Collecting Data

This notebook collects data about MLS matches from 2018 to 2023 (excludes 2020) and weather data for each match.

The notebook uses two functions that are located in gather_data.py:

df_from_fbref: This function uses the requests library to read in a webpage from FBref.com, then uses BeautifulSoup to parse a table that includes every match from a specified MLS season. The table includes the following information:

1. Date
2. Day of the week
3. Round of the season (regular season or a specific playoff round)
4. Home and away team
5. Final score
6. Stadium
7. Attendance
8. Kick off time (local time)

The function also does some processing of the data to prepare it for exploration and modeling:

1. Removes redundant header rows.
2. Splits the score into home and away goals.
3. Converts local time to a decimal.
4. Creates a new column indicating whether a match is the first home match for the home team that season.

Using supplementary data located in mls_stadiums.xlsx, the function does the following: 5. Each team name is converted to a unique integer. 6. The latitude and longitude of the stadium is added to the data. 7. A new column is created that equals attendance divided by stadium capacity. 8. A new column is created that indicates whether a match is a playoff match.

Using supplementary data located in missing_attendance.xlsx, the function does the following: 9. Adds attendance to matches where the attendance is not listed on FBref.com, but is available at the official MLS website. 10. Corrects a couple incorrect stadiums.

Using supplementary data located in mls_rivals.xlsx, the function does the following: 11. Creates a new column that indicates whether the teams in a match are rivals. 12. Creates a new column that indicates whether the teams in a match are from the same conference.

add_weather_info: This function takes the output of df_from_fbref and adds weather information. The weather information is gathered from open-meteo.com which has a convenient API for gathering historical weather information. I use the API to gather the following information:

1. Rainfall

2. Snowfall
3. Temperature
4. Cloud cover
5. Windspeed
6. Windgust speed

The weather measurements have a 1 hour resolution. I choose the hour that precedes the kick-off of each match. The function also records the amount of rain and snow that occurred on the day of each match prior to kick-off.

The data from each season is combined into a single pandas DataFrame that is used in other notebooks

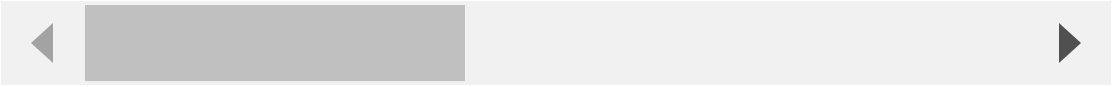
In [1]:
```
import requests
import numpy as np
import pandas as pd
from bs4 import BeautifulSoup
import re
import matplotlib.pyplot as plt
%matplotlib inline
from gather_data import *
```

In [2]:
```
# 2023 Season
mls23_df = add_weather_info(df_from_fbref(year='current'))
mls23_df
```

Out[2]:

	round	day	date	local_time	home_team	home_score	away_score	away_tea
0	Regular Season	Sat	2023-02-25	15.5	15	2	0	
1	Regular Season	Sat	2023-02-25	19.5	4	2	1	
2	Regular Season	Sat	2023-02-25	19.5	7	0	1	
3	Regular Season	Sat	2023-02-25	19.5	0	2	1	
4	Regular Season	Sat	2023-02-25	19.5	20	4	1	
...	...	...	...	...	...	...	...	
531	Round One	Sat	2023-11-11	17.0	9	1	1	
532	Round One	Fri	2023-11-10	19.0	24	1	0	
533	Round One	Sun	2023-11-12	19.0	6	4	2	
534	Conference Semifinals	Sat	2023-11-25	17.5	19	0	2	
535	Conference Semifinals	Sat	2023-11-25	20.0	4	1	0	

516 rows × 26 columns



In [3]:
```
# 2022 Season
mls22_df = add_weather_info(df_from_fbref(year=2022))
mls22_df
```

Out[3]:

	round	day	date	local_time	home_team	home_score	away_score	away_tea
0	Regular Season	Sat	2022-02-26	12.5	11	3	0	
1	Regular Season	Sat	2022-02-26	13.0	20	1	1	
2	Regular Season	Sat	2022-02-26	15.0	23	1	3	
3	Regular Season	Sat	2022-02-26	15.5	6	4	0	
4	Regular Season	Sat	2022-02-26	16.5	7	1	1	
...	...	...	...	...	...	...	...	
503	Conference Semifinals	Sun	2022-10-23	13.0	14	1	3	
504	Conference Semifinals	Sun	2022-10-23	19.0	1	2	1	
505	Conference Finals	Sun	2022-10-30	12.0	11	3	0	
506	Conference Finals	Sun	2022-10-30	20.0	20	3	1	
507	MLS Cup	Sat	2022-11-05	13.0	11	3	3	

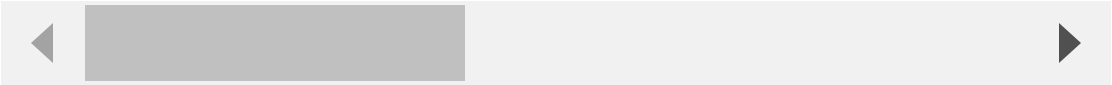
489 rows × 26 columns

In [4]:
```
# 2021 Season
mls21_df = add_weather_info(df_from_fbref(year=2021))
mls21_df
```

Out[4]:

	round	day	date	local_time	home_team	home_score	away_score	away_tea
0	Regular Season	Sat	2021-04-17	14.0	14	4	2	
1	Regular Season	Sat	2021-04-17	15.0	11	2	0	
2	Regular Season	Sat	2021-04-17	15.0	19	0	0	
3	Regular Season	Fri	2021-04-16	18.5	24	4	0	
4	Regular Season	Fri	2021-04-16	19.0	9	2	1	
...	...	...	...	...	...	...	...	
485	Conference Semifinals	Sun	2021-11-28	17.5	20	1	1	
486	Conference Semifinals	Tue	2021-11-30	19.5	16	2	2	
487	Conference Finals	Sat	2021-12-04	15.5	21	2	0	
488	Conference Finals	Sun	2021-12-05	15.0	20	1	2	
489	MLS Cup	Sat	2021-12-11	12.0	21	1	1	

472 rows × 26 columns

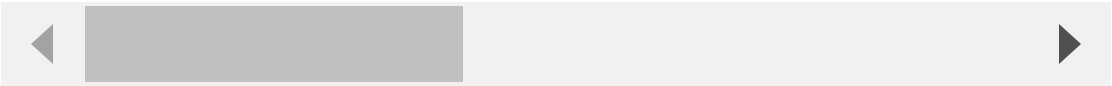


In [5]:
```
# 2019 Season
mls19_df = add_weather_info(df_from_fbref(year=2019))
mls19_df
```

Out[5]:

	round	day	date	local_time	home_team	home_score	away_score	away_te
0	Regular Season	Sat	2019-03-02	13.0	20	1	3	
1	Regular Season	Sat	2019-03-02	14.5	19	2	2	
2	Regular Season	Sat	2019-03-02	15.0	28	2	3	
3	Regular Season	Sat	2019-03-02	15.5	7	1	1	
4	Regular Season	Sat	2019-03-02	16.0	5	3	3	
...	...	...	...	...	...	...	...	...
432	Conference Semifinals	Thu	2019-10-24	19.5	11	5	3	
433	Conference Semifinals	Thu	2019-10-24	20.0	0	2	0	
434	Conference Finals	Tue	2019-10-29	19.0	11	1	3	
435	Conference Finals	Wed	2019-10-30	20.0	0	1	2	
436	MLS Cup 2019	Sun	2019-11-10	15.0	24	3	1	

421 rows × 26 columns



In [6]:
```
# 2018 Season
mls18_df = add_weather_info(df_from_fbref(year=2018))
mls18_df
```

Out[6]:

	round	day	date	local_time	home_team	home_score	away_score	away_tea
0	Regular Season	Sat	2018-03-03	13.0	27	0	2	
1	Regular Season	Sat	2018-03-03	14.5	9	4	0	
2	Regular Season	Sun	2018-03-04	14.0	24	0	1	
3	Regular Season	Sun	2018-03-04	15.0	28	2	1	
4	Regular Season	Sat	2018-03-03	19.0	20	2	0	
...	...	...	...	...	...	...	...	
419	Conference Finals	Sun	2018-11-25	17.0	0	3	0	
420	Conference Finals	Sun	2018-11-25	16.5	21	0	0	
421	Conference Finals	Thu	2018-11-29	19.5	18	1	0	
422	Conference Finals	Thu	2018-11-29	20.5	25	2	3	
423	MLS Cup 2018	Sat	2018-12-08	20.0	0	2	0	

408 rows × 26 columns

In [7]:
```
# Combine each season into a single DataFrame
mlsall_df = pd.concat([mls18_df, mls19_df, mls21_df, mls22_df, mls23_df]

# Create columns for year, month, and day
mlsall_df['date_year'] = mlsall_df['date'].apply(lambda x: int(x[0:4]))
mlsall_df['date_month'] = mlsall_df['date'].apply(lambda x: int(x[5:7]))
mlsall_df['date_day'] = mlsall_df['date'].apply(lambda x: int(x[8:10]))
mlsall_df.reset_index(drop=True, inplace=True)
mlsall_df
```

Out[7]:

	round	day	date	local_time	home_team	home_score	away_score	away_te
0	Regular Season	Sat	2018-03-03	13.0	27	0	2	
1	Regular Season	Sat	2018-03-03	14.5	9	4	0	
2	Regular Season	Sun	2018-03-04	14.0	24	0	1	
3	Regular Season	Sun	2018-03-04	15.0	28	2	1	
4	Regular Season	Sat	2018-03-03	19.0	20	2	0	
...	...	...	...	...	...	...	...	
2301	Round One	Sat	2023-11-11	17.0	9	1	1	
2302	Round One	Fri	2023-11-10	19.0	24	1	0	
2303	Round One	Sun	2023-11-12	19.0	6	4	2	
2304	Conference Semifinals	Sat	2023-11-25	17.5	19	0	2	
2305	Conference Semifinals	Sat	2023-11-25	20.0	4	1	0	

2306 rows × 29 columns

In [8]:
```
# Save data as CSV
mlsall_df.to_csv('mls_with_weather.csv')
```

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
from exploration_visuals import *
```

Data Exploration

In this notebook, I look for patterns in the data using a variety of visualizations (scatterplots, histograms, bar charts, tables). I am primarily looking for how attendance relates to other features, but I will also look for evidence of relations between features.

Read in the data that was created in collect_match_weather_data.ipynb

In [2]:
```
mlsall_df = pd.read_csv('mls_with_weather.csv', index_col=0)
mlsall_df
```

Out[2]:

	round	day	date	local_time	home_team	home_score	away_score	away_tean
0	Regular Season	Sat	2018-03-03	13.0	27	0	2	
1	Regular Season	Sat	2018-03-03	14.5	9	4	0	
2	Regular Season	Sun	2018-03-04	14.0	24	0	1	1
3	Regular Season	Sun	2018-03-04	15.0	28	2	1	1
4	Regular Season	Sat	2018-03-03	19.0	20	2	0	1
...	...	...	...	...	...	...	...	..
2280	Regular Season	Sat	2023-10-21	19.0	5	0	1	2
2281	Regular Season	Sat	2023-10-21	20.0	26	0	2	2
2282	Regular Season	Sat	2023-10-21	20.0	25	3	1	1
2283	Wild Card Round	Wed	2023-10-25	19.5	18	5	2	
2284	Wild Card Round	Wed	2023-10-25	20.5	25	0	0	2

2285 rows × 29 columns

There are some matches that have an attendance of 0. This either means that the attendance was not recorded or no fans were allowed into the stadium due to the COVID pandemic. There is no reason to include these matches, so they are dropped below.

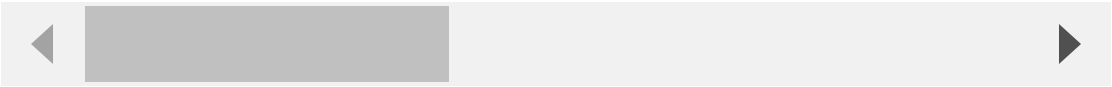
Also, there may be missing weather data for more recent matches. Those matches are also dropped.

In [3]:
```
mlsall_df.dropna(inplace=True)
mlsall_df.drop(mlsall_df[mlsall_df['attendance']==0].index, inplace=Tru
mlsall_df
```

Out[3]:

	round	day	date	local_time	home_team	home_score	away_score	away_team
0	Regular Season	Sat	2018-03-03	13.0	27	0	2	6
1	Regular Season	Sat	2018-03-03	14.5	9	4	0	0
2	Regular Season	Sun	2018-03-04	14.0	24	0	1	11
3	Regular Season	Sun	2018-03-04	15.0	28	2	1	14
4	Regular Season	Sat	2018-03-03	19.0	20	2	0	16
...	...	...	...	...	...	...	...	...
2278	Regular Season	Sat	2023-10-21	18.0	21	1	3	9
2279	Regular Season	Sat	2023-10-21	18.0	28	1	1	11
2280	Regular Season	Sat	2023-10-21	19.0	5	0	1	22
2281	Regular Season	Sat	2023-10-21	20.0	26	0	2	24
2282	Regular Season	Sat	2023-10-21	20.0	25	3	1	12

2229 rows × 29 columns



Below, I look at the datatypes of the columns

In [4]:
```
mlsall_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2229 entries, 0 to 2282
Data columns (total 29 columns):
 #   Column            Non-Null Count  Dtype  
---  ------            --------------  -----  
 0   round             2229 non-null   object 
 1   day               2229 non-null   object 
 2   date              2229 non-null   object 
 3   local_time        2229 non-null   float64
 4   home_team         2229 non-null   int64  
 5   home_score        2229 non-null   int64  
 6   away_score        2229 non-null   int64  
 7   away_team         2229 non-null   int64  
 8   attendance        2229 non-null   int64  
 9   stadium           2229 non-null   object 
 10  latitude          2229 non-null   float64
 11  longitude         2229 non-null   float64
 12  playoff           2229 non-null   int64  
 13  att_div_capacity  2229 non-null   float64
 14  real_home_team    2229 non-null   int64  
 15  same_conf         2229 non-null   int64  
 16  rivals            2229 non-null   int64  
 17  temperature       2229 non-null   float64
 18  rain              2229 non-null   float64
 19  snow              2229 non-null   float64
 20  cloudcover        2229 non-null   float64
 21  windspeed         2229 non-null   float64
 22  windgust          2229 non-null   float64
 23  rain_sum          2229 non-null   float64
 24  snow_sum          2229 non-null   float64
 25  date_year         2229 non-null   int64  
 26  date_month        2229 non-null   int64  
 27  date_day          2229 non-null   int64  
 28  home_opener       2229 non-null   int64  
dtypes: float64(12), int64(13), object(4)
memory usage: 522.4+ KB
```

There are 2,229 matches included and there is no missing data. Aside from round, day, date, and stadium, each column is numerical.

The target variable is att_div_capacity which represents the attendance divided by the capacity of the stadium.

Below, I create a list of just the numerical features. The list will be useful for using the groupby method later.

In [5]:
```
# Make list of numerical features
num_feats = ['attendance','att_div_capacity','local_time','latitude','l
             'snow','cloudcover','windspeed','windgust','rain_sum','snou
num_feats
```

Out[5]:
```
['attendance',
 'att_div_capacity',
 'local_time',
 'latitude',
 'longitude',
 'real_home_team',
 'temperature',
 'rain',
 'snow',
 'cloudcover',
 'windspeed',
 'windgust',
 'rain_sum',
 'snow_sum',
 'playoff',
 'same_conf',
 'rivals',
 'home_opener']
```

In [6]:
```
# Statistical summary of each column
mlsall_df[num_feats].describe()
```

Out[6]:

	attendance	att_div_capacity	local_time	latitude	longitude	real_home
count	2229.000000	2229.000000	2229.000000	2229.000000	2229.000000	2229.0
mean	20816.380888	0.851896	18.332473	38.792813	-93.978528	0.9
std	9397.534830	0.230473	2.084245	5.568967	17.569694	0.0
min	16.000000	0.000627	12.000000	26.190000	-123.110000	0.0
25%	15891.000000	0.729566	18.000000	33.860000	-111.890000	1.0
50%	19267.000000	0.885833	19.500000	39.810000	-87.810000	1.0
75%	22548.000000	1.000000	19.500000	42.090000	-79.420000	1.0
max	82110.000000	2.489441	22.500000	49.280000	-71.260000	1.0

The average attendance for the matches in the dataset is around 20,800. On average, MLS stadiums were 85% full. These numbers are a good baseline to keep in mind as I start slicing the data.

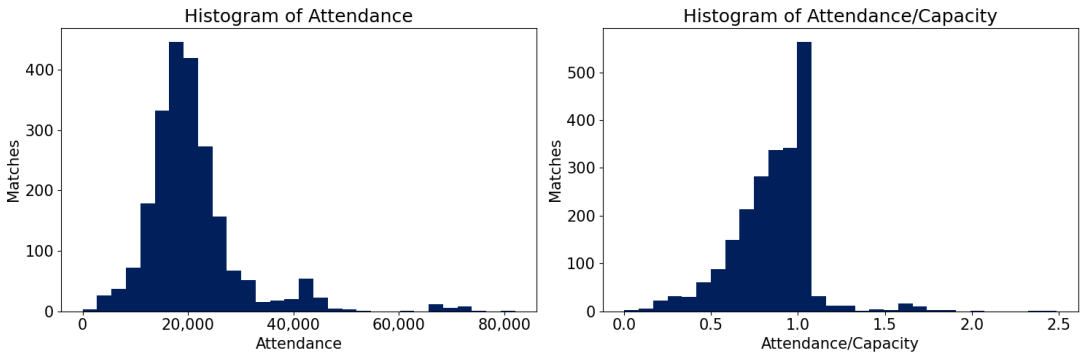
The average temperature at kick off was about 70 degrees Fahrenheit, with the values ranging from 13.7 to 106.3 degrees.

In [7]:
```
# I will use the blue and red from the MLS logo in some figures
mls_blue = '#001F5B'
mls_red = '#DF231A'
```

Target Variable: Attendance

First, I want to look at the distribution of attendance.

In [8]:
```
fig, ax = plt.subplots(ncols=2, figsize=(15,5))

mlsall_df['attendance'].plot.hist(bins=30,
                                  ax=ax[0],
                                  fontsize=15,
                                  color=mls_blue)

ax[0].set_xlabel('Attendance',fontsize=15)
ax[0].set_ylabel('Matches', fontsize=15)
ax[0].set_title('Histogram of Attendance',fontsize=18)
ax[0].set_xticks(np.arange(5)*20000)
ax[0].set_xticklabels(['0','20,000','40,000','60,000','80,000'])

mlsall_df['att_div_capacity'].plot.hist(bins=30,
                                        ax=ax[1],
                                        fontsize=15,
                                        color=mls_blue)

ax[1].set_xlabel('Attendance/Capacity',fontsize=15)
ax[1].set_ylabel('Matches', fontsize=15)
ax[1].set_title('Histogram of Attendance/Capacity',fontsize=18)

