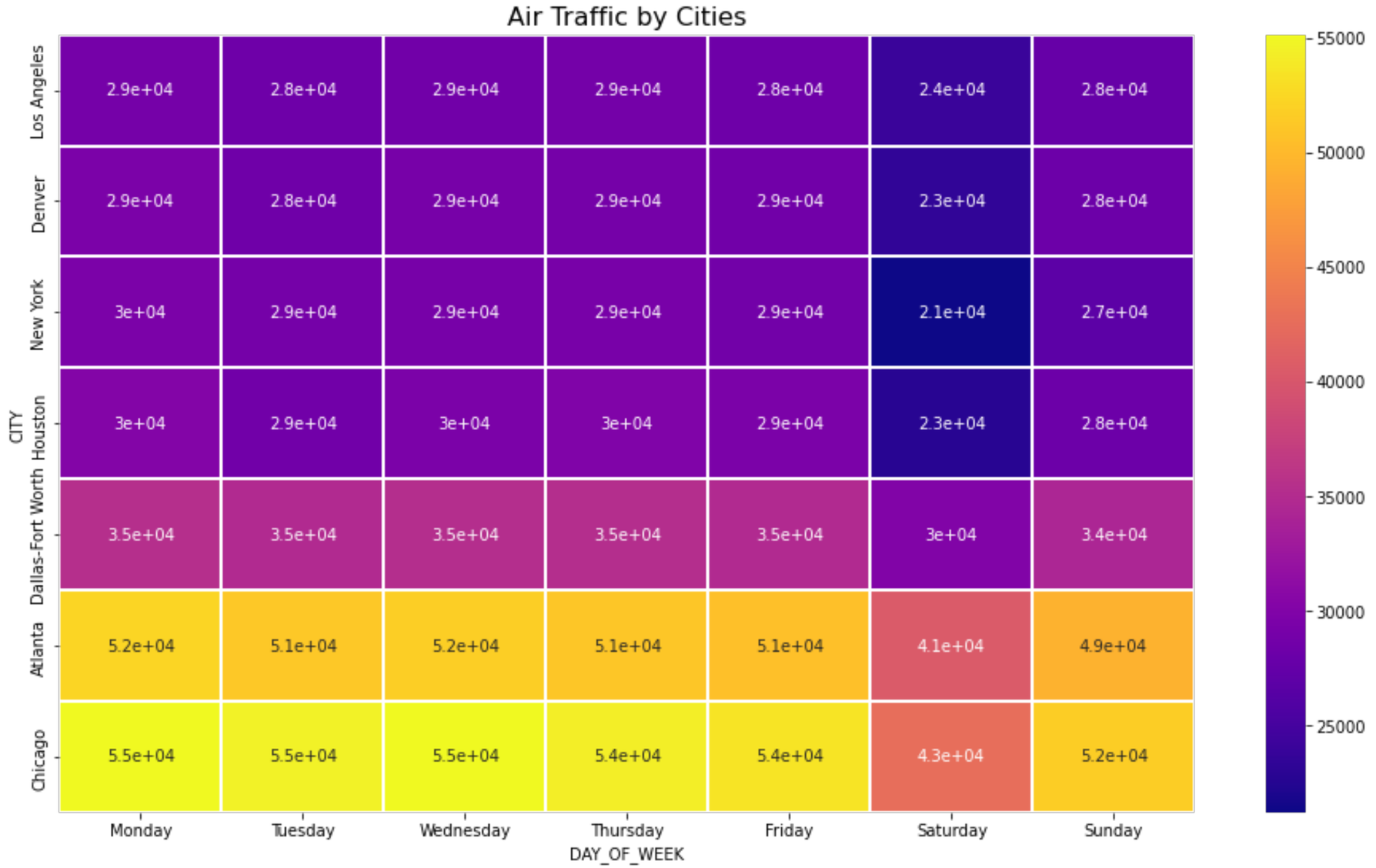
```
flight_volume = df.pivot_table(index="CITY_x",columns="DAY_OF_WEEK",
                               values="DAY",aggfunc=lambda x:x.count())
fv = flight_volume.sort_values(by=1, ascending=False)[:7]
fv = fv.iloc[::-1]

fig = plt.figure(figsize=(16,9))
sns.heatmap(fv, cmap='plasma',linecolor="w", linewidths=2, annot=True)

plt.title('Air Traffic by Cities',size=16)
plt.ylabel('CITY')
plt.xticks([0.5,1.5,2.5,3.5,4.5,5.5,6.5], ['Monday', 'Tuesday', 'Wednesday',
                                'Thursday', 'Friday', 'Saturday', 'Sunday'])
plt.show()
```

Air traffic by cities

```
1 fv
```

DAY_OF_WEEK	1	2	3	4	5	6	7
CITY_x							
Los Angeles	29027.0	28276.0	28523.0	28785.0	28317.0	24061.0	27684.0
Denver	29489.0	28392.0	29036.0	29026.0	28593.0	23369.0	28150.0
New York	29516.0	28915.0	29121.0	29173.0	28639.0	21212.0	26840.0
Houston	30193.0	28587.0	29560.0	29883.0	29489.0	22780.0	28172.0
Dallas-Fort Worth	35347.0	34762.0	35141.0	35215.0	34771.0	30042.0	34273.0
Atlanta	52261.0	51208.0	51703.0	51140.0	50583.0	40534.0	49407.0
Chicago	55136.0	54592.0	55005.0	54337.0	53539.0	42598.0	51563.0

