Brightkite

Importing Relevant Modules

In [1]:
```
import folium
import pandas as pd
import plotly.express as px

from datetime import datetime
from collections import Counter
from folium.plugins import HeatMap
```

Reading Dataset

In [2]:
```
df = pd.read_csv('loc-brightkite_totalCheckins.txt',sep='\t', header=0, names=["user", "check_in_time",
 "latitude", "longitude","location_id"])
```

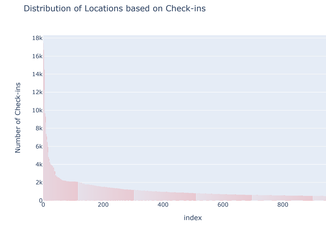
In [3]:
```
df.dropna(inplace=True)
df = df[~df.location_id.str.contains('00000000000000000000000000000000')]
df = df[df.latitude<=90]
df = df[df.latitude>=-90]
df.reset_index(inplace=True,drop=True)
```

Distribution of Locations based on Check-ins

In [4]:
```
# Resolving location check-ins counts
C = Counter(df.location_id)
location_df = pd.DataFrame()
location_df['location_id'] = [i for i in C]
location_df['count'] = [C[i] for i in C]

location_df.sort_values('count',ascending=False,inplace=True,ignore_index=True)
location_df = location_df[:1000]
```

In [5]:
```
#Plotting
fig = px.bar(location_df,y='count',title='Distribution of Locations based on Check-ins',color_discrete_
sequence=['red'])
fig.update_yaxes(title='Number of Check-ins')
fig.show()
```

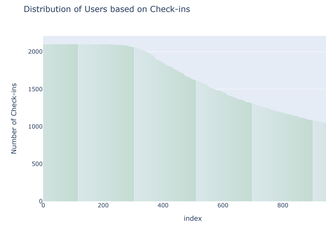


Distribution of Users based on Check-ins

In [6]:
```
# Resolving user check-ins counts
C = Counter(df.user)
user_df = pd.DataFrame()
user_df['user'] = [i for i in C]
user_df['count'] = [C[i] for i in C]

user_df.sort_values('count',ascending=False,inplace=True,ignore_index=True)
user_df = user_df[1000]
```

In [7]:
```
# Plotting
fig = px.bar(user_df,y='count',title='Distribution of Users based on Check-ins',color_discrete_sequence
=['green'])
fig.update_yaxes(title='Number of Check-ins')
fig.show()
```

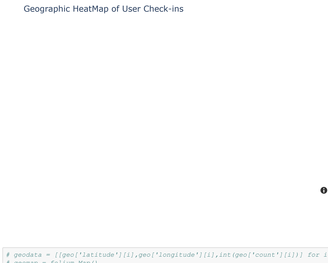


Geographic Heatmap of Check-ins

In [8]:
```
x = df.copy()
coordinates = ([x['latitude'][i],x['longitude'][i]) for i in x.index]
C = Counter(coordinates)

geo = pd.DataFrame()
geo['count'] = [C[i] for i in C]
geo['latitude'] = [i[0] for i in C]
geo['longitude'] = [i[1] for i in C]
```

In [9]:
```
fig = px.density_mapbox(data_frame=geo,lat='latitude',lon='longitude',z='count',zoom=1,radius=15,
                        title='Geographic HeatMap of User Check-ins',mapbox_style='open-street-map')
fig.show()
```



In [10]:
```
# geodata = [[geo['latitude'][i],geo['longitude'][i],int(geo['count'][i])] for i in geo.index]
# geomap = folium.Map()
# HeatMap(geodata,radius=5,blur=5).add_to(geomap,)
# geomap.save("GeographicHeatmap.html")
```

Time Series Plots

In [11]:
```
# Setting Buttons
Buttons = []
Buttons.append(dict(count=1, label="YTD", step="year", stepmode="todate"))
Buttons.append(dict(count=1, label="Day", step="day", stepmode="backward"))
Buttons.append(dict(count=7, label="Week", step="day", stepmode="backward"))
Buttons.append(dict(count=1, label="Month", step="month", stepmode="backward"))
Buttons.append(dict(label="Complete", step="all"))
```

User with Most Check-Ins

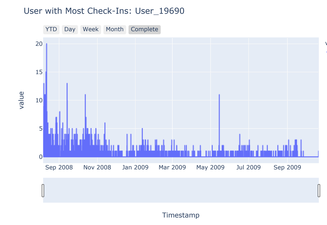
In [12]:
```
# Collecting Check-ins of User with most Check-ins
u = df['user'].value_counts().idxmax()
timestamps = df[df.user == u]['check_in_time']

C = Counter([datetime.strptime(str(i), '%Y-%m-%dT%H:%M:%SZ') for i in timestamps])

top_user_df = pd.DataFrame()
top_user_df['Timestamp'] = [i for i in C]
top_user_df['Check_ins'] = [C[i] for i in C]

top_user_df = top_user_df.sort_values('Timestamp')
top_user_df = top_user_df.resample('H', on='Timestamp').Check_ins.sum()
```

In [13]:
```
# Plotting
fig = px.line(top_user_df, title="User with Most Check-Ins: User_"+str(u))
fig.update_xaxes(rangeslider_visible=True, rangeselector = dict(buttons=Buttons))
fig.show()
```



Location with Most Check-Ins

In [14]:
```
# Collecting Check-ins of Location with most Check-ins
l = df['location_id'].value_counts().idxmax()
timestamps = df[df.location_id == l]['check_in_time']

C = Counter([datetime.strptime(str(i), '%Y-%m-%dT%H:%M:%SZ') for i in timestamps])

top_location_df = pd.DataFrame()
top_location_df['Timestamp'] = [i for i in C]
top_location_df['Check_ins'] = [C[i] for i in C]

top_location_df = top_location_df.sort_values('Timestamp')
top_location_df = top_location_df.resample('H', on='Timestamp').Check_ins.sum()
```

In [15]:
```
# Plotting
fig = px.line(top_location_df, title='Location with Most Check-Ins: '+l)
fig.update_xaxes(rangeslider_visible=True, rangeselector = dict(buttons=Buttons))
fig.show()
```