We read in the data

In [162]:

```
import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = 20, 10
import pandas as pd
import numpy as np
from sklearn import linear_model

day_hour_count = pd.read_csv("C:/Users/erinp/Desktop/mlnn/data/bikeshare_hour_count.csv")
day_hour_count
```

Out[162]:

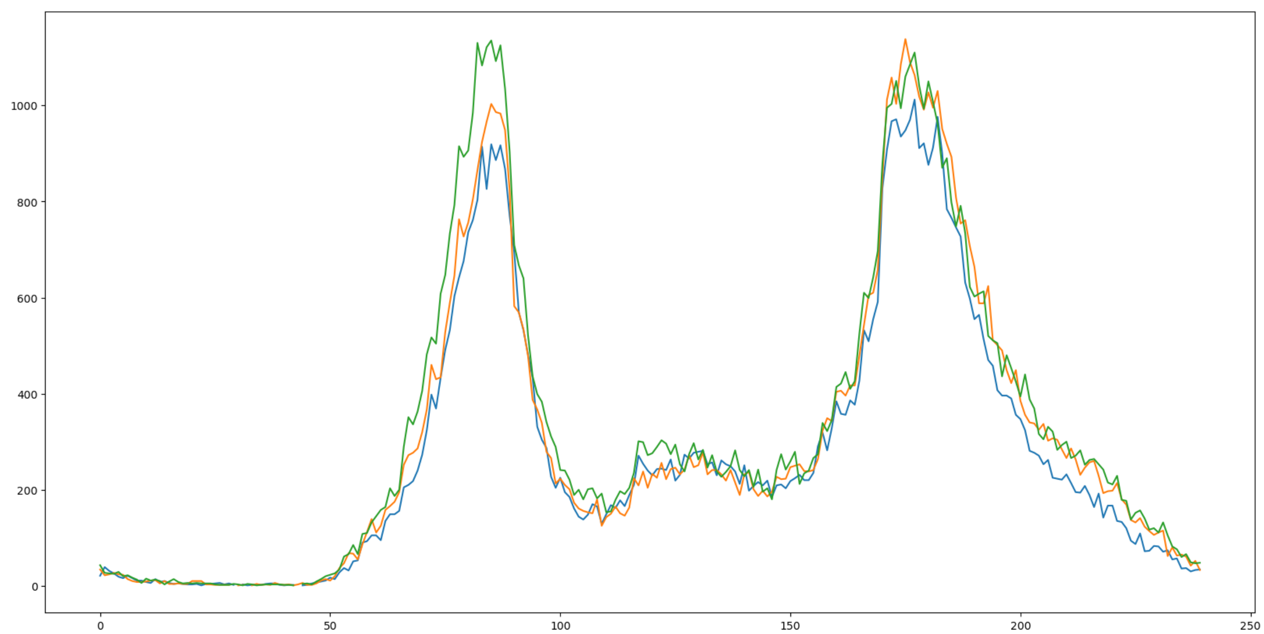
	hour	monday	tuesday	wednesday	thursday	friday	saturday	sunday
0	0.0	21.0	34.0	43.0	47.0	51.0	89.0	106.0
1	0.1	39.0	22.0	27.0	37.0	56.0	87.0	100.0
2	0.2	31.0	24.0	26.0	42.0	50.0	98.0	77.0
3	0.3	26.0	27.0	25.0	29.0	52.0	99.0	87.0
4	0.4	19.0	24.0	29.0	29.0	50.0	98.0	69.0
...	...	...	...	...	...	...	...	...
235	23.5	36.0	65.0	60.0	94.0	80.0	93.0	28.0
236	23.6	37.0	61.0	66.0	100.0	81.0	95.0	28.0
237	23.7	30.0	42.0	49.0	80.0	101.0	105.0	27.0
238	23.8	33.0	52.0	47.0	79.0	91.0	93.0	24.0
239	23.9	34.0	33.0	48.0	65.0	105.0	111.0	23.0

240 rows × 8 columns

In [163]:

```
plt.figure(figsize=(20,10))
plt.plot(day_hour_count.index, day_hour_count["monday"])
plt.plot(day_hour_count.index, day_hour_count["tuesday"])
plt.plot(day_hour_count.index, day_hour_count["wednesday"])
```

Out[163]: [<matplotlib.lines.Line2D at 0x14307277a60>]



Assignment 4

Explain the results in a **paragraph + charts** of to describe which model you'd recommend. This means show the data and the model's line on the same chart. The paragraph is a simple justification and comparison of the several models you tried.