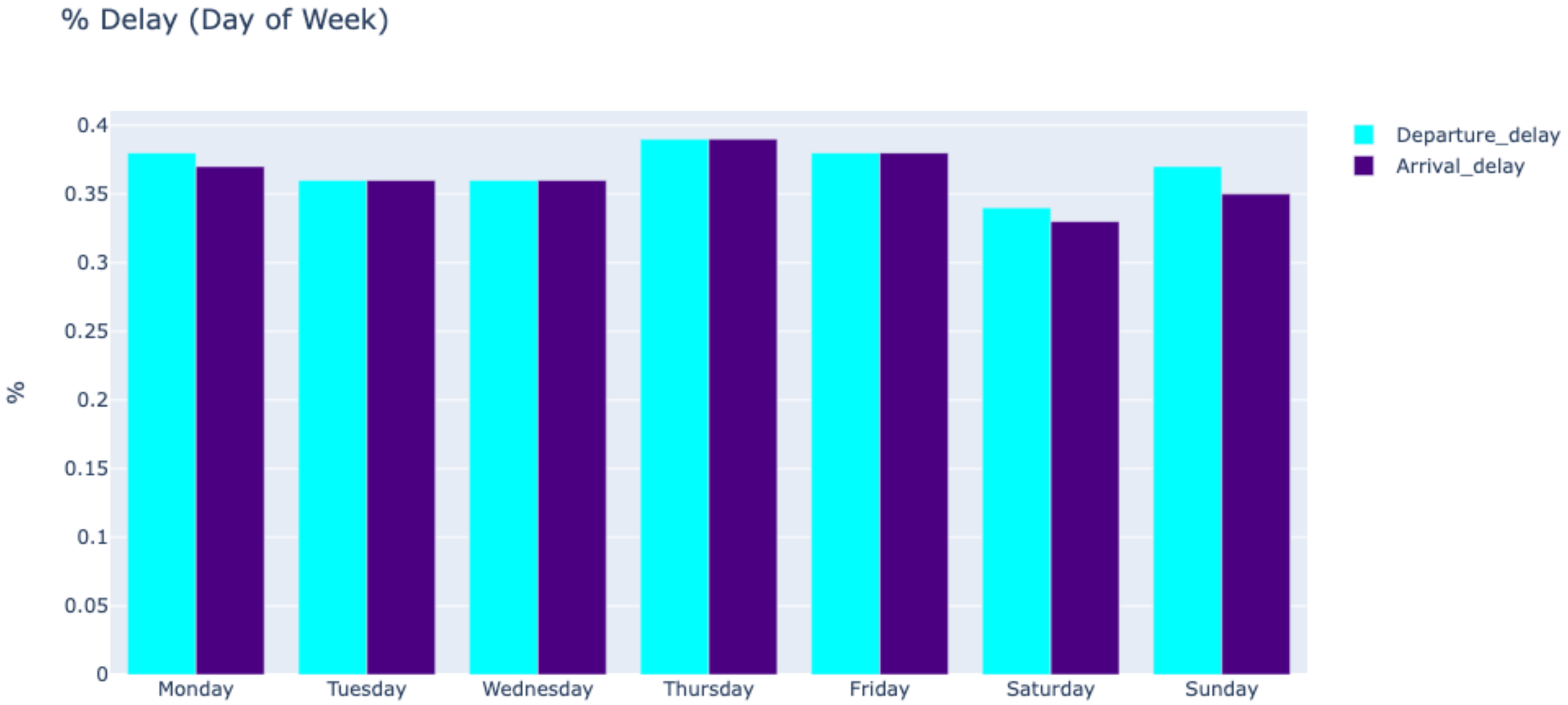
Air traffic by cities



Delay on week day

```
dff = df.groupby('DAY_OF_WEEK').dep_delay.mean().round(2)
dff.index = dff.index.map(dayOfWeek)

trace1 = go.Bar(
    x=dff.index,
    y=dff.values,
    name = 'Departure_delay',
    marker=dict( color = 'cyan' ))

dff = df.groupby('DAY_OF_WEEK').arr_delay.mean().round(2)
dff.index = dff.index.map(dayOfWeek)

trace2 = go.Bar(
    x=dff.index,
    y=dff.values,
    name='Arrival_delay',
    marker=dict( color = 'indigo' ))

data = [trace1,trace2]
layout = go.Layout(
    title='% Delay (Day of Week)',
    yaxis = dict(title = '%')
)

fig = go.Figure(data=data, layout=layout)
fig.show()
```

