Establishing a database connection and retrieving data from the BikeWeather table

In [52]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
import mysql.connector

# Allows plots to appear directly in the notebook.
%matplotlib inline

from patsy import dmatrices
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn import metrics
from pandas import DataFrame

#setting up a connection to the database
mydb = mysql.connector.connect(
    host="database-1.cx36tayg9smy.us-east-1.rds.amazonaws.com",
    user="admin",
    passwd="liamstacy",
    db='SE_Project',
)

df = pd.read_sql("""SELECT number, name, description, available_bikes, available_bike_stands, weekday, hour_of_day, (case when description in
 ("drizzle", "drizzle rain", "light intensity drizzle", "light intensity drizzle rain", "light intensity shower rain", "light rain", "mist", "m
oderate rain", "ragged shower rain", "shower rain", "shower sleet") then 1 else 0 end) as rain, (case when description in ("drizzle", "drizzle
 rain", "light intensity drizzle", "light intensity drizzle rain", "light intensity shower rain", "light rain", "mist", "moderate rain", "ragge
d shower rain", "shower rain", "shower sleet") then 0 else 1 end) as no_rain
                    FROM SE_Project.BikeWeather2
                    WHERE date_update < '2020-04-10';""", con=mydb)
```

In [46]:
```
print(len(df))
df.head(10)
```

```
2202288
```

Out[46]:

	number	name	description	available_bikes	available_bike_stands	weekday	hour_of_day	rain	no_rain
0	2	BLESSINGTON STREET	few clouds	0	20	2	16	0	1
1	2	BLESSINGTON STREET	few clouds	0	20	2	17	0	1
2	2	BLESSINGTON STREET	few clouds	0	20	2	17	0	1
3	2	BLESSINGTON STREET	few clouds	0	20	2	17	0	1
4	2	BLESSINGTON STREET	few clouds	2	18	2	17	0	1
5	2	BLESSINGTON STREET	few clouds	1	19	3	9	0	1
6	2	BLESSINGTON STREET	few clouds	0	20	3	9	0	1
7	2	BLESSINGTON STREET	few clouds	0	20	3	10	0	1
8	2	BLESSINGTON STREET	few clouds	0	20	3	10	0	1
9	2	BLESSINGTON STREET	few clouds	1	19	3	10	0	1

In [50]:
```
#Check for missing data
df.isnull().sum()
```

Out[50]:
```
number                  0
name                    0
description             0
available_bikes         0
available_bike_stands   0
weekday                 0
hour_of_day             0
rain                    0
no_rain                 0
dtype: int64
```

In [51]:
```
df.dtypes
```

Out[51]:
```
number                   int64
name                    object
description             object
available_bikes          int64
available_bike_stands    int64
weekday                  int64
hour_of_day              int64
rain                     int64
no_rain                  int64
dtype: object
```

In [47]:
```
df.available_bikes.mean()
```

Out[47]:
```
11.498141932390315
```

In [49]:
```
df.describe().T
```

Out[49]:

	count	mean	std	min	25%	50%	75%	max
number	2202288.0	60.517106	33.767496	2.0	31.0	61.0	90.0	117.0
available_bikes	2202288.0	11.498142	7.822433	0.0	6.0	11.0	16.0	40.0
available_bike_stands	2202288.0	20.626762	9.780597	0.0	13.0	20.0	28.0	40.0
weekday	2202288.0	3.037935	1.977035	0.0	1.0	3.0	5.0	6.0
hour_of_day	2202288.0	11.623759	6.896978	0.0	6.0	12.0	18.0	23.0
rain	2202288.0	0.164185	0.370444	0.0	0.0	0.0	0.0	1.0
no_rain	2202288.0	0.835815	0.370444	0.0	1.0	1.0	1.0	1.0

In [48]:
```
# Look at correlations
df[['available_bikes', 'rain', 'no_rain']].corr()
```

Out[48]:

	available_bikes	rain	no_rain
available_bikes	1.000000	0.000826	-0.000826
rain	0.000826	1.000000	-1.000000
no_rain	-0.000826	-1.000000	1.000000

As can be seen from the correlation matrix there is a very small correlation between available bikes and rain. There are more bikes available when it's raining which would be expected.

In [64]:
```
#Get stations and hours

df_stations = pd.read_sql("""SELECT number FROM SE_Project.station order by number ;""", con=mydb)


hours = []
for i in range(0,24):
    hours.append(i)

print(df_stations.head(10))
print(hours)
```

```
   number
0       2
1       3
2       4
3       5
4       6
5       7
6       8
7       9
8      10
9      11
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23]
```

Base case is when data is split by station and by hour

In [72]:
```
#for each station, calculate the mean square error and rmse when data is split by hour
#Test data should be used here which is data after 10/04/2020

#create an empty list to hold the
errors = []

for index, row in df_stations.iterrows():
    number = row["number"]
    for i in hours:
        error_squared = ((df[df.number.eq(number) & df.hour_of_day.eq(i)].available_bikes - df[df.number.eq(number) & df.hour_of_day.eq(i)].ava
ilable_bikes.mean())** 2).sum()
        tested = len(df[df.number.eq(number) & df.hour_of_day.eq(i)].available_bikes)
        errors.append([number, i, tested, error_squared])
```

```
In [74]: errors[0]

Out[74]: [2, 0, 812, 21601.02463054187]
```

```
In [75]: # calculate the total mse and rmse for each station for this model
mse = []

for index, row in df_stations.iterrows():
    number = row["number"]
    mean_squared_error = 0
    count = 0
    for e in errors:
        if e[0] == number:
            mean_squared_error += e[3]
            count += e[2]
    mse.append([number, mean_squared_error/count,(mean_squared_error/count)** 0.5])
```

Now split by station, day of the week and hour of the day

```
In [ ]: weekdays = [0, 1, 2, 3, 4, 5, 6]
errors = []

for index, row in df_stations.iterrows():
    number = row["number"]
    for i in hours:
        for w in weekdays:
            error_squared = ((df[df.number.eq(number) & df.hour_of_day.eq(i) & df.weekday.eq(w)].available_bikes - df[df.number.eq(number) & df
.hour_of_day.eq(i) & df.weekday.eq(w)].available_bikes.mean())** 2).sum()
            tested = len(df[df.number.eq(number) & df.hour_of_day.eq(i) & df.weekday.eq(w)].available_bikes)
            errors.append([number, i, tested, error_squared])
```

Ideally this would have been run on a sample of the data instead of all of it but I ran out of time

Now split by station, day of the week and hour of the day and investigate linear regression, ploting available bikes vs raining indicator

In [ ]:
```
weekdays = [0, 1, 2, 3, 4, 5, 6]
errors = []

for index, row in df_stations.iterrows():
    number = row["number"]
    for i in hours:
        for w in weekdays:
            x = [['rain']]
            y = df[df.number.eq(number) & df.hour_of_day.eq(i) & df.weekday.eq(w)].available_bikes
            linreg = LinearRegression().fit(x["rain"], y)
            linreg_predictions = linreg.predict(x["rain"])
            mean_squared_error = ((y - linreg_predictions)** 2).mean()
            rmse = mean_squared_error** 0.5
```