1. Using the `day_hour_count` dataframe create 4 dataframes `monday`, `tuesday`, `saturday` and `sunday` that represent the data for those days.

(hint: Monday is day=0)

In [189]:
```
monday = day_hour_count[["hour","monday"]].copy()
```

In [190]:
```
monday
```

Out[190]:

	hour	monday
0	0.0	21.0
1	0.1	39.0
2	0.2	31.0
3	0.3	26.0
4	0.4	19.0
...	...	...
235	23.5	36.0
236	23.6	37.0
237	23.7	30.0
238	23.8	33.0
239	23.9	34.0

240 rows × 2 columns

In [191]:
```
tuesday = day_hour_count[["hour","tuesday"]].copy()
saturday = day_hour_count[["hour", "saturday"]].copy()
sunday = day_hour_count[["hour", "sunday"]].copy()
```

2a. Create 3 models fit to (`x=hour` , `y=monday`) with varying polynomial degrees (choose from `n=5,15,20`). (Repeat for `saturday` below)

Plot all the results for each polynomial.

In [192]:
```
# check for null values in monday
monday.isnull().any()
```

Out[192]:
```
hour       False
monday      True
dtype: bool
```

In [193]:
```
# replace nulls
monday = monday.fillna(method='ffill')
```

In [194]:
```
linear = linear_model.LinearRegression()

linear.fit(monday["hour"].values.reshape(-1,1), monday["monday"])

linear.coef_, linear.intercept_
```

Out[194]:
```
(array([13.28262643]), 114.28928077455049)
```

In [195]:
```
from sklearn.preprocessing import PolynomialFeatures

poly5 = PolynomialFeatures(degree=5)

x_5 = poly5.fit_transform(monday["hour"].values.reshape(-1,1))
```

In [196]:
```
x_5.shape
```

Out[196]:
```
(240, 6)
```

In [197]:
```
x_5
```

Out[197]:
```
array([[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
        0.00000000e+00, 0.00000000e+00],
       [1.00000000e+00, 1.00000000e-01, 1.00000000e-02, 1.00000000e-03,
        1.00000000e-04, 1.00000000e-05],
       [1.00000000e+00, 2.00000000e-01, 4.00000000e-02, 8.00000000e-03,
        1.60000000e-03, 3.20000000e-04],
       ...,
       [1.00000000e+00, 2.37000000e+01, 5.61690000e+02, 1.33120530e+04,
        3.15495656e+05, 7.47724705e+06],
       [1.00000000e+00, 2.38000000e+01, 5.66440000e+02, 1.34812720e+04,
        3.20854274e+05, 7.63633171e+06],
       [1.00000000e+00, 2.39000000e+01, 5.71210000e+02, 1.36519190e+04,
        3.26280864e+05, 7.79811265e+06]])
```

In [203]:
```
linear5 = linear_model.LinearRegression()

linear5.fit(x_5, monday["monday"])

(linear5.coef_, linear.intercept_)
```

Out[203]:
```
(array([ 0.00000000e+00, -1.04112276e+02,  4.95008353e+01, -6.19337888e+00,
         3.17806232e-01, -5.80250651e-03]),
 114.28928077455049)
```

In [200]:
```
poly10 = PolynomialFeatures(degree=10)

x_10 = poly10.fit_transform(monday["hour"].values.reshape(-1,1))
```

In [201]:
```
x_10
```

Out[201]:
```
array([[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [1.00000000e+00, 1.00000000e-01, 1.00000000e-02, ...,
        1.00000000e-08, 1.00000000e-09, 1.00000000e-10],
       [1.00000000e+00, 2.00000000e-01, 4.00000000e-02, ...,
        2.56000000e-06, 5.12000000e-07, 1.02400000e-07],
       ...,
       [1.00000000e+00, 2.37000000e+01, 5.61690000e+02, ...,
        9.95375090e+10, 2.35903896e+12, 5.59092234e+13],
       [1.00000000e+00, 2.38000000e+01, 5.66440000e+02, ...,
        1.02947465e+11, 2.45014966e+12, 5.83135620e+13],
       [1.00000000e+00, 2.39000000e+01, 5.71210000e+02, ...,
        1.06459202e+11, 2.54437493e+12, 6.08105609e+13]])
```