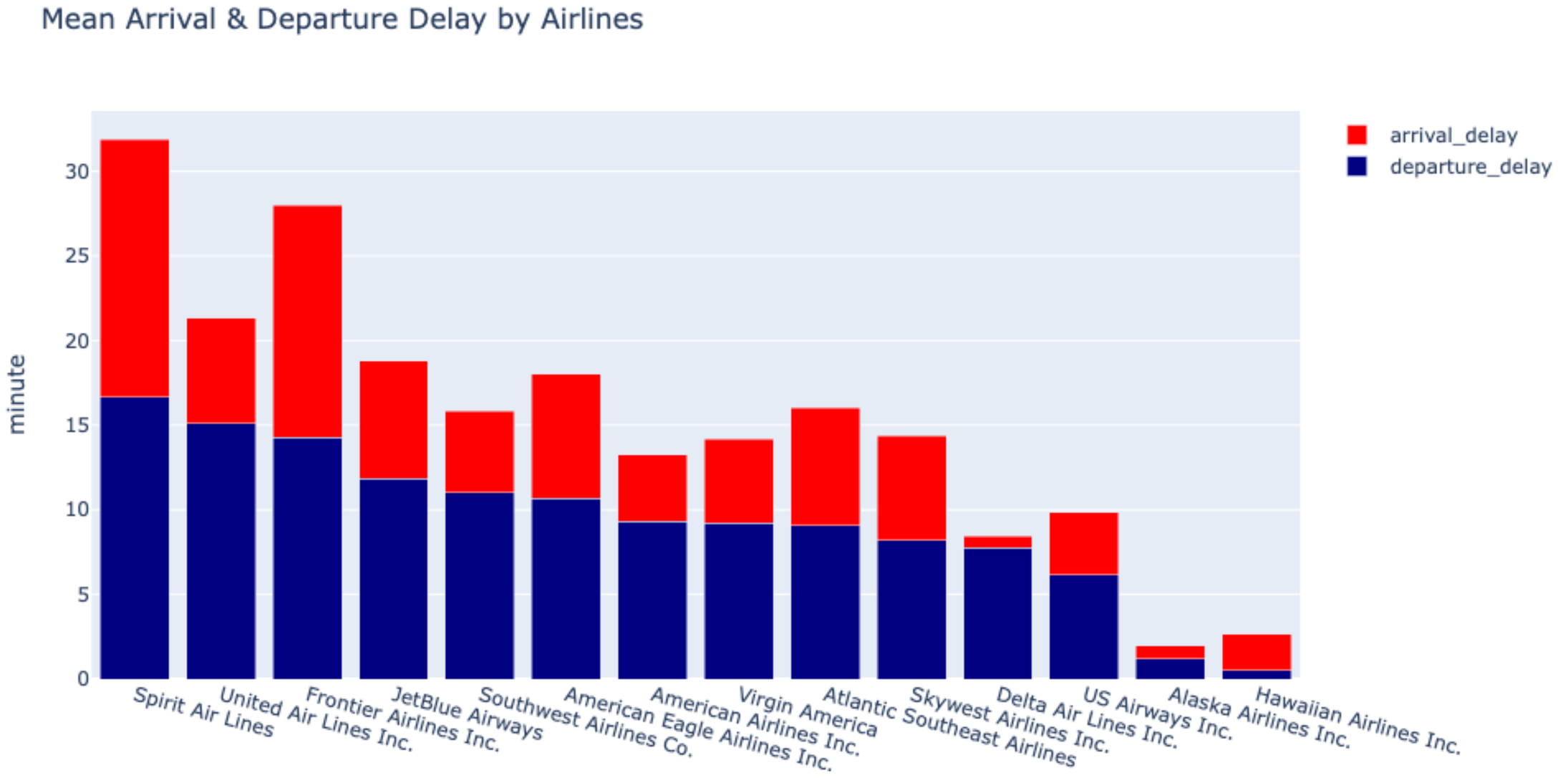
Delay on week day



Airlines' mean arrival and departure delay

```
dff = df.groupby('AIRLINE').DEPARTURE_DELAY.mean().to_frame().sort_values(by='DEPARTURE_DELAY',
                                                    ascending=False).round(2)
trace1 = go.Bar(
    x=dff.index, y=dff.DEPARTURE_DELAY,
    name='departure_delay',
    marker=dict( color = 'navy' ) )

dff = df.groupby('AIRLINE').ARRIVAL_DELAY.mean().to_frame().sort_values(by='ARRIVAL_DELAY',
                                                    ascending=False).round(2)
trace2 = go.Bar(
    x=dff.index,  y=dff.ARRIVAL_DELAY,
    name='arrival_delay',
    marker=dict( color = 'red' ) )

data = [trace1, trace2]
layout = go.Layout(xaxis=dict(tickangle=15), title='Mean Arrival & Departure Delay by Airlines',
    yaxis = dict(title = 'minute'),      barmode='stack')

fig = go.Figure(data=data, layout=layout)
fig.show()
```

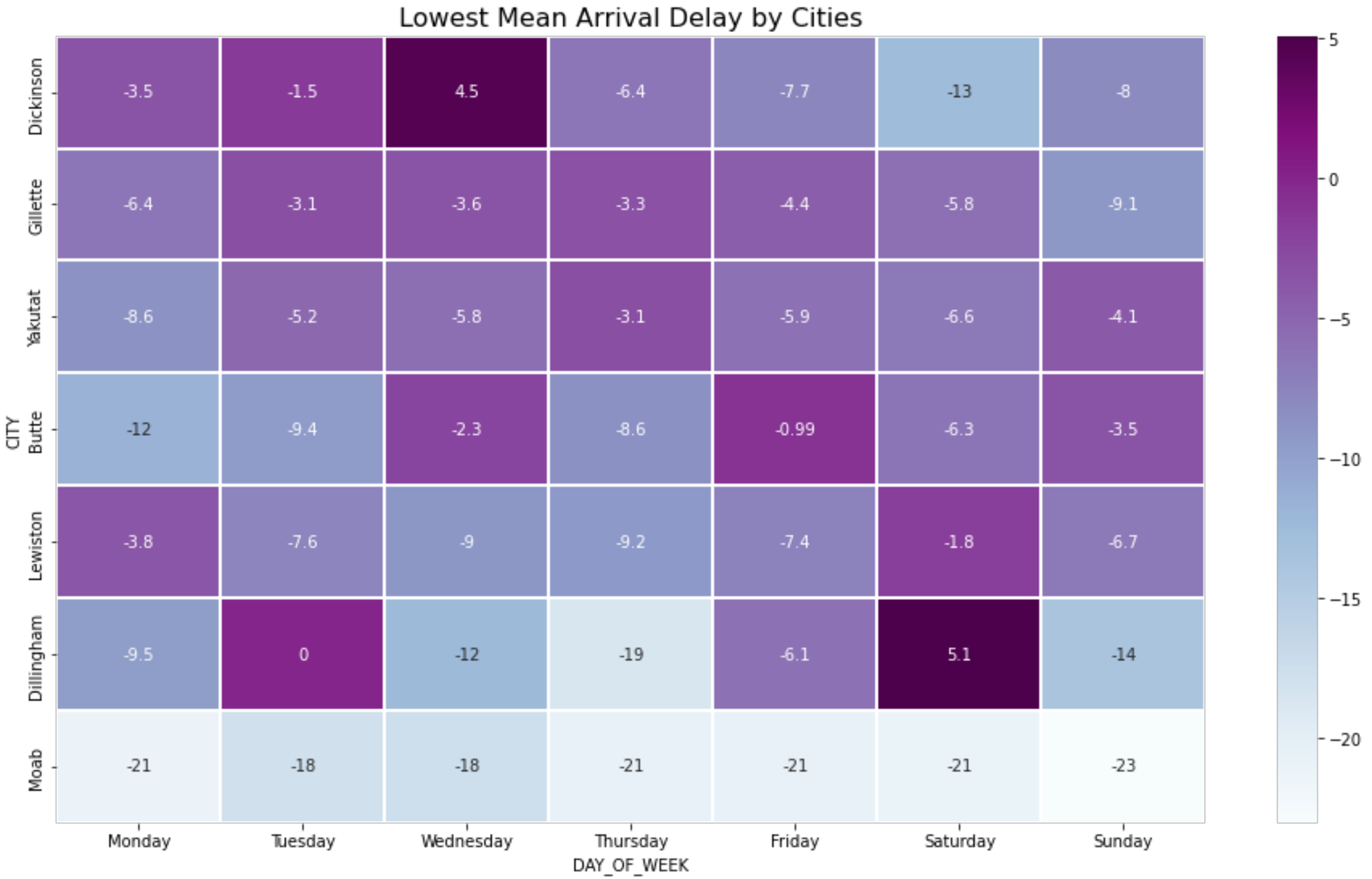
Airlines' mean arrival and departure delay



Lowest Mean Arrival Delay by Cities

```
arr = df.pivot_table(index="CITY_x",columns="DAY_OF_WEEK",values="ARRIVAL_DELAY",
                     aggfunc=lambda x:x.mean())
arr['sum'] = arr[1] + arr[2] + arr[3] + arr[4] + arr[5] + arr[6] + arr[7]

fv = arr.sort_values(by='sum')[:7]
fv = fv.iloc[::-1]
fv = fv.drop(['sum'], axis=1)
fig = plt.figure(figsize=(16,9))
sns.heatmap(fv, cmap='BuPu',linecolor="w", linewidths=2, annot=True)

plt.title('Lowest Mean Arrival Delay by Cities', size=16)
plt.ylabel('CITY')
plt.xticks([0.5,1.5,2.5,3.5,4.5,5.5,6.5], ['Monday', 'Tuesday', 'Wednesday',
                                'Thursday', 'Friday', 'Saturday', 'Sunday'])
plt.show()
```

