Title of Final Project: Communication Patterns within Amusement Park

Team Name: NL

Student Name: Na Young Moon

Student UT EID: nm29658

Student Name: Luis Kim

Student UT EID: lyk226

Date: 04/12/2023

In [ ]:
```
import pandas as pd
import numpy as np
import altair as alt
from itertools import product
from bokeh.plotting import show, output_notebook, figure
from bokeh.models import HoverTool, ColumnDataSource
from bokeh.layouts import gridplot
```

In [ ]:
```
df_fri = pd.read_csv("./MC2 2015 Data/comm-data-Fri.csv")
df_sat = pd.read_csv("./MC2 2015 Data/comm-data-Sat.csv")
df_sun = pd.read_csv("./MC2 2015 Data/comm-data-Sun.csv")
```

In [ ]:
```
df = pd.concat([df_fri, df_sat, df_sun])
df
```

Out[ ]:

	Timestamp	from	to	location
0	2014-6-06 08:03:19	439105	1053224	Kiddie Land
1	2014-6-06 08:03:19	439105	1696241	Kiddie Land
2	2014-6-06 08:03:19	439105	580064	Kiddie Land
3	2014-6-06 08:03:19	439105	1464748	Kiddie Land
4	2014-6-06 08:03:47	1836139	1593258	Entry Corridor
...	...	...	...	...
1548719	2014-6-08 23:20:37	1983198	external	Tundra Land
1548720	2014-6-08 23:20:38	1555391	857616	Tundra Land
1548721	2014-6-08 23:21:04	839736	2022346	Entry Corridor
1548722	2014-6-08 23:22:05	839736	1109589	Entry Corridor
1548723	2014-6-08 23:23:57	1222078	external	Entry Corridor

4153329 rows × 4 columns

In [ ]:
```
# Change type of Timestamp to datetime and from to object
df['from'] = df['from'].astype('object')
df['from'] = df['from'].astype('string')
df['Timestamp'] = pd.to_datetime(df['Timestamp'])
df["day"] = df["Timestamp"].dt.day
df["hour"] = df["Timestamp"].dt.hour
df["minute"] = df["Timestamp"].dt.minute
```

Description of the Dataset

Data Wrangling

In [ ]:
```
#Checks for the sum of duplicated rows
df.duplicated().sum()
```

Out[ ]:
```
1257
```

In [ ]:
```
#Drop duplicates
df = df.drop_duplicates()
```

In [ ]:
```
from_counts = df.groupby([df["from"]]).size().reset_index().rename(columns={0: "Count"})
to_counts = df.groupby([df["to"]]).size().reset_index().rename(columns={0: "Count"})
id_counts = pd.merge(from_counts,to_counts,left_on=["from"],right_on=["to"])
id_counts["TotCount"] = id_counts["Count_x"]+id_counts["Count_y"]
id_counts = id_counts.drop(columns=["Count_x","Count_y","to"])
id_counts = id_counts.rename(columns = {"from":"id"})

top20ids = id_counts.sort_values('TotCount',ascending=False).head(20)['id'].tolist()
```

Goal:

The display of Scott Jones's soccer memorabilia in the Creighton Pavilion at DinoFun World was vandalized. A number of the items of memorabilia had been damaged or had been defaced with black spray paint. Spray paint had been used to write derogatory statements about Scott Jones on the display and throughout the Pavilion. Our goal is to figure out the patterns within the communication data to figure out when this crime occurred.