In [204]:
```
linear10 = linear_model.LinearRegression()

linear10.fit(x_10, monday["monday"])

(linear10.coef_, linear.intercept_)
```

Out[204]:
```
(array([ 0.00000000e+00, -3.29458680e+02,  4.76069606e+02, -2.74960026e+02,
         7.72318763e+01, -1.17998709e+01,  1.05094815e+00, -5.60035446e-02,
         1.75333644e-03, -2.96145038e-05,  2.07006898e-07]),
 114.28928077455049)
```

In [208]:
```
poly15 = PolynomialFeatures(degree=15)

x_15 = poly15.fit_transform(monday["hour"].values.reshape(-1,1))
```

In [209]:
```
x_15
```

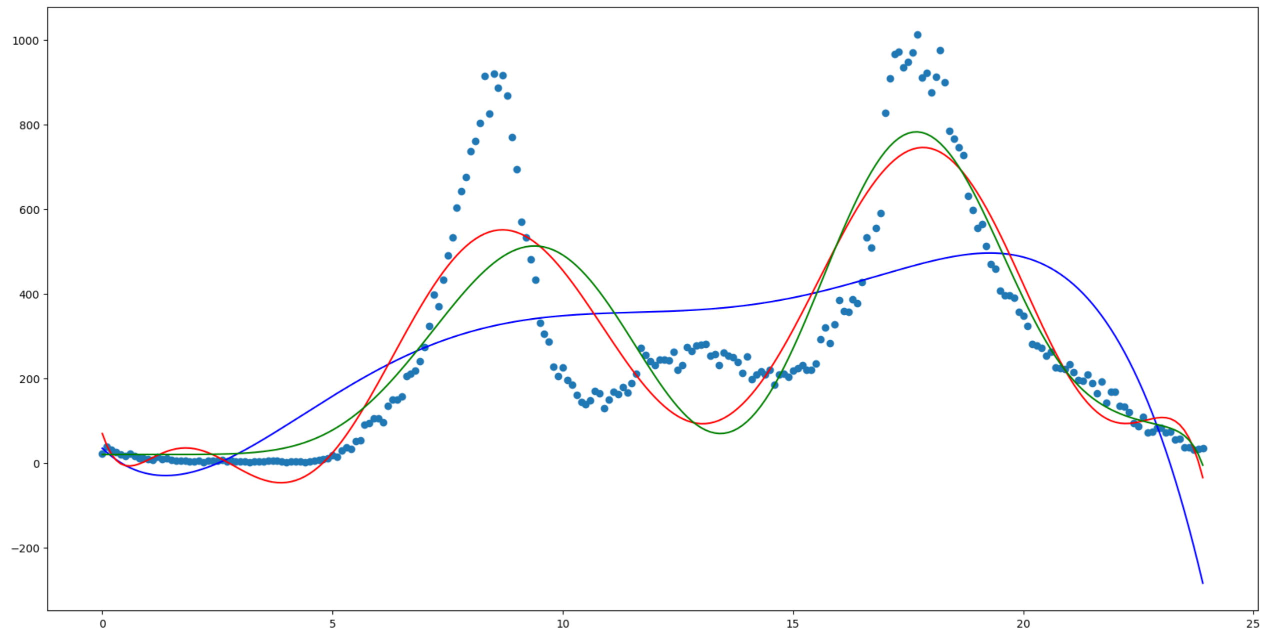
Out[209]:
```
array([[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [1.00000000e+00, 1.00000000e-01, 1.00000000e-02, ...,
        1.00000000e-13, 1.00000000e-14, 1.00000000e-15],
       [1.00000000e+00, 2.00000000e-01, 4.00000000e-02, ...,
        8.19200000e-10, 1.63840000e-10, 3.27680000e-11],
       ...,
       [1.00000000e+00, 2.37000000e+01, 5.61690000e+02, ...,
        7.44266546e+17, 1.76391171e+19, 4.18047076e+20],
       [1.00000000e+00, 2.38000000e+01, 5.66440000e+02, ...,
        7.86140991e+17, 1.87101556e+19, 4.45301703e+20],
       [1.00000000e+00, 2.39000000e+01, 5.71210000e+02, ...,
        8.30180852e+17, 1.98413224e+19, 4.74207605e+20]])
```