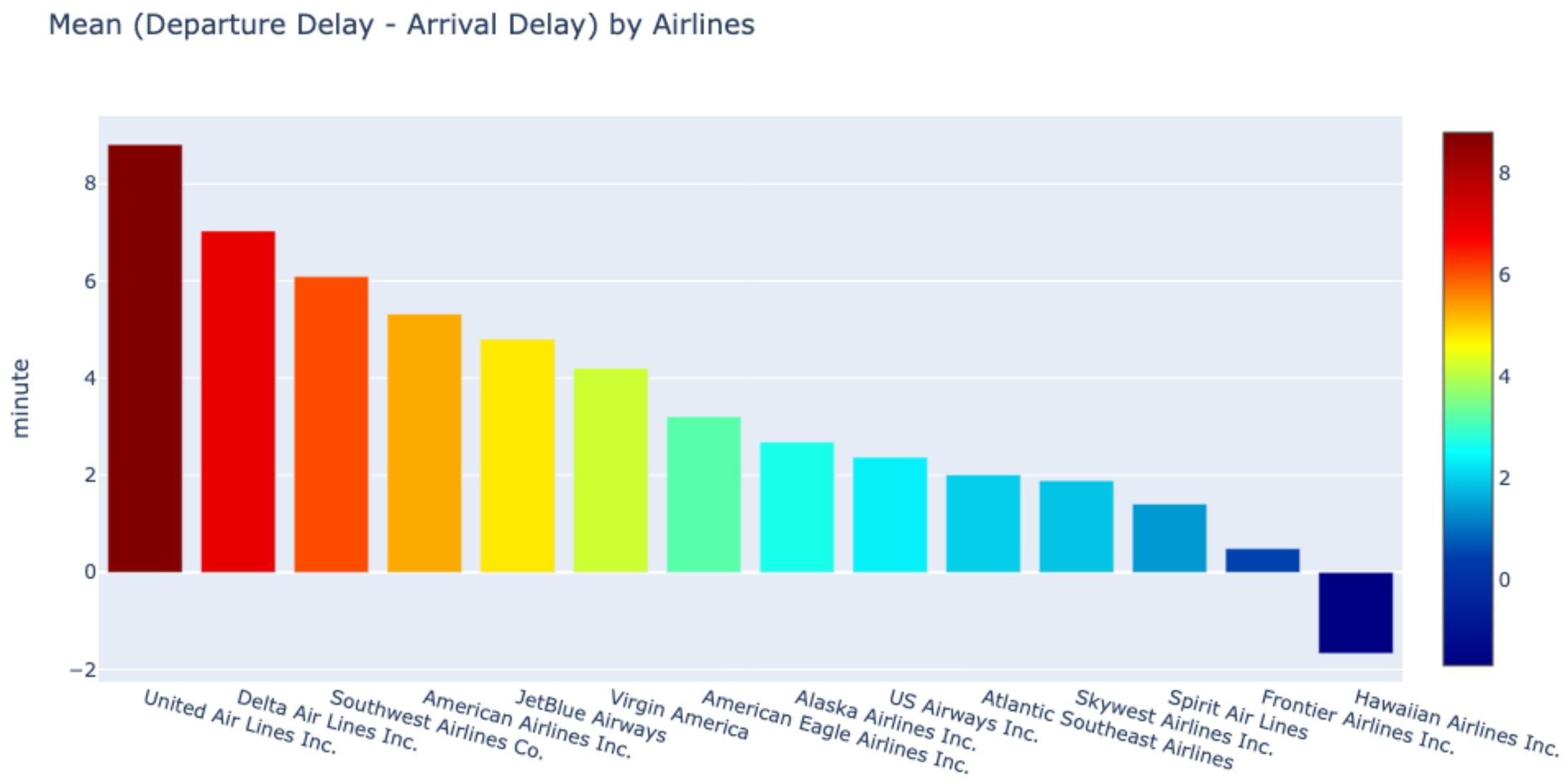
Lowest Mean Arrival Delay by Cities



Mean (Departure Delay - Arrival Delay) offset

```
df['DEP_ARR_DIFF'] = df['DEPARTURE_DELAY'] - df['ARRIVAL_DELAY']
dff = df.groupby('AIRLINE').DEP_ARR_DIFF.mean().to_frame().sort_values(by='DEP_ARR_DIFF',
                                                    ascending=False).round(2)

trace = go.Bar( x=dff.index,  y=dff.DEP_ARR_DIFF,
    marker=dict( color = dff.DEP_ARR_DIFF, colorscale='Jet', showscale=True  ) )

data = [trace]
layout = go.Layout(xaxis=dict(tickangle=15),
    title='Mean (Departure Delay - Arrival Delay) by Airlines',
                   yaxis = dict(title = 'minute') )

fig = go.Figure(data=data, layout=layout)
fig.show()
```

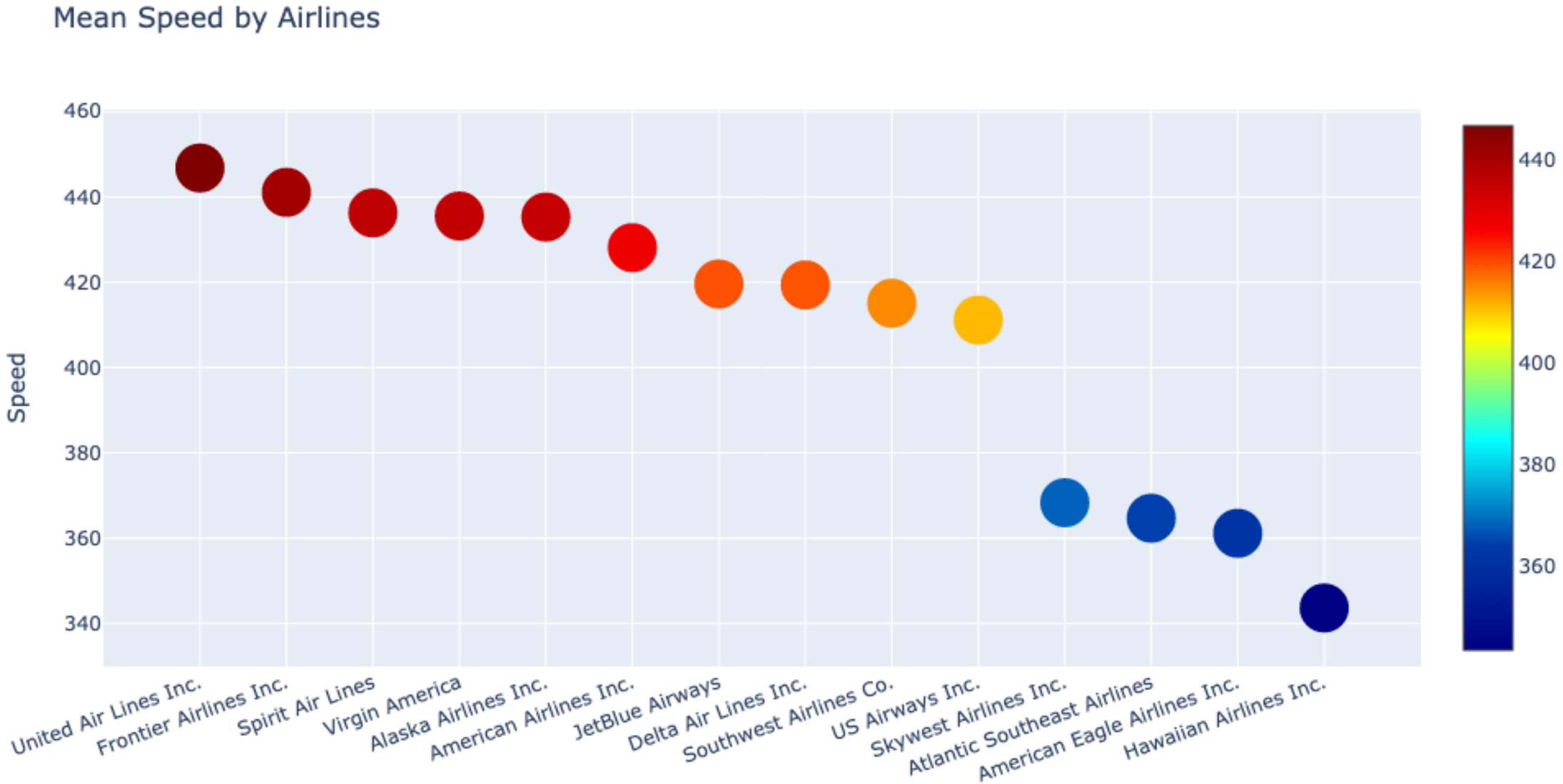
Mean (Departure Delay - Arrival Delay) offset