fig.tight_layout()
```



I have two choices for the target variable for this project: raw attendance or attendance divided by stadium capacity. Different teams have different sized stadiums, so something must be done to account for this. Using attendance divided by capacity can fix it, but I also plan on using separate parameters for each home team in the model which could also account for these differences. It will be much easier to handle raw attendance when creating the models, so I plan on using raw attendance.

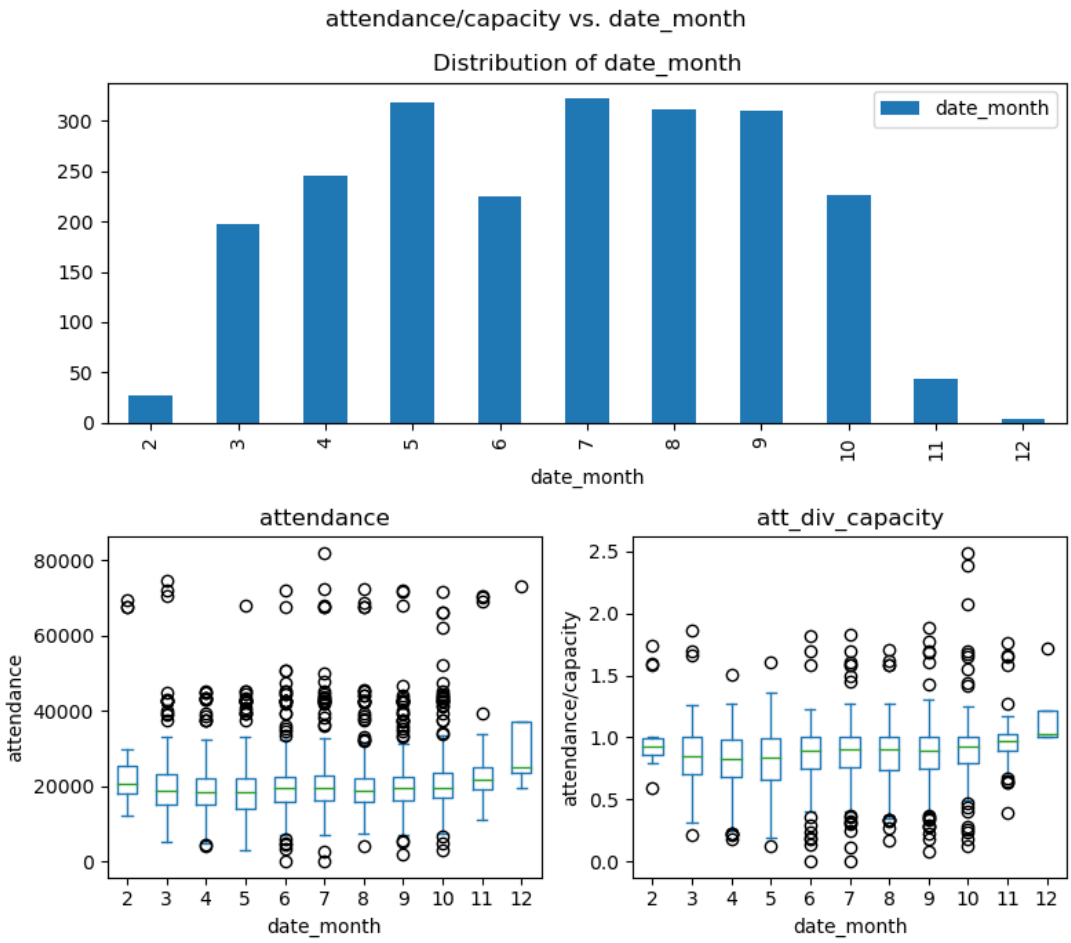
One might think that the attendance/capacity could not exceed 1 (how could the attendance be 1.5 times the capacity?), but there are a fair number of matches with values above 1. The reason for this is that some teams artifically reduce the capacity of their stadium most of the time, but on special occasions, they will use the full capacity. An example of this is the Seattle Sounders who share Lumen Field with the Seattle Seahawks of the NFL. The actual capacity of the stadium is 72,000, but the Sounders typically cap attendance at just below 38,000. However, the Sounders will use the full stadium for special occasions like playoff games or matches against rivals.

I considered rounding every value above 1 down to 1, but decided against it. The reason to round down would be to correct for the changing capacity of some stadiums from match to match. However, I figured that a match should get credit for its inflated attendance because there was likely a compelling reason why the capacity was increased. Therefore, I will leave

Attendance vs. Month

Below, I look at the distribution of matches by month and the attendance by month.

In [9]:
```
make_bar_box(mlsall_df, 'date_month', figsize=(8,7))
```



Most matches are played between March and October. A smaller number of matches are played in February, November, and December.

The reason fewer matches have been in February is because MLS currently starts the season in late February, so they don't use the whole month. In fact, MLS used to start the season in March, avoiding February entirely.

The reason there are fewer matches in November and December is because this is when the playoffs are held.

In [10]:
```
# Group by month
mlsall_df.groupby('date_month').agg({x:'mean' for x in num_feats})
```

Out[10]:

date_month	attendance	att_div_capacity	local_time	latitude	longitude	real_hom
2	26163.148148	0.979038	17.240741	36.310000	-95.855185	1
3	20765.015152	0.836999	17.329545	38.498485	-93.147727	0
4	19322.179592	0.802260	17.865306	38.843837	-92.648245	0
5	19150.311321	0.783570	18.143082	38.701730	-94.029780	0
6	21328.648889	0.858731	18.742222	39.184400	-94.173733	0
7	21225.347826	0.870895	19.078934	38.811304	-94.250280	0
8	20640.752412	0.857170	19.146034	38.742958	-93.691768	0
9	20882.954839	0.867673	18.592204	38.948742	-94.815548	0
10	22139.203540	0.903899	17.486357	38.535531	-93.726991	0
11	25237.255814	0.993807	16.906977	40.271395	-97.451395	1
12	35735.500000	1.192861	15.625000	41.155000	-101.290000	1

The highest attendance values occur in February, November, and December. The attendance is likely high in February because people are excited for the new season. It is also possible that this is just due to a small sample size since there are not that many matches in February and two of them happened to have very high attendance.

Attendance is high in November and December because that is when the playoffs are happening and the playoffs are the most exciting part of the season. In fact, the attendance in December is 119% capacity. This is because MLS Cup, the championship match, is held in December. Since that match decides the champion, there is high demand for tickets and a stadium's full capacity will be used rather than the artifically reduced capacity.

The attendance in March through May was below the average of 85% while the attendance from June to October was above the average. This seems to indicate that the attendance improves as the season progresses.

The average temperature at kick-off was pretty low for November and December (47.9 and 40.9 degrees Fahrenheit, respectively), but the attendance was still very good. This shows that people are willing to deal with low temperatures to attend playoff matches.

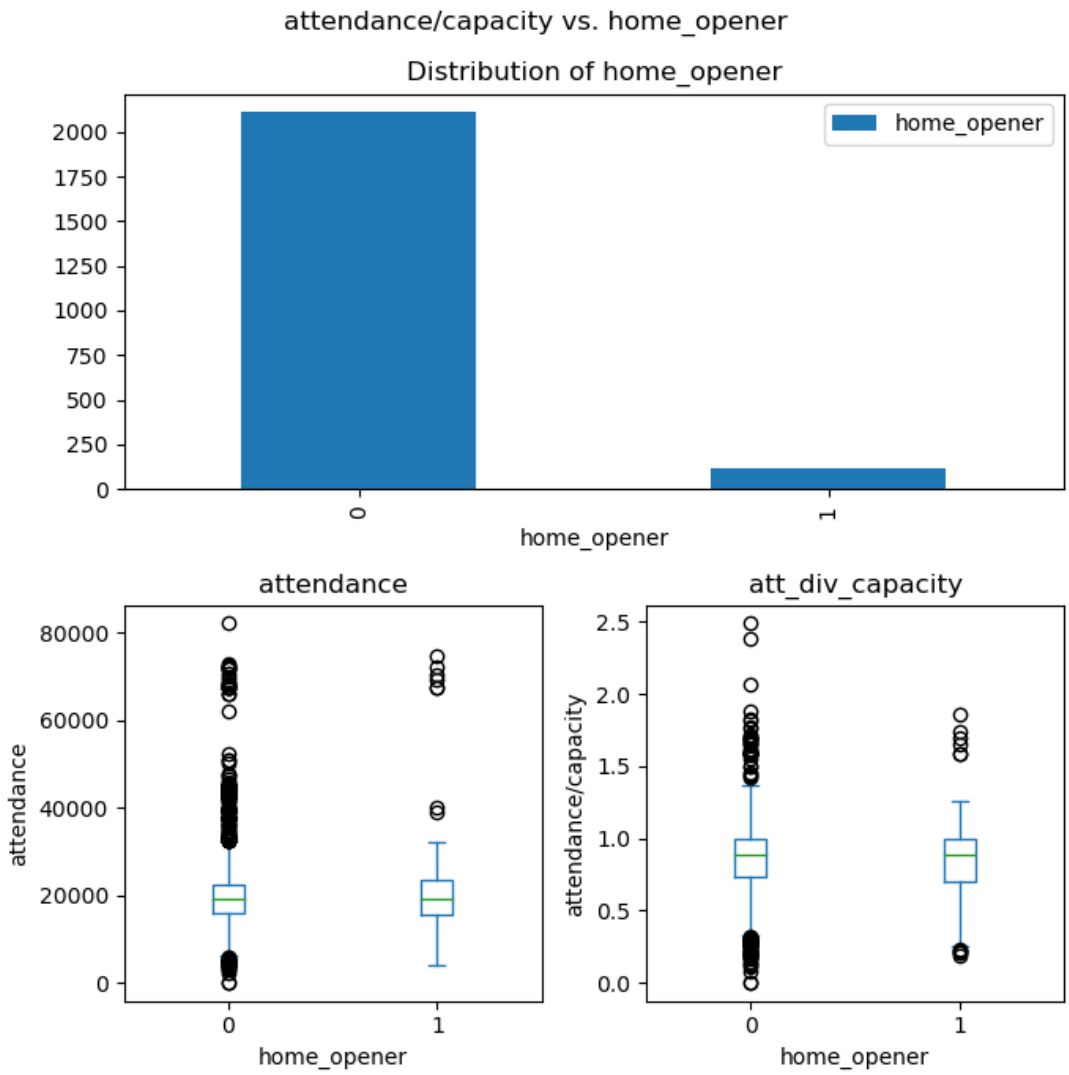
There are also a couple interesting trends I saw in the table that are not directly related to attendance.

1. The kick-off time (local_time) is earlier at the beginning and end of the season, but later in the middle of the season. I think the most likely reason for this is that this helps manage temperature. During the winter, MLS has earlier kick-off times because it gets too cold later in the night. During the summer, the kick-off times are later to allow for the temperature to drop to a more comfortable level. Importantly, MLS abandoned this strategy in 2023 by making kick-off times more consistent (see next cell).
2. The average longitude in February was about 2 degrees lower than the other months in which regular season games are played (March through October). This might be due to smaller sample size (because fewer games are played in February), but it could also be intentional. By playing February matches at locations at lower latitudes, MLS avoids playing matches in very cold conditions (Canada winters can be brutal). In fact, MLS used to start the season in mid-March to avoid playing in February altogether, but they have been forced to start earlier because they have to play so many matches between MLS and other competitions.

Attendance in Home Openers

Above, I saw that matches in February had higher attendance than any other month in which regular season matches are played. I hypothesized that the reason for this is that home openers (first home game of the season) have higher attendance because fans are excited for the start of the season. The 'home_opener' column indicates whether a match was a home opener. Let's check if that correlates with a change in attendance.

In [11]:
```
make_bar_box(mlsall_df, 'home_opener')
```



In [12]:
```
# Compare home openers to non-home openers
mlsall_df[mlsall_df['playoff']==0].groupby('home_opener').agg({x:'mean'
```

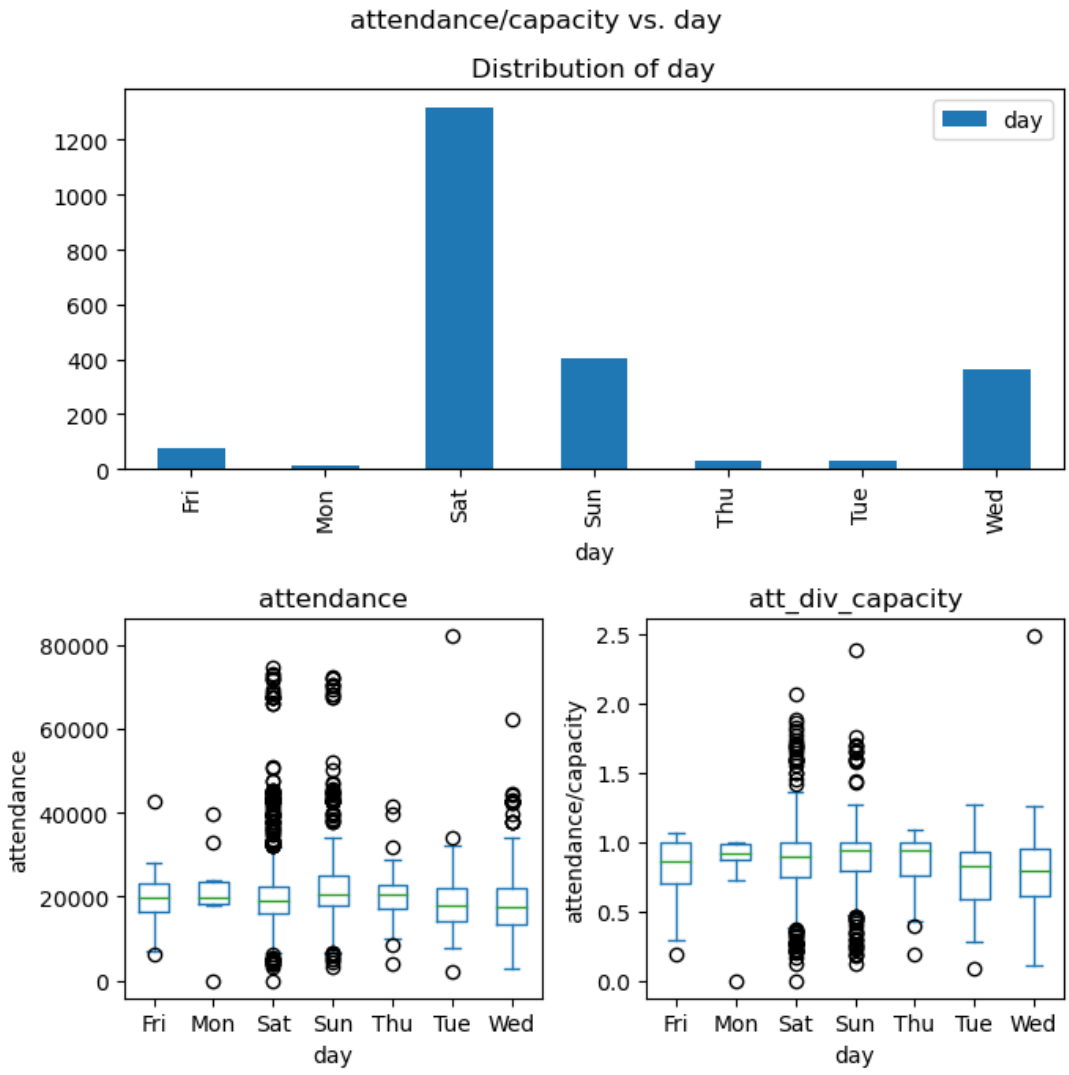
Out[12]:

home_opener	attendance	att_div_capacity	local_time	latitude	longitude	real_hor
0	20623.130604	0.848363	18.442779	38.780599	-93.961852	
1	21330.404959	0.844244	17.008264	38.668182	-93.633719	

Home openers averaged 21,330 (84.4%) attendance while other regular season matches averaged 20,623 (84.8%). There does not appear to be a strong trend here.

Attendance vs. Day of the Week

In [13]:
```
make_bar_box(mlsall_df, 'day')
```



Most matches have been held on Saturdays with Sundays and Wednesdays the next most frequent. MLS used to hold a single match on Fridays, but they abandoned that for the 2023 season. A small number of matches have been held on Mondays, Tuesdays, and Thursdays.

It looks as though matches held on weekends have higher attendance. Let's confirm that below by looking at the mean values for each day.

In [14]:
```
# Group by day
mlsall_df.groupby('day').agg({x:'mean' for x in num_feats})
```

Out[14]:

day	attendance	att_div_capacity	local_time	latitude	longitude	real_home_team
Fri	19757.041096	0.825588	19.450913	38.783151	-98.473562	1.000000
Mon	21423.800000	0.825168	18.950000	39.553000	-101.327000	0.900000
Sat	20729.835481	0.865025	18.397207	38.888688	-93.331092	0.994693
Sun	23227.210918	0.899636	16.704715	38.805037	-95.238809	0.987593
Thu	20889.000000	0.853490	19.453125	36.927500	-95.019688	1.000000
Tue	20713.600000	0.765152	19.483333	37.002000	-98.483333	0.966667
Wed	18646.784530	0.764002	19.471685	38.724116	-93.359724	0.991713

The best percentage attendance was on Sundays where the attendance was at 90% capacity on average. Saturday has the second highest attendance at 86%. Mondays and Fridays both have 82.5% attendance on average while Tuesdays and Wednesdays have 76.5%, far below average.

In terms of raw attendance, Mondays outperformed Saturdays, but that might be due to the low sample size of Monday matches.

Another interesting trend is that matches on Sundays started about an hour and a half earlier than matches on Saturdays. This might indicate that earlier matches could be a good thing if it is not a school/work day.

Does Attendance on Sundays decline when NFL starts?

The main reason MLS plays out of sync with European soccer is to avoid cold winter weather, but another argument that has been made is that MLS wants to avoid going head-to-head with the NFL as much as possible. The worry is that MLS attendance would tank if sports fans could go to NFL games instead of MLS matches. Below, I check that assumption by looking at the attendance of Sunday MLS regular season matches before and after the NFL season starts. NFL usually starts in early September, so I will use that as the cutoff. I only include regular season matches because the playoffs would boost attendance and bias the comparison.

In [15]:
```
# Sunday, regular season matches before NFL starts
mlsall_df.loc[(mlsall_df['date_month']<9)&\
              (mlsall_df['day']=='Sun')&\
              (mlsall_df['playoff']==0)].agg({'attendance':'mean','att_
```

Out[15]:
```
attendance          23683.338843
att_div_capacity        0.870035
day                   242.000000
dtype: float64
```

In [16]:
```
# Sunday, regular season matches after NFL starts
mlsall_df.loc[(mlsall_df['date_month']>=9)&\
              (mlsall_df['day']=='Sun')&\
              (mlsall_df['playoff']==0)].agg({'attendance':'mean','att_
```

Out[16]:
```
attendance          21877.615942
att_div_capacity        0.926718
day                   138.000000
dtype: float64
```

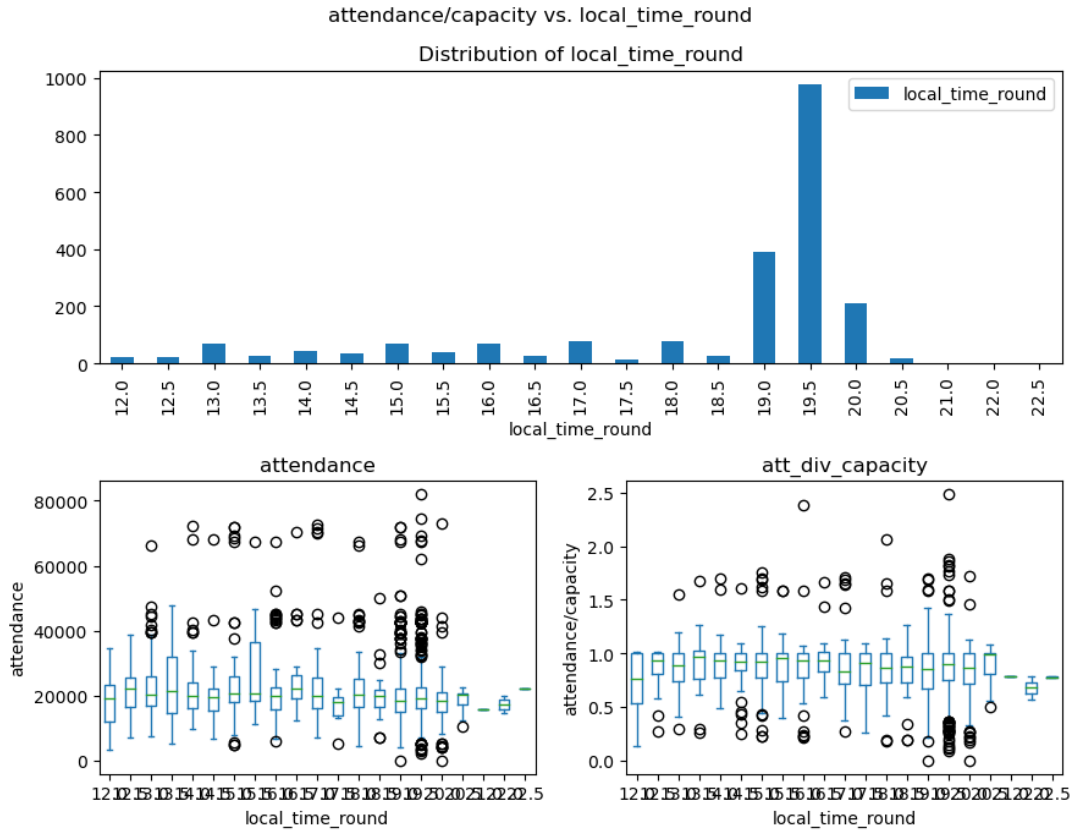
The raw and percentage attendance figures tell different stories.

The raw attendance is about 1800 people larger prior to the NFL starting, reinforcing the belief that competing with the NFL is harmful to attendance. On the other hand, the percentage attendance is 92.7% after the NFL starts and 87.0% before the NFL starts. So the reduction in raw attendance might be down to the fact that the matches held on Sundays during the NFL season just happen to be in stadiums with lower capacities (or maybe that is intentional).

Attendance vs. Kick Off Time

In [17]:
```
mlsall_df['local_time_round'] = mlsall_df['local_time'].apply(lambda x:
```

In [18]:
```
make_bar_box(mlsall_df, 'local_time_round', figsize=(9,7))
```



In [19]:
```
# Group by kick off time
mlsall_df.groupby('local_time_round').agg({x:'mean' for x in num_feats}
```

Out[19]:

local_time_round	attendance	att_div_capacity	local_time	latitude	longitude	real
12.0	18508.750000	0.725057	12.000000	40.707500	-99.953333	
12.5	22142.428571	0.843898	12.500000	39.488095	-108.310000	
13.0	23458.985915	0.853092	12.996479	41.096761	-94.143099	
13.5	23687.821429	0.900725	13.500000	40.979643	-100.031429	
14.0	23577.232558	0.905007	13.998062	39.624186	-101.294884	
14.5	20275.000000	0.870446	14.500000	37.918919	-95.589459	
15.0	24096.757143	0.905347	15.000000	39.048286	-91.895571	
15.5	26998.717949	0.918623	15.500000	38.340000	-85.632051	
16.0	22503.239437	0.878904	15.998826	38.840000	-92.995070	
16.5	26200.333333	0.951799	16.500000	38.466667	-99.596296	
17.0	23394.384615	0.865658	16.994658	38.632051	-91.985000	
17.5	18675.230769	0.841329	17.500000	37.693846	-92.087692	
18.0	22616.282051	0.860268	18.000000	38.652564	-90.963718	
18.5	20064.185185	0.809653	18.500000	39.564444	-100.783333	
19.0	20123.664103	0.813698	19.000641	40.409308	-99.099949	
19.5	20262.958035	0.861027	19.500000	38.391699	-92.099969	
20.0	18435.518868	0.819807	20.000786	36.926981	-91.762311	
20.5	18424.166667	0.892198	20.500000	33.301111	-93.997222	
21.0	15655.000000	0.782750	21.000000	38.870000	-77.010000	
22.0	17267.500000	0.677157	22.000000	28.540000	-81.390000	
22.5	22085.000000	0.778985	22.500000	43.630000	-79.420000	

The vast majority of matches started between 7 pm and 8 pm local time. However, there are a fair number of matches that started earlier than 7 pm and the attendance figures are not bad. Each kick off time from 1 pm to 5 pm had an average attendance at or above 85%, which is the average over the entire dataset.

The kick off times with below average attendance are mostly between 5:30 and 10:30 pm. Matches starting at noon also performed quite poorly, only achieving around getting 18,500 people (72.5%) on average.

The very late kick off times (9 pm or later) had particularly poor attendance, but there are two caveats to this. There are only 5 matches that started this late, so the sample size is small. Also, I expect that MLS did not originally intend for these matches to start so late. Four of the matches had rain and the fifth had snow. This indicates to me that there was a

In [20]:
```
# Matches that started at 9 pm or later
mlsall_df[mlsall_df['local_time_round']>=21]
```

Out[20]:

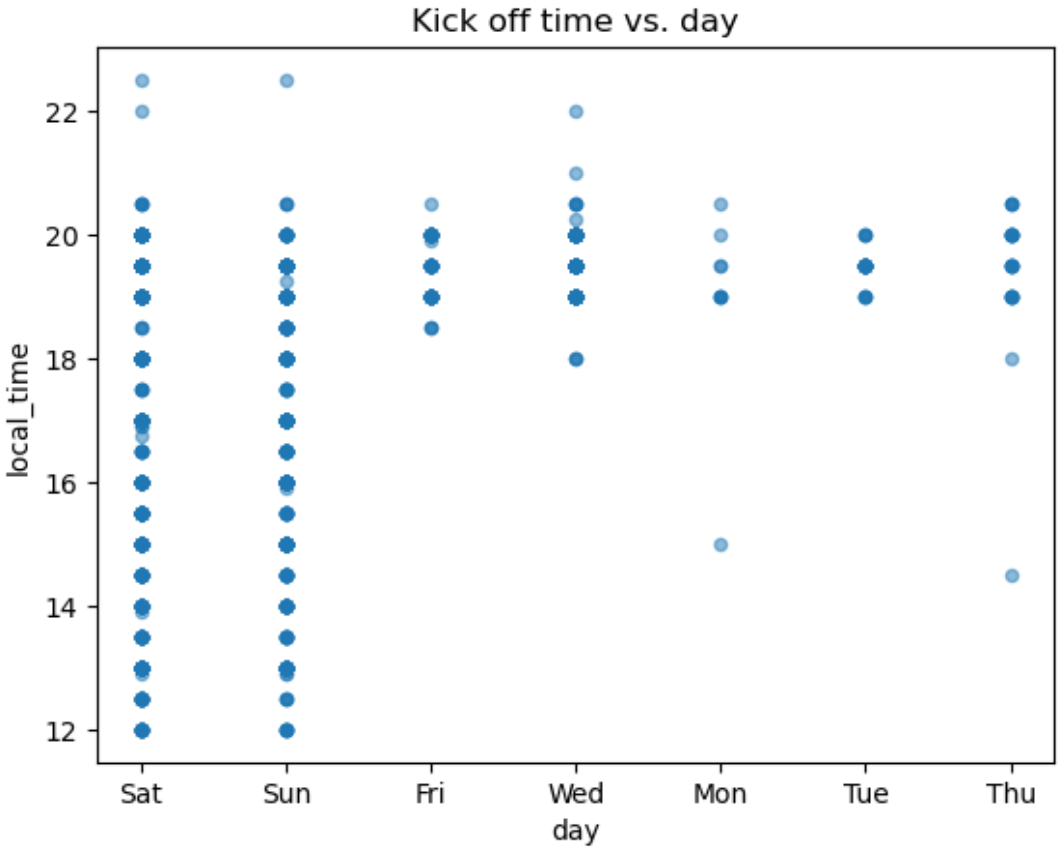
	round	day	date	local_time	home_team	home_score	away_score	away_tean
233	Regular Season	Wed	2018-07-25	21.0	8	0	1	18
1563	Regular Season	Sat	2022-07-09	22.0	19	1	0	13
1693	Regular Season	Wed	2022-08-31	22.0	19	3	2	24
1830	Regular Season	Sun	2023-03-12	22.5	11	4	0	16
1858	Regular Season	Sat	2023-03-25	22.5	11	2	1	1

5 rows × 30 columns

Kick off time vs. day

In [21]:
```
# Plot of kick off times vs. day
mlsall_df.plot.scatter(x='day',
                       y='local_time',
                       title='Kick off time vs. day',
                       alpha=0.5)
```

Out[21]:
```
<Axes: title={'center': 'Kick off time vs. day'}, xlabel='day', ylabel
='local_time'>
```



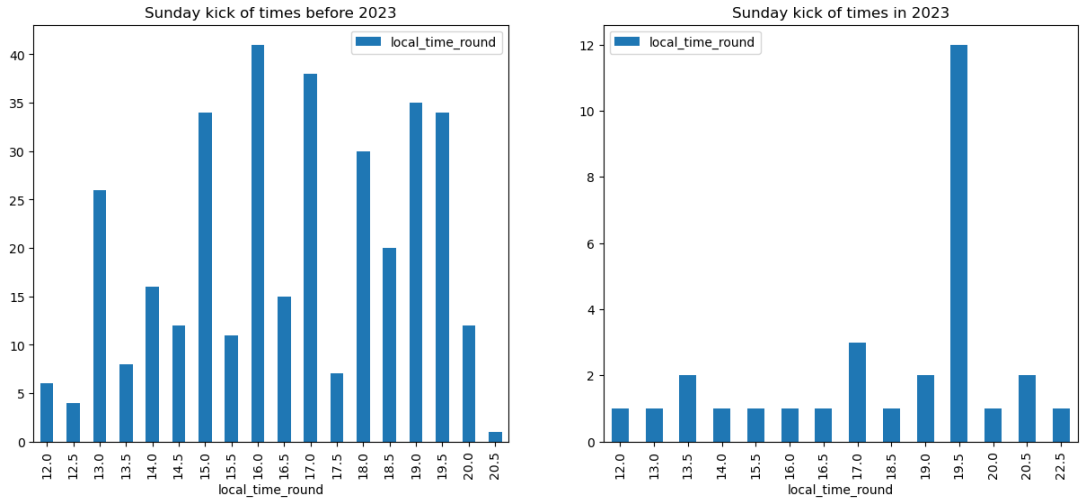
Early kick offs (between 12 and 17, or noon to 5 pm), basically only happened on weekends. Only two matches were played before 6 pm on weekdays.

Did adjustment to kick off times in 2023 help attendance?

Prior to 2023, MLS used to have a lot of variability in kick off times, but this past season, they tried to standardize it more by having most matches start at 7:30 pm local time. This can be seen in the distribution of matches played on Sundays, in particular, which I show below.

In [22]:
```
fig, ax = plt.subplots(ncols=2, figsize=(15,6))

mlsall_df[(mlsall_df['date_year']!=2023)&\
          (mlsall_df['day']=='Sun')&\
          (mlsall_df['playoff']==0)].\
groupby('local_time_round').agg({'local_time_round':'count'}).plot.bar(a

ax[0].set_title('Sunday kick of times before 2023')

mlsall_df[(mlsall_df['date_year']==2023)&\
          (mlsall_df['day']=='Sun')&\
          (mlsall_df['playoff']==0)].\
groupby('local_time_round').agg({'local_time_round':'count'}).plot.bar(a

ax[1].set_title('Sunday kick of times in 2023')
```

Out[22]: Text(0.5, 1.0, 'Sunday kick of times in 2023')



Between 2018 and 2022 (excluding 2020), Sunday kick off times were all over the place. The most common kick off time was 4 pm.

In 2023, 40% of Sunday matches started at 7:30 pm, which was by far the most common kick off time. Did this improve attendance?

In [23]:
```
# Sunday 2018,2019,2021,2022 attendance
mlsall_df.loc[(mlsall_df['date_year']!=2023)&(mlsall_df['day']=='Sun'),
              ['attendance','att_div_capacity','local_time']].describe(
```

Out[23]:

	attendance	att_div_capacity	local_time
count	373.000000	373.000000	373.000000
mean	23244.954424	0.902015	16.603217
std	11544.229492	0.254090	2.208159
min	3219.000000	0.128760	12.000000
25%	17704.000000	0.795411	15.000000
50%	20738.000000	0.940867	16.500000
75%	25117.000000	1.003633	18.500000
max	72243.000000	2.386024	20.500000

In [24]:
```
# Sunday 2023 attendance
mlsall_df.loc[(mlsall_df['date_year']==2023)&(mlsall_df['day']=='Sun'),
              ['attendance','att_div_capacity','local_time']].describe(
```

Out[24]:

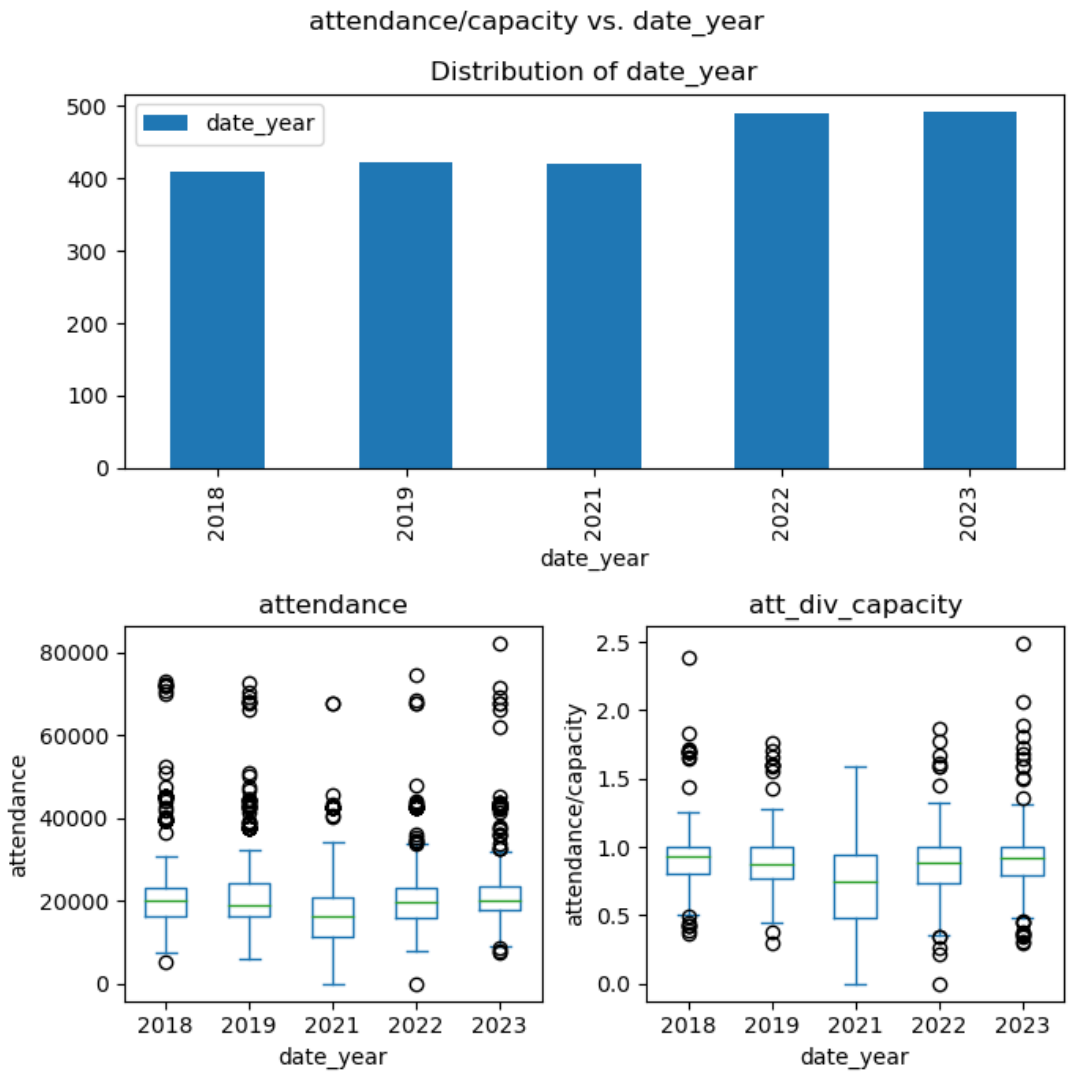
	attendance	att_div_capacity	local_time
count	30.00000	30.000000	30.000000
mean	23006.60000	0.870052	17.966667
std	7701.90772	0.194867	2.619336
min	7417.00000	0.296680	12.000000
25%	18537.00000	0.762492	16.625000
50%	21372.50000	0.983598	19.500000
75%	26747.25000	1.005186	19.500000
max	43527.00000	1.046541	22.500000

The average attendance for Sunday matches between 2018 and 2022 was 23,245 (90.2%). In 2023, when the average kick off time was about an hour and 20 minutes later, the average was 23,006 (87.0%). The percentages would seem to indicate that the move to later kick offs made attendance a little worse.

Attendance by Year

This dataset contains 5 MLS seasons: 2018, 2019, 2021, 2022, and 2023. The 2020 MLS season is not being used for this analysis because almost the entire season was played without fans in attendance due to the COVID-19 pandemic.

In [25]:
```
make_bar_box(mlsall_df, 'date_year')
```



In [26]:
```
# Group by year
mlsall_df.groupby('date_year').agg({x:'mean' for x in num_feats})
```

Out[26]:

date_year	attendance	att_div_capacity	local_time	latitude	longitude	real_home_t
2018	22174.053922	0.910684	17.938930	39.580637	-94.942990	1.00
2019	21722.232779	0.877872	18.045131	39.589406	-94.701876	0.99
2021	16846.233890	0.707067	18.356205	38.039475	-94.076181	0.98
2022	20977.822086	0.860713	18.001534	38.381329	-93.384110	0.99
2023	22136.000000	0.895493	19.213415	38.508394	-93.067398	0.99

The best attendance numbers, both in terms of percentage and raw value, was 2018 when the average attendance was 22,174 (91%).

The worst year for attendance was 2021 when the average attendance was 16,846 (70.7%). A big part of this was the pandemic. Even though I removed matches that had no fans, the dataset does include some matches where the attendance was artifically reduced for safety reasons and I think it took people a while to get comfortable to return to large gatherings.

Attendance for Conference vs. Non-Conference Matches

MLS is split into a Western and an Eastern Conference. Teams play most of their matches against teams from the same conference. The 'same_conf' column says whether the teams in a given match are from the same conference. Let's see if their is a difference in attendance between conference and non-conference matches. I will only look at regular season matches.

In [27]:
```
# Conference matches
mlsall_df[(mlsall_df['same_conf']==1)&\
          (mlsall_df['playoff']==0)][['attendance','att_div_capacity']]
```

Out[27]:
```
attendance          20525.866707
att_div_capacity        0.842100
dtype: float64
```

In [28]:
```
# Non-conferences matches
mlsall_df[(mlsall_df['same_conf']==0)&\
          (mlsall_df['playoff']==0)][['attendance','att_div_capacity']]
```

Out[28]:
```
attendance          21119.736000
att_div_capacity        0.868322
dtype: float64
```

Non-conference matches had an average attendance of 21,120 (86.8%) compared to 20,526 (84.2%) for conference matches. This might seem to indicate that non-conference matches are better attended, if only slightly. However, I think something else is going on. In 2021, the season in which attendance was low because of the pandemic, 94.5% of matches were conference matches. In other seasons, that value ranges from 67% to 81%. I think this is biasing the attendance of conference matches to look lower than it would otherwise. Let's look at the numbers without 2021 included.

In [29]:
```
# Conference matches (excluding 2021)
mlsall_df[(mlsall_df['same_conf']==1)&\
          (mlsall_df['playoff']==0)&\
          (mlsall_df['date_year']!=2021)][['attendance','att_div_capaci
```

Out[29]:
```
attendance          21697.052754
att_div_capacity        0.885558
dtype: float64
```

In [30]:
```
# Non-conference matches (excluding 2021)
mlsall_df[(mlsall_df['same_conf']==0)&\
          (mlsall_df['playoff']==0)&\
          (mlsall_df['date_year']!=2021)][['attendance','att_div_capaci
```

Out[30]:
```
attendance          21283.391213
att_div_capacity        0.875153
dtype: float64
```

If 2021 is removed, then conference games are slightly better attended. The difference between the two values is quite small, indicating that there likely isn't a significant effect on attendance.

Does attendance improve for rivalry matches?

One might expect attendance to be higher when rivals play one another. Let's see if that is true.

In [31]:
```
# Rivals playing
mlsall_df[(mlsall_df['rivals']==1)&(mlsall_df['playoff']==0)][num_feats
```

Out[31]:

	attendance	att_div_capacity	local_time	latitude	longitude	real_home_te
count	176.000000	176.000000	176.000000	176.000000	176.000000	17
mean	22485.062500	0.869651	18.385890	39.261705	-101.197727	
std	9794.993923	0.194465	2.188235	6.212876	18.972539	
min	5000.000000	0.266407	12.000000	26.190000	-123.110000	
25%	16658.000000	0.770218	18.500000	33.860000	-122.010000	
50%	20738.000000	0.909840	19.500000	39.970000	-97.720000	
75%	25389.250000	1.000000	19.500000	45.520000	-81.390000	
max	82110.000000	1.270250	22.000000	49.280000	-73.550000	

In [32]:
```
# Not rivals playing
mlsall_df[(mlsall_df['rivals']==0)&(mlsall_df['playoff']==0)][num_feats
```

Out[32]:

	attendance	att_div_capacity	local_time	latitude	longitude	real_home
count	1997.000000	1997.000000	1997.000000	1997.000000	1997.000000	1997.0
mean	20501.888833	0.846237	18.360875	38.731387	-93.304256	0.9
std	9084.442176	0.231042	2.040484	5.525733	17.247786	0.0
min	16.000000	0.000627	12.000000	26.190000	-123.110000	0.0
25%	15621.000000	0.726293	18.000000	33.860000	-104.890000	1.0
50%	19096.000000	0.879176	19.500000	39.810000	-87.620000	1.0
75%	22423.000000	1.000000	19.500000	42.090000	-79.420000	1.0
max	74479.000000	2.489441	22.500000	49.280000	-71.260000	1.0

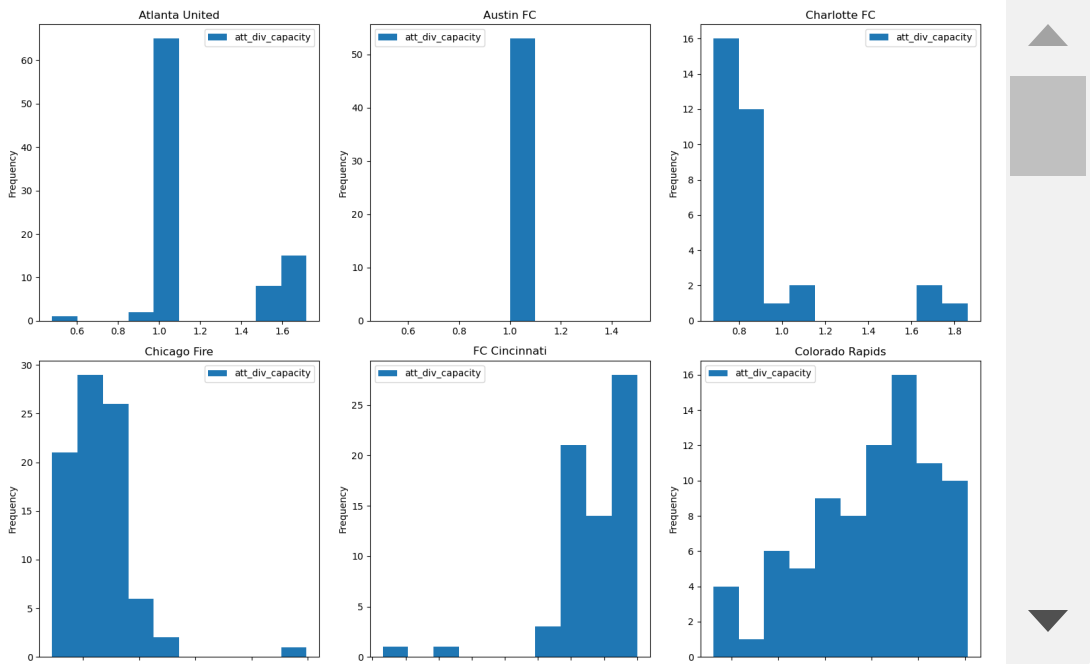
The attendance for rivalry games is 22,485 (87.0%). For non-rivalry games, it is 20,502 (84.6%). This indicates that rivalry games are better attended.

Attendance by Team

Below, I look at the distribution of attendance for each team. For the sake of labeling, I create a dictionary that changes the team ID number into the team name.

In [33]:
```
team_names = {0: 'Atlanta United',
              1: 'Austin FC',
              2: 'Charlotte FC',
              3: 'Chicago Fire',
              4: 'FC Cincinnati',
              5: 'Colorado Rapids',
              6: 'Columbus Crew',
              7: 'FC Dallas',
              8: 'D.C. United',
              9: 'Houston Dynamo',
              10: 'Los Angeles Galaxy',
              11: 'Los Angeles FC',
              12: 'Minnesota United',
              13: 'Inter Miami',
              14: 'CF Montreal',
              15: 'Nashville SC',
              16: 'New England Revolution',
              17: 'New York City FC',
              18: 'New York Red Bulls',
              19: 'Orlando City',
              20: 'Philadelphia Union',
              21: 'Portland Timbers',
              22: 'Real Salt Lake',
              23: 'San Jose Earthquakes',
              24: 'Seattle Sounders',
              25: 'Sporting Kansas City',
              26: 'St. Louis FC',
              27: 'Toronto FC',
              28: 'Vancouver Whitecaps'}
```

In [34]:
```
# Histograms of attendance/capacity for each home team
attendance_histograms(mlsall_df, 'home_team', label_dict=team_names)
```



First, there are a couple teams who have sold out every home match: Austin FC and St. Louis FC. It might be worth excluding their home matches from the analysis because there are no trends to be found.

The following teams have matches at above 100% capacity: Atlanta United, Charlotte FC, Chicago Fire, Minnesota United, New England Revolution, New York Red Bulls, Seattle Sounders, Sporting Kansas City, and Vancouver Whitecaps.

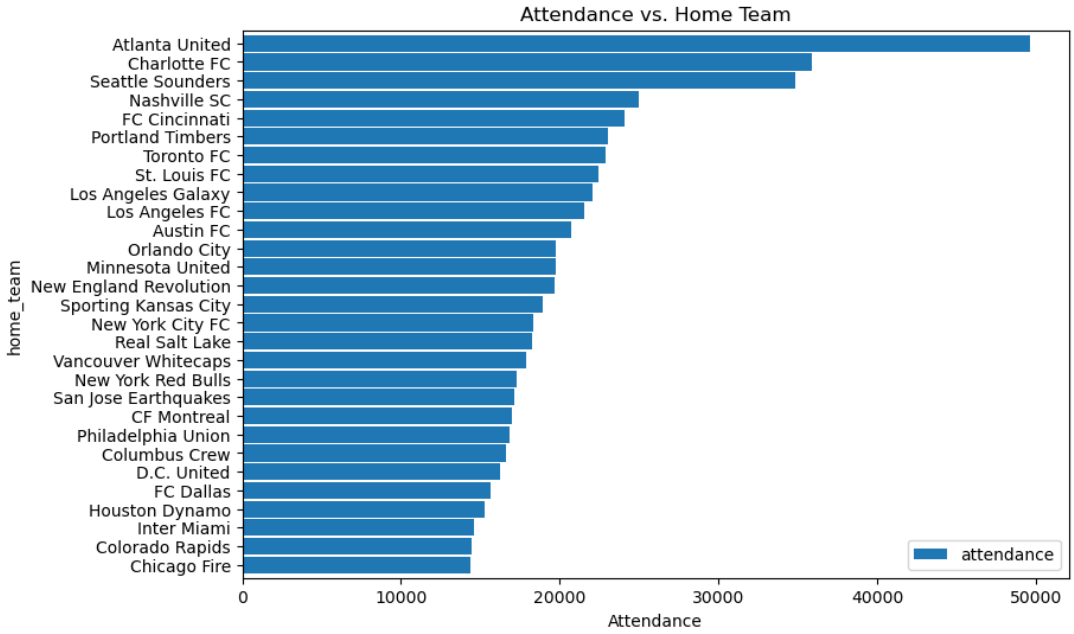
Besides the two teams that always sell out their home matches, there are some teams that nearly sell out each match: Atlanta United, Los Angeles FC, Minnesota United, and Portland Timbers.

In [35]:
```
# Group by home team
mlsall_df.groupby('home_team').agg({x:'mean' for x in num_feats})
```

Out[35]:

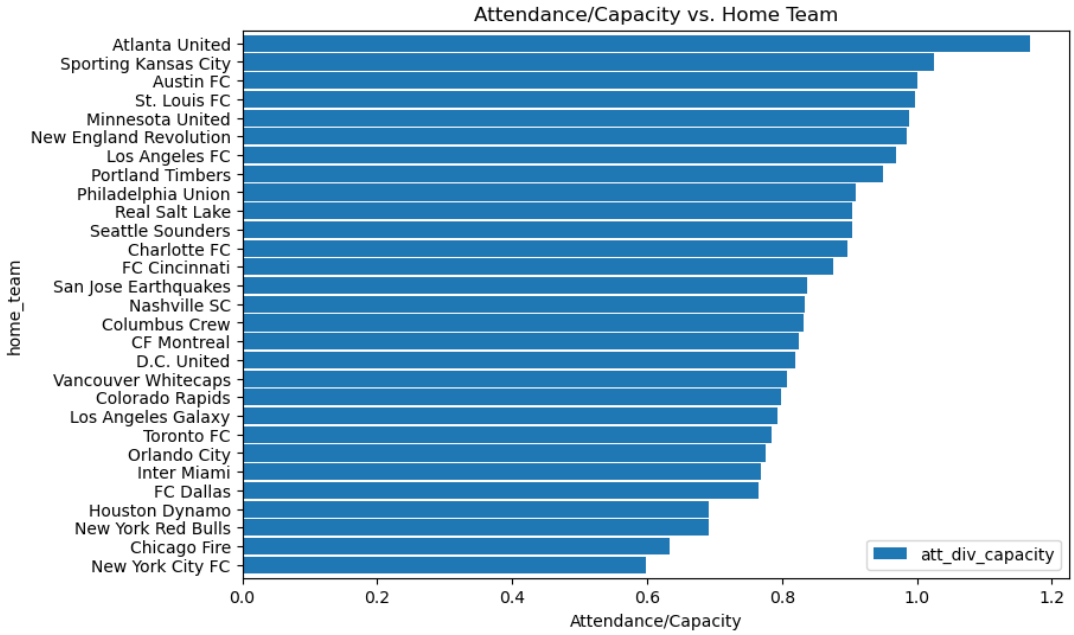
home_team	attendance	att_div_capacity	local_time	latitude	longitude	real_hom
0	49617.934066	1.167481	17.444139	33.750000	-84.400000	1
1	20738.000000	1.000000	19.136792	30.390000	-97.720000	1
2	35916.382353	0.897910	18.279412	33.230000	-81.020000	1
3	14370.211765	0.633156	17.764706	41.817647	-87.700471	1
4	24091.191176	0.875644	18.845588	39.115000	-84.520000	1
5	14413.036585	0.798020	18.524390	39.810000	-104.890000	1
6	16643.406977	0.832355	18.482558	39.988140	-83.006395	1
7	15673.022989	0.764538	18.623563	33.150000	-96.840000	1
8	16262.139535	0.819084	18.984496	38.874535	-77.007674	1
9	15251.047059	0.692003	18.769608	29.750000	-95.350000	1
10	22056.604651	0.794221	18.063953	33.865233	-118.259186	0
11	21513.098901	0.968377	18.114469	34.325495	-116.998681	0
12	19717.139535	0.988594	17.854651	44.955930	-93.171860	1
13	14629.860000	0.768126	18.800000	26.190000	-80.160000	1
14	17003.577465	0.824471	18.218310	45.560000	-73.550000	1
15	24992.019231	0.833067	18.346154	36.130000	-86.770000	1
16	19690.341463	0.984517	18.780488	42.090000	-71.260000	1
17	18355.613636	0.597915	17.567235	40.808977	-73.950000	0
18	17266.023810	0.690641	18.350198	40.740000	-74.150000	1
19	19780.247059	0.775696	18.688235	28.540000	-81.390000	1
20	16825.604396	0.909492	18.494505	39.830000	-75.380000	1
21	23022.141176	0.949036	18.022549	45.520000	-122.690000	1
22	18287.436782	0.904736	19.051724	40.580000	-111.890000	1
23	17130.271605	0.837747	18.450617	37.355556	-121.926173	1
24	34872.666667	0.904692	16.893519	47.600000	-122.330000	1
25	18930.581395	1.025103	18.563953	39.120000	-94.820000	1
26	22423.000000	0.996578	19.205882	38.630000	-90.210000	1
27	22875.146341	0.785485	18.646341	43.630000	-79.420000	1
28	17854.794872	0.807835	18.096154	49.168462	-122.966154	1

In [36]:
```
# Plot of attendance vs. home team
ax = mlsall_df.groupby('home_team').\
agg({x:'mean' for x in num_feats}).sort_values(by='attendance').\
plot.barh(y='attendance',
          title='Attendance vs. Home Team',
          xlabel='Attendance',
          width=0.9,
          figsize=(9,6))