In [210]:
```
linear15 = linear_model.LinearRegression()

linear15.fit(x_15, monday["monday"])

(linear15.coef_, linear.intercept_)
```

Out[210]:
```
(array([ 0.00000000e+00,  1.71547065e-05,  1.16674235e-07,  9.21414900e-07,
         5.83839485e-06,  3.24582329e-05,  1.50346643e-04,  5.17640560e-04,
         9.62317580e-04, -3.95821169e-04,  6.38949590e-05, -5.57055075e-06,
         2.86664837e-07, -8.74650163e-09,  1.46850393e-10, -1.04821342e-12]),
 114.28928077455049)
```

In [211]:
```
plt.scatter(monday["hour"].values.reshape(-1,1),monday["monday"])
plt.plot(monday["hour"].values.reshape(-1,1), linear5.predict(x_5), c='b')
plt.plot(monday["hour"].values.reshape(-1,1), linear10.predict(x_10), c='r')
plt.plot(monday["hour"].values.reshape(-1,1), linear15.predict(x_15), c='g')
```

Out[211]: [<matplotlib.lines.Line2D at 0x14307e312b0>]



Based on the chart above, the x^15 degree model seems to fit the model best, as shown by the green line. The model denoted by the blue line shows the x^5 degree model and notably does not pick up on the spikes occurring between hours 5 & 10 and 15 & 20. The fitted model fits the underlying data closer when performing a polynomial regression with a higher number of degrees.

2b. Repeat `2a` for `saturday`

In [212]:
```
# check for null values in monday
saturday.isnull().any()
```

Out[212]:
```
hour        False
saturday    False
dtype: bool
```

In [213]:
```
linear = linear_model.LinearRegression()

linear.fit(saturday["hour"].values.reshape(-1,1), saturday["saturday"])

linear.coef_, linear.intercept_
```

Out[213]: (array([10.13721158]), 91.97282157676354)

In [217]:
```
x_5 = poly5.fit_transform(saturday["hour"].values.reshape(-1,1))
x_5
```

Out[217]:
```
array([[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, 0.00000000e+00,
        0.00000000e+00, 0.00000000e+00],
       [1.00000000e+00, 1.00000000e-01, 1.00000000e-02, 1.00000000e-03,
        1.00000000e-04, 1.00000000e-05],
       [1.00000000e+00, 2.00000000e-01, 4.00000000e-02, 8.00000000e-03,
        1.60000000e-03, 3.20000000e-04],
       ...,
       [1.00000000e+00, 2.37000000e+01, 5.61690000e+02, 1.33120530e+04,
        3.15495656e+05, 7.47724705e+06],
       [1.00000000e+00, 2.38000000e+01, 5.66440000e+02, 1.34812720e+04,
        3.20854274e+05, 7.63633171e+06],
       [1.00000000e+00, 2.39000000e+01, 5.71210000e+02, 1.36519190e+04,
        3.26280864e+05, 7.79811265e+06]])
```

In [215]:
```
linear5 = linear_model.LinearRegression()

linear5.fit(x_5, saturday["saturday"])

(linear5.coef_, linear.intercept_)
```

Out[215]:
```
(array([ 0.00000000e+00, -7.69357325e+01,  8.78980568e+00,  7.64304295e-01,
        -9.33173938e-02,  2.15983799e-03]),
 91.97282157676354)
```

In [218]:
```
x_10 = poly10.fit_transform(saturday["hour"].values.reshape(-1,1))
x_10
```

Out[218]:
```
array([[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [1.00000000e+00, 1.00000000e-01, 1.00000000e-02, ...,
        1.00000000e-08, 1.00000000e-09, 1.00000000e-10],
       [1.00000000e+00, 2.00000000e-01, 4.00000000e-02, ...,
        2.56000000e-06, 5.12000000e-07, 1.02400000e-07],
       ...,
       [1.00000000e+00, 2.37000000e+01, 5.61690000e+02, ...,
        9.95375090e+10, 2.35903896e+12, 5.59092234e+13],
       [1.00000000e+00, 2.38000000e+01, 5.66440000e+02, ...,
        1.02947465e+11, 2.45014966e+12, 5.83135620e+13],
       [1.00000000e+00, 2.39000000e+01, 5.71210000e+02, ...,
        1.06459202e+11, 2.54437493e+12, 6.08105609e+13]])
```

