In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import altair as alt
import plotly.graph_objects as go
```

In [ ]:
```
import warnings
warnings.filterwarnings("ignore")
# warnings.resetwarnings()
```

In [ ]:
```
firewall_06 = pd.read_csv('/Users/stevenyoo/Desktop/CS329E - Data Visualization/Project/Data/Firewall-0406
firewall_07 = pd.read_csv('/Users/stevenyoo/Desktop/CS329E - Data Visualization/Project/Data/Firewall-0407
```

In [ ]:
```
# convert Date/time to datetime structure
firewall_06['Date/time'] = pd.to_datetime(firewall_06['Date/time'])
firewall_06['Date/time'] = firewall_06['Date/time'].dt.strftime('%m/%d/%Y %H:%M:%S')

firewall_07['Date/time'] = pd.to_datetime(firewall_07['Date/time'])
firewall_07['Date/time'] = firewall_07['Date/time'].dt.strftime('%m/%d/%Y %H:%M:%S')
```

Argument 1

In [ ]:
```
firewall_06_normal = firewall_06[(firewall_06['Date/time'] >= "04/05/2012 17:51:00") &
                                  (firewall_06['Date/time'] < "04/05/2012 20:25:00")]

firewall_06_malware = firewall_06[firewall_06['Date/time'] >= "04/05/2012 20:25:00"]
```

In [ ]:
```
firewall_06_normal_hourly = firewall_06_normal
firewall_06_normal_hourly['Date/time'] = pd.to_datetime(firewall_06_normal_hourly['Date/time'])
firewall_06_normal_hourly['Date/time'] = firewall_06_normal_hourly['Date/time'].dt.strftime('%m/%d %H')

firewall_07_hourly = firewall_07
firewall_07_hourly['Date/time'] = pd.to_datetime(firewall_07_hourly['Date/time'])
firewall_07_hourly['Date/time'] = firewall_07_hourly['Date/time'].dt.strftime('%m/%d %H')
```

In [ ]:

```
# create pivot table
data1 = firewall_06_normal_hourly
data1 = data1[['Date/time', 'Operation']]
grouped_data_06_normal = data1.groupby(['Date/time', 'Operation']).size().reset_index(name = 'Count')
pivot_06_normal_operation = grouped_data_06_normal.pivot(index='Date/time',
                                                         columns='Operation',
                                                         values='Count').fillna(0).astype(int).reset_index
pivot_06_normal_operation.drop('(empty)', axis=1, inplace=True)

# repeat for malware data
data2 = firewall_07_hourly
data2 = data2[['Date/time', 'Operation']]
grouped_data_07 = data2.groupby(['Date/time', 'Operation']).size().reset_index(name = 'Count')
pivot_07_operation = grouped_data_07.pivot(index='Date/time',
                                           columns='Operation',
                                           values='Count').fillna(0).astype(int).reset_index()
pivot_07_operation.drop('(empty)', axis=1, inplace=True)
pivot_07_operation.drop('Command executed', axis = 1, inplace=True)
```

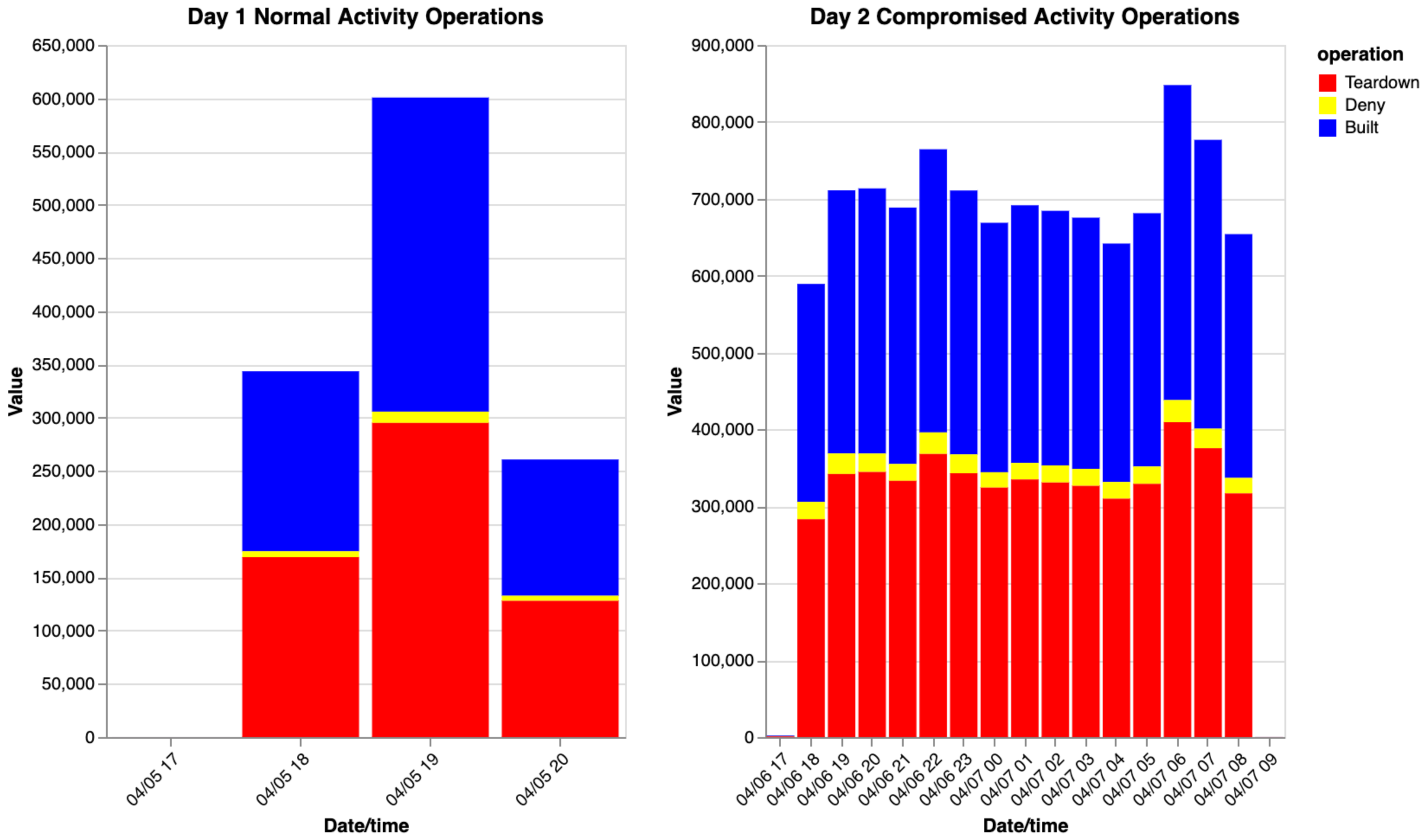
In [ ]:

```
melted_06_normal_operation = pivot_06_normal_operation.melt(id_vars='Date/time',
                                                            var_name='operation',
                                                            value_name='Value')
melted_07_operation = pivot_07_operation.melt(id_vars='Date/time',
                                              var_name='operation',
                                              value_name='Value')
```

In [ ]:

```
color_scale = alt.Scale(domain = ('Teardown', 'Deny', 'Built'), range = ['#FF0000', '#FFFF00', '#0000FF'])

chart1 = alt.Chart(melted_06_normal_operation).mark_bar().encode(
    x=alt.X('Date/time:O', axis=alt.Axis(labelAngle=-45)),
    y='Value:Q',
    color=alt.Color('operation:N', scale = color_scale),
    tooltip=['Date/time:O', 'operation:N', 'Value:Q']
).properties(
    width=300,
    height=400,
    title='Day 1 Normal Activity Operations'
)

chart2 = alt.Chart(melted_07_operation).mark_bar().encode(
    x=alt.X('Date/time:O', axis=alt.Axis(labelAngle=-45)),
    y='Value:Q',
    color=alt.Color('operation:N', scale = color_scale),
    tooltip=['Date/time:O', 'operation:N', 'Value:Q']
).properties(
    width=300,
    height=400,
    title='Day 2 Compromised Activity Operations'
)

chart1 | chart2
```

Out[ ]:



Argument 2

In [ ]:

```
firewall_06 = pd.read_csv('/Users/stevenyoo/Desktop/CS329E - Data Visualization/Project/Data/Firewall-0406
firewall_07 = pd.read_csv('/Users/stevenyoo/Desktop/CS329E - Data Visualization/Project/Data/Firewall-0407

# convert Date/time to datetime structure
firewall_06['Date/time'] = pd.to_datetime(firewall_06['Date/time'])
firewall_06['Date/time'] = firewall_06['Date/time'].dt.strftime('%m/%d/%Y %H:%M:%S')

firewall_07['Date/time'] = pd.to_datetime(firewall_07['Date/time'])
firewall_07['Date/time'] = firewall_07['Date/time'].dt.strftime('%m/%d/%Y %H:%M:%S')
```

In [ ]:

```
# convert date/time to hourly
firewall_06_hourly = firewall_06
firewall_06_hourly['Date/time'] = pd.to_datetime(firewall_06['Date/time'])
firewall_06_hourly['Date/time'] = firewall_06['Date/time'].dt.strftime('%m/%d %H')

firewall_07_hourly = firewall_07
firewall_07_hourly['Date/time'] = pd.to_datetime(firewall_07['Date/time'])
firewall_07_hourly['Date/time'] = firewall_07['Date/time'].dt.strftime('%m/%d %H')
```

In [ ]:

```
# create pivot table
pivoted_data_06_port = firewall_06_hourly.pivot_table(index='Date/time',
                                                      columns='Destination port',
                                                      aggfunc='count',
                                                      fill_value=0).reset_index()

pivoted_data_07_port = firewall_07_hourly.pivot_table(index='Date/time',
                                                      columns='Destination port',
                                                      aggfunc='count',
                                                      fill_value=0).reset_index()
```

In [ ]:

```
data = firewall_06_hourly[firewall_06_hourly['Destination port'] == '21']
data = data[['Date/time', 'Destination port']]
grouped_data_06 = data.groupby('Date/time').size().reset_index(name='Port_21')

data = firewall_07_hourly[firewall_07_hourly['Destination port'] == '21']
data = data[['Date/time', 'Destination port']]
```

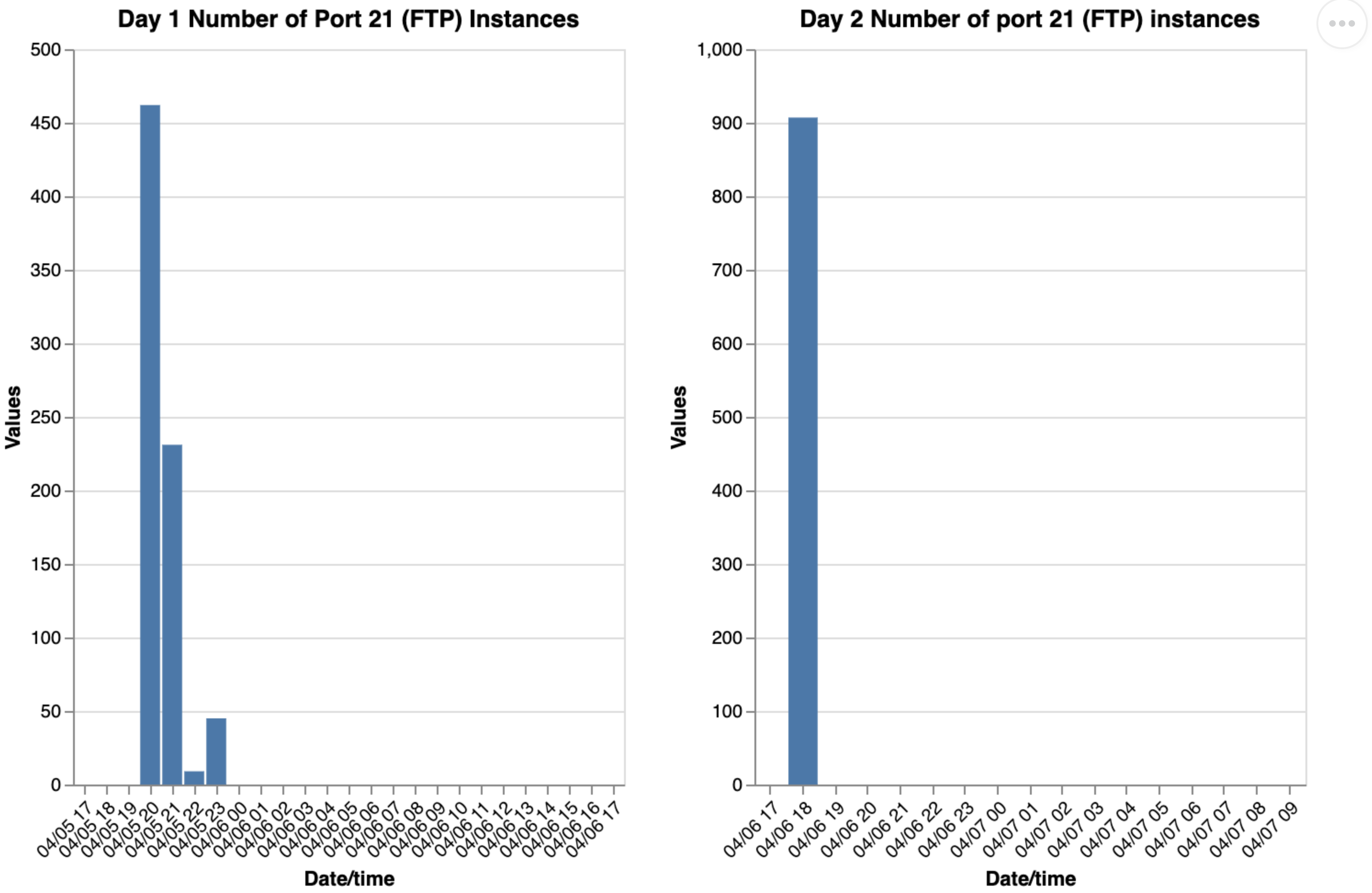
```
grouped_data_07 = data.groupby('Date/time').size().reset_index(name='Port_21')

pivoted_data_06_port = pd.merge(pivoted_data_06_port, grouped_data_06, on='Date/time', how='left')
pivoted_data_07_port = pd.merge(pivoted_data_07_port, grouped_data_07, on='Date/time', how='left')

# replace NaN with 0
pivoted_data_06_port['Port_21'].fillna(0, inplace=True)
pivoted_data_07_port['Port_21'].fillna(0, inplace=True)
```

In [ ]:

```
data1 = pd.DataFrame({
    'Date/time': pivoted_data_06_port['Date/time'],
    'Values': pivoted_data_06_port['Port_21']
})

data2 = pd.DataFrame({
    'Date/time': pivoted_data_07_port['Date/time'],
    'Values': pivoted_data_07_port['Port_21']
})

# Create the Altair chart
chart1 = alt.Chart(data1).mark_bar().encode(
    x=alt.X('Date/time:N', axis=alt.Axis(labelAngle=-45)),
    y='Values:Q',
    tooltip=['Date/time:N', 'Values:Q']
).properties(
    width=300,
    height=400,
    title='Day 1 Number of Port 21 (FTP) Instances'
)

# 2nd dataset
chart2 = alt.Chart(data2).mark_bar().encode(
    x=alt.X('Date/time:N', axis = alt.Axis(labelAngle=-45)),
    y='Values:Q',
    tooltip=['Date/time:N', 'Values:Q']
).properties(
    width=300,
    height=400,
    title='Day 2 Number of port 21 (FTP) instances'
)
```

```
chart1 | chart2
```

Out[ ]:



Argument 3

In [ ]:

```
# create dataframe only where destination port is 22
data = firewall_06_hourly[firewall_06_hourly['Destination port'] == '22']
# only keep relavent columns
data = data[['Date/time', 'Destination port']]
# group by day/hour and find count
```

```
grouped_data_06 = data.groupby('Date/time').size().reset_index(name='Port_22')

# repeat for second dataframe
data = firewall_07_hourly[firewall_07_hourly['Destination port'] == '22']
data = data[['Date/time', 'Destination port']]
grouped_data_07 = data.groupby('Date/time').size().reset_index(name='Port_22')

# merge this to pivot table to add the counts (instances) where date/time match
pivoted_data_06_port = pd.merge(pivoted_data_06_port, grouped_data_06, on='Date/time', how='left')
pivoted_data_07_port = pd.merge(pivoted_data_07_port, grouped_data_07, on='Date/time', how='left')

# replace NaN with 0
pivoted_data_06_port['Port_22'].fillna(0, inplace=True)
pivoted_data_07_port['Port_22'].fillna(0, inplace=True)
```

In [ ]:

```
data1 = pd.DataFrame({
    'Date/time': pivoted_data_06_port['Date/time'],
    'Values': pivoted_data_06_port['Port_22']
})

sns.set(style="whitegrid")
plt.figure(figsize=(10, 6))

fig = go.Figure()
fig.add_trace(go.Bar(
    x=data1['Date/time'],
    y=data1['Values'],
    name='Bar Plot',
    marker_color='skyblue'
))

fig.add_trace(go.Scatter(
    x=data1['Date/time'],
    y=data1['Values'],
    mode='lines+markers',
    name='Line Plot',
    line=dict(color='orange')
))

fig.update_layout(
    title='Day 1 Number of Port 22 (SSH) Instances',
    xaxis_title='Date/time',
    yaxis_title='Number of Port 22 Instances',
    xaxis=dict(tickangle=-45)
)

fig.show()
```

```
<Figure size 1000x600 with 0 Axes>
```

Selection available as well as hover as interactions for plot

Argument 4

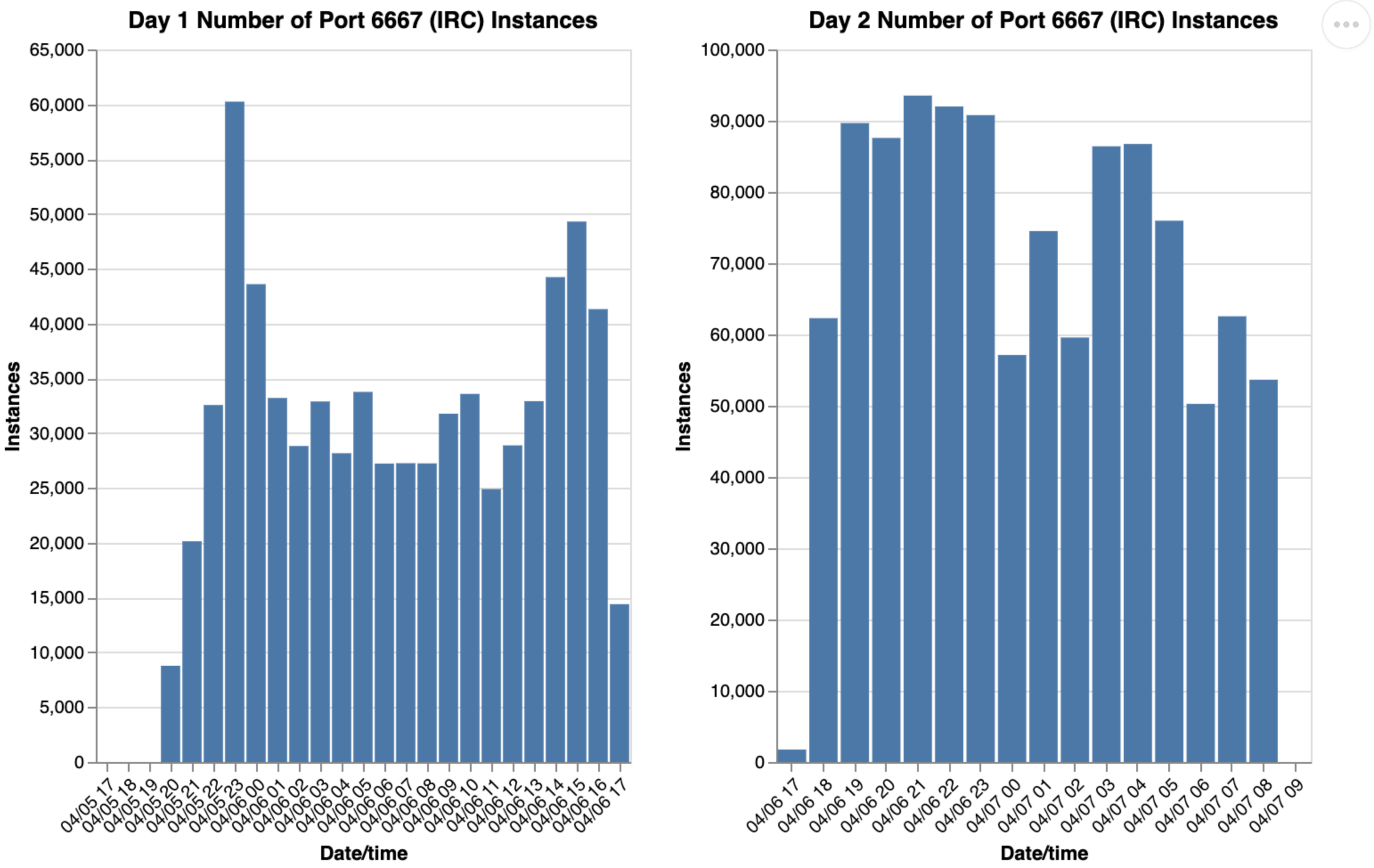
In [ ]:

```
# create dataframe only where destination port is 6667
data = firewall_06_hourly[firewall_06_hourly['Destination port'] == '6667']
# only keep relavent columns
data = data[['Date/time', 'Destination port']]
# group by day/hour and find count
grouped_data_06 = data.groupby('Date/time').size().reset_index(name='Port_6667')

# repeat for second dataframe
data = firewall_07_hourly[firewall_07_hourly['Destination port'] == '6667']
data = data[['Date/time', 'Destination port']]
grouped_data_07 = data.groupby('Date/time').size().reset_index(name='Port_6667')

# merge this to pivot table to add the counts (instances) where date/time match
pivoted_data_06_port = pd.merge(pivoted_data_06_port, grouped_data_06, on='Date/time', how='left')
pivoted_data_07_port = pd.merge(pivoted_data_07_port, grouped_data_07, on='Date/time', how='left')

# replace NaN with 0
pivoted_data_06_port['Port_6667'].fillna(0, inplace=True)
pivoted_data_07_port['Port_6667'].fillna(0, inplace=True)
```

In [ ]:

```
data1 = pd.DataFrame({
    'Date/time': pivoted_data_06_port['Date/time'],
    'Instances': pivoted_data_06_port['Port_6667']
})

data2 = pd.DataFrame({
    'Date/time': pivoted_data_07_port['Date/time'],
    'Instances': pivoted_data_07_port['Port_6667']
})

# Create the Altair chart
chart1 = alt.Chart(data1).mark_bar().encode(
    x=alt.X('Date/time:N', axis=alt.Axis(labelAngle=-45)),
    y='Instances:Q',
    tooltip=['Date/time:N', 'Instances:Q']
).properties(
    width=300,
    height=400,
    title='Day 1 Number of Port 6667 (IRC) Instances'
)

# 2nd dataset
chart2 = alt.Chart(data2).mark_bar().encode(
    x=alt.X('Date/time:N', axis = alt.Axis(labelAngle=-45)),
    y='Instances:Q',
    tooltip=['Date/time:N', 'Instances:Q']
).properties(
    width=300,
    height=400,
    title='Day 2 Number of Port 6667 (IRC) Instances'
)

chart1 | chart2
```

Out[ ]:



Argument 5

In [ ]:

```
pivoted_data_06_source = firewall_06_hourly.pivot_table(index='Date/time', columns='Source IP',
                                                         aggfunc='count', fill_value=0).reset_index()
pivoted_data_06_dest = firewall_06_hourly.pivot_table(index='Date/time', columns='Destination IP',
                                                       aggfunc='count', fill_value=0).reset_index()
pivoted_data_07_source = firewall_07_hourly.pivot_table(index='Date/time', columns='Source IP',
                                                         aggfunc='count', fill_value=0).reset_index()
```

```
pivoted_data_07_dest = firewall_07_hourly.pivot_table(index='Date/time', columns='Destination IP',
                                                      aggfunc='count', fill_value=0).reset_index()
```

In [ ]:

```
# create grouped by dataframe to count instances of each source IP at each date/time
data = firewall_06_hourly
data = data[['Date/time', 'Source IP']]
grouped_data_06.groupby('Date/time').size().reset_index(name = 'SourceIP')
grouped_data_06_source = data.groupby(['Date/time', 'Source IP']).size().reset_index(name='Count')

# create pivoted table to make each source IP a column and the count as it's values
pivot_df_source = grouped_data_06_source.pivot(index='Date/time',
                                               columns='Source IP',
                                               values='Count').fillna(0).astype(int).reset_index()
pivot_df_source.drop('(empty)', axis = 1, inplace=True) # remove empty values
pivot_df_source.loc['Total'] = pivot_df_source.drop('Date/time', axis=1).sum() # calculate totals

# only store the top 10 source IPs by total count
sorted_columns = pivot_df_source.drop('Date/time',
                                      axis=1).sum().sort_values(ascending=False).head(10).index
filtered_06_source = pivot_df_source[['Date/time'] + list(sorted_columns)]


# repeat for destination IP
data = firewall_06_hourly
data = data[['Date/time', 'Destination IP']]
grouped_data_06.groupby('Date/time').size().reset_index(name = 'DestinationIP')
grouped_data_06_dest = data.groupby(['Date/time', 'Destination IP']).size().reset_index(name='Count')

# create pivoted table to make each source IP a column and the count as it's values
pivot_df_dest = grouped_data_06_dest.pivot(index='Date/time',
                                           columns='Destination IP',
                                           values='Count').fillna(0).astype(int).reset_index()
pivot_df_dest.drop('(empty)', axis = 1, inplace=True) # remove empty values
pivot_df_dest.loc['Total'] = pivot_df_dest.drop('Date/time', axis=1).sum() # calculate totals

# only store the top 10 source IPs by total count
sorted_columns = pivot_df_dest.drop('Date/time', axis=1).sum().sort_values(ascending=False).head(10).index
filtered_06_dest = pivot_df_dest[['Date/time'] + list(sorted_columns)]
```

In [ ]:

```
# repeat for second dataframe
# create grouped by dataframe to count instances of each source IP at each date/time
```

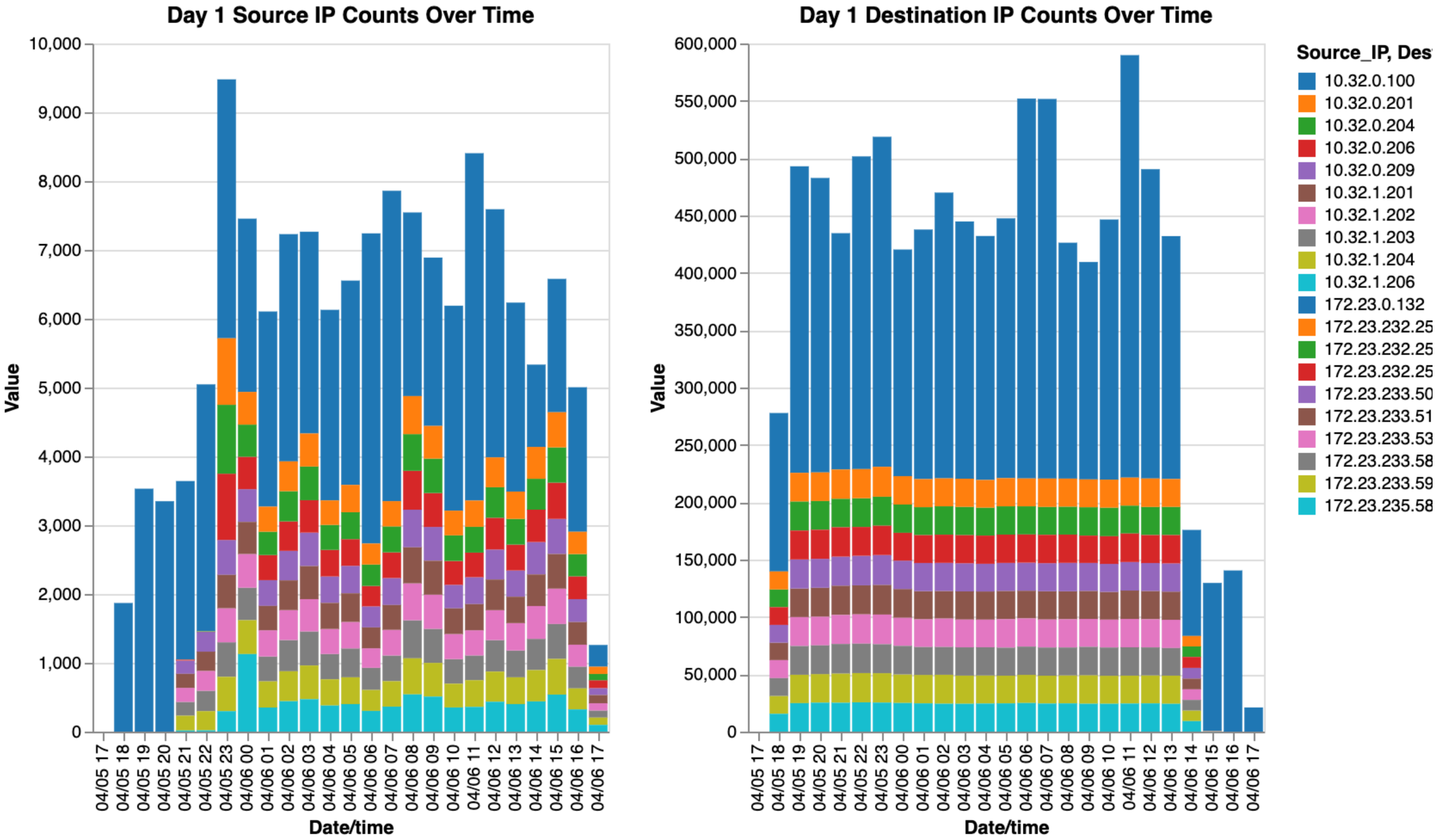
```
data = firewall_07_hourly
data = data[['Date/time', 'Source IP']]
grouped_data_07.groupby('Date/time').size().reset_index(name = 'SourceIP')
grouped_data_07_source = data.groupby(['Date/time', 'Source IP']).size().reset_index(name='Count')

# create pivoted table to make each source IP a column and the count as it's values
pivot_df_source = grouped_data_07_source.pivot(index='Date/time',
                                               columns='Source IP',
                                               values='Count').fillna(0).astype(int).reset_index()
pivot_df_source.drop('(empty)', axis = 1, inplace=True) # remove empty values
pivot_df_source.loc['Total'] = pivot_df_source.drop('Date/time', axis=1).sum() # calculate totals

# only store the top 10 source IPs by total count
sorted_columns = pivot_df_source.drop('Date/time',
                                      axis=1).sum().sort_values(ascending=False).head(10).index
filtered_07_source = pivot_df_source[['Date/time'] + list(sorted_columns)]


# repeat for destination IP
data = firewall_07_hourly
data = data[['Date/time', 'Destination IP']]
grouped_data_07.groupby('Date/time').size().reset_index(name = 'DestinationIP')
grouped_data_07_dest = data.groupby(['Date/time', 'Destination IP']).size().reset_index(name='Count')

# create pivoted table to make each source IP a column and the count as it's values
pivot_df_dest = grouped_data_07_dest.pivot(index='Date/time',
                                           columns='Destination IP',
                                           values='Count').fillna(0).astype(int).reset_index()
pivot_df_dest.drop('(empty)', axis = 1, inplace=True) # remove empty values
pivot_df_dest.loc['Total'] = pivot_df_dest.drop('Date/time', axis=1).sum() # calculate totals

# only store the top 10 source IPs by total count
sorted_columns = pivot_df_dest.drop('Date/time', axis=1).sum().sort_values(ascending=False).head(10).index
filtered_07_dest = pivot_df_dest[['Date/time'] + list(sorted_columns)]
```

In [ ]:

```
# plot for Source IPs and Destination IPs for firewall_06
melted_06_source = filtered_06_source.melt(id_vars='Date/time', var_name='Source_IP', value_name='Value')
melted_06_source = melted_06_source.dropna(subset=['Date/time'])

melted_06_dest = filtered_06_dest.melt(id_vars='Date/time', var_name='Dest_IP', value_name='Value')
melted_06_dest = melted_06_dest.dropna(subset=['Date/time'])
```

```
chart1 = alt.Chart(melted_06_source).mark_bar().encode(
    x='Date/time:O',
    y='Value:Q',
    color=alt.Color('Source_IP:N', scale = alt.Scale(scheme='category10')),
    tooltip=['Date/time:O', 'Source_IP:N', 'Value:Q']
).properties(
    width=300,
    height=400,
    title='Day 1 Source IP Counts Over Time'
)

chart2 = alt.Chart(melted_06_dest).mark_bar().encode(
    x='Date/time:O',
    y='Value:Q',
    color=alt.Color('Dest_IP:N', scale = alt.Scale(scheme='plasma')),
    tooltip=['Date/time:O', 'Dest_IP:N', 'Value:Q']
).properties(
    width=300,
    height=400,
    title='Day 1 Destination IP Counts Over Time'
)

chart1 | chart2
```

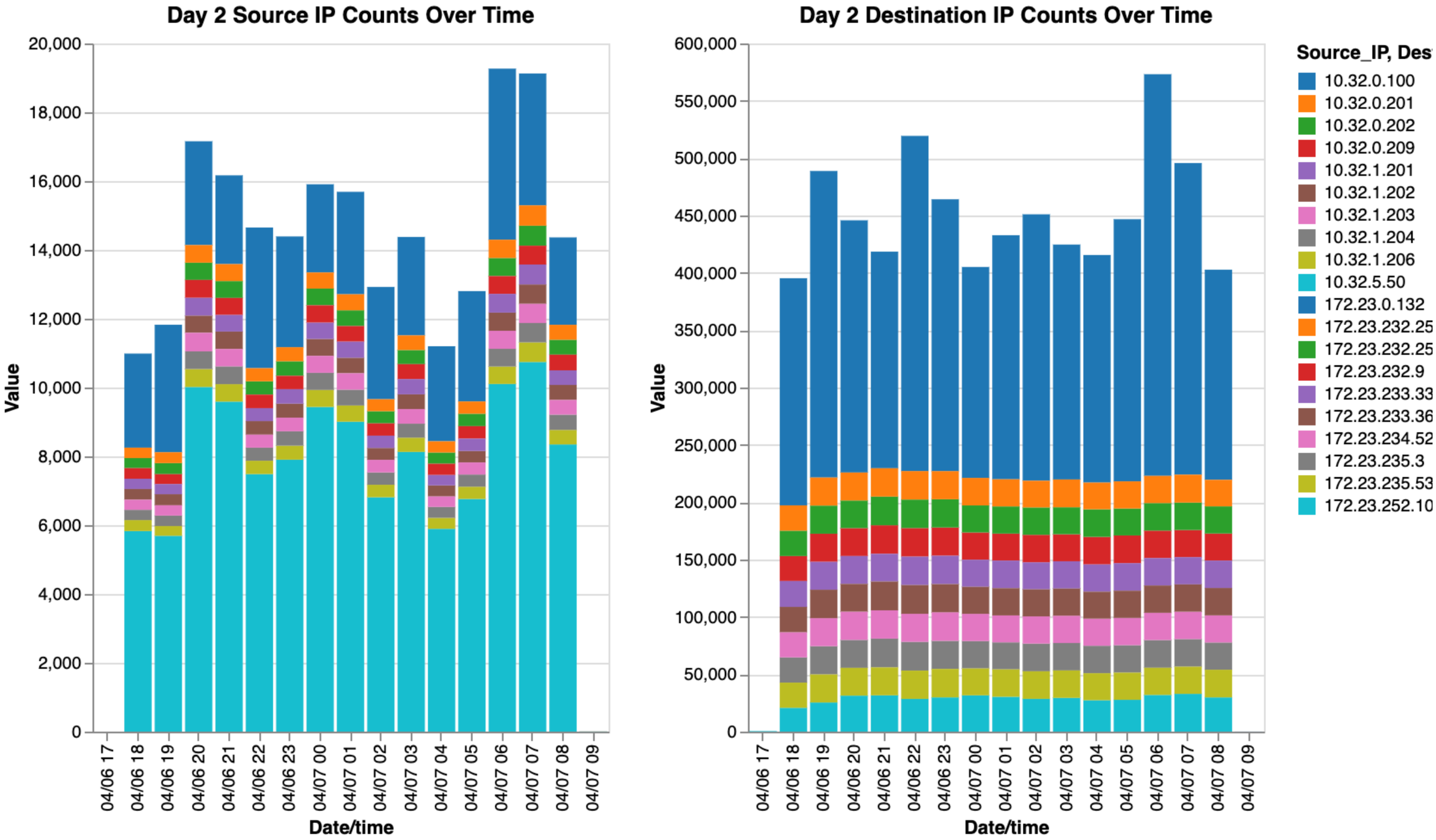
Out[ ]:



In [ ]:

```
# plot for Source IPs and Destination IPs for firewall_07

melted_07_source = filtered_07_source.melt(id_vars='Date/time', var_name='Source_IP', value_name='Value')
melted_07_source = melted_07_source.dropna(subset=['Date/time'])

melted_07_dest = filtered_07_dest.melt(id_vars='Date/time', var_name='Dest_IP', value_name='Value')
melted_07_dest = melted_07_dest.dropna(subset=['Date/time'])

chart1 = alt.Chart(melted_07_source).mark_bar().encode(
    x='Date/time:O',
    y='Value:Q',
    color=alt.Color('Source_IP:N', scale = alt.Scale(scheme='category10')),
    tooltip=['Date/time:O', 'Source_IP:N', 'Value:Q']
).properties(
    width=300,
    height=400,
    title=' Day 2 Source IP Counts Over Time'
)

chart2 = alt.Chart(melted_07_dest).mark_bar().encode(
    x='Date/time:O',
    y='Value:Q',
    color=alt.Color('Dest_IP:N', scale = alt.Scale(scheme='plasma')),
    tooltip=['Date/time:O', 'Dest_IP:N', 'Value:Q']
).properties(
    width=300,
    height=400,
    title='Day 2 Destination IP Counts Over Time'
)

chart1 | chart2
```

Out[ ]:


Day 2 Source IP Counts Over Time
Value
0
2,000
4,000
6,000
8,000
10,000
12,000
14,000
16,000
18,000
20,000
Date/time
04/06 17
04/06 18
04/06 19
04/06 20
04/06 21
04/06 22
04/06 23
04/07 00
04/07 01
04/07 02
04/07 03
04/07 04
04/07 05
04/07 06
04/07 07
04/07 08
04/07 09
Day 2 Destination IP Counts Over Time
Value
0
50,000
100,000
150,000
200,000
250,000
300,000
350,000
400,000
450,000
500,000
550,000
600,000
Date/time
04/06 17
04/06 18
04/06 19
04/06 20
04/06 21
04/06 22
04/06 23
04/07 00
04/07 01
04/07 02
04/07 03
04/07 04
04/07 05
04/07 06
04/07 07
04/07 08
04/07 09
Source_IP, Des
10.32.0.100
10.32.0.201
10.32.0.202
10.32.0.209
10.32.1.201
10.32.1.202
10.32.1.203
10.32.1.204
10.32.1.206
10.32.5.50
172.23.0.132
172.23.232.25
172.23.232.25
172.23.232.9
172.23.233.33
172.23.233.36
172.23.234.52
172.23.235.3
172.23.235.53
172.23.252.10