ax.set_yticklabels([team_names[x] for x in np.argsort(mlsall_df.groupby
                                                      agg({'attendance'
```



Atlanta United, Seattle Sounders, and Charlotte FC lead the way in terms of overall attendance. These teams all share a large stadium with an NFL team, allowing these teams to have average attendance values higher than the stadium capacity of most other teams.

In [37]:
```
# Plot of attendance/capacity vs. home team
ax = mlsall_df.groupby('home_team').\
agg({x:'mean' for x in num_feats}).sort_values(by='att_div_capacity').\
plot.barh(y='att_div_capacity',
          title='Attendance/Capacity vs. Home Team',
          xlabel='Attendance/Capacity',
          width=0.9,
          figsize=(9,6))

ax.set_yticklabels([team_names[x] for x in np.argsort(mlsall_df.groupby
                                                      agg({'att_div_capa
```



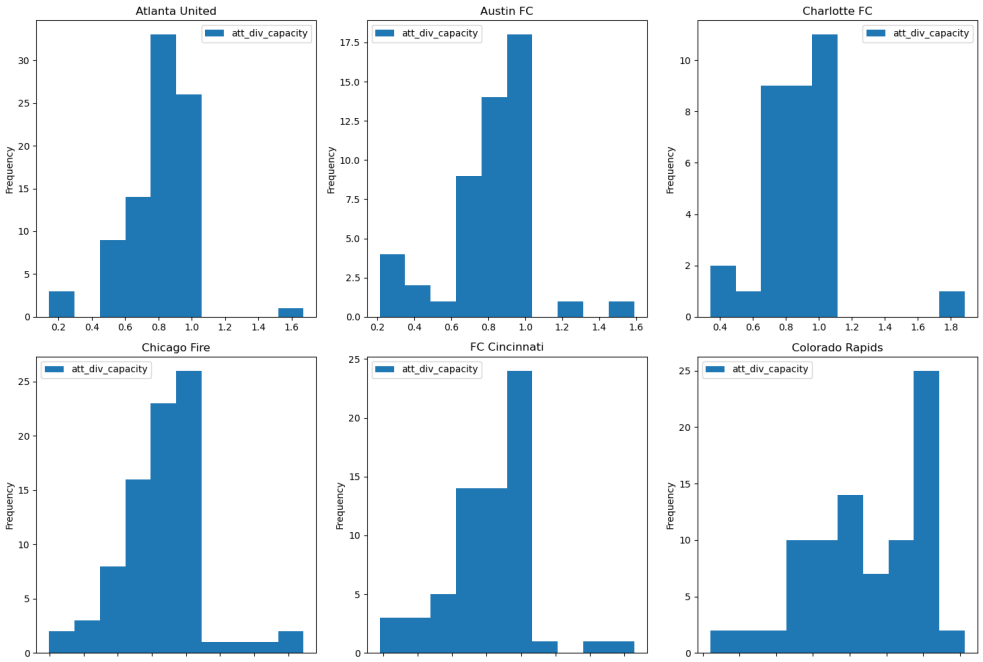
In terms of percentage attendance, Atlanta United still leads the way. Their average attendance is higher than the artificially reduced capacity they usually use thanks to the matches where they do use the entire capacity.

The teams with the worst attendance are the Chicago Fire and New York City FC. Both of these teams are in the process of building new stadiums and have had to use other stadiums in the meantime, which has not helped with attendance.

Attendance vs. Away Team

It is possible that certain away teams drive attendance more than others. This could be because they have a star player that people want to see (think Lionel Messi in 2023), or that people just want to see matches against stronger opponents.

In [38]:
```
# Histograms of attendance/capacity vs. away team
attendance_histograms(mlsall_df, 'away_team', label_dict=team_names)
```

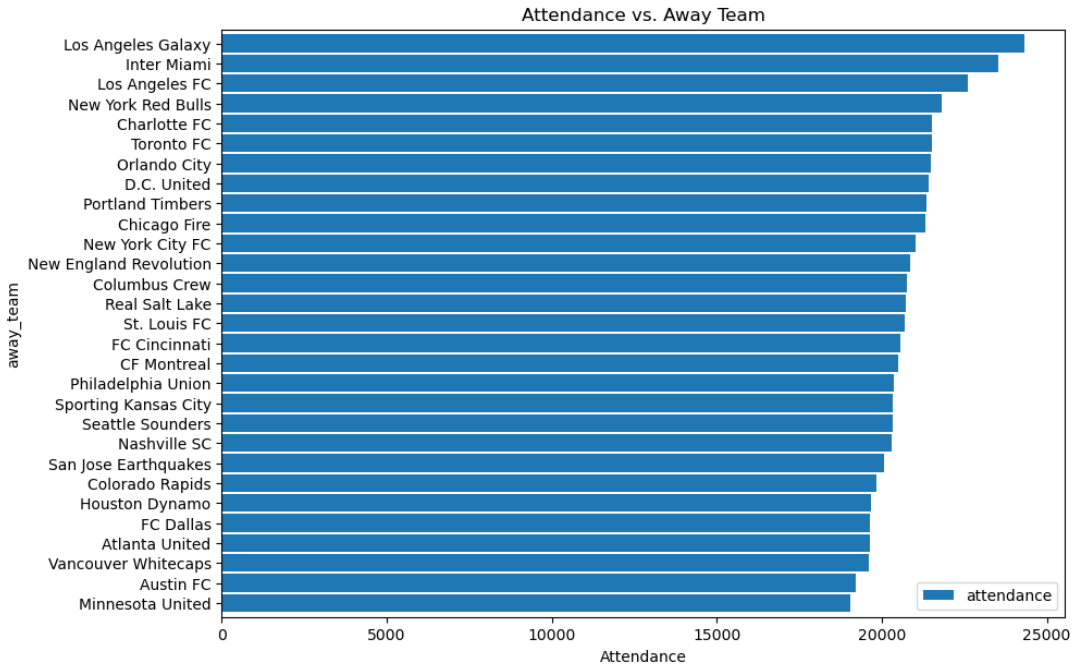


In [39]:
```
# Group by away team
mlsall_df.groupby('away_team').agg({x:'mean' for x in num_feats})
```

Out[39]:

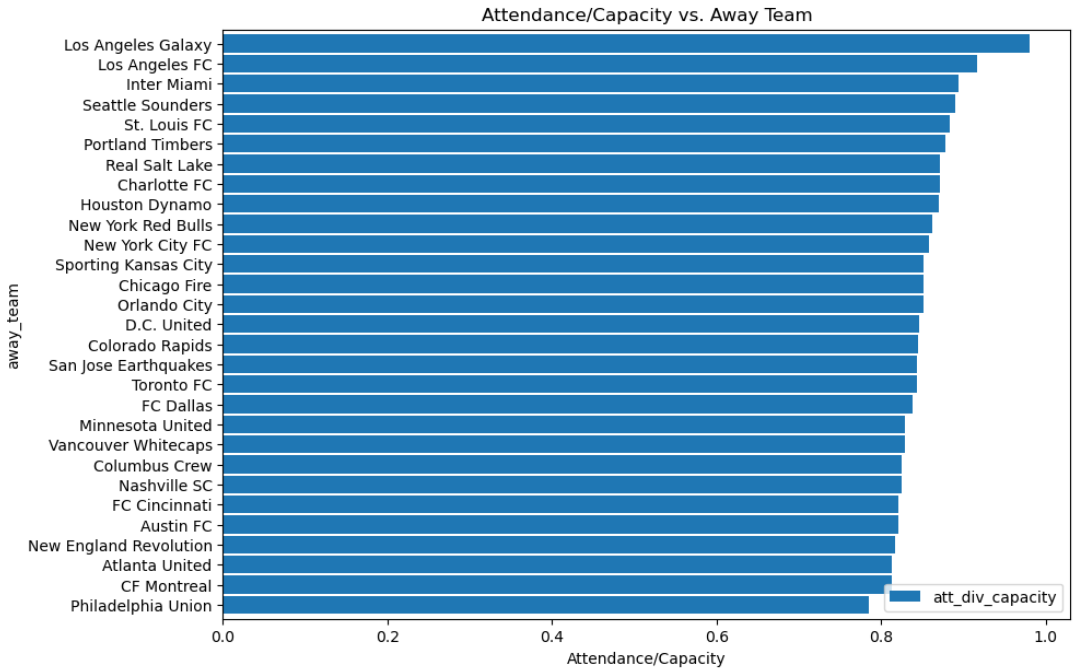
away_team	attendance	att_div_capacity	local_time	latitude	longitude	real_hom
0	19634.779070	0.813420	17.975775	38.980581	-86.190349	0.
1	19212.340000	0.821436	18.370000	38.762400	-103.591000	1.
2	21520.636364	0.871524	19.196970	38.665455	-85.441818	0.
3	21309.867470	0.851394	18.445783	38.718434	-85.968313	1.
4	20562.484848	0.821742	18.772727	38.603939	-84.932424	0.
5	19850.035714	0.844569	18.479167	38.751548	-101.210238	1.
6	20776.869048	0.825024	18.392857	38.518571	-85.830476	0.
7	19645.211765	0.838161	18.445098	39.093176	-102.007059	0.
8	21419.084337	0.845700	18.559237	38.754940	-86.046145	0.
9	19656.654321	0.869849	18.759259	39.364074	-101.500988	1.
10	24338.682353	0.981295	18.111765	39.173294	-101.376235	1.
11	22611.423529	0.916365	17.694118	38.804235	-102.488471	1.
12	19032.964286	0.828991	18.641865	38.805238	-102.406905	1.
13	23538.588235	0.894030	18.549020	38.350000	-84.101765	0.
14	20484.821429	0.812810	18.142857	38.073571	-86.120476	0.
15	20299.384615	0.824637	18.346154	37.951154	-89.791346	0.
16	20852.764706	0.816671	18.211765	38.730588	-86.132471	0.
17	21025.344444	0.857716	17.813889	38.914889	-86.811778	1.
18	21820.597701	0.862420	17.936782	38.535287	-86.364253	1.
19	21494.987952	0.851115	18.355422	39.497229	-86.332771	0.
20	20379.160920	0.785299	18.514368	38.588736	-86.831034	0.
21	21359.651685	0.878907	18.241573	38.704045	-100.482135	0.
22	20717.097826	0.871532	18.508152	39.300978	-102.522283	1.
23	20067.529412	0.843893	18.347059	39.160706	-101.902824	1.
24	20321.709302	0.890036	18.337209	38.606860	-102.762326	1.
25	20341.583333	0.852222	18.095238	39.185833	-102.578810	1.
26	20696.705882	0.883772	18.911765	38.560588	-103.321176	1.
27	21510.678571	0.843496	18.196429	38.343333	-86.667619	0.
28	19615.047619	0.828714	18.547619	38.662143	-101.926310	1.

In [40]:
```
# Plot of attendance vs. away team
ax = mlsall_df.groupby('away_team').\
agg({x:'mean' for x in num_feats}).sort_values(by='attendance').\
plot.barh(y='attendance',
          title='Attendance vs. Away Team',
          xlabel='Attendance',
          width=0.9,
          figsize=(10,7))

ax.set_yticklabels([team_names[x] for x in np.argsort(mlsall_df.groupby
                                                      agg({'attendance'
```



Los Angeles Galaxy, Inter Miami, and Los Angeles FC have the highest attendance values when they play on the road. Inter Miami may have gotten a bit of a boost late in 2023 because of Lionel Messi.

In [41]:
```
# Plot of attendance/capacity vs. away team
ax = mlsall_df.groupby('away_team').\
agg({x:'mean' for x in num_feats}).sort_values(by='att_div_capacity').\
plot.barh(y='att_div_capacity',
          title='Attendance/Capacity vs. Away Team',
          xlabel='Attendance/Capacity',
          width=0.9,
          figsize=(10,7))

ax.set_yticklabels([team_names[x] for x in np.argsort(mlsall_df.groupby
                                                      agg({'att_div_capa
```



The two teams that get the highest attendance for their away matches are the Los Angeles Galaxy (98.1%) and Los Angeles FC (91.6%). No other teams average over 90%.

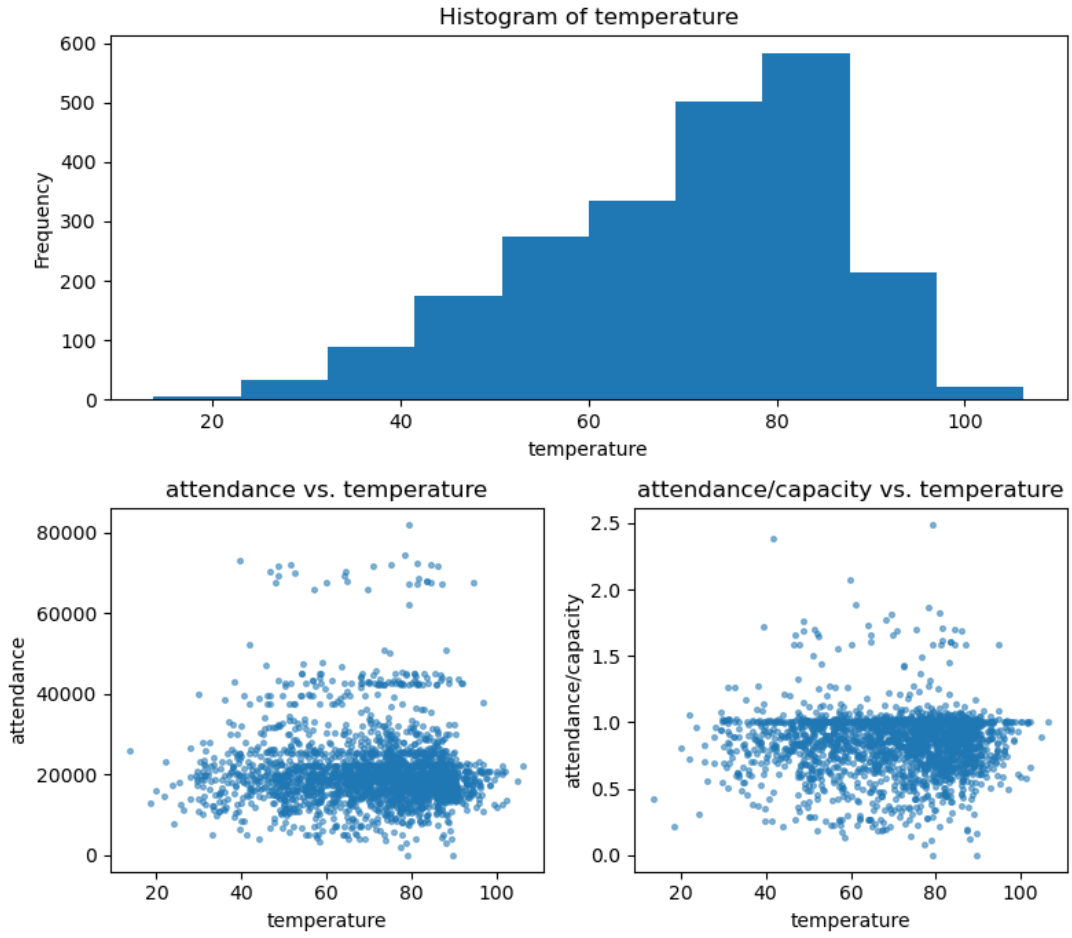
The team with the lowest attendance for its away matches is the Philadelphia Union (78.5%). Despite being a title contender the last few years, they are the only team that averages less than 80% attendance for away matches.

Attendance vs. Weather

Next, I will look at how each weather feature (temperature, rain, snow, cloud cover, windspeed, windgust speed) relate to attendance.

Temperature

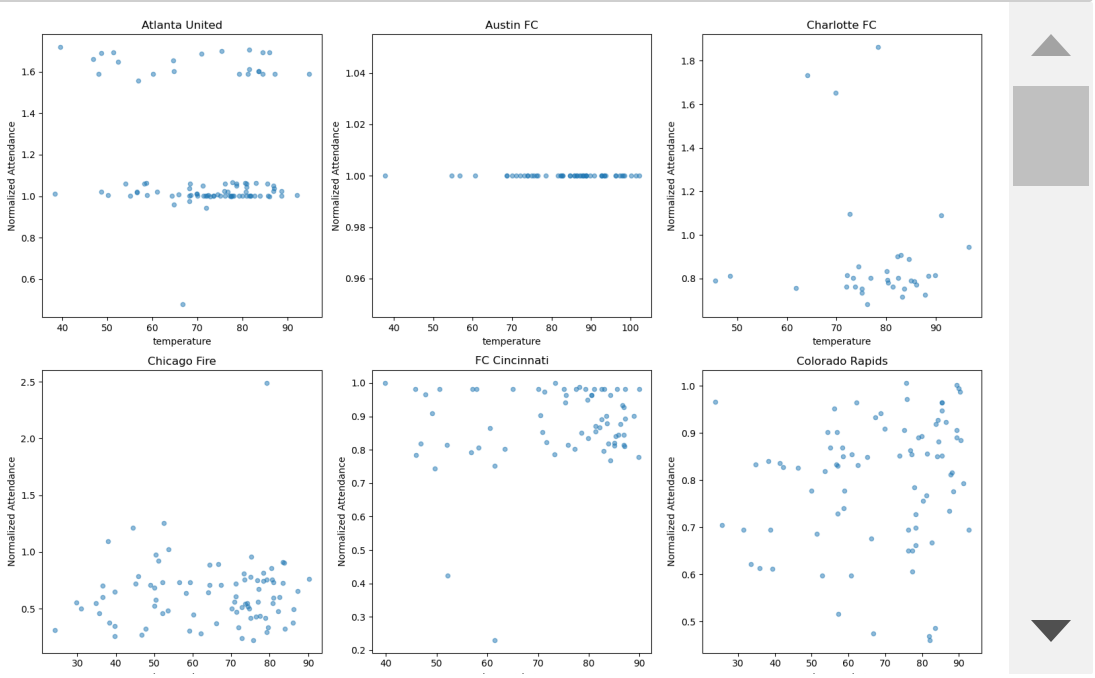
In [42]:
```
# Temperature
make_hist_scatter(mlsall_df, 'temperature', figsize=(8,7))
```



There is not a super clear trend between attendance and temperature.

I also want to look at how these scatter plot looks for each individual team, which I do below.

In [43]:
```
#Plot attendance vs. temperature
make_scatter_plots(mlsall_df, feature='temperature', split_by_team=True
```



While there is not a clear trend between temperature and attendance for any team, there is some evidence that extremely cold temperatures reduce attendance:

CF Montreal: Montreal had a few matches below 40 degrees that were very poorly attended.

New England Revolution: None of the 7 matches with a temperature below 40 degrees reach 80% attendance.

That said, several other teams have had cold matches that still had high attendance.

Below, I look at the stats for regular season matches with a temperature below 40 degrees. The average attendance was about 80% of the capacity, 5% lower than the overall average, indicating that cold temperatures might have a slight effect.

In [44]:
```
# Regular season matches with temperature <= 40
mlsall_df[(mlsall_df['temperature']<=40)&(mlsall_df['playoff']==0)][num_
```

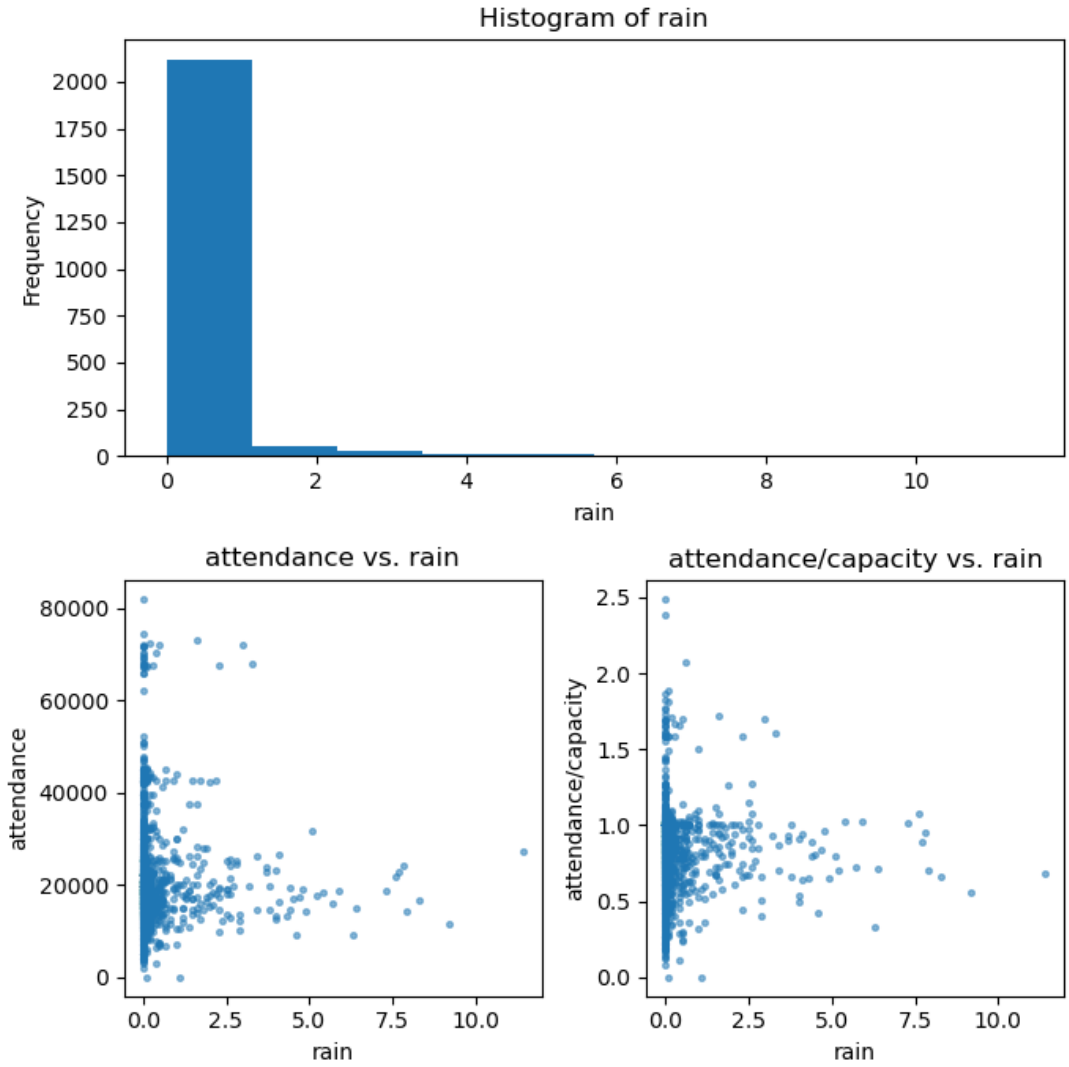
Out[44]:

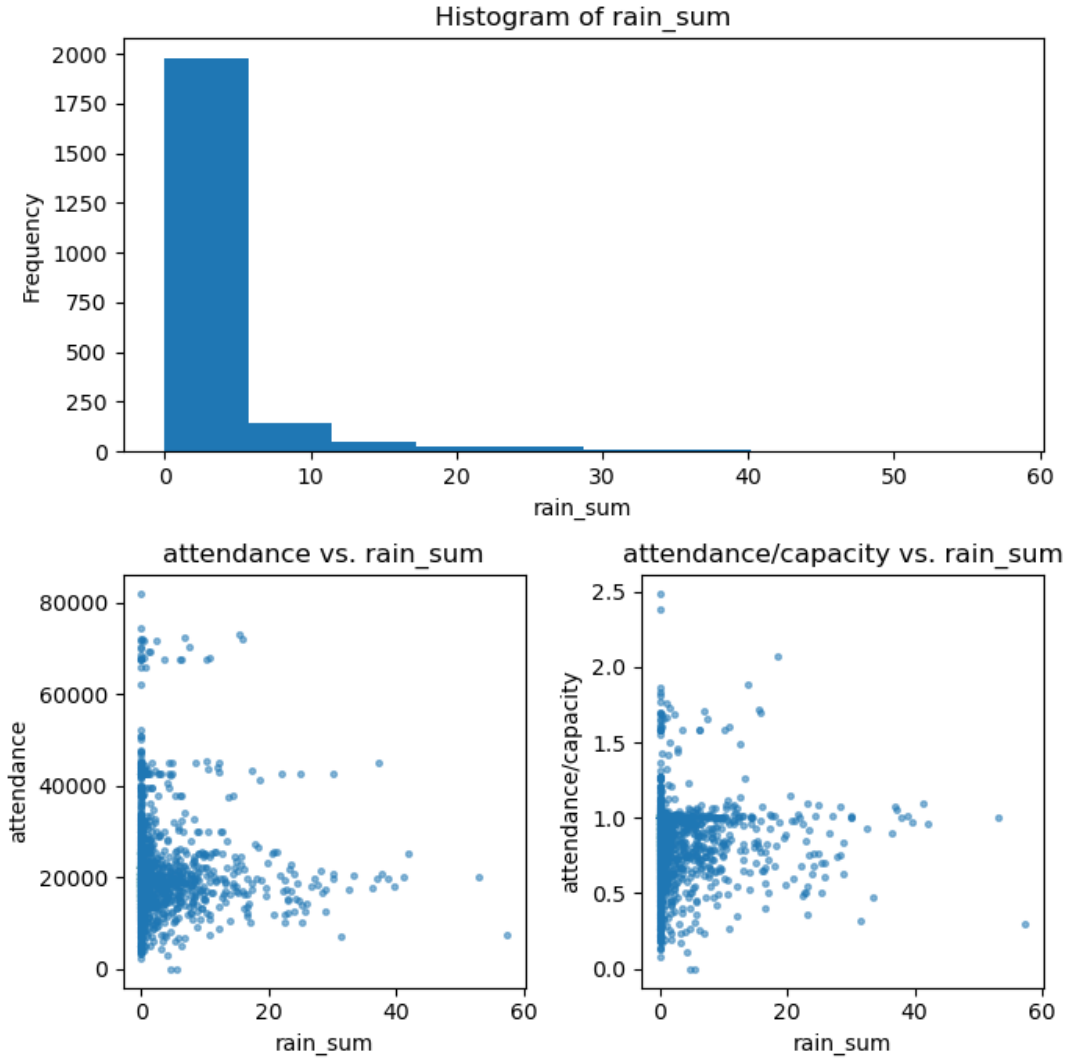
	attendance	att_div_capacity	local_time	latitude	longitude	real_home_te
count	103.000000	103.000000	103.000000	103.000000	103.000000	103.000
mean	18897.174757	0.799095	15.907767	42.167864	-92.699612	0.980
std	7039.546989	0.210286	2.675916	3.703361	17.577162	0.138
min	5128.000000	0.214199	12.000000	30.390000	-123.110000	0.000
25%	13240.500000	0.662025	13.500000	39.830000	-104.890000	1.000
50%	18632.000000	0.828270	15.000000	41.860000	-87.810000	1.000
75%	23023.000000	0.968999	19.000000	44.950000	-77.160000	1.000
max	43055.000000	1.258454	22.500000	49.280000	-71.260000	1.000

For the 103 matches that had temperature below 40 degrees at kick off, the average attendance was 18,897 (80%). This is below the average, indicating that very low temperatures do affect attendance.

Rain

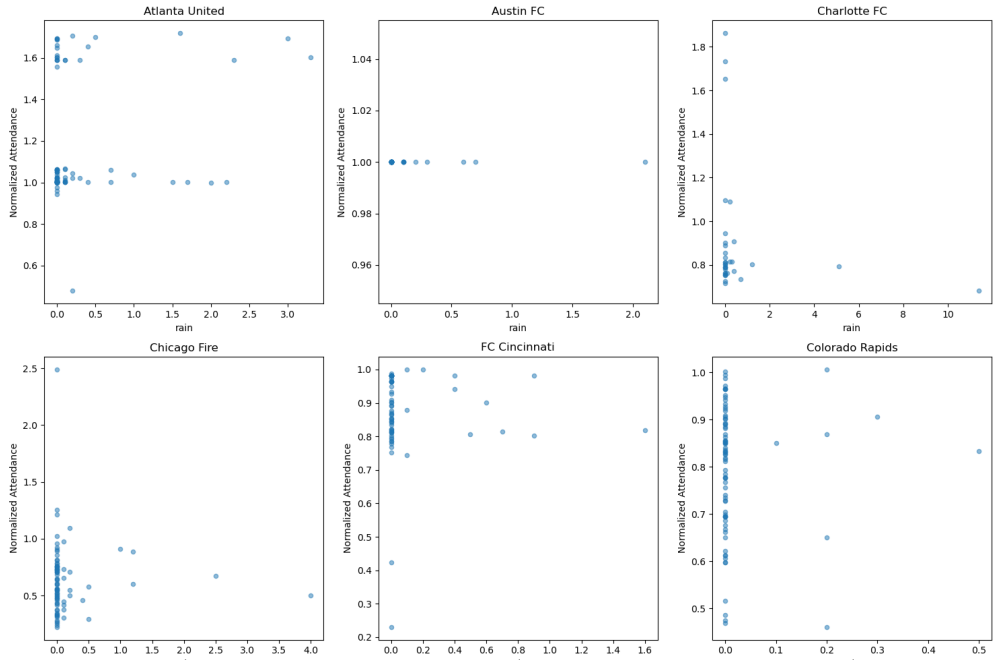
In [45]:
```
# Rain at kick off
make_hist_scatter(mlsall_df, 'rain')