In [219]:
```
linear10 = linear_model.LinearRegression()

linear10.fit(x_10, saturday["saturday"])

(linear10.coef_, linear.intercept_)
```

Out[219]:
```
(array([ 0.00000000e+00, -1.06443321e+02,  7.71686602e+01, -2.76567416e+01,
         4.22940967e+00, -1.56794605e-01, -2.77618174e-02,  3.76528108e-03,
        -1.97444255e-04,  4.94027096e-06, -4.88275206e-08]),
 91.97282157676354)
```

In [220]:
```
x_15 = poly15.fit_transform(saturday["hour"].values.reshape(-1,1))
x_15
```

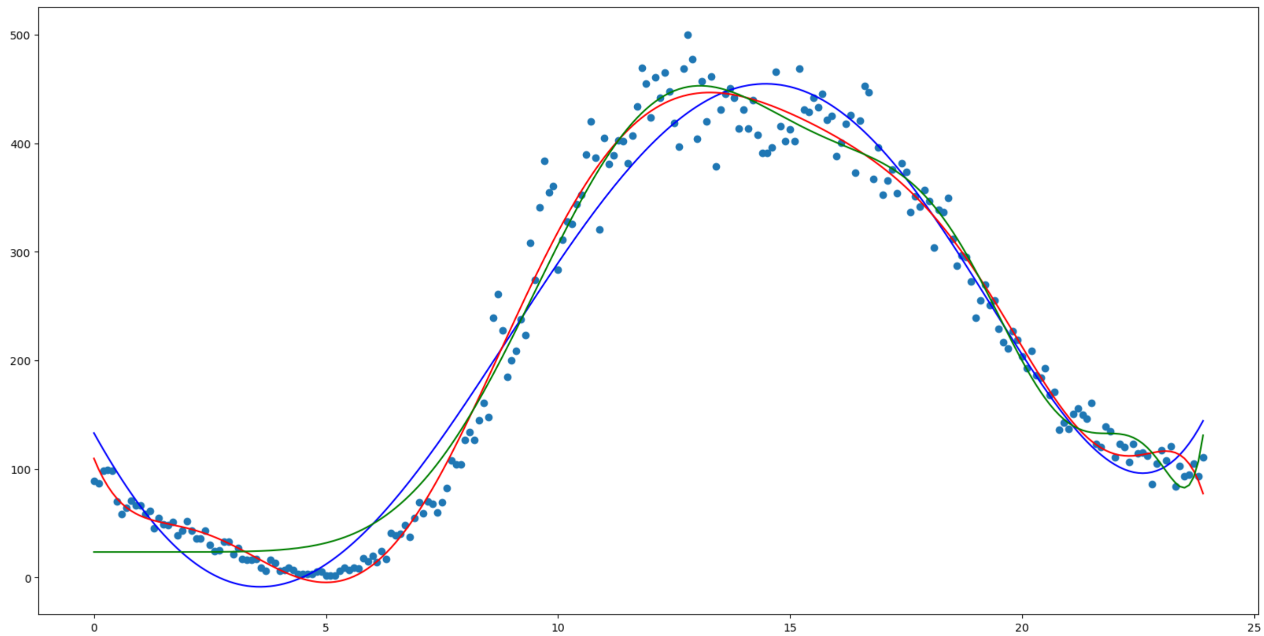
Out[220]:
```
array([[1.00000000e+00, 0.00000000e+00, 0.00000000e+00, ...,
        0.00000000e+00, 0.00000000e+00, 0.00000000e+00],
       [1.00000000e+00, 1.00000000e-01, 1.00000000e-02, ...,
        1.00000000e-13, 1.00000000e-14, 1.00000000e-15],
       [1.00000000e+00, 2.00000000e-01, 4.00000000e-02, ...,
        8.19200000e-10, 1.63840000e-10, 3.27680000e-11],
       ...,
       [1.00000000e+00, 2.37000000e+01, 5.61690000e+02, ...,
        7.44266546e+17, 1.76391171e+19, 4.18047076e+20],
       [1.00000000e+00, 2.38000000e+01, 5.66440000e+02, ...,
        7.86140991e+17, 1.87101556e+19, 4.45301703e+20],
       [1.00000000e+00, 2.39000000e+01, 5.71210000e+02, ...,
        8.30180852e+17, 1.98413224e+19, 4.74207605e+20]])
```