Mean Speed by Airline

```
df['SPEED'] = 60*df['DISTANCE']/df['AIR_TIME']
dff = df.groupby('AIRLINE').SPEED.mean().to_frame().sort_values(by='SPEED',
                                                    ascending=False).round(2)

trace = go.Scatter( x=dff.index, y=dff.SPEED, mode='markers',
    marker=dict( sizemode = 'diameter',  sizeref = 1, size = 30,
        color = dff.SPEED.values, colorscale='Jet', showscale=True ) )

data = [trace]
layout = go.Layout(xaxis=dict(tickangle=-20),
    title='Mean Speed by Airlines', yaxis = dict(title = 'Speed') )

fig = go.Figure(data=data, layout=layout)
fig.show()
```

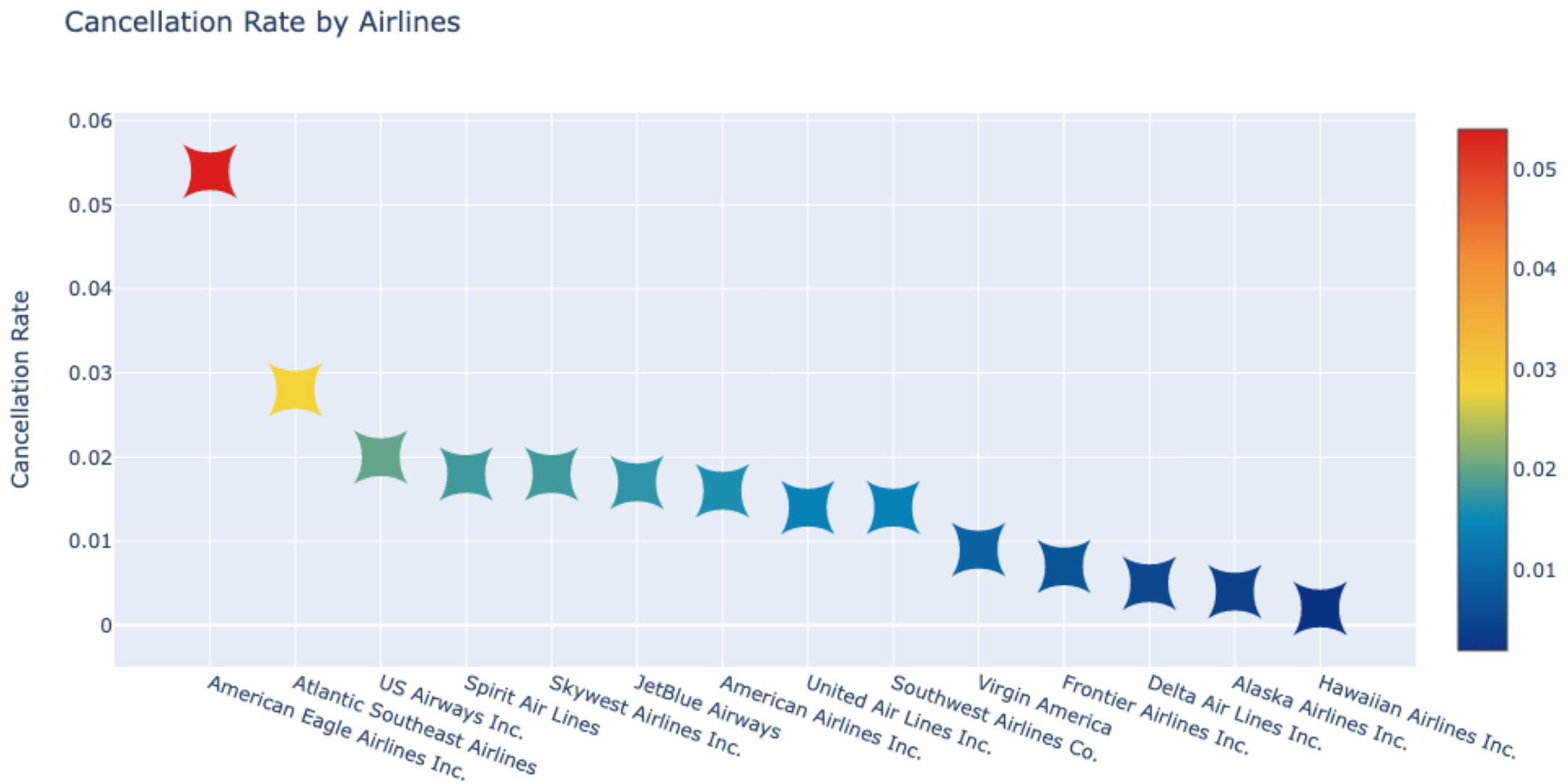
Mean Speed by Airline



Cancellation Rate by Airlines

```
dff = df.groupby('AIRLINE')[['CANCELLED']].mean().sort_values(by='CANCELLED',
                                                    ascending=False).round(3)

trace1 = go.Scatter( x=dff.index, y=dff.CANCELLED,
    mode='markers',  marker=dict( symbol = 'star-square', sizemode = 'diameter',
        sizeref = 1,  size = 30,  color = dff.CANCELLED, colorscale='Portland',
        showscale=True     ) )

data = [trace1]
layout = go.Layout(xaxis=dict(tickangle=20),
    title='Cancellation Rate by Airlines', yaxis = dict(title = 'Cancellation Rate' ) )

fig = go.Figure(data=data, layout=layout)
fig.show()
```

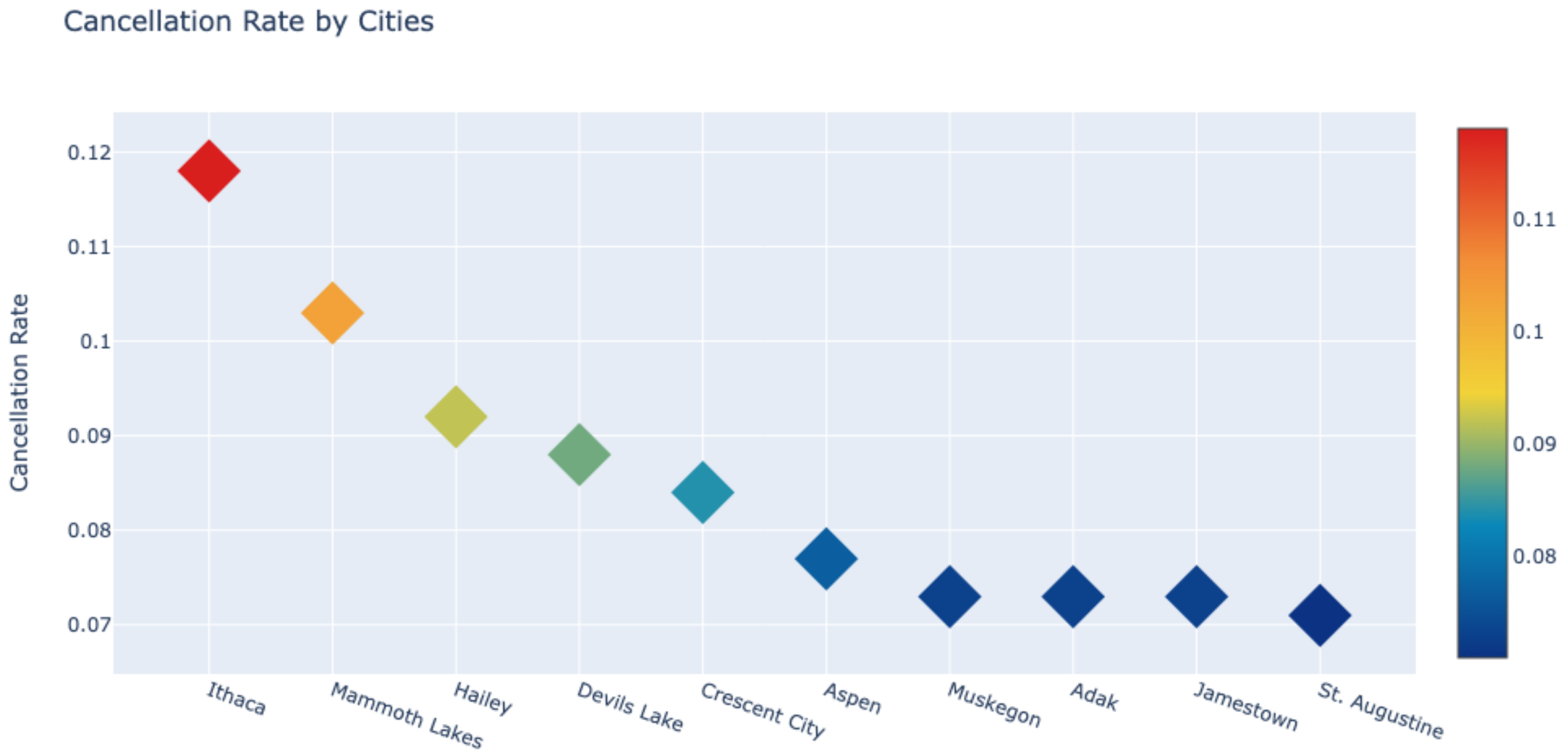
Cancellation Rate by Airlines



Cancellation Rate by Cities

```
dff = df.groupby('CITY_x')[['CANCELLED']].mean().sort_values(by='CANCELLED',
                                                    ascending=False)[:10].round(3)
trace2 = go.Scatter( x=dff.index,  y=dff.CANCELLED,
    mode='markers', marker=dict(symbol = 'diamond', sizemode = 'diameter',
        sizeref = 1, size = 30, color = dff.CANCELLED,
        colorscale='Portland', showscale=True ) )

data = [trace2]
layout = go.Layout(xaxis=dict(tickangle=20),
    title='Cancellation Rate by Cities',
                    yaxis = dict(title = 'Cancellation Rate' ) )

fig = go.Figure(data=data, layout=layout)
fig.show()
```

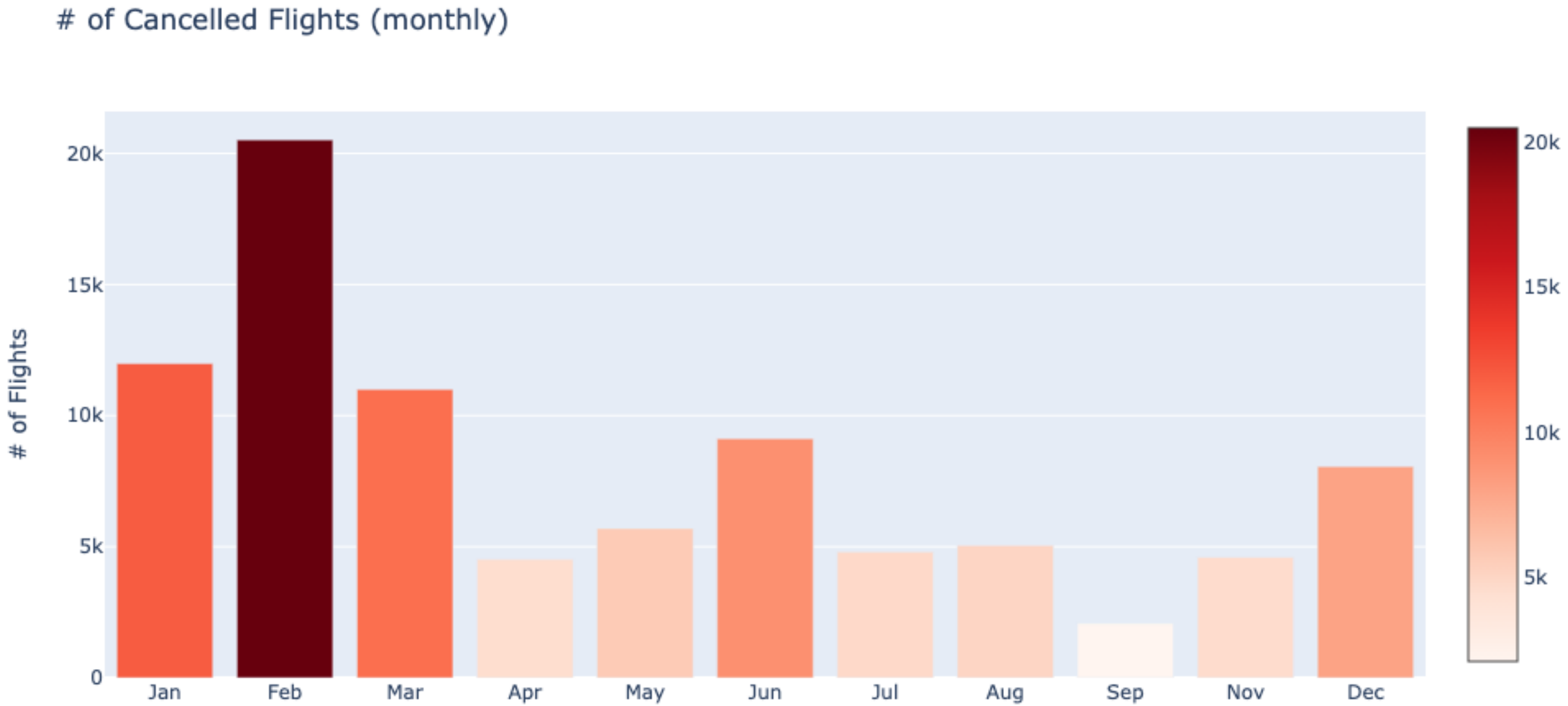
Cancellation Rate by Cities



Number of Cancelled Flights per Month

```
reason={'A':'Airline/Carrier', 'B':'Weather', 'C':'National Air System', 'D':'Security'}
df.CANCELLATION_REASON = df.CANCELLATION_REASON.map(reason)

dff = df[df.CANCELLED==1]['MONTH'].value_counts().reset_index().sort_values(by='index')
dff.columns = ['month', 'flight_num']
dff.month = dff.month.map(month)

trace = go.Bar(  x=dff.month,   y=dff.flight_num,
    marker=dict( color = dff.flight_num,  colorscale='Reds', showscale=True ) )

data = [trace]
layout = go.Layout(
    title='# of Cancelled Flights (monthly)',
    yaxis = dict(title = '# of Flights'  ) )

fig = go.Figure(data=data, layout=layout)
fig.show()
```

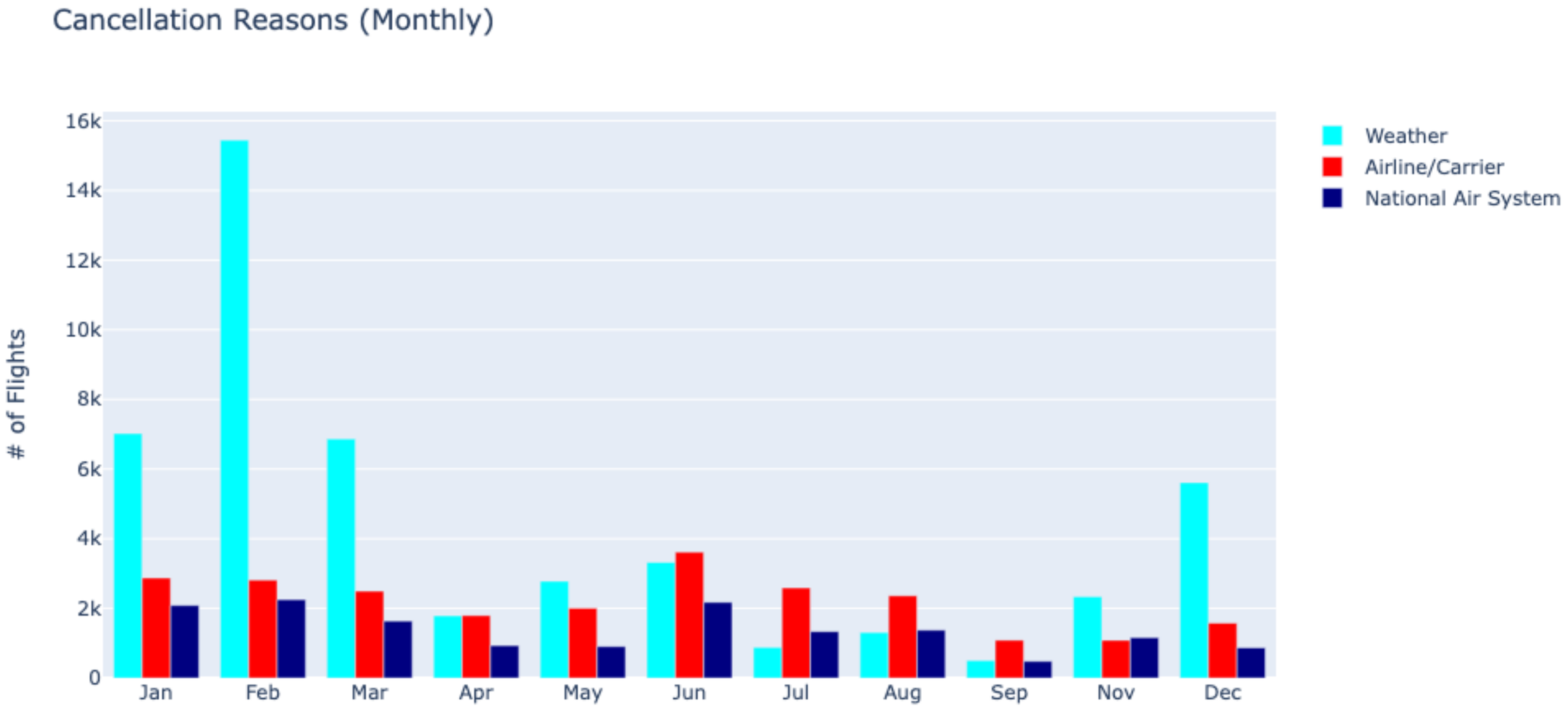
Number of Cancelled Flights per Month



Cancellation Reasons per month

```
dff = df[df.CANCELLATION_REASON=='Weather'].MONTH.value_counts()
dff = dff.to_frame().sort_index()
dff.index = dff.index.map(month)

trace1 = go.Bar( x=dff.index,  y=dff.MONTH,  name = 'Weather',
    marker=dict(  color = 'aqua'   ))

dff = df[df.CANCELLATION_REASON=='Airline/Carrier'].MONTH.value_counts()
dff = dff.to_frame().sort_index()
dff.index = dff.index.map(month)

trace2 = go.Bar(  x=dff.index,  y=dff.MONTH,   name='Airline/Carrier',
    marker=dict(  color = 'red'  ))

dff = df[df.CANCELLATION_REASON=='National Air System'].MONTH.value_counts()
dff = dff.to_frame().sort_index()
dff.index = dff.index.map(month)

trace3 = go.Bar( x=dff.index,  y=dff.MONTH,   name='National Air System',
    marker=dict(  color = 'navy'  ))

data = [trace1,trace2,trace3]
layout = go.Layout( title='Cancellation Reasons (Monthly)',
    yaxis = dict(title = '# of Flights'  ))

fig = go.Figure(data=data, layout=layout)
fig.show()
```

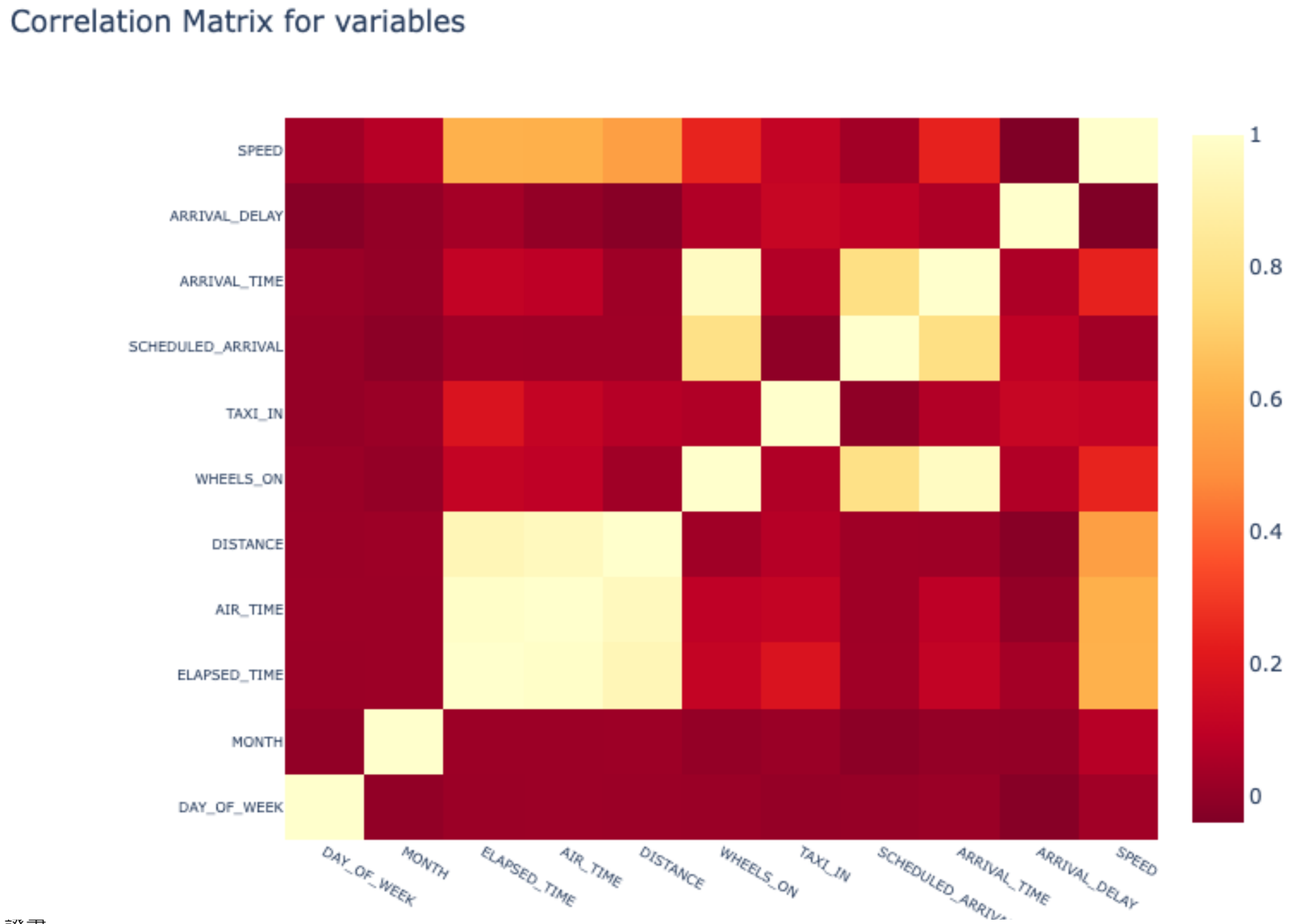
Cancellation Reasons per month



Correlation of variables

```
correlation = df[['DAY_OF_WEEK','MONTH','ELAPSED_TIME', 'AIR_TIME', 'DISTANCE',
                  'WHEELS_ON', 'TAXI_IN', 'SCHEDULED_ARRIVAL', 'ARRIVAL_TIME',
                  'ARRIVAL_DELAY','SPEED']].fillna(0).corr()
cols = correlation.columns.values
corr  = correlation.values
trace = go.Heatmap(z = corr,  x = cols, y = cols,
                   colorscale = "YlOrRd", reversescale = True  )

data = [trace]
layout = go.Layout(dict(title = "Correlation Matrix for variables",
                        autosize = False,  height  = 600, width   = 800,
                        margin  = dict(l = 200 ),
                        yaxis   = dict(tickfont = dict(size = 8)),
                        xaxis   = dict(tickfont = dict(size = 8))
                       )
                  )
fig = go.Figure(data=data,layout=layout)
fig.show()
```

Correlation of variables



Chapter Wrap Up

Plotting a visual is always not the difficult part.

The challenge are:

- what kind of plot to visualise
- prepare extra data for visual. i.e. count, sum, mean, etc.
- choose better parameter
- choose suitable visualization package
- how to interpret those plots
- any suggestion to the scenario

Reference & Resources

Official Website:

https://plotly.com/python/

Plotly Graph Objects:

https://plotly.com/python/graph-objects/

Kaggle dataset:

https://www.kaggle.com/datasets

WorldBank API:

- https://blogs.worldbank.org/opendata/introducing-wbgapi-new-python-package-accessing-world-bank-data
- https://nbviewer.org/github/tgherzog/wbgapi/blob/master/examples/wbgapi-cookbook.ipynb
- https://pypi.org/project/wbgapi/

GitHub Open Source Code:

https://github.com/plotly/plotly.py