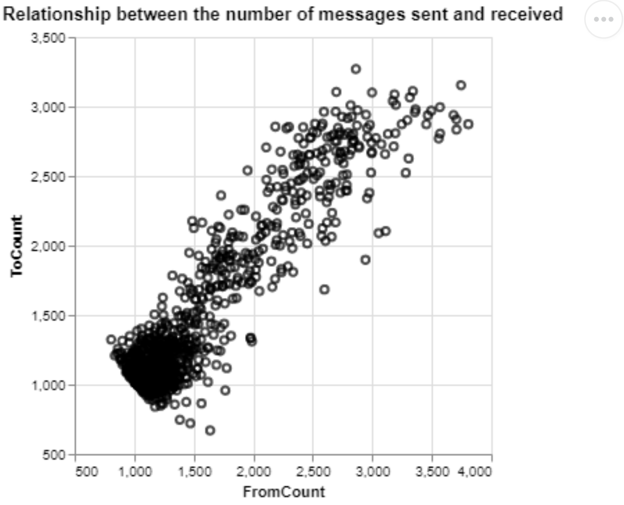
Visualizations

Visualization 1: Relationship between the number of messages sent and received

In [ ]:
```
from_counts = df.groupby(df["from"]).size().reset_index().rename(columns = {0: "FromCount"})
to_counts = df.groupby(df["to"]).size().reset_index().rename(columns = {0: "ToCount"})
totfromto_counts = pd.merge(left = from_counts, right = to_counts, left_on = 'from',right_on = 'to')
totfromto_counts["TotalCount"] = totfromto_counts["FromCount"]+totfromto_counts["ToCount"]
totfromto_counts = totfromto_counts.sort_values("TotalCount",ascending=False)
totfromto_counts = totfromto_counts.iloc[:1000]
totfromto_counts["ID"]=totfromto_counts["from"]

toptexters = totfromto_counts[(totfromto_counts["TotalCount"]>5000) & (totfromto_counts["TotalCount"]<7000)]["ID"].tolist()
```

In [ ]:
```
multi = alt.selection_point(on='mouseover', nearest=True)

alt.Chart(totfromto_counts.iloc[2:,]).mark_point().encode(
    alt.X('FromCount:Q',scale = alt.Scale(domain=[500, 4000])),
    alt.Y('ToCount:Q',scale = alt.Scale(domain = [500,3500])),
    tooltip=['ID:N'],
    color=alt.condition(multi, alt.value('black'), alt.value('lightgray'))
).add_params(
    multi
).properties(
    title='Relationship between the number of messages sent and received'
).interactive()
```

Out[ ]:



The nature of individual IDs can be observed through this plot. The number of messages sent and received is approximately equal, so this plot suggests that there is a balanced exchange of information between the parties involved.

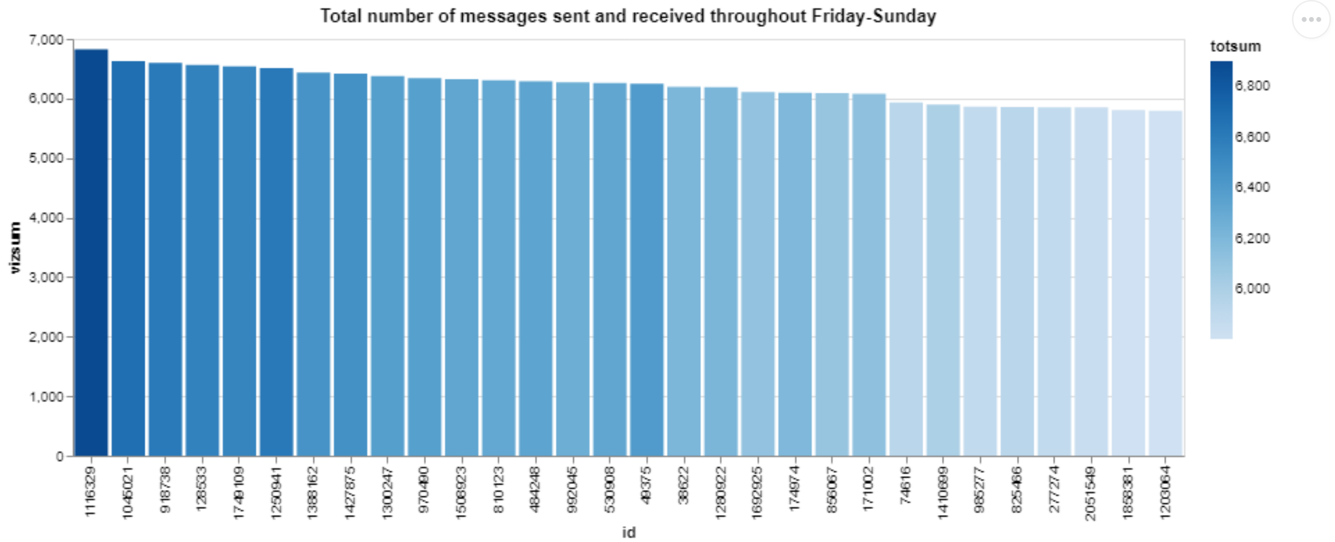
Visualization 2: Total number of messages sent and received per ID