In [221]:
```
linear15 = linear_model.LinearRegression()

linear15.fit(x_15, saturday["saturday"])

(linear15.coef_, linear.intercept_)
```

Out[221]:
```
(array([ 0.00000000e+00,  9.21241655e-07,  6.27357281e-09,  4.96634090e-08,
         3.15670049e-07,  1.76324017e-06,  8.23224751e-06,  2.88040056e-05,
         5.63718582e-05, -9.20996361e-06, -3.58321506e-07,  1.76707058e-07,
        -1.61512317e-08,  7.01197827e-10, -1.52188993e-11,  1.33139858e-13]),
 91.97282157676354)
```

In [222]:
```
plt.scatter(saturday["hour"].values.reshape(-1,1),saturday["saturday"])
plt.plot(saturday["hour"].values.reshape(-1,1), linear5.predict(x_5), c='b')
plt.plot(saturday["hour"].values.reshape(-1,1), linear10.predict(x_10), c='r')
plt.plot(saturday["hour"].values.reshape(-1,1), linear15.predict(x_15), c='g')
```

Out[222]: [<matplotlib.lines.Line2D at 0x14308397e50>]




```
Based on the chart above, the x^10 degree model seems to fit the model best, as shown by the red line. The model denoted by the blue line shows the x^5 degree model and underestimates several values before hour 5, while overestimating at the end. Likewise, the x^15 degree model shown by the green line underestimates before hour 5 and overestimates between hours 5 and 10. The x^10 degree model best follows the pattern of the underlying data on the lower and higher ends of the x axis.
```

3. Using the best `monday` model's prediction, determine the errors (MSE, MAE, MAPE) between the prediction with the `monday` and `tuesday` datasets

Repeat for `saturday` / `sunday`

In [234]:
```
# check for null values in tuesday
tuesday.isnull().any()
```

Out[234]:
```
hour       False
tuesday     True
dtype: bool
```

In [235]:
```
# replace nulls
tuesday = tuesday.fillna(method='ffill')
```

In [247]:
```
from sklearn import metrics

x_15 = poly15.fit_transform(monday["hour"].values.reshape(-1,1))
linear15 = linear_model.LinearRegression()

linear15.fit(x_15, monday["monday"])

#Monday/Tuesday
metrics.mean_squared_error(monday["monday"], linear15.predict(x_15))
```
Out[247]: 19251.749485949582

In [246]:
```
metrics.mean_squared_error(tuesday["tuesday"], linear15.predict(x_15))
```
Out[246]: 23671.79871034551

In [248]:
```
metrics.mean_absolute_error(monday["monday"], linear15.predict(x_15))
```
Out[248]: 97.44988294581307

In [249]:
```
metrics.mean_absolute_error(tuesday["tuesday"], linear15.predict(x_15))
```
Out[249]: 105.02168692603242

In [250]:
```
metrics.mean_absolute_percentage_error(monday["monday"], linear15.predict(x_15))
```
Out[250]: 1.9979152575007362

In [251]:
```
metrics.mean_absolute_percentage_error(tuesday["tuesday"], linear15.predict(x_15))
```
Out[251]: 1.921287113990311

In [252]:
```
# check for null values in sunday
sunday.isnull().any()
```
Out[252]:
```
hour      False
sunday    False
dtype: bool
```

In [253]:
```
x_10 = poly10.fit_transform(saturday["hour"].values.reshape(-1,1))
linear10 = linear_model.LinearRegression()

linear10.fit(x_10, saturday["saturday"])

#Saturday/Sunday
metrics.mean_squared_error(saturday["saturday"], linear10.predict(x_10))
```
Out[253]: 475.432117731917

In [254]:
```
metrics.mean_squared_error(sunday["sunday"], linear10.predict(x_10))
```
Out[254]: 1366.0930279055237

In [255]:
```
metrics.mean_absolute_error(saturday["saturday"], linear10.predict(x_10))
```
Out[255]: 15.803728740895234

In [257]:
```
metrics.mean_absolute_error(sunday["sunday"], linear10.predict(x_10))
```
Out[257]: 28.09846879707431

In [258]:
```
metrics.mean_absolute_percentage_error(saturday["saturday"], linear10.predict(x_10))
```
Out[258]: 0.22012829723849595

In [259]:
```
metrics.mean_absolute_percentage_error(sunday["sunday"], linear10.predict(x_10))
```
Out[259]: 0.4082860111367491