# Rain prior to kick off
make_hist_scatter(mlsall_df, 'rain_sum')
```





Most matches did not have any rain at kick off, but there are 461 that did.

In [46]:
```
#Plot attendance vs. rain at kick off
make_scatter_plots(mlsall_df, feature='rain', split_by_team=True, label_
```



In [47]:
```
# Regular season matches with non-zero rain at kick off
mlsall_df[(mlsall_df['rain']>0)&(mlsall_df['playoff']==0)][num_feats].de
```

Out[47]:

	attendance	att_div_capacity	local_time	latitude	longitude	real_home_te
count	463.000000	463.000000	463.000000	463.000000	463.000000	463.000
mean	20841.779698	0.836921	18.598092	37.915054	-91.038618	0.991
std	10546.856937	0.256007	1.910734	6.322014	16.351152	0.092
min	16.000000	0.000627	12.000000	26.190000	-123.110000	0.000
25%	15108.000000	0.687520	19.000000	33.150000	-96.840000	1.000
50%	18725.000000	0.851996	19.500000	39.130000	-84.520000	1.000
75%	22882.500000	1.000000	19.500000	41.975000	-79.420000	1.000
max	72548.000000	2.067750	22.500000	49.280000	-71.260000	1.000

In [48]:
```
# Regular season matches with non-zero rain prior to kick off
mlsall_df[(mlsall_df['rain_sum']>0)&(mlsall_df['playoff']==0)][num_feats
```

Out[48]:

	attendance	att_div_capacity	local_time	latitude	longitude	real_home_te
count	873.000000	873.000000	873.000000	873.000000	873.000000	873.000
mean	20765.518900	0.838986	18.535987	38.561856	-89.954021	0.993
std	9929.557158	0.243007	1.891265	5.935955	15.763722	0.082
min	16.000000	0.000627	12.000000	26.190000	-123.110000	0.000
25%	15412.000000	0.708000	19.000000	33.750000	-96.840000	1.000
50%	19027.000000	0.860930	19.500000	39.830000	-84.400000	1.000
75%	22651.000000	1.000000	19.500000	42.090000	-77.010000	1.000
max	72548.000000	2.067750	22.500000	49.280000	-71.260000	1.000

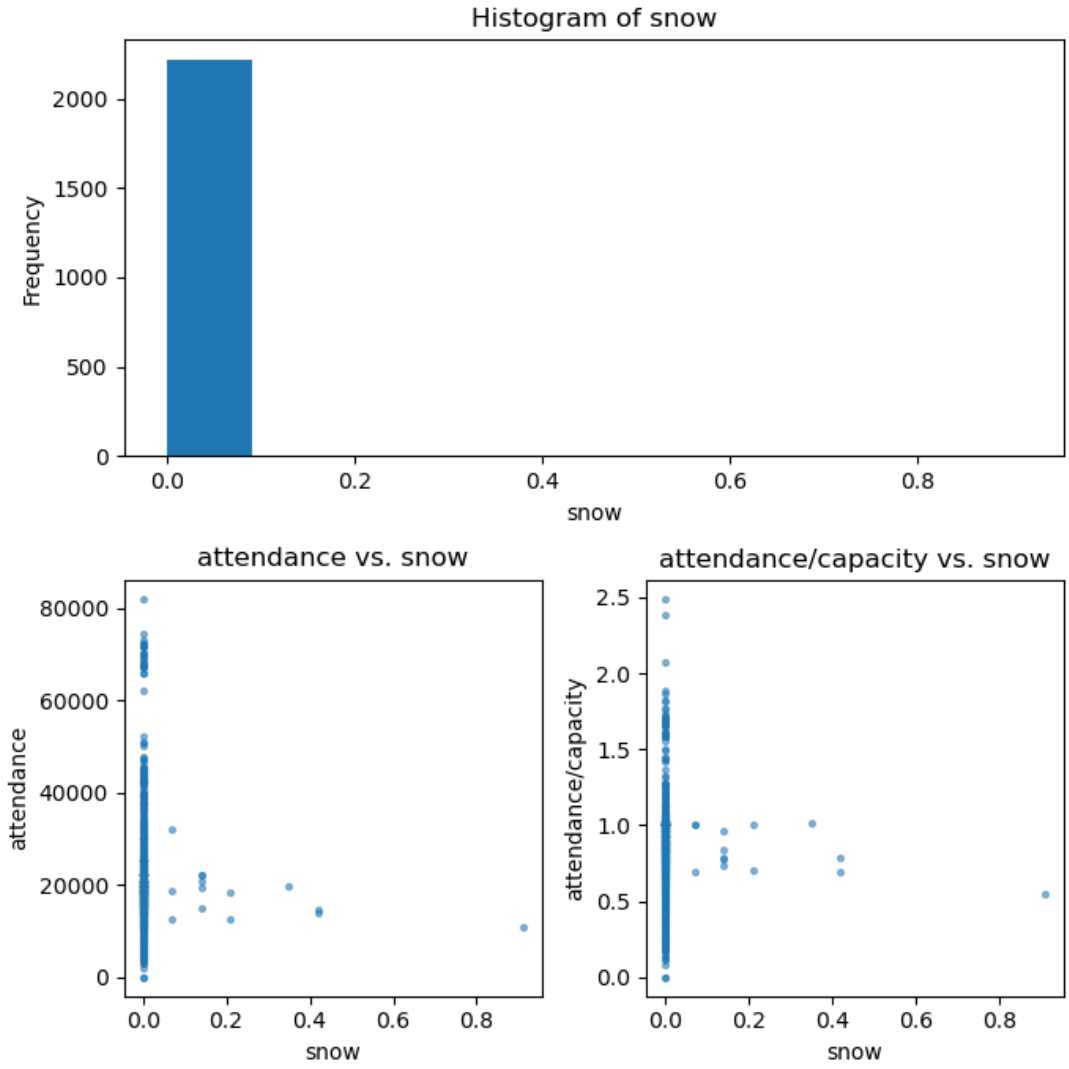
The average attendance for the 461 regular season matches that had rain at kick off was 83.4%.

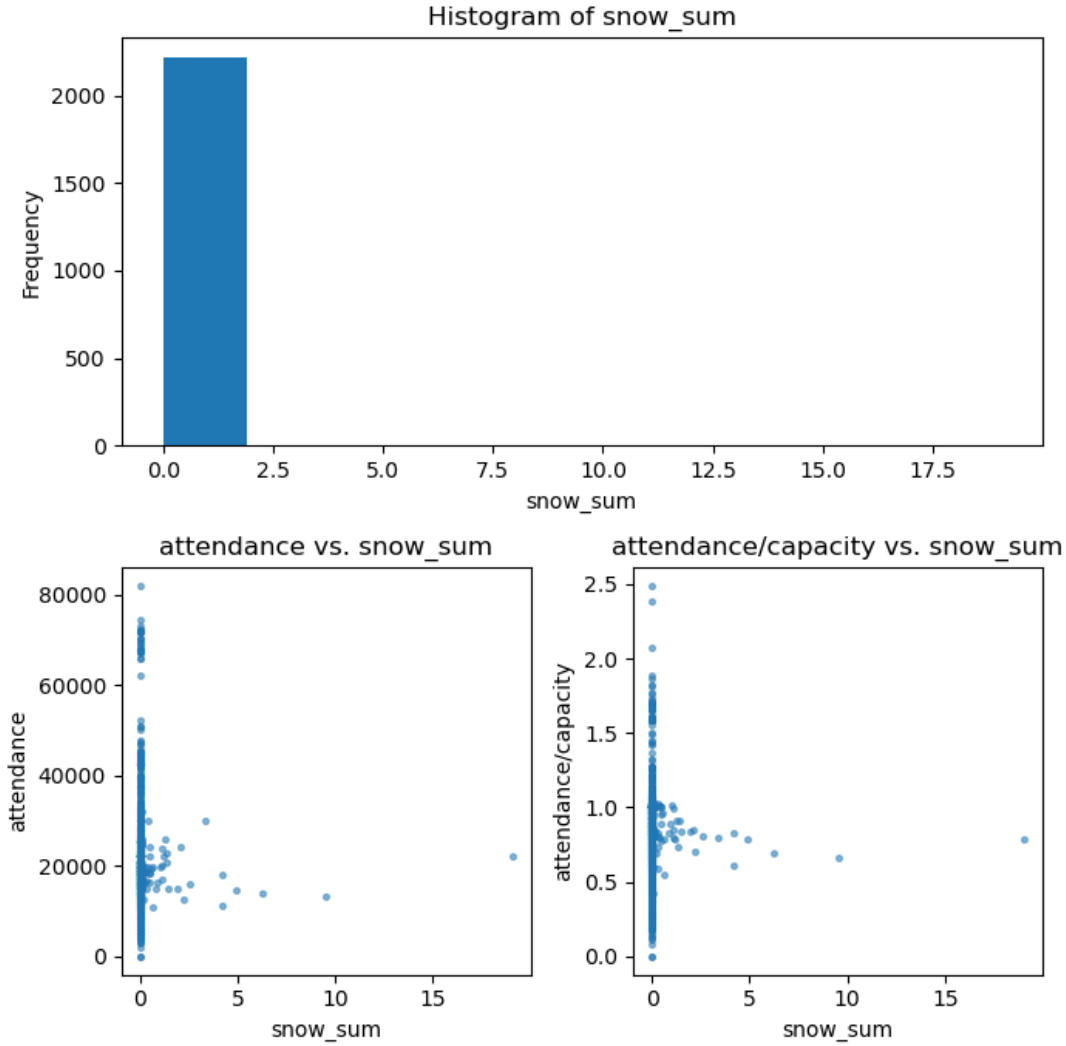
For the 868 regular season matches that were preceded by rain, the average attendance was 83.8%.

These are both a little smaller than the 85% average for the entire dataset.

Snow

In [49]:
```
# Snow at kick off
make_hist_scatter(mlsall_df, 'snow')

# Snow prior to kick off
make_hist_scatter(mlsall_df, 'snow_sum')
```





Very few matches had snow. That is not surprising since MLS tries to avoid playing matches in winter and will typically suspend matches if the weather is poor.

In [50]:
```
# Regular season matches with non-zero snow at kick off
mlsall_df[(mlsall_df['snow']>0)&(mlsall_df['playoff']==0)][num_feats].de
```

Out[50]:

	attendance	att_div_capacity	local_time	latitude	longitude	real_home_tea
count	14.000000	14.000000	14.000000	14.000000	14.000000	14.00000
mean	18068.571429	0.823370	18.214286	41.051429	-88.310714	0.85714
std	5519.320230	0.149977	2.524604	2.079848	13.730457	0.36313
min	10908.000000	0.545400	13.500000	38.870000	-111.890000	0.00000
25%	13979.500000	0.710627	16.250000	39.810000	-102.372500	1.00000
50%	18571.000000	0.784340	19.500000	39.830000	-81.970000	1.00000
75%	20429.250000	0.991280	19.500000	43.245000	-77.612500	1.00000
max	32250.000000	1.011031	22.500000	44.950000	-71.260000	1.00000

In [51]:
```
# Regular season matches with non-zero snow prior to kick off
mlsall_df[(mlsall_df['snow_sum']>0)&(mlsall_df['playoff']==0)][num_feats
```

Out[51]:

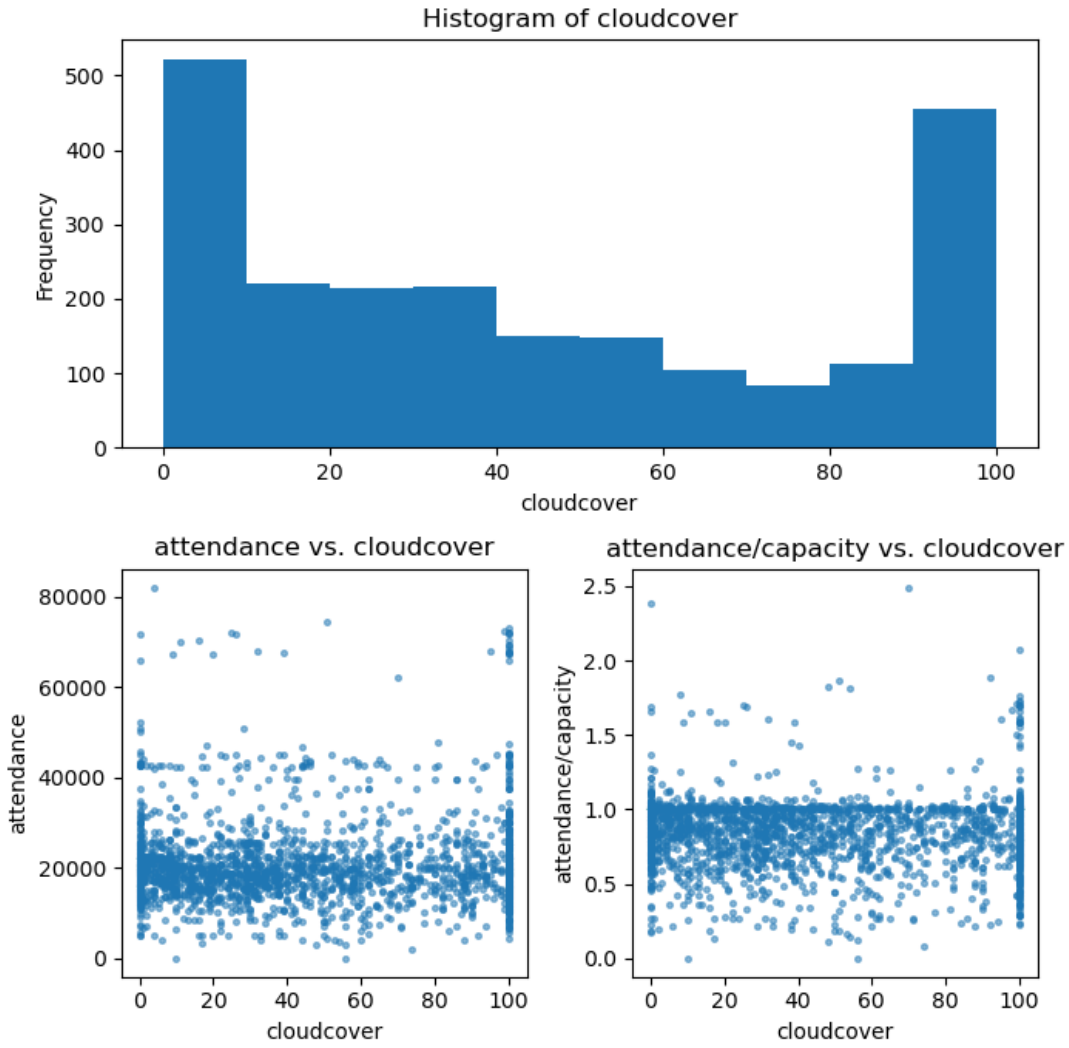
	attendance	att_div_capacity	local_time	latitude	longitude	real_home_tea
count	47.000000	47.000000	47.000000	47.000000	47.000000	47.00000
mean	19217.170213	0.824803	18.021277	41.855106	-91.110000	0.93617
std	4931.293486	0.136969	2.340470	2.482731	16.365078	0.24709
min	10908.000000	0.426284	13.000000	38.870000	-122.690000	0.00000
25%	15612.000000	0.778985	16.000000	39.820000	-104.890000	1.00000
50%	18510.000000	0.827733	19.000000	40.740000	-87.620000	1.00000
75%	22182.500000	0.929424	19.500000	43.630000	-76.195000	1.00000
max	32250.000000	1.020412	22.500000	47.600000	-71.260000	1.00000

There have been 14 matches with snow at kick off and 47 that had snow prior to kick off (there is overlap between the two).

The average attendance for these matches was 82.3% and 82.4%, respectively. This is a little less than the overall average.

Cloud cover

In [52]:
```
# Cloud cover at kick off
make_hist_scatter(mlsall_df, 'cloudcover')
```



The cloud cover describes the percentage of the sky covered by clouds. Most matches had either 0 or 100% cloud cover, but values in between are well represented.

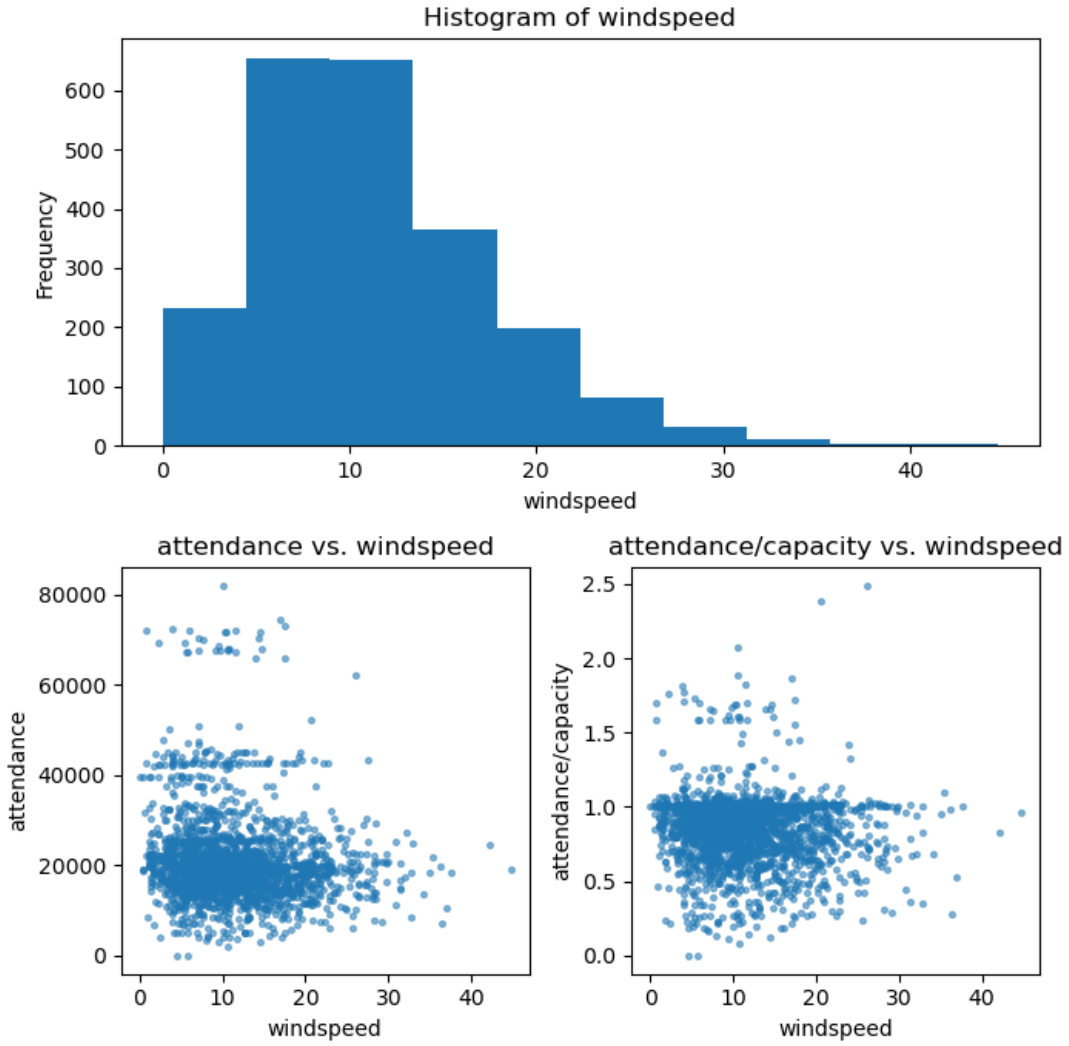
There is no obvious trend between attendance and cloud cover.

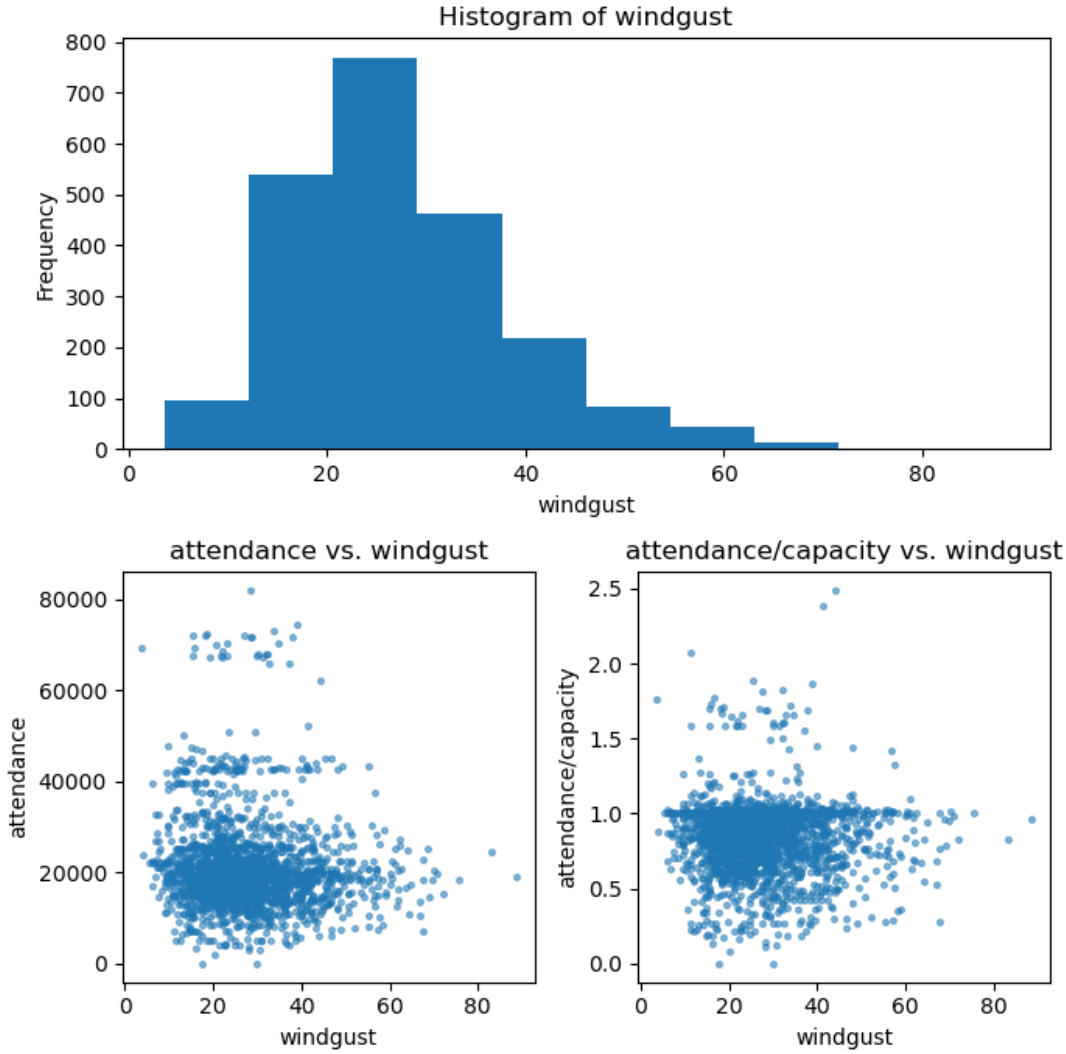
Windspeed and windgust speed

Windspeed is the average speed of the wind in km/h.

Windgust speed is the speed of the gusts of wind in km/h.

In [53]:
```
# Windspeed at kick off
make_hist_scatter(mlsall_df, 'windspeed')

# Windgust speed at kick off
make_hist_scatter(mlsall_df, 'windgust')
```



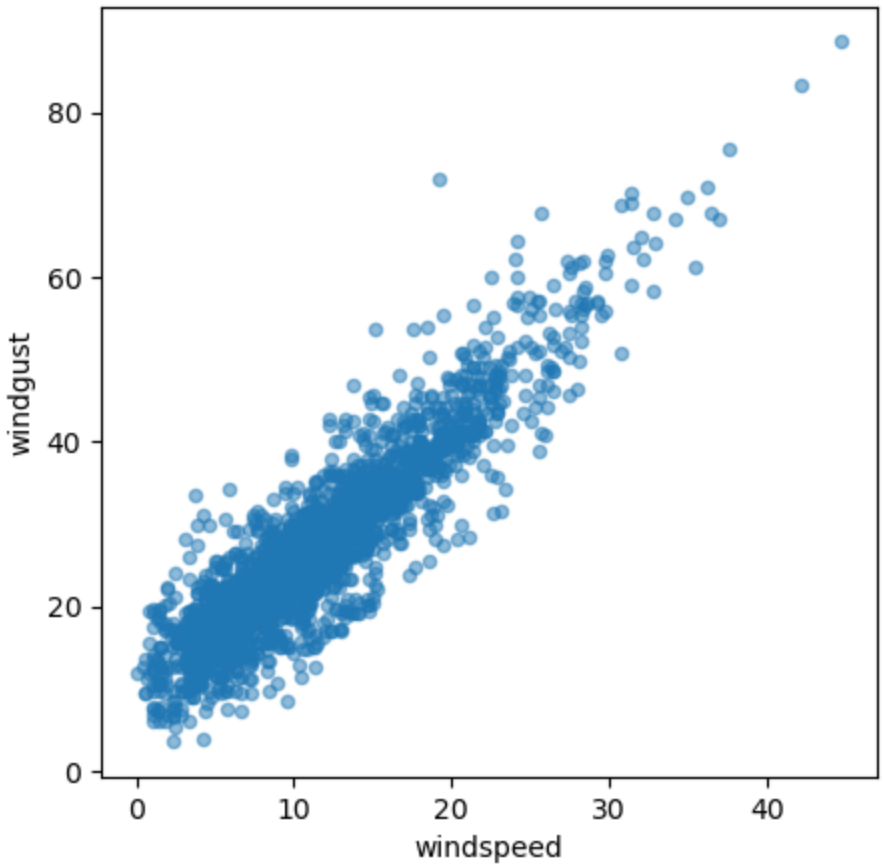


The scatterplots don't seem to have a strong trend with attendance.

The distributions and scatterplots for windspeed and windgust speed look very similar. Let's check out the scatterplot of windspeed vs windgust below.

In [54]:
```
fig, ax = plt.subplots(figsize=(5,5))

mlsall_df.plot.scatter(x='windspeed',
                       y='windgust',
                       ax=ax,
                       alpha=0.5)
```

Out[54]: <Axes: xlabel='windspeed', ylabel='windgust'>

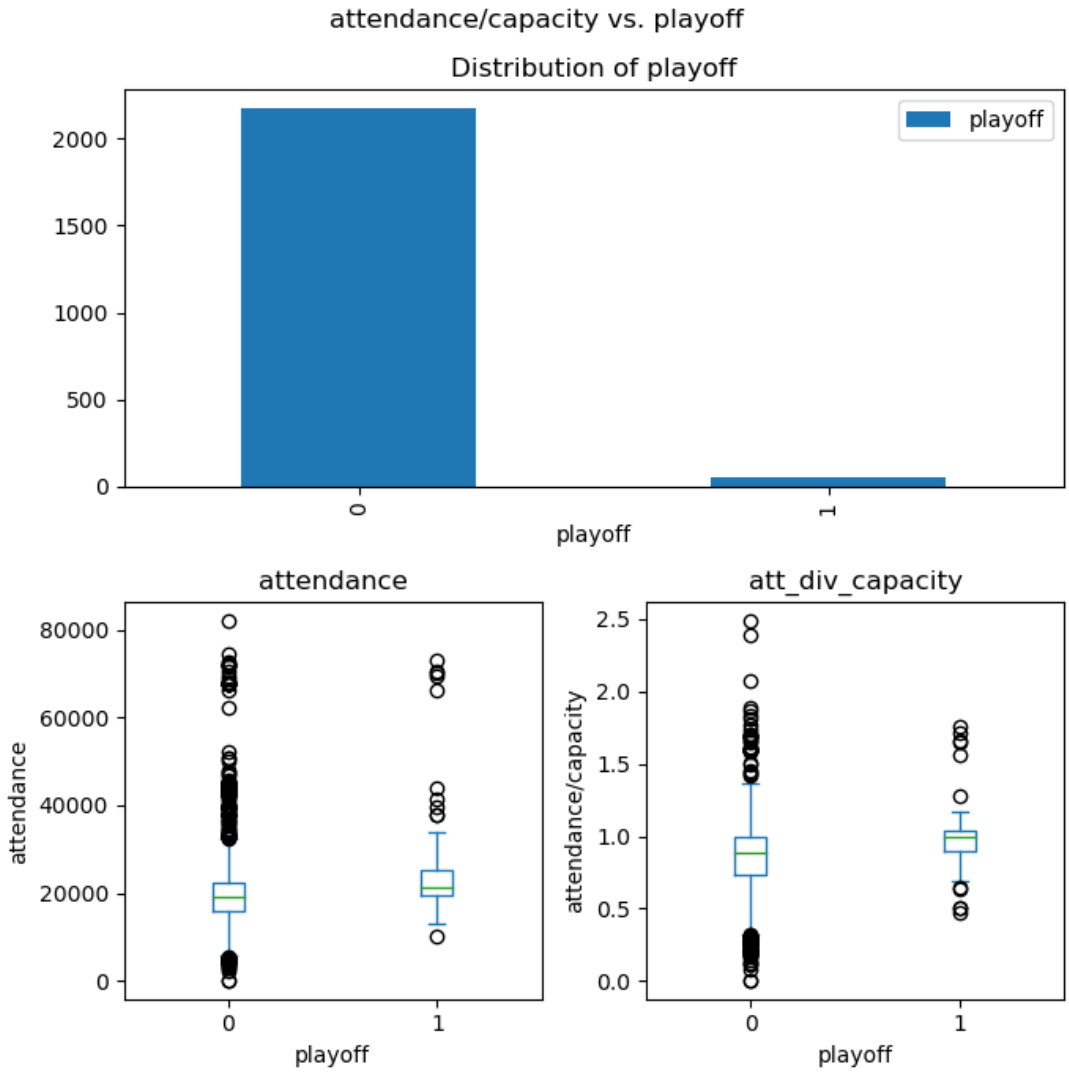


There is a strong correlation between windspeed and windgust, so at most one of these features would be used for the model.

Attendance vs. Regular Season/Playoffs

Below, I look at how attendance changes from the regular season to the playoffs. There are far more regular season matches, but still a solid sample of playoff matches.

In [55]:
```
make_bar_box(mlsall_df, 'playoff')
```



In [56]:
```
mlsall_df.groupby('playoff').agg({x:'mean' for x in num_feats})
```

Out[56]:

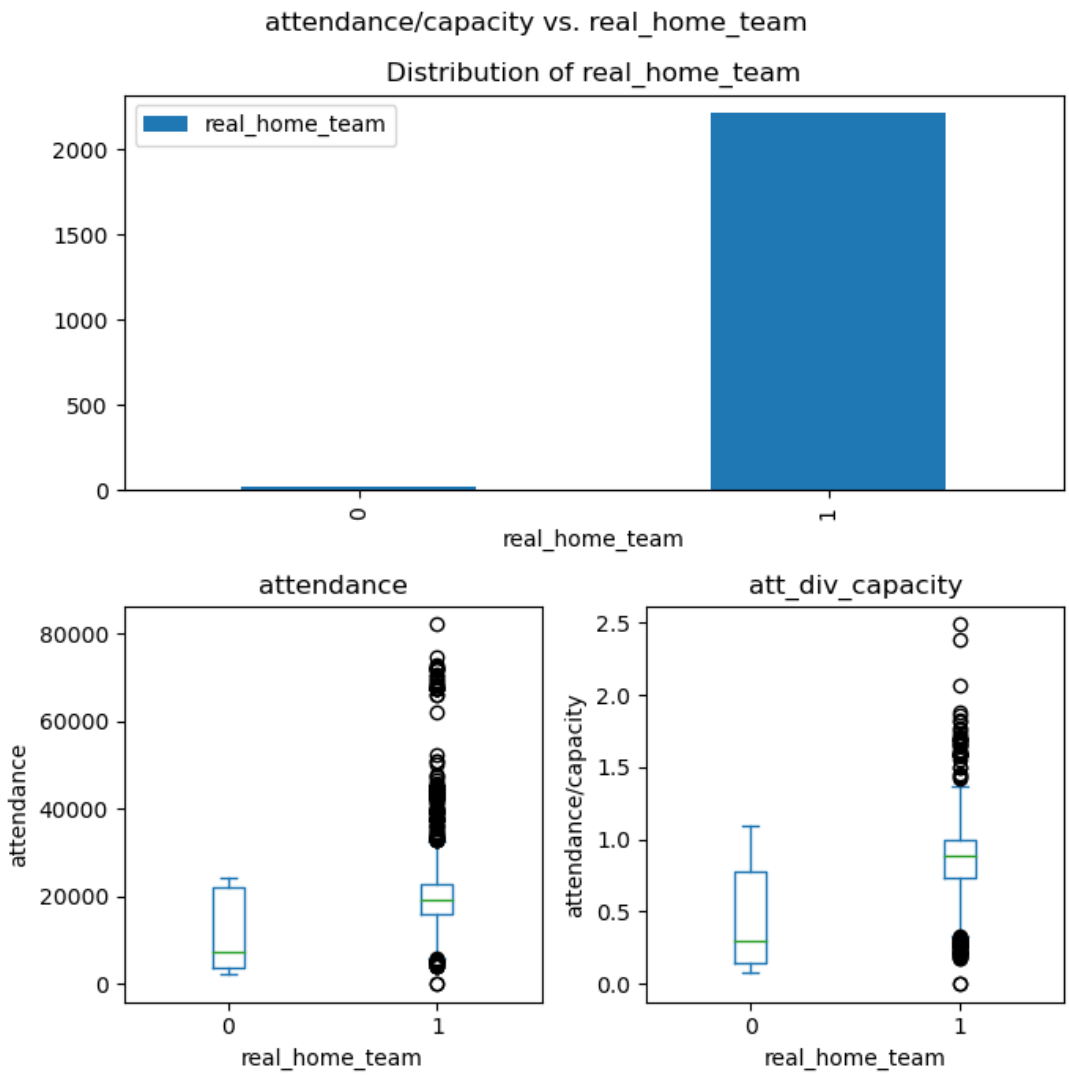
playoff	attendance	att_div_capacity	local_time	latitude	longitude	real_home_tea
0	20662.514036	0.848134	18.362901	38.774340	-93.943580	0.99263
1	26786.964286	0.997887	17.151786	39.509643	-95.334643	0.98214

Playoff matches have very good attendance figures. They average nearly 100% attendance compared to 84.7% for regular season matches. They get over 6000 more people on average.

Attendance vs. real home team

On some occasions, teams were forced to play "home" matches away from their actual home stadium. Most of these were Canadian teams that were forced to play some of their home matches in the US (Montreal and Toronto had to play in Orlando and Vancouver had to play in Utah) during the pandemic. New York City FC also had to play a few matches in the home stadium of the New York Red Bulls.

In [57]:
```
make_bar_box(mlsall_df, 'real_home_team')
```



In [58]:
```
mlsall_df.groupby('real_home_team').agg({x:'mean' for x in num_feats})
```

Out[58]:

real_home_team	attendance	att_div_capacity	local_time	latitude	longitude	real_l
0	11088.588235	0.428912	18.235294	40.454706	-80.257647	
1	20891.142405	0.855147	18.333220	38.780041	-94.083978	

Not surprisingly, the attendance drops dramatically when neither team is really the home team. On average, matches without a real home team only averaged 42.9% attendance. When there was no real home team, the average attendance was just 11,089, which is awful.

Summary

Trends I noticed in the data:

Months: There is some evidence that attendance does change from month to month. Other factors might be responsible, but it does look like attendance might improve as the season progresses.

Day of the week: It looks like Sundays might be the best for attendance while Wednesdays are the worst. However, it is possible that Sundays had the highest average attendance because more marquee matches were scheduled for Sundays.

Time of day: There is some evidence that early/mid afternoon matches might be slightly better than late afternoon or night matches.

Rivals: It does look like there could be a significant increase in attendance when two rivals are playing.

Home team, away team: The models will definitely need to take into account which teams are playing as there is significant variation between them.

Weather: Cold temperatures do appear to be correlated with lower attendance. Other than that, there were not any clear trends with other weather features.

The models that I create should do a better job of finding real trends because the models will do a better job of accounting for correlations between features. For example, it is possible that once the relative averages of each home team are taken into account, some trends seen here might disappear while others not seen here become apparent.

Modeling

In this notebook, I create different types of models that attempt to predict attendance of MLS matches.

Types of models:

1. Linear Regression
2. XGBoost Regression
3. Random Forest Regression
4. K-Nearest Neighbors Regression

Business Problem

Major League Soccer (MLS) is the top professional soccer league in the United States and Canada. It began play in 1996 and has been a driving force in growing the sport of soccer in the U.S. ever since.

A major source of income for a league and its teams is ticket sales. MLS exists in a crowded sports landscape in which it competes with both other professional sports and college sports. Avoiding competing head-to-head with the National Football League (NFL) is a major reason that MLS starts its season in late February and ends in early December, out of phase with most other soccer leagues worldwide which start in August and end in May.

The goal of this project is to determine when MLS should hold their matches in order to maximize attendance. I will use seasons ranging from 2018 to 2023 to perform my analysis, but I will exclude 2020 because almost the entire season was played without fans in attendance due to the COVID-19 pandemic.

I will look at the following factors related to time:

1. Time of day. Matches start as early as noon and as late as 10 pm.
2. Day of the week. Most matches are played on weekends, a fair number are played on Wednesdays, and a smaller number are played on other weekdays.
3. Month. The season starts in February and ends in December, though the regular season ends in October.
4. Year.

I will also look at these factors related to weather:

1. Temperature. I am especially looking to see if especially hot or cold temperatures negatively affect attendance.
2. Rain and snow. I hypothesize that precipitation could reduce attendance.
3. Windspeed.

The models will also incorporate these factors:

1. Regular season vs. playoffs: I expect playoff matches to have higher attendance.

2. Home openers: In data_exploration.ipynb, there was some evidence that home openers tend to have better attendance than average.
3. Rivalry matches: MLS has quite a few rivalries that could help drive fans to attend matches.
4. Matches without a real home team: Each team has a stadium that it uses for its home matches. However, there have been cases where matches had to be held in neutral locations. There is reason to believe these matches have MUCH lower attendance than average.

Note about the models

While I am creating regression models that could make predictions of attendance for single matches, the goal of this work is not to use the model to make match-by-match predictions. I expect there to be a lot of variance in attendance that is not related to the factors listed above, making a single prediction somewhat unreliable. Instead, the goal is to look at the way the factors are incorporated into the models to see what is important and how it affects attendance.

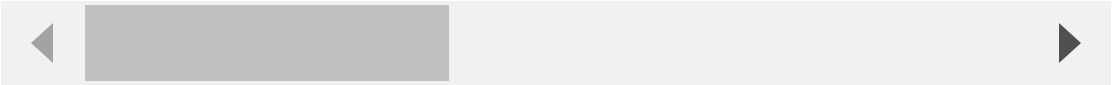
In [1]:
```
import numpy as np
import scipy.stats as sp
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import statsmodels.api as sm
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.metrics import mean_squared_error as mse
from sklearn.metrics import r2_score
from xgboost import XGBRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.neighbors import KNeighborsRegressor
import warnings
warnings.filterwarnings('ignore', category=FutureWarning)
from model_funcs import *
```

In [2]:
```
# Read in data
mlsall_df = pd.read_csv('mls_with_weather.csv', index_col=0)
mlsall_df
```

Out[2]:

	round	day	date	local_time	home_team	home_score	away_score	away_tean
0	Regular Season	Sat	2018-03-03	13.0	27	0	2	
1	Regular Season	Sat	2018-03-03	14.5	9	4	0	
2	Regular Season	Sun	2018-03-04	14.0	24	0	1	1
3	Regular Season	Sun	2018-03-04	15.0	28	2	1	1
4	Regular Season	Sat	2018-03-03	19.0	20	2	0	1
...	...	...	...	...	...	...	...	..
2280	Regular Season	Sat	2023-10-21	19.0	5	0	1	2
2281	Regular Season	Sat	2023-10-21	20.0	26	0	2	2
2282	Regular Season	Sat	2023-10-21	20.0	25	3	1	1
2283	Wild Card Round	Wed	2023-10-25	19.5	18	5	2	
2284	Wild Card Round	Wed	2023-10-25	20.5	25	0	0	2

2285 rows × 29 columns



Below, I drop matches for which weather data is missing and the attendance is 0. If the attendance is 0, it either means the information was missing or fans were not allowed due to the pandemic.

In [3]:
```
# Drop anything with missing data
mlsall_df.dropna(inplace=True)
# Drop matches with attendance=0
mlsall_df.drop(mlsall_df[mlsall_df['attendance']==0].index, inplace=Tru
mlsall_df
```

Out[3]:

	round	day	date	local_time	home_team	home_score	away_score	away_team
0	Regular Season	Sat	2018-03-03	13.0	27	0	2	6
1	Regular Season	Sat	2018-03-03	14.5	9	4	0	0
2	Regular Season	Sun	2018-03-04	14.0	24	0	1	11
3	Regular Season	Sun	2018-03-04	15.0	28	2	1	14
4	Regular Season	Sat	2018-03-03	19.0	20	2	0	16
...	...	...	...	...	...	...	...	...
2278	Regular Season	Sat	2023-10-21	18.0	21	1	3	9
2279	Regular Season	Sat	2023-10-21	18.0	28	1	1	11
2280	Regular Season	Sat	2023-10-21	19.0	5	0	1	22
2281	Regular Season	Sat	2023-10-21	20.0	26	0	2	24
2282	Regular Season	Sat	2023-10-21	20.0	25	3	1	12

2229 rows × 29 columns

After dropping no weather, no attendance matches, there are 2229 matches remaining.

In [4]:
```
# Get data types of each column
mlsall_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2229 entries, 0 to 2282
Data columns (total 29 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   round             2229 non-null   object
 1   day               2229 non-null   object
 2   date              2229 non-null   object
 3   local_time        2229 non-null   float64
 4   home_team         2229 non-null   int64
 5   home_score        2229 non-null   int64
 6   away_score        2229 non-null   int64
 7   away_team         2229 non-null   int64
 8   attendance        2229 non-null   int64
 9   stadium           2229 non-null   object
 10  latitude          2229 non-null   float64
 11  longitude         2229 non-null   float64
 12  playoff           2229 non-null   int64
 13  att_div_capacity  2229 non-null   float64
 14  real_home_team    2229 non-null   int64
 15  same_conf         2229 non-null   int64
 16  rivals            2229 non-null   int64
 17  temperature       2229 non-null   float64
 18  rain              2229 non-null   float64
 19  snow              2229 non-null   float64
 20  cloudcover        2229 non-null   float64
 21  windspeed         2229 non-null   float64
 22  windgust          2229 non-null   float64
 23  rain_sum          2229 non-null   float64
 24  snow_sum          2229 non-null   float64
 25  date_year         2229 non-null   int64
 26  date_month        2229 non-null   int64
 27  date_day          2229 non-null   int64
 28  home_opener       2229 non-null   int64
dtypes: float64(12), int64(13), object(4)
memory usage: 522.4+ KB
```

Target variable

There are two options for the target variable: attendance and attendance/capacity. I plan on using home team and away team as categorical parameters in the model, so even if I choose to use raw attendance, there will be parameters that effectively normalize the attendance per team. Therefore, I will use raw attendance as the target variable.

In [5]:
```
# Target variable
#y = mlsall_df['att_div_capacity']
y = mlsall_df['attendance']
```

Linear Regression

The way I am going to create the linear regression model is by iteratively trying different combinations of inputs and evaluating the success of the model based on the RMSE of the residuals and the R-squared of the fit.

Model 1

To start, I will just fit a single constant to the data.

In [6]:
```
X = create_X(mlsall_df, columns=[], add_constant=True)
linreg = sm.OLS(y,X).fit()
linreg.summary()
```

Out[6]:

OLS Regression Results

Dep. Variable:	attendance	**R-squared:**	0.000
Model:	OLS	**Adj. R-squared:**	0.000
Method:	Least Squares	**F-statistic:**	nan
Date:	Sun, 26 Nov 2023	**Prob (F-statistic):**	nan
Time:	12:49:22	**Log-Likelihood:**	-23554.
No. Observations:	2229	**AIC:**	4.711e+04
Df Residuals:	2228	**BIC:**	4.712e+04
Df Model:	0		
Covariance Type:	nonrobust		

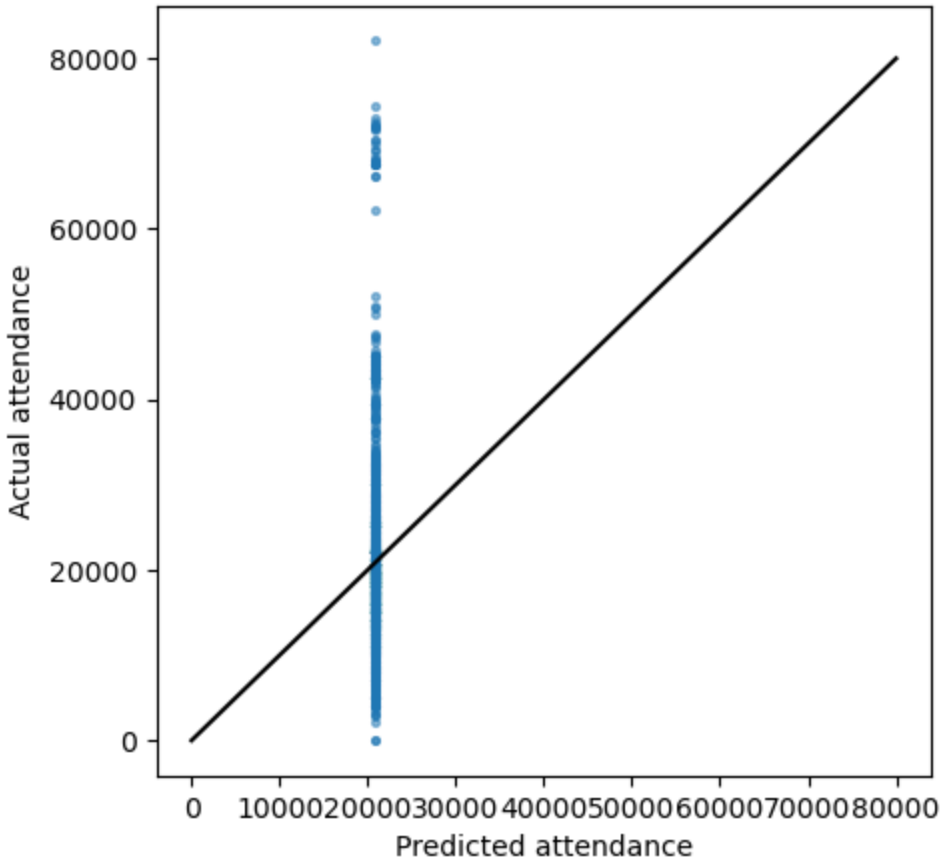
	coef	std err	t	P>|t|	[0.025	0.975]
const	2.082e+04	199.049	104.579	0.000	2.04e+04	2.12e+04

Omnibus:	1176.355	**Durbin-Watson:**	1.850
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	9640.591
Skew:	2.370	**Prob(JB):**	0.00
Kurtosis:	12.019	**Cond. No.**	1.00

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

In [7]:
```
evaluate_linear_model(linreg, X, y)
```

```
RMSE of residuals: 9397.534829728174
R-squared: 1.1102230246251565e-16
```



This model predicts 20,816 for each match. Not surprisingly, this is a pretty bad fit as it does not account for any of the trends in the data.

The RMSE for this fit was 9,398. The goal of subsequent models will be to reduce this number.

Model 2

This model will add home and away team parameters. I expect this to improve the fit quite a lot because it will account for differences in stadium capacities.

In [8]:
```
X = create_X(mlsall_df, columns=['home_team','away_team'], add_constant=
linreg = sm.OLS(y,X).fit()
linreg.summary()
```

Out[8]:

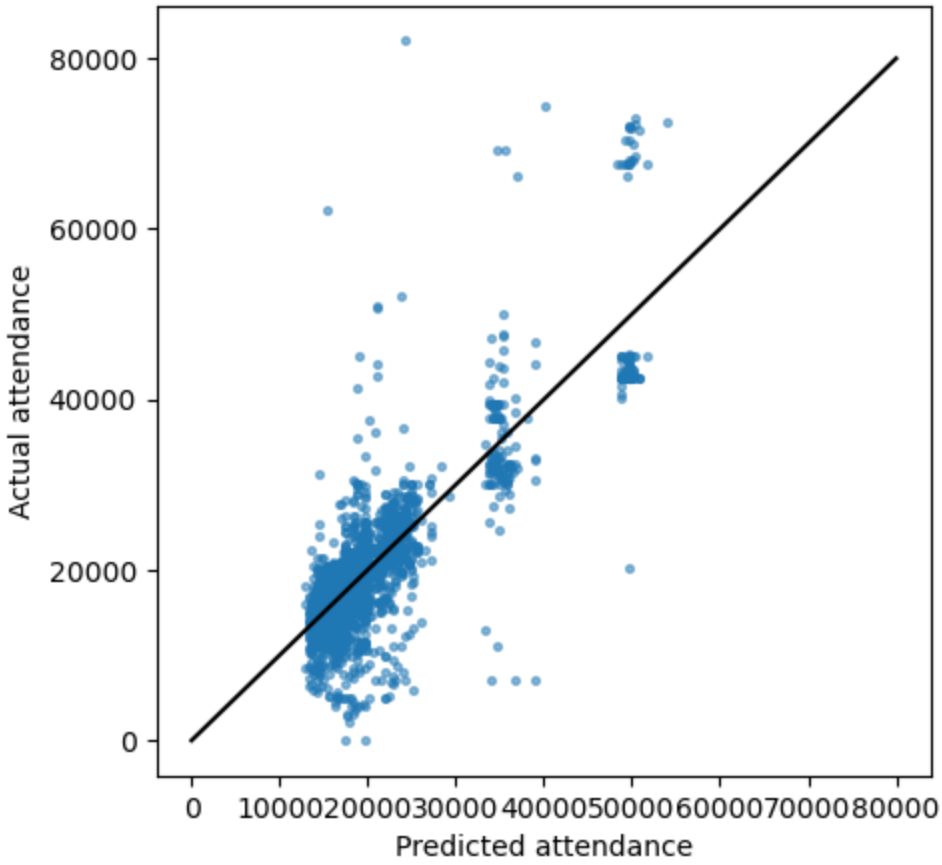
OLS Regression Results

Dep. Variable:	attendance	**R-squared:**	0.662
Model:	OLS	**Adj. R-squared:**	0.653
Method:	Least Squares	**F-statistic:**	75.95
Date:	Sun, 26 Nov 2023	**Prob (F-statistic):**	0.00
Time:	12:49:23	**Log-Likelihood:**	-22345.
No. Observations:	2229	**AIC:**	4.480e+04
Df Residuals:	2172	**BIC:**	4.513e+04
Df Model:	56		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	4.969e+04	845.914	58.746	0.000	4.8e+04	5.14e+04

In [9]:
```
evaluate_linear_model(linreg, X, y)
```

```
RMSE of residuals: 5533.834547035362
R-squared: 0.6619592263521832
```



Accounting for the teams involved in the match improved the fit. The RMSE of the residuals dropped from 9,398 to 5,534.

The R-squared is now at 0.662. The parameters for home and away team will be kept for all subsequent models.

For note, the constant in the fit (49,694) is a lot higher than it was for model 1. This is because the new variables are all in reference to Atlanta United which has the highest average attendance for its home matches. This does strangely mean that the constant assumes a match between Atlanta and Atlanta, but that is not actually a problem.

Model 3

In this model, I incorporate a few time-based parameters:

1. Day of the week
2. Month (reference case is a match in July)
3. Local time of kick off (categorical)
4. Year (2018 is reference case)

In [10]:
```
X = create_X(mlsall_df,
             columns=['home_team','away_team',
                      'day','date_month','local_time','date_year'],
             add_constant=True)
linreg = sm.OLS(y,X).fit()
linreg.summary()
```

Out[10]:

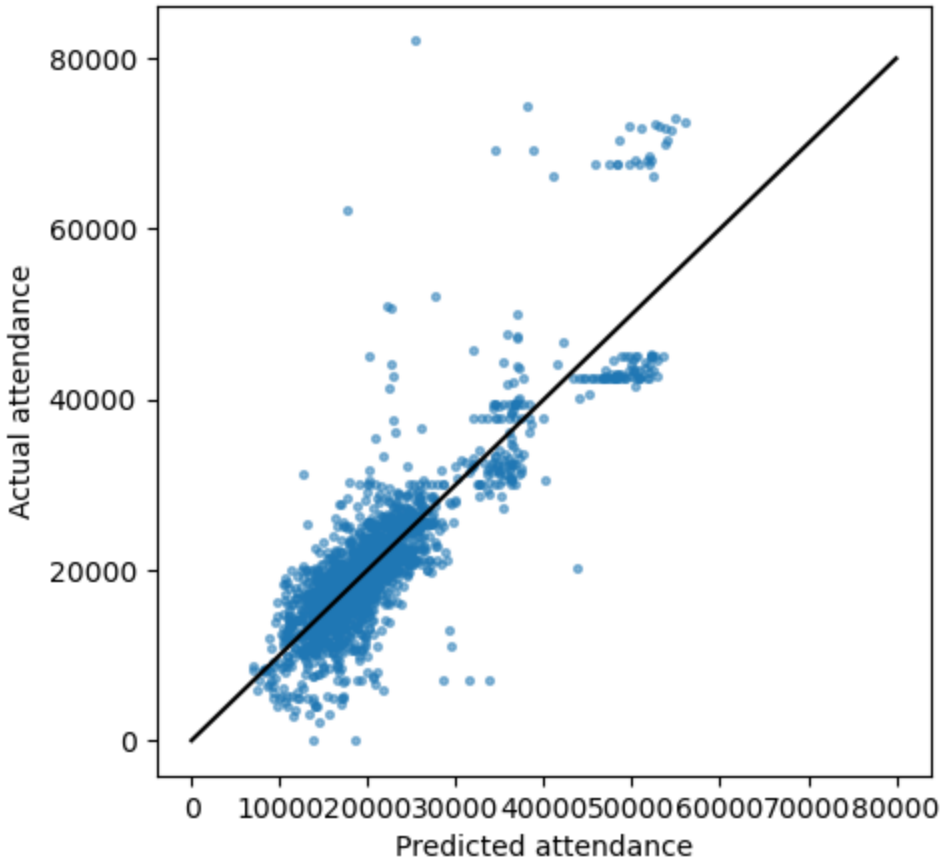
OLS Regression Results

Dep. Variable:	attendance	**R-squared:**	0.728
Model:	OLS	**Adj. R-squared:**	0.719
Method:	Least Squares	**F-statistic:**	82.59
Date:	Sun, 26 Nov 2023	**Prob (F-statistic):**	0.00
Time:	12:49:24	**Log-Likelihood:**	-22102.
No. Observations:	2229	**AIC:**	4.435e+04
Df Residuals:	2158	**BIC:**	4.475e+04
Df Model:	70		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
date_month	543.4748	49.962	10.878	0.000	445.495	641.454

In [11]:
```
evaluate_linear_model(linreg, X, y)
```

```
RMSE of residuals: 4978.200192318331
R-squared: 0.7281977294597555
```



The fit improved a bit. The R-squared is now 0.728 and the RMSE is 4,978.

Day of the week: All of the parameters for day of the week were negative, indicating that Saturday is the best day to hold matches. However, only Tuesday and Wednesday had p-values below 0.05. The result for Wednesday is quite telling. MLS holds a lot of matches on Wednesdays, but the model here indicates that those matches expect over 3000 fewer attendees than on Saturdays.

Month: The one month parameter is statistically significant according to the p-value. The value of the parameter is 543, which means that attendance goes up by 543 people on average for every month that goes by. That is a significant increase for a season that spans 11 months. However, this number might be biased high because the model does not yet account for the effect of playoff matches which happen at the end of the season. Once that is incorporated into the model, I expect the value of the month parameter to decrease a bit.

Kick off time: None of the parameters were statistically significant at the 0.05 level. The numbers indicate that matches played between 2 and 5 did the best. It is important to keep in mind that almost all of the matches like this were played on the weekend.

Year: Both 2021 and 2022 had significantly lower attendance than 2018. The effect of the pandemic was quite clear in 2021 when the average attendance was about 6000 lower than

Model 4

This model incorporates a few binary factors:

1. Playoff or not
2. Home opener or not
3. Rivals or not
4. Real home team or not

For now, I am going to keep all of the parameters from Model 3, but I am going to keep my eye on the local_time variables to see if they are significant or not.

In [12]:
```
X = create_X(mlsall_df,
             columns=['home_team','away_team',
                      'day','date_month','local_time','date_year',
                      'playoff','home_opener','rivals','real_home_team'],
             add_constant=True)
linreg = sm.OLS(y,X).fit()
linreg.summary()
```

Out[12]:

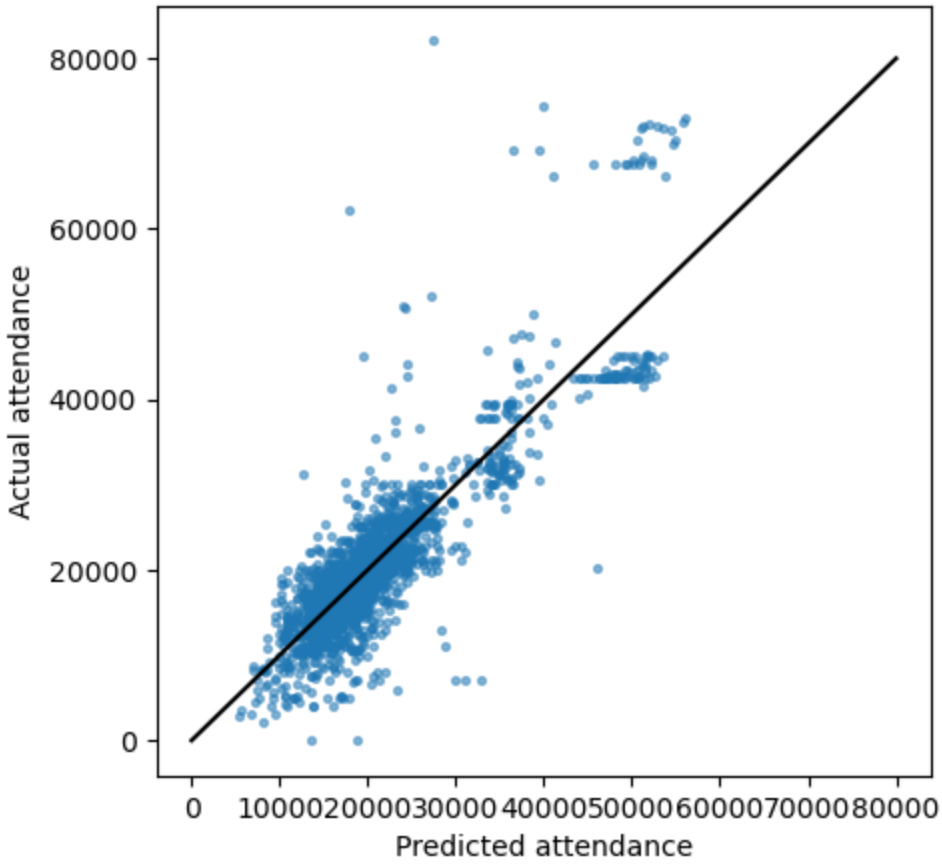
OLS Regression Results

Dep. Variable:	attendance	**R-squared:**	0.739
Model:	OLS	**Adj. R-squared:**	0.730
Method:	Least Squares	**F-statistic:**	82.30
Date:	Sun, 26 Nov 2023	**Prob (F-statistic):**	0.00
Time:	12:49:24	**Log-Likelihood:**	-22058.
No. Observations:	2229	**AIC:**	4.427e+04
Df Residuals:	2154	**BIC:**	4.469e+04
Df Model:	74		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
date_month	581.6798	55.051	10.566	0.000	473.722	689.638

In [13]:
```
evaluate_linear_model(linreg, X, y)
```

```
RMSE of residuals: 4885.263804900885
R-squared: 0.7387365449922851
```



The fit improved slightly. The R-squared is up to 0.739 and the RMSE is down to 4,885.

Playoff: I was surprised that the parameter for indicating whether a match was a playoff match was not significant at the 0.05 level.

Home openers: The parameter for home openers was significant and it indicates that home openers attract about 2,200 more people than would otherwise be expected.

Rivals: This parameter was also significant. Rivalry matches attract about 2,300 more people on average. Certain teams in MLS don't have rivalries, yet. Can we get Minnesota and Nashville to hate each other?

Real home team: Matches in which the home team was actually the home team averaged nearly 7,100 more fans.

The parameter for month is actually higher now (582).

Wednesdays and Thursdays are both statistically worse than Saturdays for attendance and the effect is quite large. Both get over 2000 fewer people per match.

None of the parameters for kick off time are significant still.

Model 5

This model will incorporate weather parameters

1. Whether it rained prior to match or at start of match.
2. Whether it snowed prior to match or at start of match.
3. Whether it was below 40 degrees Fahrenheit.
4. Whether it was above 90 degrees Fahrenheit.
5. Windspeed.

In [14]:
```
X = create_X(mlsall_df,
             columns=['home_team','away_team',
                      'day','date_month','local_time','date_year',
                      'playoff','home_opener','rivals','real_home_team',
                      'rain','snow','temperature','windspeed'],
             add_constant=True)
linreg = sm.OLS(y,X).fit()
linreg.summary()
```

Out[14]:

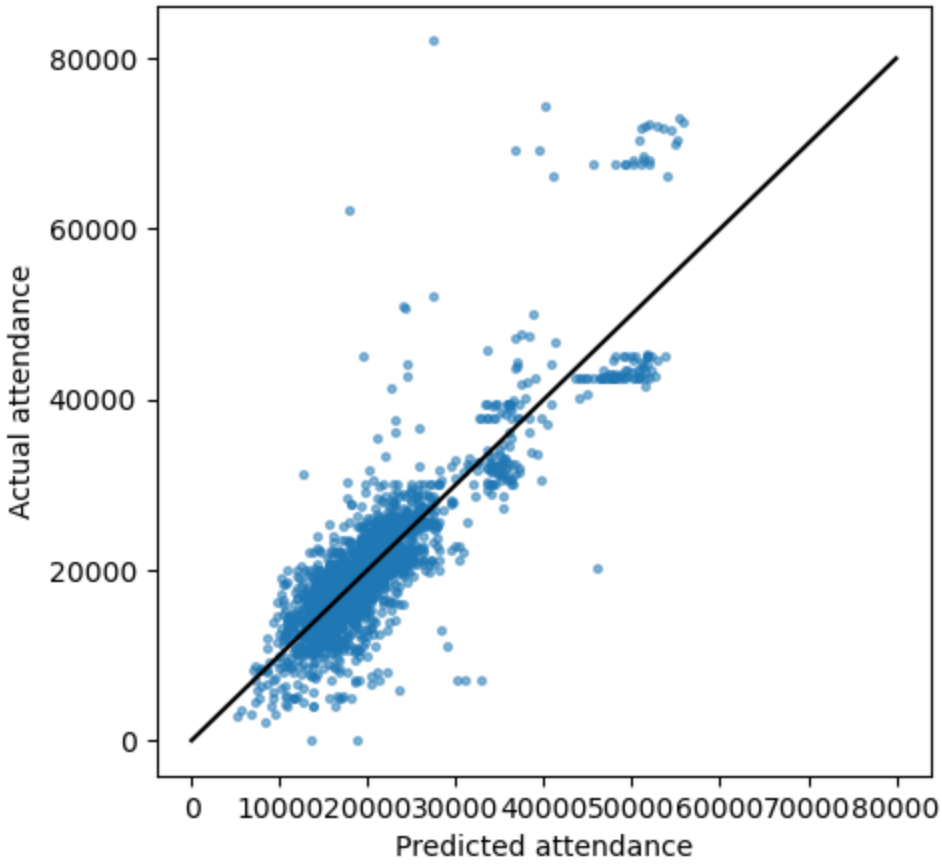
OLS Regression Results

Dep. Variable:	attendance	**R-squared:**	0.739
Model:	OLS	**Adj. R-squared:**	0.729
Method:	Least Squares	**F-statistic:**	77.02
Date:	Sun, 26 Nov 2023	**Prob (F-statistic):**	0.00
Time:	12:49:25	**Log-Likelihood:**	-22057.
No. Observations:	2229	**AIC:**	4.427e+04
Df Residuals:	2149	**BIC:**	4.473e+04
Df Model:	79		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
date_month	572.1905	56.687	10.094	0.000	461.023	683.358

In [15]:
```
evaluate_linear_model(linreg, X, y)
```

```
RMSE of residuals: 4888.42769570239
R-squared: 0.7390052731571286
```



There was no improvement to the fit and no indication that any of the weather parameters had an effect on the attendance. Weather parameters will not be included in the fit for subsequent models.

Model 6

For this model, I am going to remove parameters that were not significant. This means I will remove the following:

1. All weather parameters.
2. Kick off time parameters.

In [16]:
```
X = create_X(mlsall_df,
             columns=['home_team','away_team',
                      'day','date_month','date_year',
                      'playoff','home_opener','rivals','real_home_team'],
             add_constant=True)
linreg = sm.OLS(y,X).fit()
linreg.summary()
```

Out[16]:

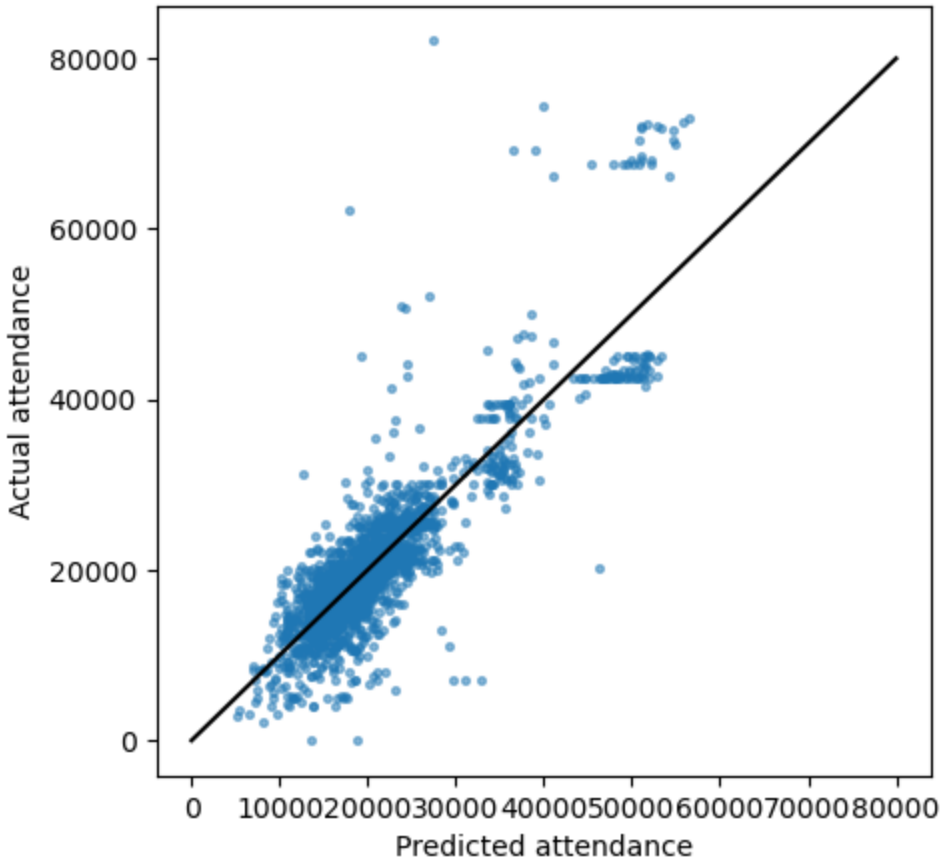
OLS Regression Results

Dep. Variable:	attendance	**R-squared:**	0.738
Model:	OLS	**Adj. R-squared:**	0.730
Method:	Least Squares	**F-statistic:**	85.76
Date:	Sun, 26 Nov 2023	**Prob (F-statistic):**	0.00
Time:	12:49:26	**Log-Likelihood:**	-22059.
No. Observations:	2229	**AIC:**	4.426e+04
Df Residuals:	2157	**BIC:**	4.467e+04
Df Model:	71		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
date_month	585.3170	54.818	10.678	0.000	477.816	692.818

In [17]:
```
evaluate_linear_model(linreg, X, y)
```

```
RMSE of residuals: 4884.843861590792
R-squared: 0.7384176459978459
```



This model has an R-squared of 0.738 and the RMSE is 4,885.

This is the best performing linear model. Below, I dig into the results a little more to see how well the model performed.

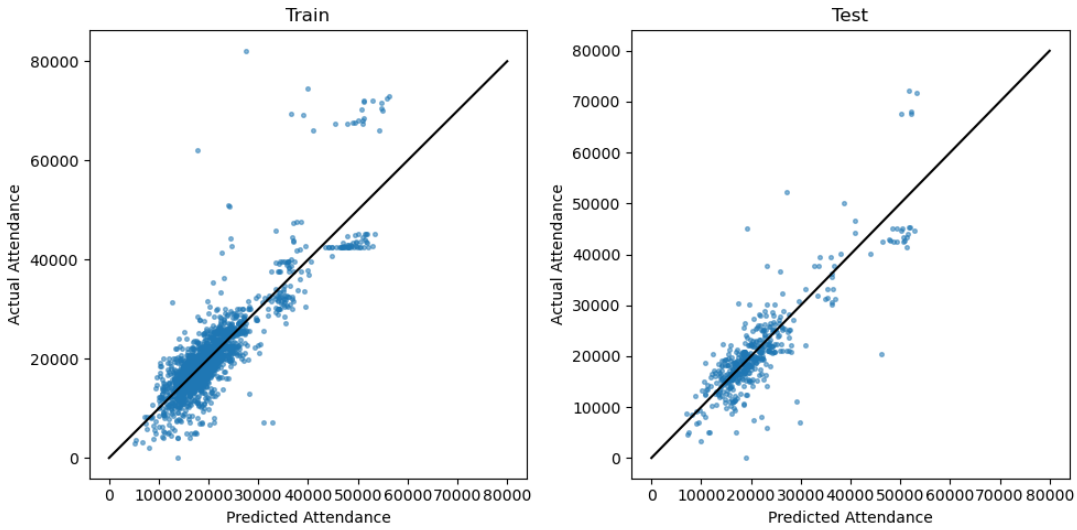
First, I am going to be using a train-test split for the other types of models I will try. In order to make a more direct comparison to those models, I would like to evaluate the linear regression model on those splits.

In [18]:
```
# Set random seed that will be used for ALL train-test splits
rando = 23
```

In [19]:
```
# Split the data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=r
```

In [20]:
```
evaluate_model_split(linreg, X_train, y_train, X_test, y_test)
```

```
RMSE:
Train: 4752.0
Test: 5012.6

R-squared:
Train: 0.74
Test: 0.733
```



Despite fitting all of the data without the split, there does not appear to be an issue with overfitting. This is not surprising as a linear regression model is likely too simple to overfit the data.

For the other types of models, the benchmark performance for the test data is the following:

RMSE = 5,012.6

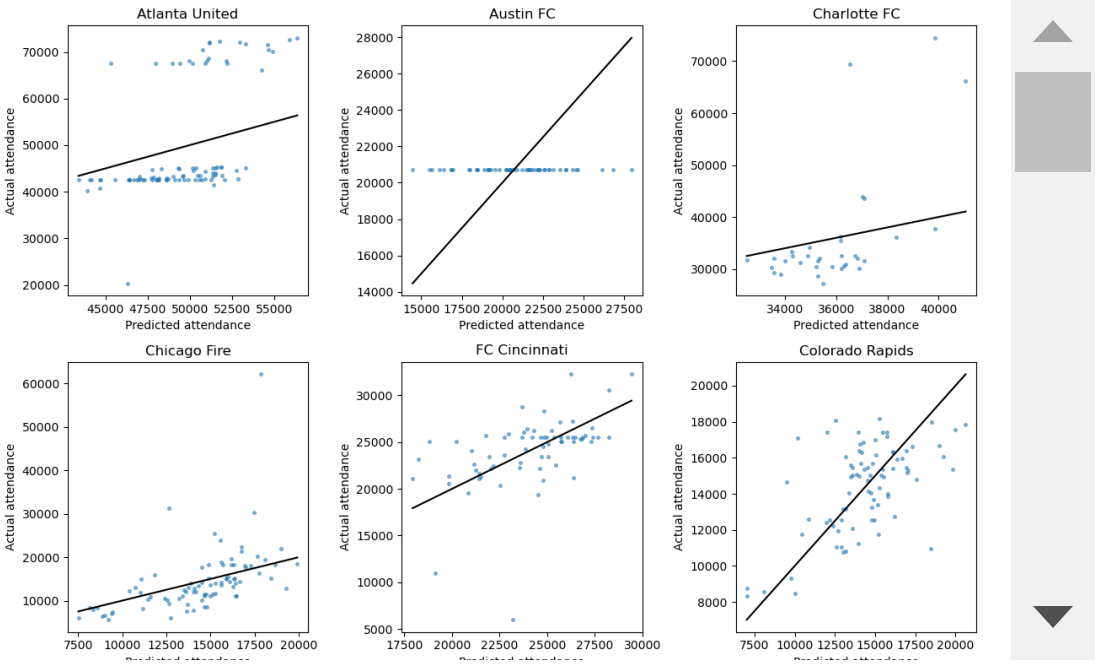
R-squared = 0.733

I want to look at how the fit performed on a team-by-team basis. The dictionary below turns the IDs in home_team and away_team into the actual team names.

In [21]:
```
team_names = {0: 'Atlanta United',
              1: 'Austin FC',
              2: 'Charlotte FC',
              3: 'Chicago Fire',
              4: 'FC Cincinnati',
              5: 'Colorado Rapids',
              6: 'Columbus Crew',
              7: 'FC Dallas',
              8: 'D.C. United',
              9: 'Houston Dynamo',
              10: 'Los Angeles Galaxy',
              11: 'Los Angeles FC',
              12: 'Minnesota United',
              13: 'Inter Miami',
              14: 'CF Montreal',
              15: 'Nashville SC',
              16: 'New England Revolution',
              17: 'New York City FC',
              18: 'New York Red Bulls',
              19: 'Orlando City',
              20: 'Philadelphia Union',
              21: 'Portland Timbers',
              22: 'Real Salt Lake',
              23: 'San Jose Earthquakes',
              24: 'Seattle Sounders',
              25: 'Sporting Kansas City',
              26: 'St. Louis FC',
              27: 'Toronto FC',
              28: 'Vancouver Whitecaps'}
```

In [22]:
```
# Get predictions from the model
y_pred_lr = linreg.predict(X)
```

In [23]:
```
# Plot actual attendance vs. predicted attendance for each team's home r
fig, ax = plt.subplots(ncols=3, nrows=10, figsize=(12,40))

for ht in range(29):
    r = ht//3
    c = ht%3

    y_pred_ht = y_pred_lr[mlsall_df['home_team']==ht]

    ax[r,c].scatter(y_pred_ht,
                    y[mlsall_df['home_team']==ht],
                    s=7,
                    alpha=0.5)

    ax[r,c].set_title(team_names[ht])
    ax[r,c].set_xlabel('Predicted attendance')
    ax[r,c].set_ylabel('Actual attendance')

    ax[r,c].plot([np.min(y_pred_ht),np.max(y_pred_ht)],
                 [np.min(y_pred_ht),np.max(y_pred_ht)],
                 color='black')

fig.tight_layout()
```



The predictions are quite solid for some teams, but not for others.

The model struggles the most with teams that change the capacity of their stadiums. Atlanta United is a good example. Usually, they use a capacity around 45,000, but then expand it to over 70,000 for other matches. The changing capacity means that the model struggles to fit the data which is grouped into two parts of the graph. In the case of Atlanta, I can tell pretty clearly what capacity was being used in a given match, but that is not the case for all teams.

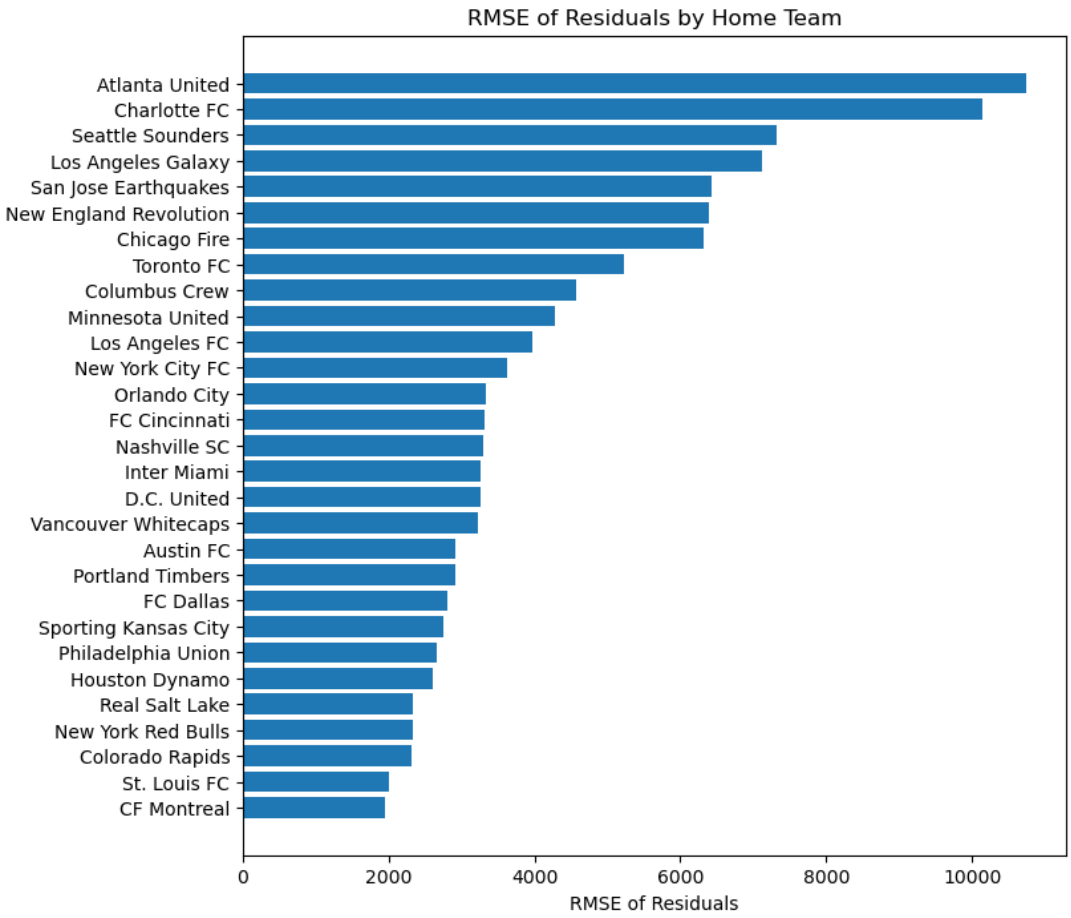
The artifically reduced capacity of Gillette Stadium, used by the New England Revolution, is listed as 20,000, but many of their matches have a higher attendance than that. In contrast to teams like Atlanta United, there seems to be a continuum of values above 20,000 rather than one unique, higher capacity. This implies that New England uses a less rigid capacity for its home matches, making it impossible to tell what capacity they used, if they used one at all.

The model also struggled with teams that had very consistent home attendance. The most obvious cases are Austin FC and St. Louis FC who have had the same (or nearly the same attendance) for every home match they have played. The model expects the attendance to vary from match to match depending on changes in the parameters, but there just isn't any variance for these teams. Similar issues are present for these teams that just have less variance in attendance: Los Angeles FC, Minnesota United, and Sporting Kansas City.

These teams outline a second problem with the model which is that there is a true maximum capacity for each stadium that cannot be exceeded, but the linear model does not

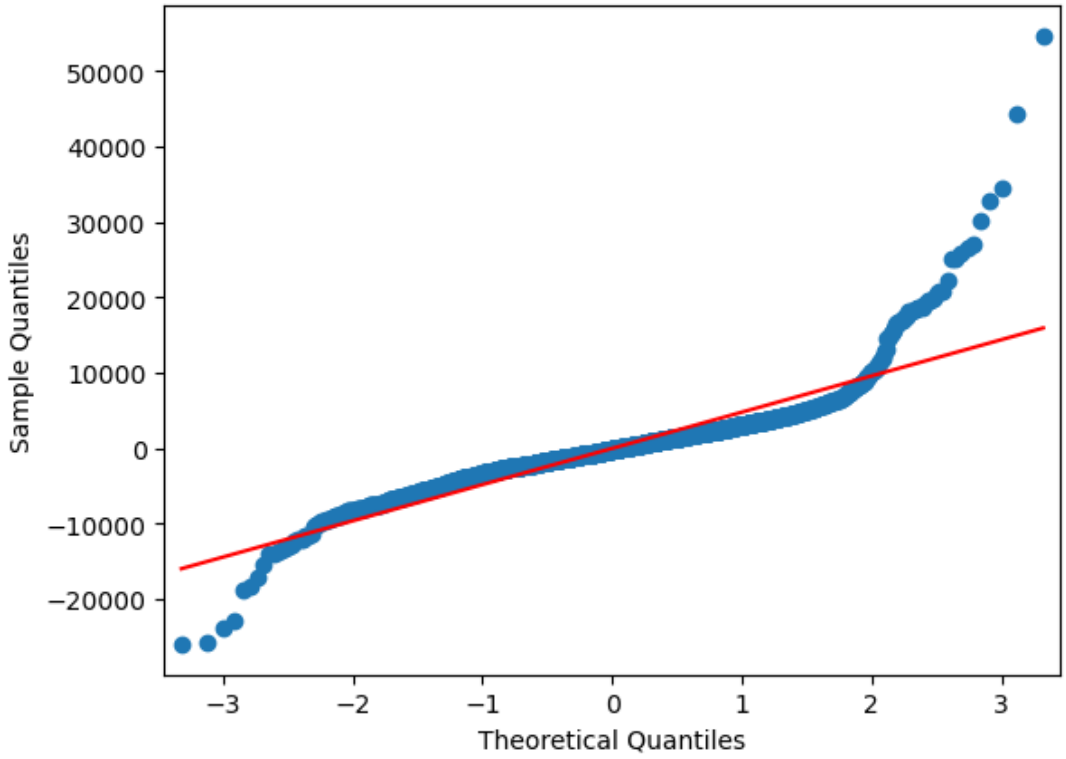
One metric for judging how well the model fits each team is the RMSE. Below, I plot the RMSE for each teams' home matches.

In [24]:
```
# RMSE of residuals for each home team
fig, ax = plt.subplots(figsize=(8,8))

rmse_team = []
for ht in range(29):
    rmse_team.append(np.sqrt(np.var(linreg.resid[mlsall_df['home_team']=

ax.barh(y=np.arange(29), width=np.sort(rmse_team))

ax.set_yticks(np.arange(29))
ax.set_yticklabels([team_names[x] for x in np.argsort(rmse_team)])
ax.set_xlabel('RMSE of Residuals')
ax.set_title('RMSE of Residuals by Home Team');
```



Unsurprisingly, the highest RMSE values are for the 3 teams that often change their stadium capacity (Atlanta, Charlotte, and Seattle). For 19 of the 29 teams, the RMSE is below 4,000.

In [25]:
```
# Make Quantile-Quantile plot to compare to normal distribution
sm.qqplot(linreg.resid,line='s');
```



The Q-Q plot above shows that the distribution of residuals is close to normal except at the edges.

XGBoost Regression

Next, I use an extreme gradient boosting regression model, XGBRegressor.

Model 1

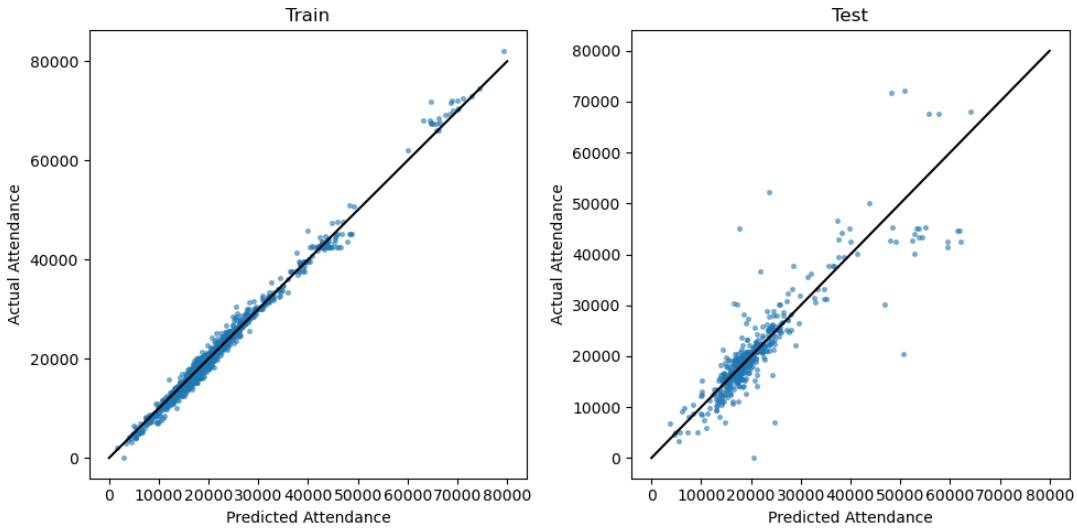
First, I want to see how well XGBRegressor performs with the default hyperparameters.

I am going to include all of the possible columns in my input data.

In [26]:
```
# Create input DataFrame, X
X = create_X(mlsall_df, columns=['home_team','away_team',
                                 'day','date_month','date_year','local_t
                                 'playoff','home_opener','rivals','real_
                                 'temperature','rain','snow','windspeed
             drop_first=False)
# Apply the same train-test split that was used earlier
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=r
```

In [27]:
```
# Instantiate XGBClassifier
xgb = XGBRegressor()

# Fit model
xgb.fit(X_train, y_train)

# Evaluate XGBRegressor model
evaluate_model_split(xgb, X_train, y_train, X_test, y_test)
```

```
RMSE:
Train: 971.3
Test: 4814.8

R-squared:
Train: 0.989
Test: 0.754
```



The model is overfitting quite a bit as the R-squared value is 0.989 for the train dataset, but just 0.754 for the test dataset. The RMSE is also higher for the test data. Some hyperparameters will need to be adjusted to improve the fit.

Both the RMSE and R-squared values are better for the test data than they were for the linear regression model. Below, I start adjusting hyperparameters to improve the model further.

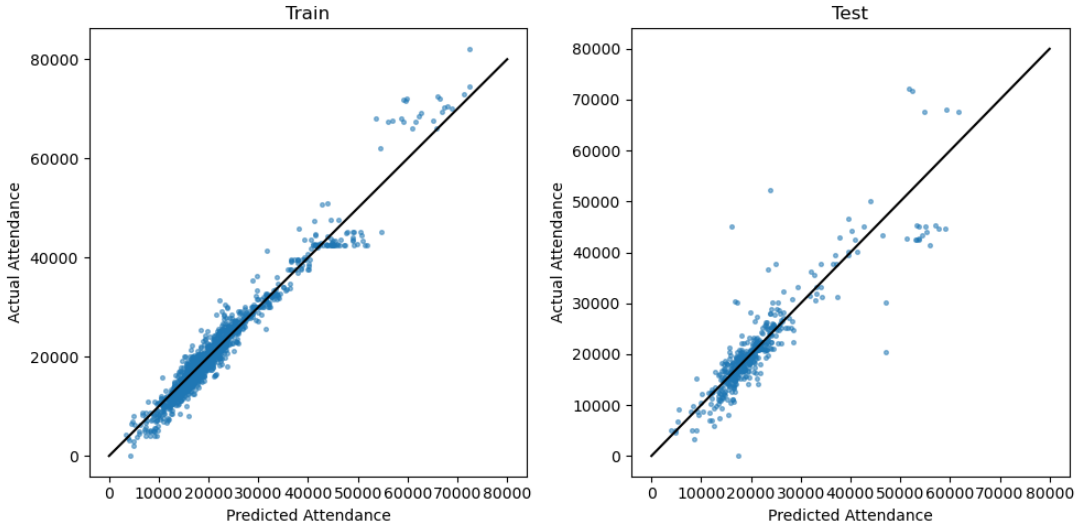
Model 2

Below, I use a custom function, xgb_gridsearch, which can be found in model_funcs.py. First, I just want to try out different tree depths and number of estimators.

In [28]:
```
# Create parameter grid
param_grid_xgb = {
    'max_depth': [3,4,5,6,7,8,9,None],
    'n_estimators': [50,100,200,250,300,350,400],
    'booster': ['gbtree']
}

# Search for optimal parameter combination
xgb_best, best_params, params, scores = model_gridsearch('xgb',param_gr
```

In [29]:
```
# Evaluate XGBRegressor model
print(best_params)
evaluate_model_split(xgb_best, X_train, y_train, X_test, y_test)
```

```
{'max_depth': 4, 'n_estimators': 100, 'booster': 'gbtree'}
RMSE:
Train: 2026.9
Test: 4467.4

R-squared:
Train: 0.953
Test: 0.788
```



The best model had a max_depth of 4 and 100 estimators.

The performance improved a bit from the last model. The RMSE and R-squared are now 4,467.4 and 0.788, respectively.

The model did overfit as the metrics for the training data was significantly better than the test data. Some regularization might come in handy.

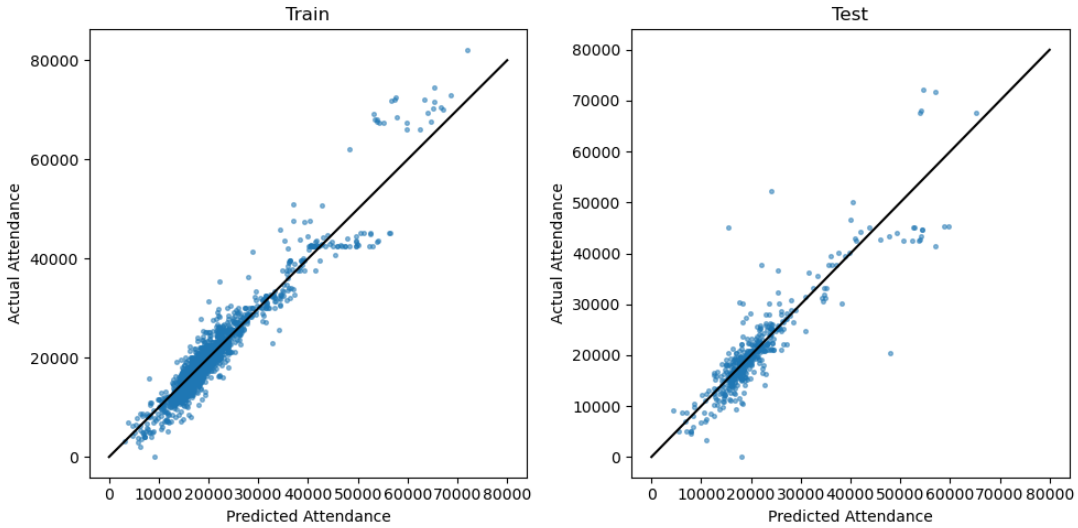
Model 3

Next, I introduce learning_rate, reg_lambda, and subsample to the parameter grid. The hope is that reg_lambda and subsample will help with the overfitting.

In [30]:
```
# Create parameter grid
param_grid_xgb = {
    'max_depth': [3,4,5,6],
    'n_estimators': [25,50,75,100,125,150,200,250,300],
    'booster': ['gbtree'],
    'learning_rate':[0.3,0.4,0.5,0.6],
    'reg_alpha':[0.01,0.1,1,10],
    'subsample':[0.7,0.8,0.9,1]
}

# Search for optimal parameter combination
xgb_best, best_params, params, scores = model_gridsearch('xgb',param_gr
```

In [31]:
```
# Evaluate XGBRegressor model
print(best_params)
evaluate_model_split(xgb_best, X_train, y_train, X_test, y_test)
```

```
{'max_depth': 4, 'n_estimators': 25, 'booster': 'gbtree', 'learning_ra
te': 0.6, 'reg_alpha': 10, 'subsample': 0.9}
RMSE:
Train: 2685.7
Test: 4408.7

R-squared:
Train: 0.917
Test: 0.793
```



The fit once again improved a bit. The new RMSE and R-squared values are 4,408.7 and 0.793.

The best parameter values were:

1. max_depth = 4
2. n_estimators = 25
3. learning_rate = 0.6
4. reg_alpha = 10
5. subsample = 0.9

The overfitting is still present.

Number of estimators

Below, I want to check to see how the performance changes as the number of estimators increases. All other hyperparameters will take on the values they had in the best model so far.

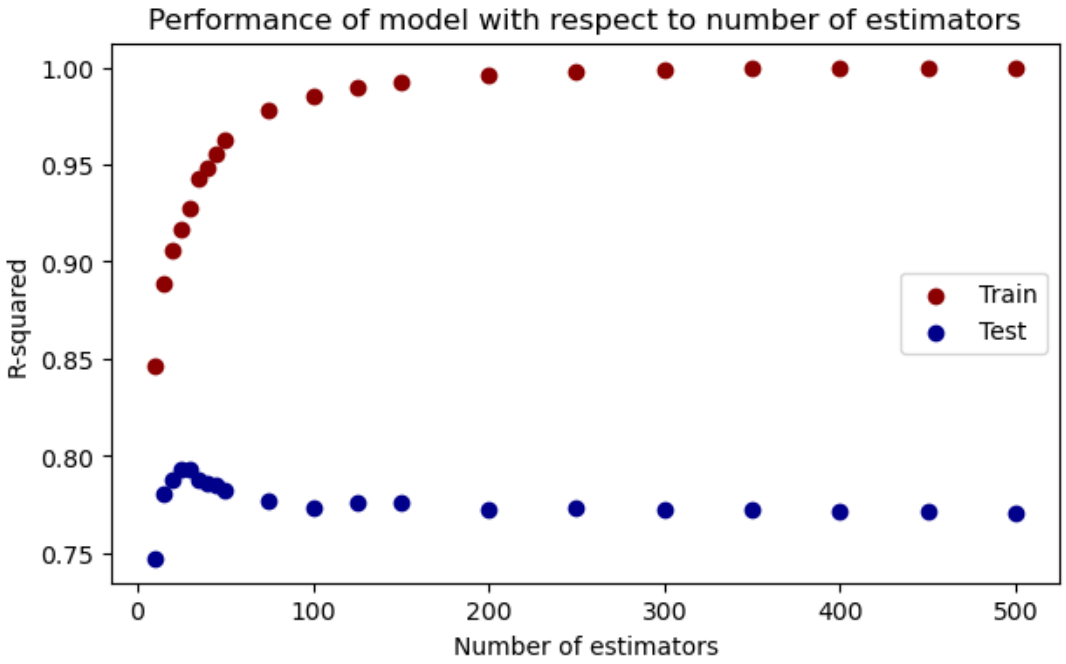
In [32]:
```
# Create parameter grid
param_grid_xgb = {
    'max_depth': [4],
    'n_estimators': [10,15,20,25,30,35,40,45,50,75,100,125,150,200,250,
    'booster': ['gbtree'],
    'learning_rate':[0.6],
    'reg_alpha':[10],
    'subsample':[0.9]
}

# Search for optimal parameter combination
xgb_best, best_params, params, scores = model_gridsearch('xgb',param_gr
```

In [33]:
```
best_params
```

Out[33]:
```
{'max_depth': 4,
 'n_estimators': 25,
 'booster': 'gbtree',
 'learning_rate': 0.6,
 'reg_alpha': 10,
 'subsample': 0.9}
```

In [34]:
```
# plot R-squared values vs. number of estimators
fig, ax = plt.subplots(figsize=(7,4))

ax.scatter(param_grid_xgb['n_estimators'],scores[:,0],label='Train', co
ax.scatter(param_grid_xgb['n_estimators'],scores[:,1],label='Test',colo
ax.set_xlabel('Number of estimators')
ax.set_ylabel('R-squared')
ax.set_title('Performance of model with respect to number of estimators

ax.legend()
```

Out[34]: <matplotlib.legend.Legend at 0x1e015423880>



The R-squared value for the test data maxes out at 0.793 when the number of estimators is 25. After that, the R-squared value begins to drop a bit.

Summary for XGBRegressor: The XGB model was able to outperform the linear regression model. The best R-squared value was just below 0.8 and the best RMSE was 4,408.7. Next, I will try random forest regression to see if it can outperform these metrics.

Random Forest Regression

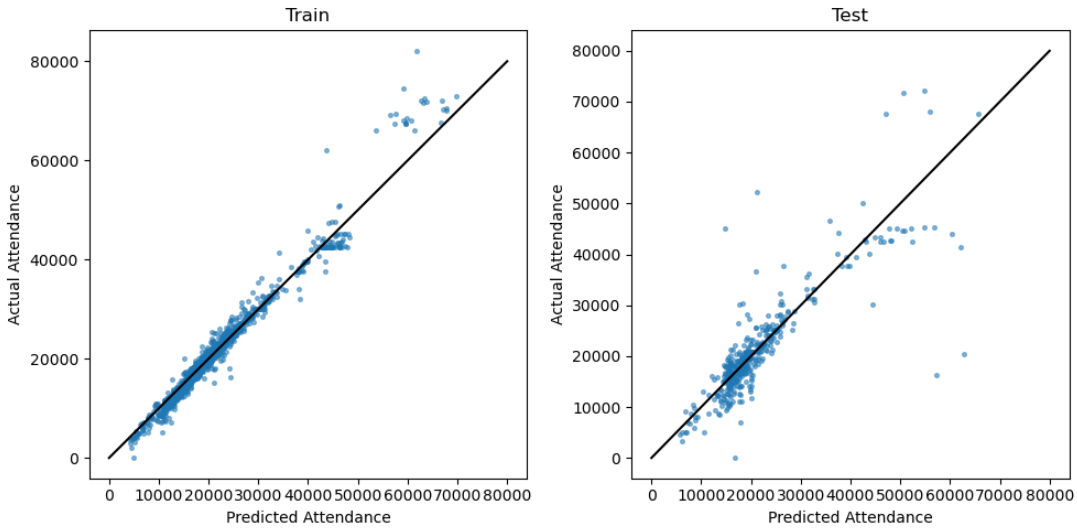
The random forest regressor will use the same training and test data as the XGB regressor.

Model 1

First, I will try the default hyperparameters for RandomForestRegressor.

In [35]:
```
# Create instance of RandomForestRegressor
rfr = RandomForestRegressor()

# Fit the model
rfr.fit(X_train, y_train)

# Evaluate the fit
evaluate_model_split(rfr, X_train, y_train, X_test, y_test)
```

```
RMSE:
Train: 1622.3
Test: 5137.9

R-squared:
Train: 0.97
Test: 0.72
```



The R-squared for this model is 0.725 and the RMSE is 5,084.0. Like the default XGBRegressor model, this is significantly overfitting.

Model 2

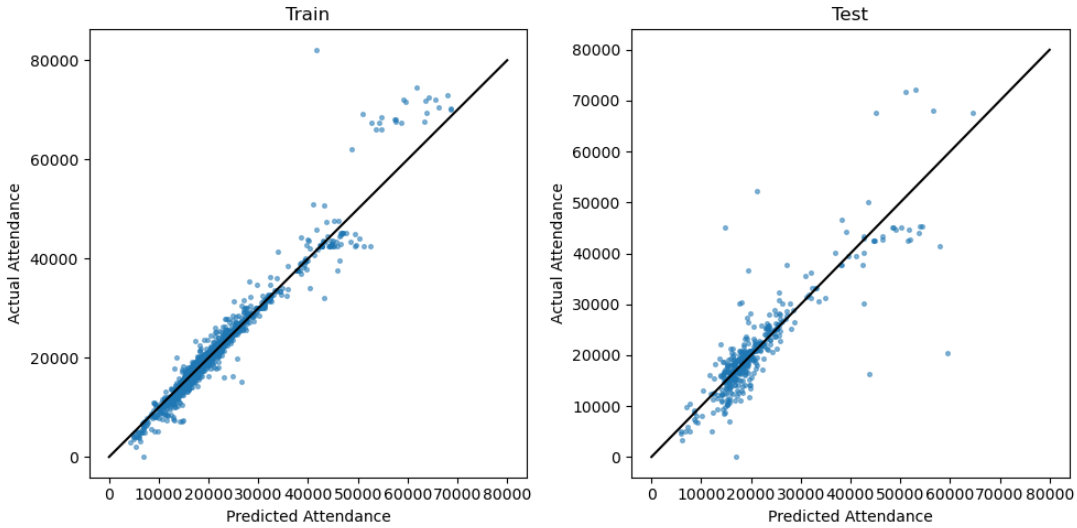
Next, I do a grid search on the following parameters:

1. Number of estimators
2. Max depth
3. Minimum samples to split a node

In [36]:
```
# Create parameter grid
param_grid_rfr = {
    'n_estimators': [5,10,15,20,25,30,40,50],
    'max_depth': [20,30,40,50,60,70,80],
    'min_samples_split': [2,3,4,5,6,7]
}

# Search for optimal parameter combination
rfr_best, best_params, params, scores = model_gridsearch('rfr', param_gr
```

In [37]:
```
print(best_params)
evaluate_model_split(rfr_best, X_train, y_train, X_test, y_test)
```

```
{'n_estimators': 15, 'max_depth': 40, 'min_samples_split': 3}
RMSE:
Train: 2100.5
Test: 4802.9

R-squared:
Train: 0.949
Test: 0.755
```



The fit on the test data did improve a bit. The RMSE is 4,650.8 and the R-squared is 0.77. The model is still overfitting quite a bit.

The performance of this model is worse than the best XGB model. This is not surprising since XGB also uses decision trees, but uses a more sophisticated algorithm to improve the fit each iteration.

K Nearest Neighbors

The last type of model I will try is a K-Nearest Neighbors model. Specifically, I will use KNeighborsRegressor.

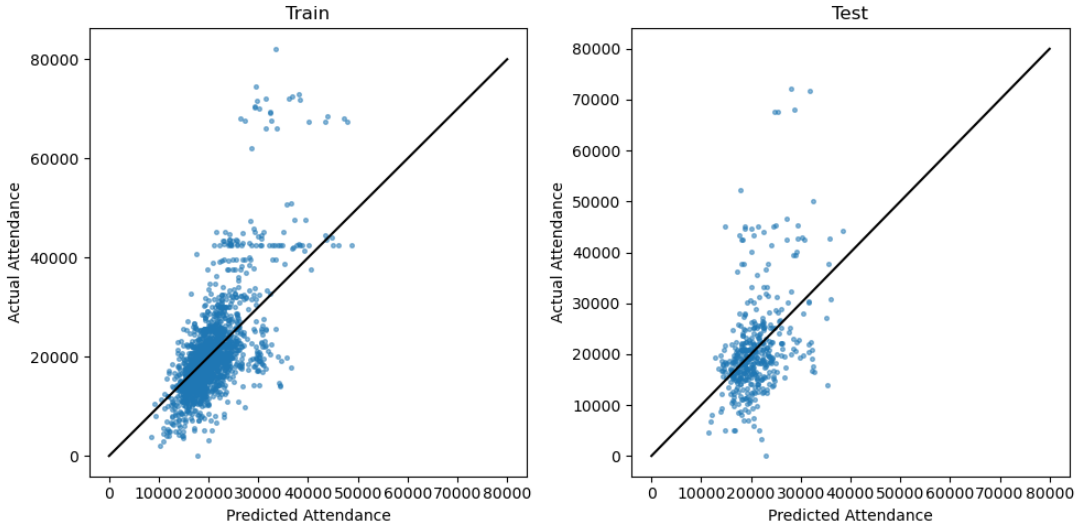
In theory, I think this type of model could be quite effective if there was enough data. However, I expect that this model will struggle to match the performance of the other models because certain features have less data coverage.

Model 1

I start with the default hyperparameters for KNeighborsRegressor.

In [38]:
```
# Create instance of KNeighborsRegressor
knn = KNeighborsRegressor()

# Fit the model
knn.fit(X_train, y_train)

# Evaluate the fit
evaluate_model_split(knn, X_train, y_train, X_test, y_test)
```

```
RMSE:
Train: 7040.4
Test: 8853.3

R-squared:
Train: 0.429
Test: 0.167
```



The model performed very poorly on both the training and test data. The hyperparameters need some tuning to improve the model.

Both the XGBRegressor and RandomForestRegressor performed quite well without tuning anything. This might confirm what I said above about this model struggling.

Model 2

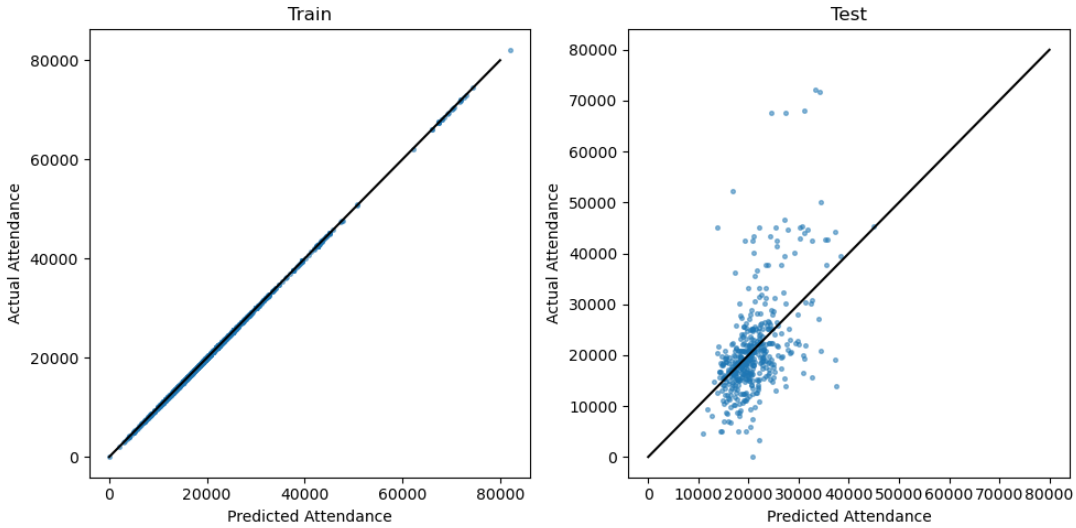
Next, I adjust values for the following parameters:

1. Number of neighbors
2. Minkowski metric exponent, p.
3. Whether to weight with distance or not.

In [39]:
```
# Create parameter grid
param_grid_knn = {
    'n_neighbors': [5,10,15,20,25,30,40,50],
    'p': [1,1.5,2,2.5,3],
    'weights': ['uniform','distance']
}

# Search for optimal parameter combination
knn_best, best_params, params, scores = model_gridsearch('knn', param_gr
```

In [40]:
```
# Evaluate the fit
print(best_params)
evaluate_model_split(knn_best, X_train, y_train, X_test, y_test)
```

```
{'n_neighbors': 5, 'p': 1, 'weights': 'distance'}
RMSE:
Train: 0.0
Test: 8131.9

R-squared:
Train: 1.0
Test: 0.297
```



The model suffers from severe overfitting and the performance on the test data is still very bad (RMSE=8,131.9 and R-squared=0.297). Since I already have other types of models that perform well, I am going to stop here for K-Nearest Neighbors.

Summary of Models

The best performing instance of each model is summarized in the table below.

Model	RMSE (Test)	R^2 (Test)	RMSE (Training)	R^2 (Training)
XGBoost	4,408.7	0.793	2,685.7	0.917
Random Forest	4,650.8	0.770	2,289.8	0.940
Linear Regression	5,012.6	0.733	4,752.0	0.740
K-Nearest Neighbors	8,131.9	0.207	0.0	0.000

The RMSE and R^2 values are for the test dataset which comprises 20% of the data. For each model except linear regression, the model was trained on the other 80% of the data. The train-test split was not used for linear regression because there was no concern that the model would overfit.

Overall performance

The first three types of models (linear regression, XGBoost, and Random Forest) each achieved RMSE values around or below 5,000 and R-squared values between 0.73 and 0.8. However, the K-Nearest Neighbors model struggled to match this level of performance. There might have been some room for improvement by tweaking the scaling of the input variables, but I doubt the predictions would have been as good as the other three.

As stated at the top of this notebook, the match-by-match predictions are not stellar. The best RMSE was around 4,400. This means that we expect a typical prediction to be off by about this much on average. This is not a problem, though. The goal of this project was to look for trends that MLS could use to improve attendance over the course of a season, not to target specific matches to try to improve attendance. There are trends that can be useful which will be discussed in the **Exploring the Model** section of this notebook.

The Best Performing Model

The model that performed the best based on having the lowest RMSE and highest R-squared was the XGBoost Regression model. The model did suffer from some overfitting,

Exploring the Model

In this section, I primarily explore the best-fitting XGBoost model, but I am also going to mention the linear regression model sometimes for comparison.

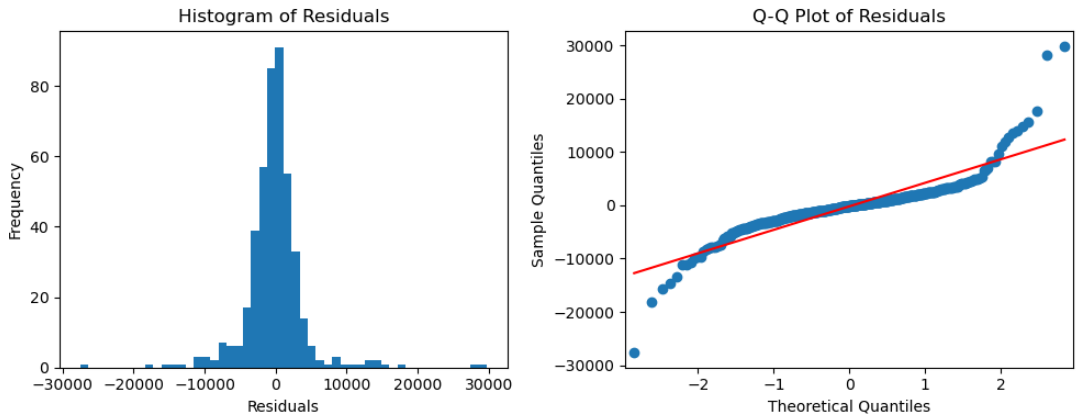
Residuals

First, I want to look into the residuals. I already know that the RMSE is around 4,400 for the best model, but I want to look at how it is distributed.

In [41]:
```
# I will use the blue and red from the MLS logo in some figures
mls_blue = '#001F5B'
mls_red = '#DF231A'
```

In [42]:
```
# Calculate residuals of test data
xgb_resid = y_test-xgb_best.predict(X_test)

# Kurtosis
print(f"Kurtosis of Residuals: {sp.kurtosis(xgb_resid)}")
print(f"Proportion of predictions within 1000 of actual value: {np.sum(
print(f"Proportion of predictions within 2000 of actual value: {np.sum(
print(f"Proportion of predictions within 3000 of actual value: {np.sum(
print(f"Proportion of predictions within 4000 of actual value: {np.sum(

# Plot residuals and a Q-Q plot
fig, ax = plt.subplots(ncols=2, figsize=(10,4))

# Residual distribution
ax[0].hist(xgb_resid, bins=50)
ax[0].set_xlabel('Residuals')
ax[0].set_ylabel('Frequency')
ax[0].set_title('Histogram of Residuals')

# Q-Q plot
sm.qqplot(xgb_resid,line='s',ax=ax[1])
ax[1].set_title('Q-Q Plot of Residuals')

fig.tight_layout();
```

```
Kurtosis of Residuals: 13.481246785624222
Proportion of predictions within 1000 of actual value: 0.3452914798206
278
Proportion of predictions within 2000 of actual value: 0.6008968609865
47
Proportion of predictions within 3000 of actual value: 0.7399103139013
453
Proportion of predictions within 4000 of actual value: 0.8363228699551
569
```



The distribution of the residuals appears pretty symmetric, but more centrally concentrated than a normal distribution. This is verified by the kurtosis value of 13.5. A normal distribution would have a kurtosis of 0, while positive values imply more central concentration.

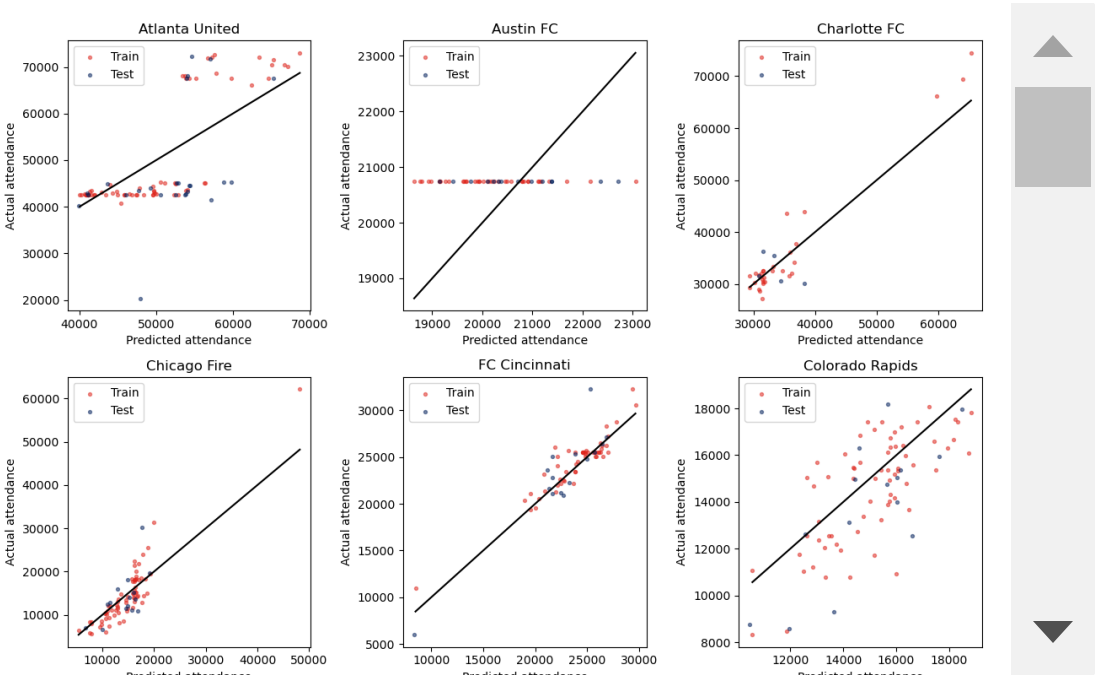
The Q-Q plot on the right shows how there are some large outliers on the edges. This is seen in the histogram as there are some residuals that are above 10,000, even as high as almost 30,000. However, the vast majority of the residuals are smaller than that. Even

Performance by Team

Below, I plot the actual attendance vs. the predicted attendance for each team's home matches. The goal is to see whether the model performed consistently well for each team, or if it was better for some teams compared to others.

In [43]:

```
# Get predictions from the best XGB model
y_pred_train_xgb = xgb_best.predict(X_train)
y_pred_test_xgb = xgb_best.predict(X_test)

# Plot actual attendance vs. predicted attendance for each team's home m
fig, ax = plt.subplots(ncols=3, nrows=10, figsize=(12,40))

for ht in range(29):
    r = ht//3
    c = ht%3

    train_select = X_train[f"home_team_{ht}"]==1
    test_select = X_test[f"home_team_{ht}"]==1

    # Training data
    ax[r,c].scatter(y_pred_train_xgb[train_select],
                    y_train[train_select],
                    s=8,
                    alpha=0.5,
                    label='Train',
                    color=mls_red)

    # Test data
    ax[r,c].scatter(y_pred_test_xgb[test_select],
                    y_test[test_select],
                    s=8,
                    alpha=0.5,
                    label='Test',
                    color=mls_blue)

    ax[r,c].set_title(team_names[ht])
    ax[r,c].set_xlabel('Predicted attendance')
    ax[r,c].set_ylabel('Actual attendance')