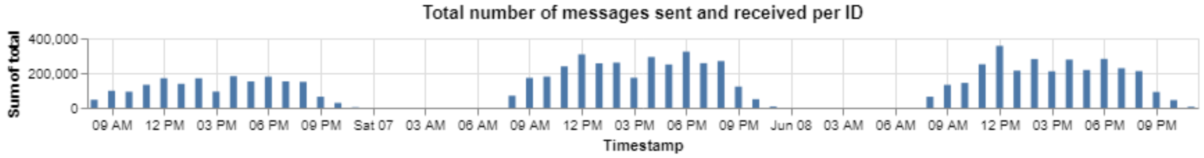
In [ ]:
```
fromhl_counts = df.groupby([df["day"],df["hour"],df["from"]]).size().reset_index().rename(columns={0: "Count"})
tohl_counts = df.groupby([df["day"],df["hour"],df["to"]]).size().reset_index().rename(columns={0: "Count"})
dfviz1 = pd.merge(fromhl_counts,tohl_counts,left_on=["day","hour","from"],right_on=["day","hour","to"])
dfviz1["id"]=dfviz1["from"]
dfviz1["total"]=dfviz1["Count_x"]+dfviz1["Count_y"]
dfviz1 = dfviz1.drop(columns=["from","to"])
#dfviz1 = dfviz1[dfviz1["id"].isin(toptexters)]
dfviz1["Timestamp"] = pd.to_datetime(dict(year=2014, month=6, day=dfviz1["day"], hour=dfviz1["hour"]))
dfviz1 = dfviz1.rename(columns={"Count_x":"Count_from","Count_y":"Count_to"})
dfviz1 = dfviz1.drop(columns=["day","hour"])
dfviz1 = pd.merge(dfviz1,id_counts,left_on=["id"],right_on=["id"])

dfviz1
```

```
Out[ ]:
        Count_from  Count_to       id  total            Timestamp  TotCount
     0           1         4  1000708      5  2014-06-06 08:00:00       101
     1           3         3  1000708      6  2014-06-06 09:00:00       101
     2           1         1  1000708      2  2014-06-06 10:00:00       101
     3           1         1  1000708      2  2014-06-06 11:00:00       101
     4           1         4  1000708      5  2014-06-06 12:00:00       101
   ...         ...       ...      ...    ...                  ...       ...
136053           1         2   158818      3  2014-06-08 20:00:00         6
136054           2         1  1594937      3  2014-06-08 20:00:00         7
136055           3         1   162882      4  2014-06-08 20:00:00         7
136056           1         1  1895812      2  2014-06-08 20:00:00         9
136057           1         1   941716      2  2014-06-08 23:00:00        15

136058 rows × 6 columns
```

```
In [ ]:
alt.data_transformers.disable_max_rows()

interval = alt.selection_interval(encodings=['x'])

selector = alt.Chart(dfviz1).mark_bar().encode(
    x = "Timestamp:T",
    y = "sum(total)"
).properties(
    width=800,
    height=50
).add_params(
    interval
).properties(
    title='Total number of messages sent and received per ID'
)

chart = alt.Chart(dfviz1).mark_bar().transform_filter(
    interval
).transform_aggregate(
    vizsum ='sum(total)',
    totsum = 'median(TotCount)',
    groupby=['id']
).encode(
    alt.X('id:N').sort('-y'),
    y = 'vizsum:Q',
    color = 'totsum:Q'
).transform_filter(
    '(datum.id != "1278894") & (datum.id != "839736")'
).transform_window(
    window=[{'op': 'rank', 'as': 'rank'}],
    sort=[{'field': 'vizsum', 'order': 'descending'}]
).transform_filter('datum.rank <= 30').properties(
    width=800,
    height=300,
).properties(
    title='Total number of messages sent and received throughout Friday-Sunday'
)

chart & selector
```

Out[ ]:


Total number of messages sent and received throughout Friday-Sunday


Total number of messages sent and received per ID

Visualization 3: Top 10 Total number of messages SENT and RECEIVED per ID by hour and day

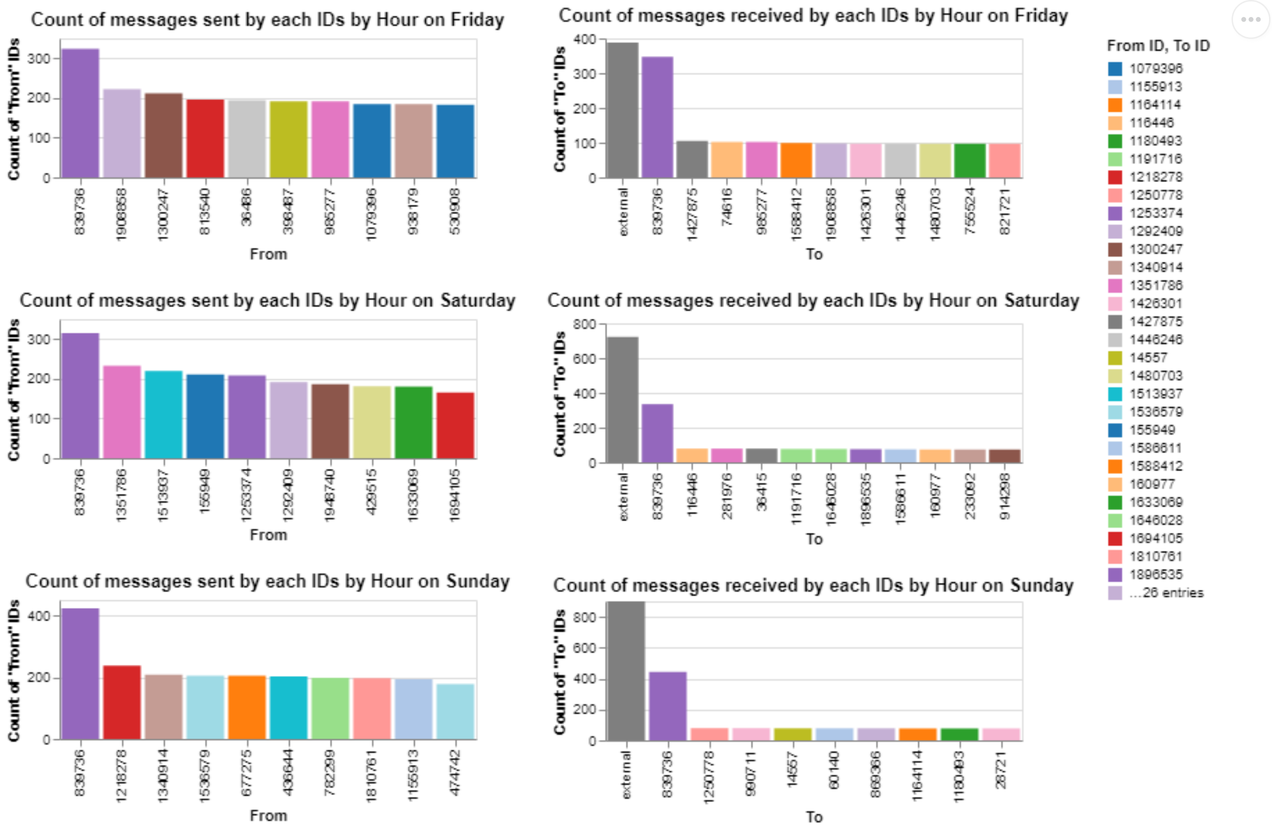
In [ ]:

```
top_ids_from = df.groupby(['hour', 'from', 'day']).size().reset_index(name='count')
top_ids_from = top_ids_from[top_ids_from['count']>10]

fri_from = top_ids_from[top_ids_from['day'] == 6]
sat_from = top_ids_from[top_ids_from['day'] == 7]
sun_from = top_ids_from[top_ids_from['day'] == 8]

top_ids_to = df.groupby(['hour', 'to', 'day']).size().reset_index(name='count')
top_ids_to = top_ids_to[top_ids_to['count']>10]

fri_to = top_ids_to[top_ids_to['day'] == 6]
sat_to = top_ids_to[top_ids_to['day'] == 7]
sun_to = top_ids_to[top_ids_to['day'] == 8]
```

In [ ]:

```
slider = alt.binding_range(min=8, max=23, step=1, name='Select Hour:')
selector = alt.selection_point(name="Hour", bind=slider, fields=['hour'], value = 8)

fri_chart_from = alt.Chart(fri_from).mark_bar().encode(
    x=alt.X('from:O', sort='-y', axis=alt.Axis(title='From')),
    y=alt.Y('count:Q', axis=alt.Axis(title='Count of "from" IDs')),
    color=alt.Color('from:N', legend=alt.Legend(title='From ID'), scale=alt.Scale(scheme='category20')),
    tooltip=['from', 'count']
).add_params(
    selector
).transform_filter(
    selector
).properties(
    width=300,
    height=100,
    title='Count of messages sent by each IDs by Hour on Friday'
).transform_window(
    window=[{'op': 'rank', 'as': 'rank'}],
    sort=[{'field': 'count', 'order': 'descending'}]
).transform_filter('datum.rank <= 10')

sat_chart_from = alt.Chart(sat_from).mark_bar().encode(
    x=alt.X('from:O', sort='-y', axis=alt.Axis(title='From')),
    y=alt.Y('count:Q', axis=alt.Axis(title='Count of "from" IDs')),
    color=alt.Color('from:N', legend=alt.Legend(title='From ID'), scale=alt.Scale(scheme='category20')),
    tooltip=['from', 'count']
).add_params(
    selector
).transform_filter(
    selector
).properties(
    width=300,
    height=100,
    title='Count of messages sent by each IDs by Hour on Saturday'
).transform_window(
    window=[{'op': 'rank', 'as': 'rank'}],
    sort=[{'field': 'count', 'order': 'descending'}]
).transform_filter('datum.rank <= 10')

sun_chart_from = alt.Chart(sun_from).mark_bar().encode(
    x=alt.X('from:O', sort='-y', axis=alt.Axis(title='From')),
    y=alt.Y('count:Q', axis=alt.Axis(title='Count of "from" IDs')),
    color=alt.Color('from:N', legend=alt.Legend(title='From ID'), scale=alt.Scale(scheme='category20')),
    tooltip=['from', 'count']
).add_params(
    selector
).transform_filter(
    selector
).properties(
    width=300,
    height=100,
    title='Count of messages sent by each IDs by Hour on Sunday'
).transform_window(
    window=[{'op': 'rank', 'as': 'rank'}],
    sort=[{'field': 'count', 'order': 'descending'}]
).transform_filter('datum.rank <= 10')

fri_chart_to = alt.Chart(fri_to).mark_bar().encode(
    x=alt.X('to:O', sort='-y', axis=alt.Axis(title='To')),
    y=alt.Y('count:Q', axis=alt.Axis(title='Count of "To" IDs')),
    color=alt.Color('to:N', legend=alt.Legend(title='To ID'), scale=alt.Scale(scheme='category20')),
    tooltip=['to', 'count']
).add_params(
    selector
).transform_filter(
    selector
).properties(
    width=300,
    height=100,
    title='Count of messages received by each IDs by Hour on Friday'
).transform_window(
    window=[{'op': 'rank', 'as': 'rank'}],
    sort=[{'field': 'count', 'order': 'descending'}]
).transform_filter('datum.rank <= 10')

sat_chart_to = alt.Chart(sat_to).mark_bar().encode(
    x=alt.X('to:O', sort='-y', axis=alt.Axis(title='To')),
    y=alt.Y('count:Q', axis=alt.Axis(title='Count of "To" IDs')),
    color=alt.Color('to:N', legend=alt.Legend(title='To ID'), scale=alt.Scale(scheme='category20')),
    tooltip=['to', 'count']
).add_params(
    selector
).transform_filter(
    selector
).properties(
    width=300,
    height=100,
    title='Count of messages received by each IDs by Hour on Saturday'
).transform_window(
    window=[{'op': 'rank', 'as': 'rank'}],
    sort=[{'field': 'count', 'order': 'descending'}]
).transform_filter('datum.rank <= 10')

sun_chart_to = alt.Chart(sun_to).mark_bar().encode(
    x=alt.X('to:O', sort='-y', axis=alt.Axis(title='To')),
    y=alt.Y('count:Q', axis=alt.Axis(title='Count of "To" IDs')),
    color=alt.Color('to:N', legend=alt.Legend(title='To ID'), scale=alt.Scale(scheme='category20')),
    tooltip=['to', 'count']
).add_params(
    selector
).transform_filter(
    selector
).properties(
    width=300,
    height=100,
    title='Count of messages received by each IDs by Hour on Sunday'
).transform_window(
    window=[{'op': 'rank', 'as': 'rank'}],
    sort=[{'field': 'count', 'order': 'descending'}]
).transform_filter('datum.rank <= 10')

(fri_chart_from & sat_chart_from & sun_chart_from) | (fri_chart_to & sat_chart_to & sun_chart_to)
```

Out[ ]:



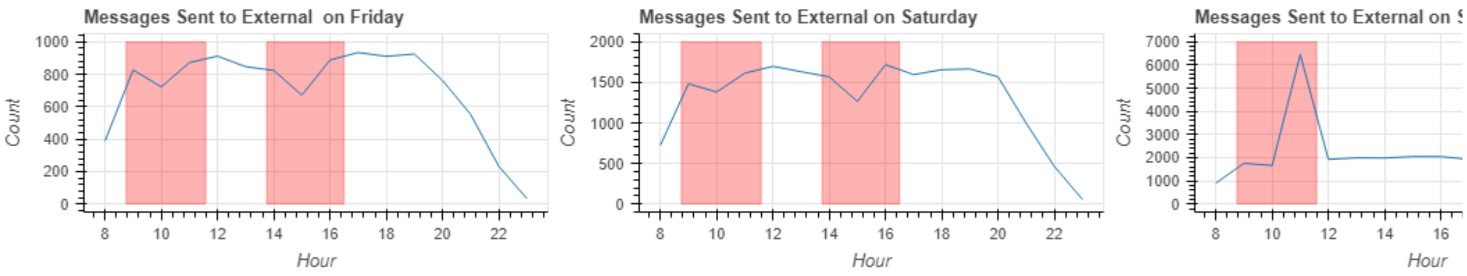
Select Hour: 8

At noon, we see an unusual amount of 839736 on Sunday compared to the normal communication patterns we see on Friday and Saturday.

Visualization 4: Communication pattern of External ID when Scott Jones is at the park

In [ ]:
```python
six = ColumnDataSource(fri_to[(fri_to['day'] == 6) & (fri_to['to'] == 'external')])
seven = ColumnDataSource(sat_to[(sat_to['day'] == 7) & (sat_to['to'] == 'external')])
eight = ColumnDataSource(sun_to[(sun_to['day'] == 8) & (sun_to['to'] == 'external')])


fri_fig = figure(title="Messages Sent to External  on Friday", x_axis_label='Hour', y_axis_label='Count', width=400, height=200)
sat_fig = figure(title="Messages Sent to External on Saturday", x_axis_label='Hour', y_axis_label='Count', width=400, height=200)
sun_fig = figure(title="Messages Sent to External on Sunday", x_axis_label='Hour', y_axis_label='Count', width=400, height=200)


x = np.array([8.75, 11.5833, 11.5833, 8.75])
y = np.array([0, 0, 1000, 1000])
x1 = np.array([13.75, 16.5, 16.5, 13.75])
y1 = np.array([0, 0, 1000, 1000])
fri_fig.patch(x, y, color='red', alpha=0.3)
fri_fig.patch(x1, y1, color='red', alpha=0.3)

x = np.array([8.75, 11.5833, 11.5833, 8.75])
y = np.array([0, 0, 2000, 2000])
x1 = np.array([13.75, 16.5, 16.5, 13.75,])
y1 = np.array([0, 0, 2000, 2000])
sat_fig.patch(x, y, color='red', alpha=0.3)
sat_fig.patch(x1, y1, color='red', alpha=0.3)

x = np.array([8.75, 11.5833, 11.5833, 8.75])
y = np.array([0,0,7000, 7000])
sun_fig.patch(x, y, color='red', alpha=0.3)


hover = HoverTool(tooltips=[('Hour', '@hour'), ('Count', '@count')])
fri_fig.line(x='hour', y='count', source=six)
sat_fig.line(x='hour', y='count', source=seven)
sun_fig.line(x='hour', y='count', source=eight)
fri_fig.add_tools(hover)
sat_fig.add_tools(hover)
sun_fig.add_tools(hover)

grid = gridplot([[fri_fig, sat_fig, sun_fig]])
output_notebook()
show(grid)
```

BokehJS 3.1.0 successfully loaded.



The common ID we see that are frequently placing first in terms of the amount of messages sent or recieved are: 1278894, 839736, and external.

Specifically, external places first in specific time frames. To preface, Scott Jones visits the amusement park between the times 8:45-11:35 each day and 13:45-16:30 on Friday and Saturday, which is labeled by the red segments. We can see that when he shows up, the amount of external communication decreased followed by a spike in the communication numbers specifically at 10.

From this we can assume that the show starts at 8:45 and 13:45 and conclude that there is a reduced external messaging when Scott Jones shows are in progress. We can also conclude that the show ends at 10 pm and 3 pm, as the number of external messaging spikes during those times.

Another thing we want to mention is the unusual spike on Sunday where we hypothesized that some incident occurred after the first show on that day.

Visualization 4: Timeline of messages sent and received by each ID

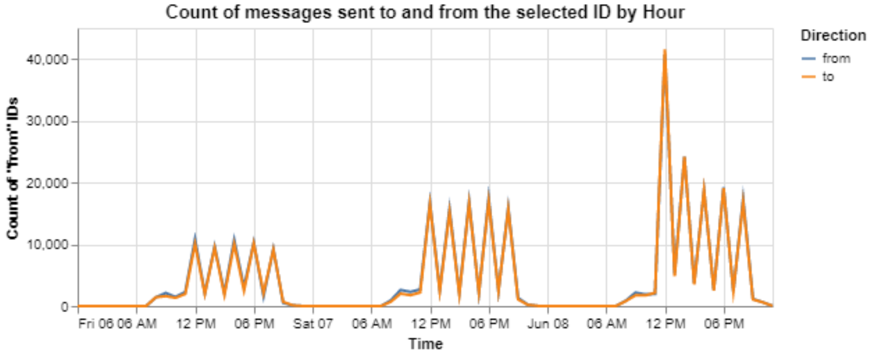
In [ ]:

```
top20idsfrom = top_ids_from[top_ids_from['from'].isin(top20ids)]
top20idsto = top_ids_to[top_ids_to['to'].isin(top20ids)]

hour = [i for i in range(24)]
day = [i for i in range(6,9)]

dfviz3 = pd.DataFrame(list(product(top20ids,hour,day)), columns=['id','hour','day'])
dfviz3 = pd.merge(dfviz3,top20idsfrom,left_on=['id','hour','day'],right_on = ['from','hour','day'],how='left')
dfviz3 = dfviz3.drop(columns=['from']).rename(columns={'count':'from'})
dfviz3['from']=dfviz3['from'].fillna(0).astype('int')

dfviz3 = pd.merge(dfviz3,top20idsto,left_on=['id','hour','day'],right_on = ['to','hour','day'],how='left')
dfviz3 = dfviz3.drop(columns=['to']).rename(columns={'count':'to'})
dfviz3['to']=dfviz3['to'].fillna(0).astype('int')

dfviz3["Timestamp"] = pd.to_datetime(dict(year=2014, month=6, day=dfviz3["day"], hour=dfviz3["hour"]))
dfviz3 = dfviz3.drop(columns=["day","hour"])

dfviz3 = pd.melt(dfviz3,id_vars=['id','Timestamp'],value_vars=['from','to']).rename(columns={'variable':'Direction','value':'count'})

dfviz3tot = dfviz3.groupby(['Timestamp','Direction']).sum('count').reset_index()
dfviz3tot['id']='Total'

dfviz3 = pd.concat([dfviz3,dfviz3tot])

top20ids = ['Total']+top20ids
```

In [ ]:

```
input_dropdown = alt.binding_select(options=top20ids, name='Select ID: ')
selector = alt.selection_point(bind=input_dropdown, fields=['id'],  value='Total')

chart = alt.Chart(dfviz3).mark_line().encode(
    x=alt.X('Timestamp:T', axis=alt.Axis(title='Time')),
    y=alt.Y('count:Q', axis=alt.Axis(title='Count of "from" IDs')),
    color = 'Direction:N'
).add_params(
    selector
).transform_filter(
    selector
).properties(
    width=500,
    height=200,
    title='Count of messages sent to and from the selected ID by Hour'
)

chart
```

Out[ ]:



Select ID: Total ▾

Two specific IDs, namely 1278894 and 839736, have been identified for having a high volume of communication activity. ID 1278894 is observed to send and receive messages to multiple visitors in the park. The communication pattern shows a regular interval of activity for a certain period, followed by a period of no activity.

On the other hand, ID 839736 is observed to communicate with several park visitors throughout the day, with no fixed interval of activity. However, an anomalous pattern is also noticed where there is a spike in activity at Sunday 12 pm.

Visualization 5: Total Number of messages sent and received per hour by location & Total Number of messages sent and received by location

In [ ]:

```
hdl_counts = df.groupby([df["day"],df["hour"],df["location"]]).size().reset_index().rename(columns={0: "Count"})
hdl_counts.head()
```

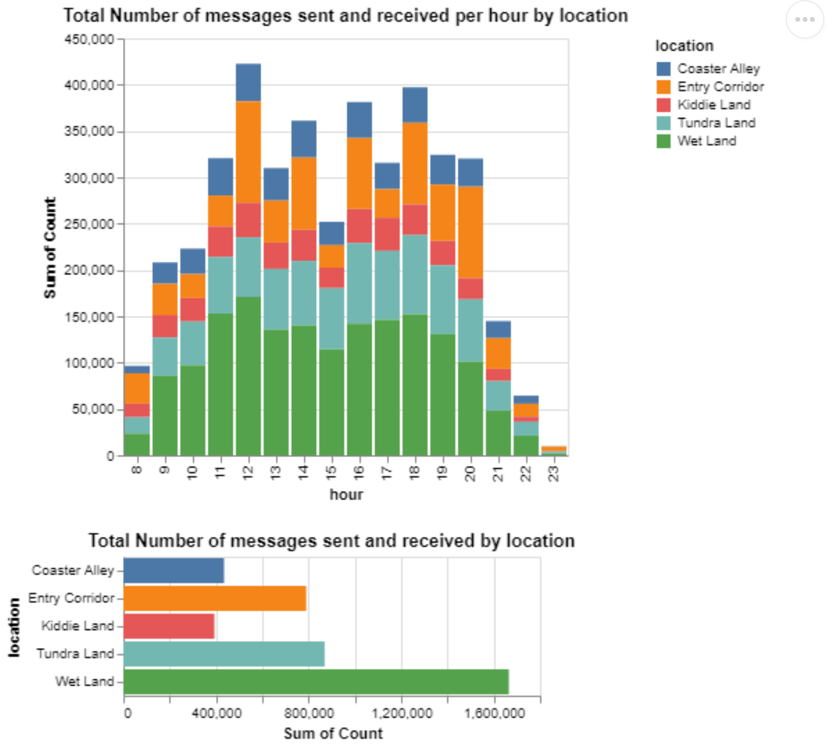
Out[ ]:

	day	hour	location	Count
0	6	8	Coaster Alley	3534
1	6	8	Entry Corridor	4109
2	6	8	Kiddie Land	4047
3	6	8	Tundra Land	6194
4	6	8	Wet Land	6799

In [ ]:

```
click = alt.selection_point(encodings=['y'])

chart1 = alt.Chart(hdl_counts).mark_bar().encode(
    x = 'hour:O',
    y = 'sum(Count)',
    color = 'location:N'
).transform_filter(
    click
).properties(
    title='Total Number of messages sent and received per hour by location'
)

chart2 = alt.Chart(hdl_counts).mark_bar().encode(
    y = 'location:N',
    x = 'sum(Count)',
    color = alt.condition(click,'location:N',alt.value('lightgray'))
).add_params(
    click
).properties(
    title='Total Number of messages sent and received by location'
)

chart1 & chart2
```

Out[ ]:

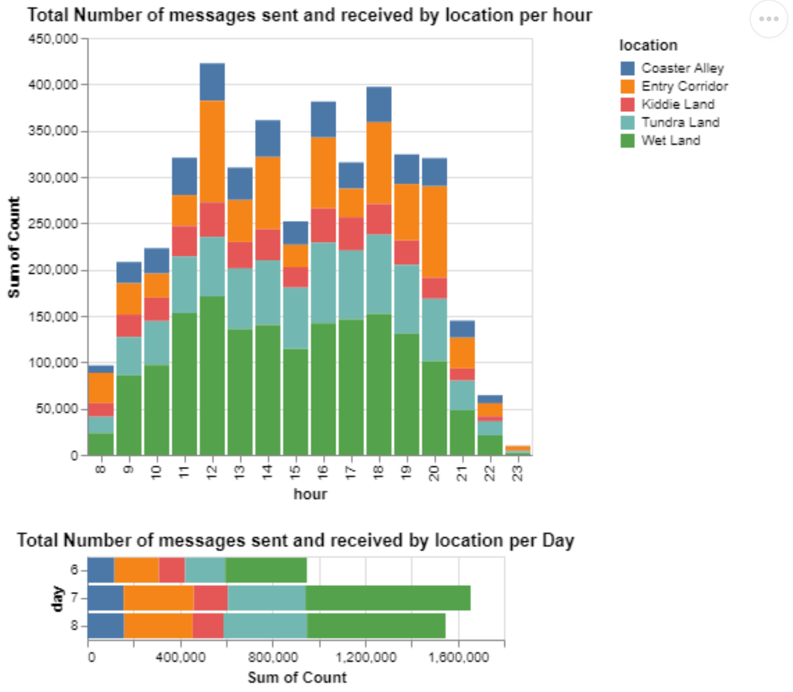


Visualization 6: Total number of messages sent and received per hour and day by Location

In [ ]:

```
click = alt.selection_point(encodings=['y','color'])

chart1 = alt.Chart(hdl_counts).mark_bar().encode(
    x = 'hour:O',
    y = 'sum(Count)',
    color = 'location:N'
).transform_filter(
    click
).properties(
    title='Total Number of messages sent and received by location per hour'
)

chart2 = alt.Chart(hdl_counts).mark_bar().encode(
    y = 'day:O',
    x = 'sum(Count)',
    color = alt.condition(click,'location:N',alt.value('lightgray'))
).add_params(
    click
).properties(
    title='Total Number of messages sent and received by location per Day'
)

chart1 & chart2
```

Out[ ]:



This visulization was to locate where the crime could have occurred. We can see that the entry corridor had an an unusual amount of total messages so we can hypothesize that the crime had occurred near the entry cooridor on Sunday.

Conclusion

It can be inferred from the analysis that the crime took place during the first and only show on Sunday around 12 PM. After the show, visitors increased their communication with external contacts and other IDs (most likely help desks), indicating that the vandalism was discovered soon after the show ended.