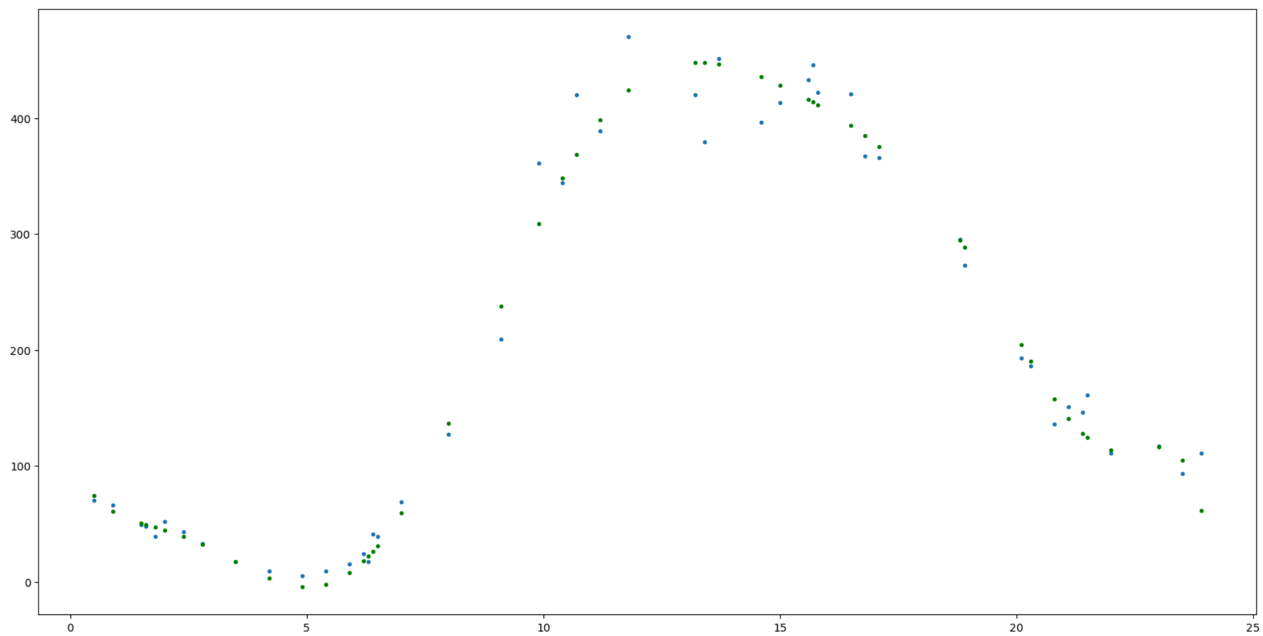
4. With `saturday`, use `train_test_split` to create training and test sets and build a model. Create predictions using the xtest from and determine the errors between these predictions and the ytest (MSE, MAE, MAPE).

repeat for `monday`

In [265]:
```
from sklearn.model_selection import train_test_split

xtrain, xtest, ytrain, ytest = train_test_split(saturday["hour"].values.reshape(-1,1), saturday["saturday"], test_size=0.

xtrain10 = PolynomialFeatures(degree=10).fit_transform(xtrain)
xtest10 = PolynomialFeatures(degree=10).fit_transform(xtest)

linear10 = linear_model.LinearRegression().fit(xtrain10, ytrain)
```

In [267]:
```
size = 8
plt.scatter(xtest, ytest, s=size)
plt.scatter(xtest, linear10.predict(xtest10), c='g', s=size)
```

Out[267]: <matplotlib.collections.PathCollection at 0x143084f4970>



Using degree = 10 helps fit the predicted values to the underlying data for Saturday.

In [268]:
```
metrics.mean_squared_error(ytest, linear10.predict(xtest10))
```

Out[268]: 512.2358329511585

In [269]:
```
metrics.mean_absolute_error(ytest, linear10.predict(xtest10))
```

Out[269]: 15.928887062924021

In [270]:
```
metrics.mean_absolute_percentage_error(ytest, linear10.predict(xtest10))
```