    # Plot x=y line
    ax[r,c].plot([np.min(y_pred_train_xgb[train_select]),np.max(y_pred_t
                 [np.min(y_pred_train_xgb[train_select]),np.max(y_pred_t
                 color='black')

    ax[r,c].legend()

fig.tight_layout();
```

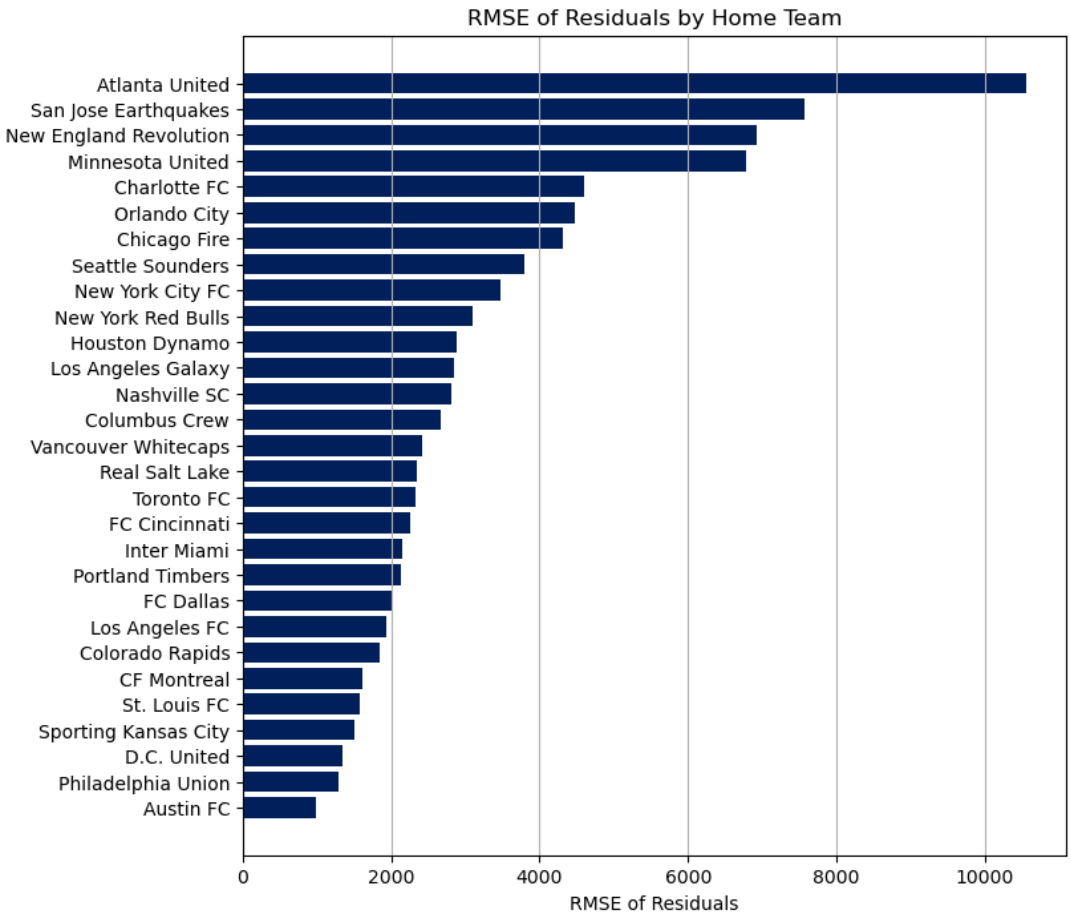


The first thing I notice is that the model struggled with the teams who change the capacity of their stadiums, like Atlanta United (first plot in top left). Even on the training data, it tends to overestimate the attendance when the lower capacity is used and underestimate it when the larger capacity is used.

It can be a little hard to tell how good the different fits are because not all of the plots are on the same scale.

Below, I look at the RMSE of the residuals for each team's home matches. Only the matches in the test dataset are used.

In [44]:
```
# RMSE of residuals for each home team
fig, ax = plt.subplots(figsize=(8,8))

rmse_team = []
for ht in range(29):
    rmse_team.append(np.sqrt(np.var(xgb_resid[mlsall_df['home_team']==ht

ax.barh(y=np.arange(29), width=np.sort(rmse_team), color=mls_blue)
ax.grid(axis='x')

ax.set_yticks(np.arange(29))
ax.set_yticklabels([team_names[x] for x in np.argsort(rmse_team)])
ax.set_xlabel('RMSE of Residuals')
ax.set_title('RMSE of Residuals by Home Team');
```



Not surprisingly, the highest RMSE was for Atlanta United whose trouble fitting I discussed earlier.

The second highest RMSE belongs to San Jose, but I think that is primarily caused by a single match that had a much higher attendance than predicted because the match was staged in a special location with much higher capacity.

The third highest RMSE belongs to New England. This also changes its capacity from match to match. The only other team with an RMSE above 5,000 was Minnesota United that also had a match with a higher than normal capacity.

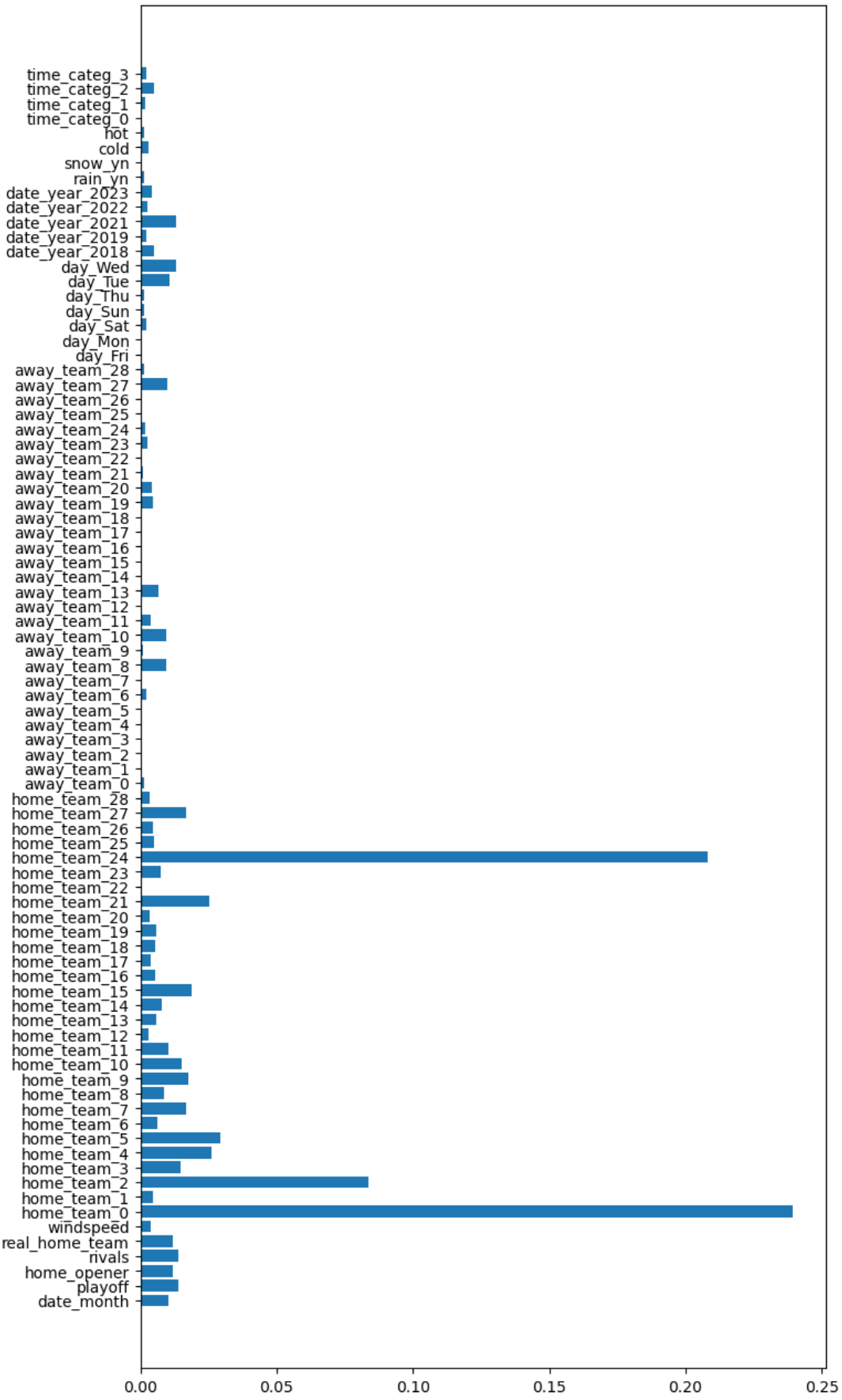
Other than the 4 teams with the highest RMSE values, all the others have RMSE values below 5,000 and most are at 3,000 or below.

Feature Importance

The main goal of this project was to figure out what factors had the greatest effect on attendance. I start by looking at the feature importance calculated by XGBoost below.

In [45]:

```
# Feature importance bar chart
fig, ax = plt.subplots(figsize=(8,16))

ax.barh(y=range(len(xgb_best.feature_importances_)), width=xgb_best.fea

ax.set_yticks(range(len(xgb_best.feature_importances_)))
ax.set_yticklabels(xgb_best.feature_names_in_);
```


time_categ_3
time_categ_2
time_categ_1
time_categ_0
hot
cold
snow_yn
rain_yn
date_year_2023
date_year_2022
date_year_2021
date_year_2019
date_year_2018
day_Wed
day_Tue
day_Thu
day_Sun
day_Sat
day_Mon
day_Fri
away_team_28
away_team_27
away_team_26
away_team_25
away_team_24
away_team_23
away_team_22
away_team_21
away_team_20
away_team_19
away_team_18
away_team_17
away_team_16
away_team_15
away_team_14
away_team_13
away_team_12
away_team_11
away_team_10
away_team_9
away_team_8
away_team_7
away_team_6
away_team_5
away_team_4
away_team_3
away_team_2
away_team_1
away_team_0
home_team_28
home_team_27
home_team_26
home_team_25
home_team_24
home_team_23
home_team_22
home_team_21
home_team_20
home_team_19
home_team_18
home_team_17
home_team_16
home_team_15
home_team_14
home_team_13
home_team_12
home_team_11
home_team_10
home_team_9
home_team_8
home_team_7
home_team_6
home_team_5
home_team_4
home_team_3
home_team_2
home_team_1
home_team_0
windspeed
real_home_team
rivals
home_opener
playoff
date_month
0.00
0.05
0.10
0.15
0.20
0.25

The most important features are those related to which team is at home. Part of the reason for this is simply that some teams have larger stadiums. The largest value in the plot above belongs to home_team_0, which is Atlanta United who play their matches in a big NFL stadium.

The trouble with the built-in feature_importances that are provided by the model is that they are normalized to add to 1 and it is hard to tell what actual effect each feature has on the attendance, especially because it does not indicate whether it has a positive or negative effect. I wrote my own function, get_feature_importance, that will attempt to measure the effect of each feature in more practical terms. The way I do this is by using the model on fake data, changing the input to see how the average attendance changes.

The fake data is generated below using the gen_fake_data function. The data contains each possible matchup between the 29 teams in MLS so that everyone plays everyone else once at home and once away (812 matches). The matches all assume the same default values for the other inputs at first. Those defaults are:

1. Match held on a Saturday in July, 2018 at 7:30 pm.
2. Not a playoff match, home opener, or rivalry match.
3. No rain or snow and the temperature is not considered hot or cold.
4. The real home team is present.

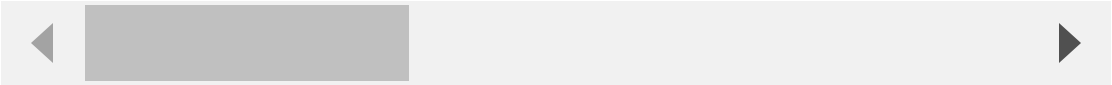
For each feature, I change the feature to compare it to the predicted attendance given the default values. For example, if I want to know the effect of playing matches on Wednesday instead of Saturday, I change all the matches to Wednesday and get the average of the predictions so it can be compared to the default average.

In [46]:
```
# Generate fake data
X_fake = gen_fake_data(X_train)
X_fake
```

Out[46]:

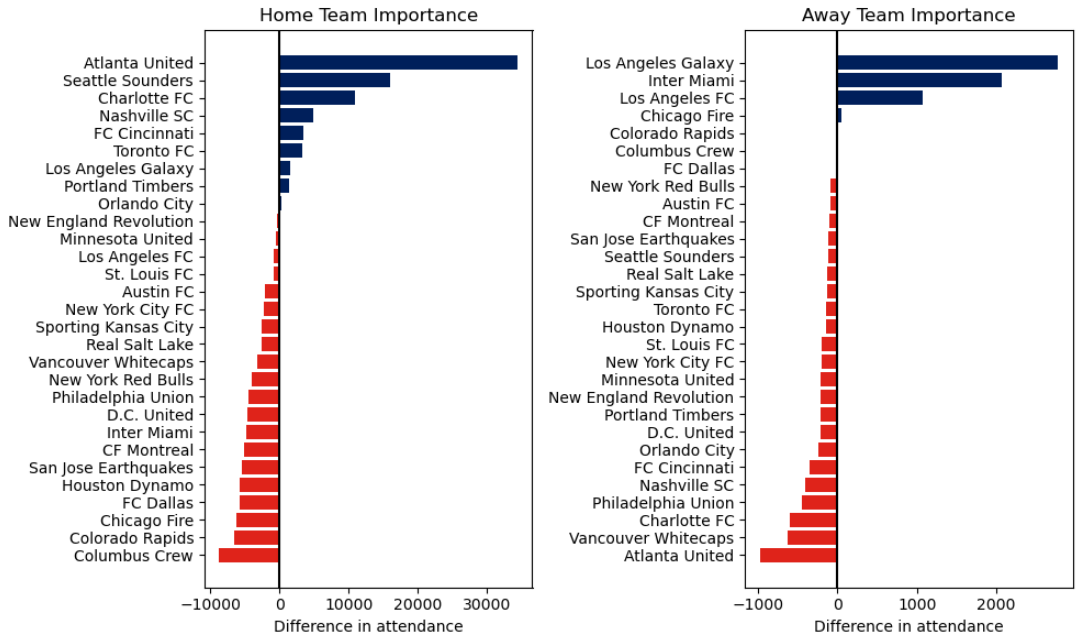
	date_month	playoff	home_opener	rivals	real_home_team	windspeed	home_team
1	0	0	0	0	1	11	
2	0	0	0	0	1	11	
3	0	0	0	0	1	11	
4	0	0	0	0	1	11	
5	0	0	0	0	1	11	
...	...	...	...	...	...	...	
835	0	0	0	0	1	11	
836	0	0	0	0	1	11	
837	0	0	0	0	1	11	
838	0	0	0	0	1	11	
839	0	0	0	0	1	11	

812 rows × 84 columns



In [47]:
```
# Get feature importance for the best XGB model
xgb_importance, xgb_import_ht = get_feature_importance(xgb_best, X_fake
```

In [48]:
```
# Home team and away team feature importance
fig, ax = plt.subplots(ncols=2, figsize=(10,6))

ht_importance = [xgb_importance[key] for key in xgb_importance.keys() i
ax[0].barh(y=range(29),
           width=np.sort(ht_importance),
           color=[mls_blue if x>=0 else mls_red for x in np.sort(ht_impo
ax[0].set_title('Home Team Importance')
ax[0].set_yticks(range(29))
ax[0].set_yticklabels([team_names[x] for x in np.argsort(ht_importance)]
ax[0].set_xlabel('Difference in attendance')
ax[0].axvline(0,color='black')

at_importance = [xgb_importance[key] for key in xgb_importance.keys() i
ax[1].barh(y=range(29),
           width=np.sort(at_importance),
           color=[mls_blue if x>=0 else mls_red for x in np.sort(at_impo
ax[1].set_title('Away Team Importance')
ax[1].set_yticks(range(29))
ax[1].set_yticklabels([team_names[x] for x in np.argsort(at_importance)]
ax[1].set_xlabel('Difference in attendance')
ax[1].axvline(0,color='black')

fig.tight_layout();
```



The bars in the graphs above show how much more or less attendance each team gets in its home matches (left) and away matches (right) compared to the average match.

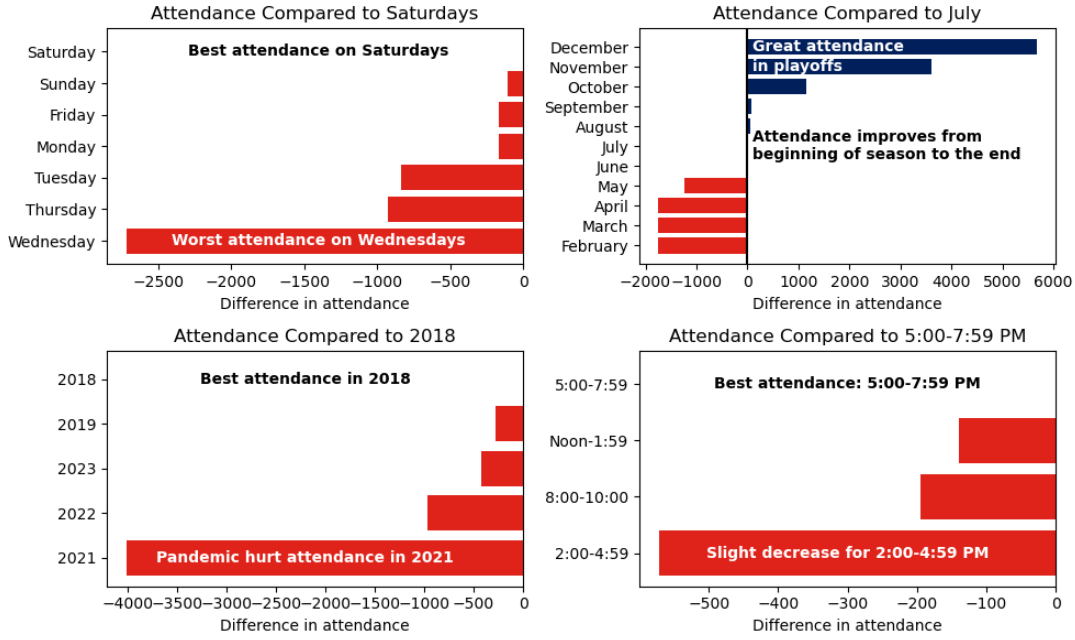
The graph on the left mostly accounts for differences in size between each teams' stadiums, it also incorporates information about how well those teams fill their stadiums. For example, the Columbus Crew get about 9,000 fewer people per match than the average. This is mostly because their stadium only holds 20,011 people, but also because the Columbus Crew don't always sell out their matches.

The three teams that tend to get the highest attendance when they go on the road are the two Los Angeles teams and Inter Miami.

In [49]:

```
#Day, month, year, and kick off time feature importance
fig, ax = plt.subplots(ncols=2, nrows=2, figsize=(10,6))

# Day of the week
day_names = ['Sunday','Monday','Tuesday','Wednesday','Thursday','Friday
day_importance = [xgb_importance[key] for key in xgb_importance.keys() :

ax[0,0].barh(y=range(7),
             width=np.sort(day_importance),
             color=[mls_blue if x>=0 else mls_red for x in np.sort(day_i
ax[0,0].set_yticks(range(7))
ax[0,0].set_yticklabels(day_names[x] for x in np.argsort(day_importance
ax[0,0].set_title('Attendance Compared to Saturdays')
ax[0,0].set_xlabel('Difference in attendance')
ax[0,0].text(-1400,6,'Best attendance on Saturdays',fontsize=10,fontweig
             horizontalalignment='center',verticalalignment='center')
ax[0,0].text(-1400,0,'Worst attendance on Wednesdays',fontsize=10,fontwe
             horizontalalignment='center',verticalalignment='center',co

# Month
month_names = ['February','March','April','May','June','July','August',
month_importance = [xgb_importance[key] for key in xgb_importance.keys(

ax[0,1].barh(y=range(11),
             width=np.sort(month_importance),
             color=[mls_blue if x>=0 else mls_red for x in np.sort(montl
ax[0,1].set_yticks(range(11))
ax[0,1].set_yticklabels(month_names[x] for x in np.argsort(month_importa
ax[0,1].axvline(0,color='black')
ax[0,1].set_title('Attendance Compared to July')
ax[0,1].set_xlabel('Difference in attendance')
ax[0,1].text(100,5,'Attendance improves from \nbeginning of season to th
             horizontalalignment='left',verticalalignment='center')
ax[0,1].text(100,10,'Great attendance',fontsize=10,fontweight='bold',
             horizontalalignment='left',verticalalignment='center',colo
ax[0,1].text(100,9,'in playoffs',fontsize=10,fontweight='bold',
             horizontalalignment='left',verticalalignment='center',colo

# Year
year_names = ['2018','2019','2021','2022','2023']
year_importance = [xgb_importance[key] for key in xgb_importance.keys()

ax[1,0].barh(y=range(5),
             width=np.sort(year_importance),
             color=[mls_blue if x>=0 else mls_red for x in np.sort(year_
ax[1,0].set_yticks(range(5))
ax[1,0].set_yticklabels(year_names[x] for x in np.argsort(year_importanc
ax[1,0].set_title('Attendance Compared to 2018')
ax[1,0].set_xlabel('Difference in attendance')
ax[1,0].text(-2200,4,'Best attendance in 2018',fontsize=10,fontweight='b
             horizontalalignment='center',verticalalignment='center')
ax[1,0].text(-2200,0,'Pandemic hurt attendance in 2021',fontsize=10,font
             horizontalalignment='center',verticalalignment='center',co

# Kick off time
kotime_names = ['Noon-1:59','2:00-4:59','5:00-7:59','8:00-10:00']
kotime_importance = [xgb_importance[key] for key in xgb_importance.keys
```

```
ax[1,1].barh(y=range(4),
             width=np.sort(kotime_importance),
             color=[mls_blue if x>=0 else mls_red for x in np.sort(kotir
ax[1,1].set_yticks(range(4))
ax[1,1].set_yticklabels(kotime_names[x] for x in np.argsort(kotime_impor
ax[1,1].set_title('Attendance Compared to 5:00-7:59 PM')
ax[1,1].set_xlabel('Difference in attendance')
ax[1,1].text(-300,3,'Best attendance: 5:00-7:59 PM',fontsize=10,fontweig
             horizontalalignment='center',verticalalignment='center')
ax[1,1].text(-300,0,'Slight decrease for 2:00-4:59 PM',fontsize=10,fontw
             horizontalalignment='center',verticalalignment='center',col

fig.tight_layout();
```



Day of the Week: All the bars showing the change in attendance compared to matches on Saturdays (hence why there is no bar for Saturday). The most important trend here is that Wednesday matches get over 2,700 fewer people than an average Saturday match. To some extent, MLS has to play some midweek matches, but there are changes they can make to reduce the number of those matches.

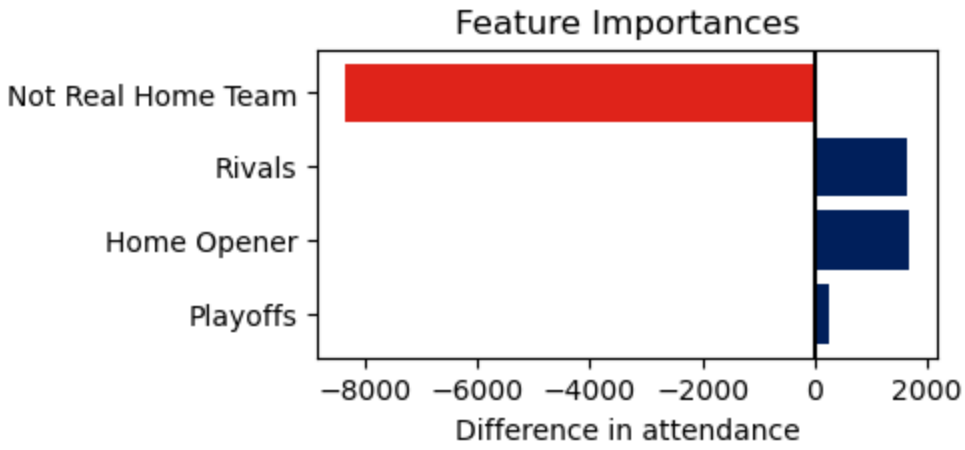
Month: There is a clear trend that attendance tends to improve as the season progresses from February to December. The most likely reason for this is that matches become more important as the playoff picture becomes clearer and then the playoffs actually start in late October. The big takeaway for me is that people want to see matches where the stakes are high.

Year: The fact that 2021 got about 4,000 fewer people per match was mostly due to the pandemic.

Kick Off Time: The best time to kick off a match according to the model is between 5:00 pm and 7:59 pm. However, the change in attendance does not appear to be huge as the kick off time changes. In fact, the linear regression model found similar numbers, but they were not

significant in that model

In [50]:
```
# Playoffs, home openers, rivals, and real home teams importance
fig, ax = plt.subplots(figsize=(4,2))

feat_importance = [xgb_importance[key] for key in xgb_importance.keys()



feat_names = ['Playoffs','Home Opener','Rivals','Not Real Home Team']
ax.barh(y=range(4),
        width=feat_importance,
        color=[mls_blue if x>=0 else mls_red for x in feat_importance])
ax.axvline(0, color='black')
ax.set_title('Feature Importances')
ax.set_yticks(range(4))
ax.set_yticklabels(feat_names)
ax.set_xlabel('Difference in attendance');
```



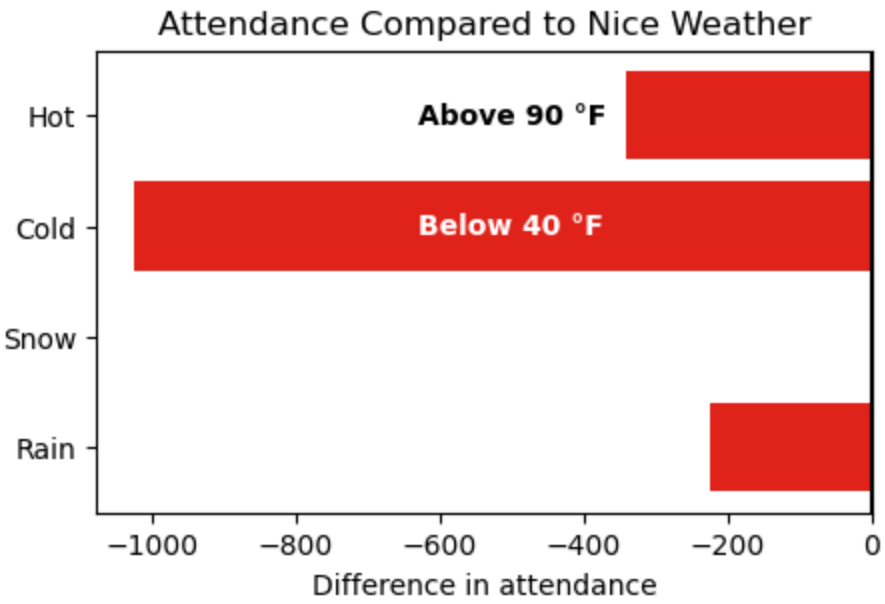
Not Real Home Team: The biggest effect is not having a real home team in a match. When that happened, the average attendance is expected to be 8,000 lower than average, a huge decrease.

Rivals: When rivals play, the attendance is expected to go up about 2,000.

Home Openers: While February and March are predicted to have the worst attendance, home openers are predicted to have a significant boost in attendance, around 2,000.

Playoffs: This one is interesting because the model says that the attendance is not expected to go up because a match is a playoff match. However, I think this is wrong. Earlier, we saw that November and December had the highest predicted attendance. The thing that makes those months special is that the playoffs happen during those months. I think the reason the playoff parameter in the model has such a small effect is because that effect is already built in to the model through the month parameters.

In [51]:
```
# Weather parameters
fig, ax = plt.subplots(figsize=(5,3))

feat_importance = [xgb_importance[key] for key in xgb_importance.keys()




feat_names = ['Rain','Snow','Cold','Hot']
ax.barh(y=range(4),
        width=feat_importance,
        color=[mls_blue if x>=0 else mls_red for x in feat_importance])

ax.axvline(0, color='black')
ax.set_title('Attendance Compared to Nice Weather')
ax.set_yticks(range(4))
ax.set_yticklabels(feat_names)
ax.set_xlabel('Difference in attendance')
ax.text(-500,3,'Above 90 \N{DEGREE SIGN}F',fontsize=10,fontweight='bold
            horizontalalignment='center',verticalalignment='center')
ax.text(-500,2,'Below 40 \N{DEGREE SIGN}F',fontsize=10,fontweight='bold
            horizontalalignment='center',verticalalignment='center',co
```



Hot and cold weather: According to the model, cold and hot temperatures don't seem to have a major effect. The model does say that cold weather (below 40 degrees Fahrenheit) is expected to get about 1,000 fewer people on average, but that is a smaller effect than many of the other factors.

Rain and snow: The model does not indicate there is much of an effect on attendance when it rains or snows. However, I think this might be mostly due to low sample size.

Recommendations

Weekends better than midweek: The model indicated that matches played on Wednesdays get about 2,700 fewer fans than Saturday matches and Tuesdays/Thursdays get almost 1,000 fewer fans. Attendance would improve if MLS could play more of their matches on weekends rather than midweek.

It is impossible to avoid playing some midweek matches because there simply aren't enough weeks during the season for the teams to play all their matches on the weekend, but there are steps MLS can take to improve the situation. Currently, MLS teams also compete in a few other competitions: the US Open Cup, the CONCACAF Champions Cup, and the Leagues Cup. The latter two competitions are somewhat redundant since they are both competitions that include teams from other North American countries (the Champions Cup is all of North America while the Leagues Cup is an MLS vs. Liga MX competition). At least in 2023, a big reason teams had to play so many midweek matches is because the Leagues Cup forced MLS to pause its season for a full month. This wouldn't be such a problem if the Leagues Cup was well attended, but the average was just 17,257, significantly lower than MLS matches. Even worse, some teams were eliminated from the Leagues Cup after the first week, meaning they didn't play a match for about three weeks. I think MLS should reconsider the format of the Leagues Cup, particularly if it is worth it to pause MLS for such a long time. Without the Leagues Cup, MLS would have gotten back about 5 weekends.

Make regular season matter: The attendance improved from the beginning of the season to the end of the season. I think a big part of this trend is that matches become more intriguing as the playoffs approach because the stakes of each match get bigger. People want to watch matches that matter. The problem is that many fans feel like most of the regular season does not matter that much because so many teams make the playoffs, so a team does not actually have to perform that well to qualify. In most European leagues, a team has to finish with the best overall record to be crowned champion, but in MLS, over half of the teams qualify for the playoffs (in 2023, 18 teams out of 29 make the playoffs). Sporting Kansas City provides a great example of how low the bar is. In their first 10 matches, they got 0 wins, 3 ties, and 7 losses. This was the worst 10 match beginning to a season in history, and yet they recovered to make the playoffs, then proceeded to beat the 1-seed in the West, St. Louis FC. This shows that a team can perform quite poorly, but still make the playoffs. Then the team just needs to go on a hot streak to make a run in the playoffs.

The relative importance of regular season matches would increase if fewer teams made the playoffs, but this also comes with a downside. The fewer playoff spots there are, then the earlier teams will be eliminated from playoff contention. Once a team is unable to make the playoffs, their matches cease to matter.

I do have an idea for a compromise between number of teams making the playoffs and rewarding teams for finishing higher in the standings. In the Australian Football League (AFL), which plays Australian rules football, they have a format in which the top 8 teams make the playoffs, but the top 4 start with a pretty significant advantage (click this link for the exact details: https://en.wikipedia.org/wiki/AFL_final_eight_system

(https://en.wikipedia.org/wiki/AFL_final_eight_system)). I think this format would do a good job of allowing enough teams into the playoffs while still rewarding the best-performing teams.

Favor warmer cities in winter: The model indicated that matches with temperatures below 40 degrees Fahrenheit had about 1,000 fewer fans on average. This seems to support the decision made by MLS to not play matches during most of the winter. The conditions would be even worse if MLS tried to hold matches in January and early February. Presumably, the attendance would also drop further.

MLS is forced to play some matches in February. Otherwise, they would not have enough time to actually play all of the matches they have scheduled. MLS could improve overall attendance by favoring warmer cities, meaning cities at lower latitude, during the first couple weeks of the year. This means playing February matches in places like Los Angeles, Houston, and Miami rather than places like Minnesota, New York, and Toronto.

Other considerations:

Sharing stadium with NFL team: The three teams with the highest overall attendance (Atlanta United, Seattle Sounders, and Charlotte FC) all share a stadium with an NFL team. Their advantage is pretty obvious: they have a larger stadium that they can utilize when demand is high. One argument against having a huge stadium is that the fan experience is not as fun when the stadium is less than half full, but this is not a big issue in my opinion. When there is not enough demand to fill the whole stadium, teams just sell tickets for the sections closest to the field, making for a similar environment as a smaller stadium.

While the top three teams in attendance indicate that having a larger stadium is better, it is not a guarantee of success. The Chicago Fire moved out of their smaller stadium in 2020 to move back into Soldier Field, which they share with the Chicago Bears of the NFL. The move has not led to a significant increase in attendance. This could possibly be due to the stadium being less easily accessible than the stadiums in Atlanta or Seattle or possibly just because the Fire have not been competitive for a long time.

I would not recommend that a team that already has their own stadium abandon it to move into a larger stadium. However, I do think this is an option that future expansion teams should explore. For example, if Phoenix or Las Vegas are awarded expansion teams, they should strongly consider using the stadiums owned by the Cardinals and Raiders of the NFL. They would have the potential for higher attendance and they would not have to pay to build a new stadium.

Promote rivalries: The model indicates that rivalry matches get about 2,000 more people on average. I think MLS would benefit from promoting these rivalries, but I would caution that they should not try to manufacture rivalries. Instead, they should make sure announcers and articles on their website highlight the history between teams. A good recent example of this is FC Cincinnati and the New York Red Bulls. They are not close enough geographically to be automatic rivals, but a rivalry was kindled when a player switched from Cincinnati to New York, leaving with some not so kind words for his former club. These are the kinds of

In [52]:
```
# Remake plot of actual attendance vs. predicted attendance for presenta
fig, ax = plt.subplots(figsize=(5,5))

ax.scatter(xgb_best.predict(X_test)/1000,
           y_test/1000,
           s=7,
           alpha=0.3,
           color=mls_blue)
ax.plot([0,80],[0,80], color='black')
ax.set_title('Actual Attendance vs. Predicted Attendance')
ax.set_xlabel('Predicted Attendance (Thousands)')
ax.set_ylabel('Actual Attendance (Thousands)')
ax.text(0,80,f"R$^2$ = {np.round(r2_score(y_test, xgb_best.predict(X_tes
ax.text(0,75,f"RMSE = {np.round(mse(y_test, xgb_best.predict(X_test), sq
ax.text(50,5,'This data not \nused to train model')
```

Out[52]:
```
Text(50, 5, 'This data not \nused to train model')
```