Out[270]: 0.18540896616412517

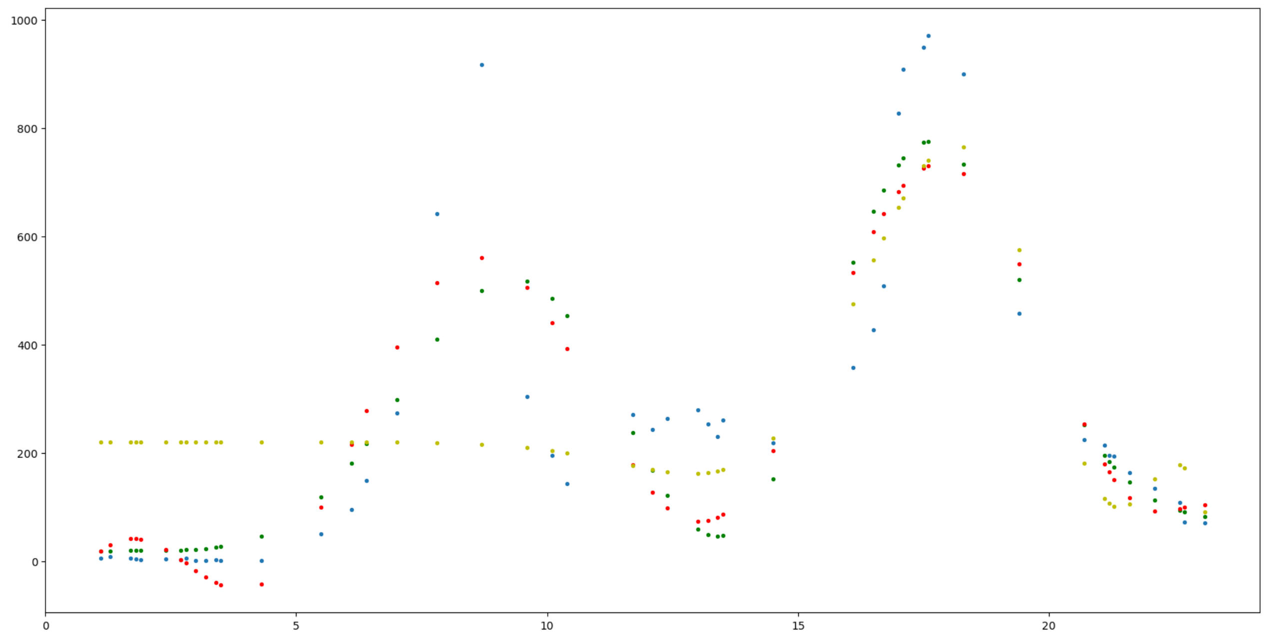
In [279]:
```
# repeat using monday
xtrain, xtest, ytrain, ytest = train_test_split(monday["hour"].values.reshape(-1,1), monday["monday"], test_size=0.2)

xtrain15 = PolynomialFeatures(degree=15).fit_transform(xtrain)
xtest15 = PolynomialFeatures(degree=15).fit_transform(xtest)

linear15 = linear_model.LinearRegression().fit(xtrain15, ytrain)
```

In [280]:
```
# testing out different degree levels
xtrain10 = PolynomialFeatures(degree=10).fit_transform(xtrain)
xtest10 = PolynomialFeatures(degree=10).fit_transform(xtest)

linear10 = linear_model.LinearRegression().fit(xtrain10, ytrain)
```

In [282]:
```
# testing out different degree levels
xtrain20 = PolynomialFeatures(degree=20).fit_transform(xtrain)
xtest20 = PolynomialFeatures(degree=20).fit_transform(xtest)

linear20 = linear_model.LinearRegression().fit(xtrain20, ytrain)
```

In [284]:
```
size = 8
plt.scatter(xtest, ytest, s=size)
plt.scatter(xtest, linear15.predict(xtest15), c='g', s=size)
plt.scatter(xtest, linear10.predict(xtest10), c='r', s=size)
plt.scatter(xtest, linear20.predict(xtest20), c='y', s=size)
```

Out[284]:
```
<matplotlib.collections.PathCollection at 0x1430cbf2b20>
```



Using degree = 15 seems to be the best fit for Monday's underlying data.

In [285]:
```
metrics.mean_squared_error(ytest, linear15.predict(xtest15))
```

Out[285]: 19415.491755440653

In [286]:
```
metrics.mean_absolute_error(ytest, linear15.predict(xtest15))
```

Out[286]: 98.12959098618182

In [287]:
```
metrics.mean_absolute_percentage_error(ytest, linear15.predict(xtest15))
```

Out[287]: 2.505578412594556

In [ ]: