Instructions

The following Cells need to be executed.

They are used to download and generate a dataset that has an aggregated count of bike trips per hundredth of an hour through the 24 hours in a day.

The assignment is in the last cell.

This cell automatically downloads Capital Bikeshare data

And here we read in the data

The datasets represents DC Bikeshare usage. Shows when bikes are checked out and checked back in.

In [1]:
```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = 20, 10 # this line sets the size of the figures
import pandas as pd
import numpy as np
bikes = pd.read_csv('../data/bikeshare.csv.gz') # the .gz used for compressed tex
bikes.head()
bikes['start'] = pd.to_datetime(bikes['Start date'], infer_datetime_format=True)
bikes['end'] = pd.to_datetime(bikes['End date'], infer_datetime_format=True) #cor
bikes["dur"] = (bikes['Duration (ms)']/1000).astype(int) #converst milliseconds t
bikes.head()
```

Out[1]:

	Duration (ms)	Start date	End date	Start station number	Start station	End station number	End station	Bike number	Member Type	s
0	301295	3/31/2016 23:59	4/1/2016 0:04	31280	11th & S St NW	31506	1st & Rhode Island Ave NW	W00022	Registered	2(0: 23:5!
1	557887	3/31/2016 23:59	4/1/2016 0:08	31275	New Hampshire Ave & 24th St NW	31114	18th St & Wyoming Ave NW	W01294	Registered	2(0: 23:5!
2	555944	3/31/2016 23:59	4/1/2016 0:08	31101	14th & V St NW	31221	18th & M St NW	W01416	Registered	2(0: 23:5!
3	766916	3/31/2016 23:57	4/1/2016 0:09	31226	34th St & Wisconsin Ave NW	31214	17th & Corcoran St NW	W01090	Registered	2(0: 23:5
4	139656	3/31/2016 23:57	3/31/2016 23:59	31011	23rd & Crystal Dr	31009	27th & Crystal Dr	W21934	Registered	2(0: 23:5

In [2]:
```
bikes.dur.mean()
```

Out[2]: 992.8716543657755

In [3]:
```
bikes.dur.std()
```

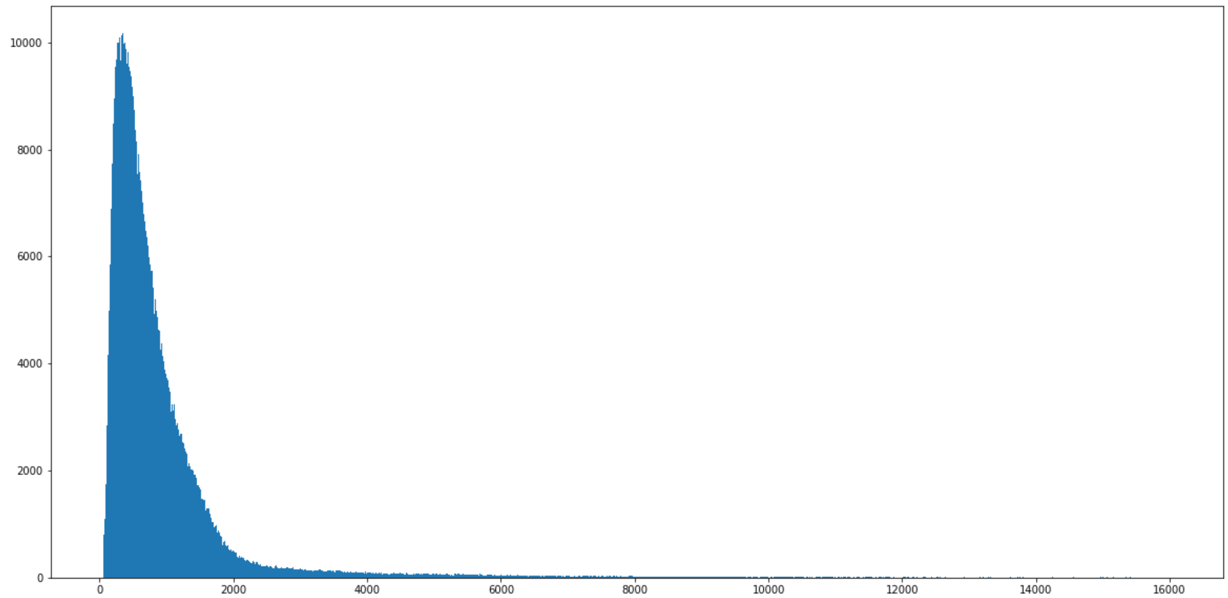
Out[3]: 2073.9809135296514

In [4]:
```
bikes[bikes.dur>16000].shape
```

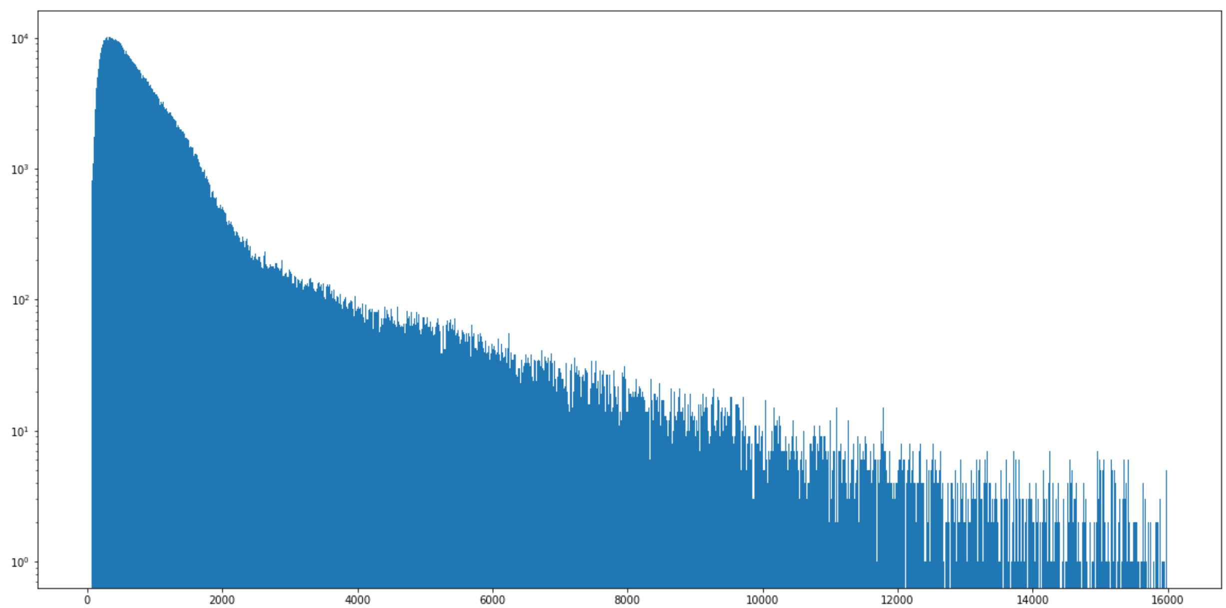
Out[4]: (973, 12)

In [5]:
```
plt.rcParams['figure.figsize'] = 20, 10
```

In [6]:
```
_=plt.hist(bikes[bikes.dur<16000].dur, log=False, bins=1000) #plot without log sc
```

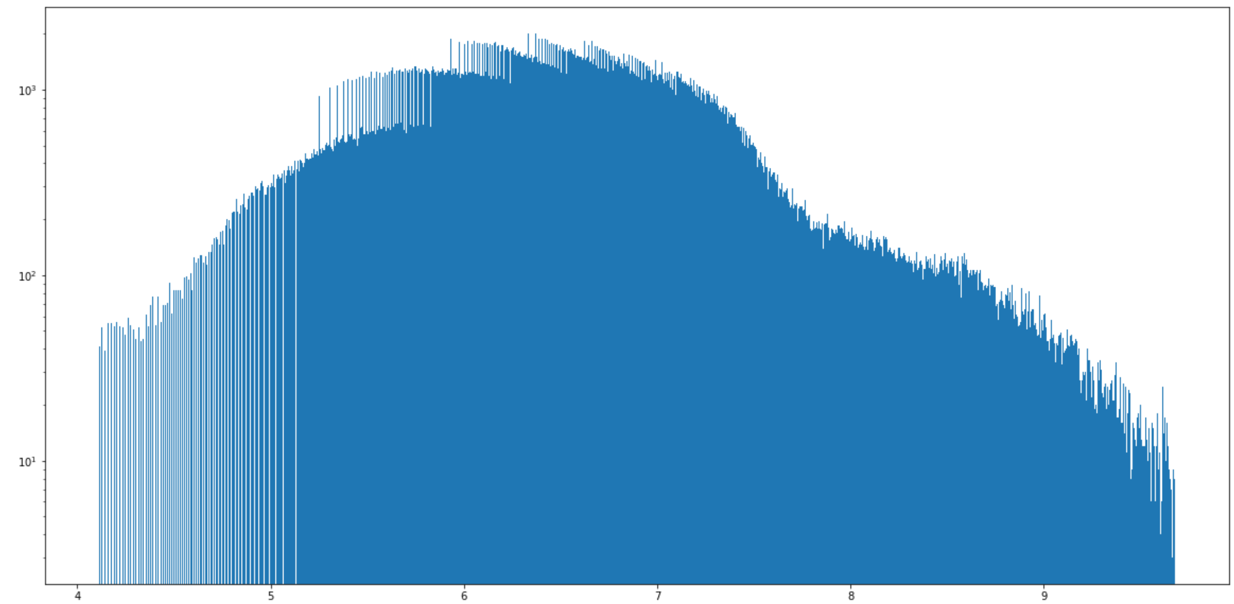


In [7]:
```
_=plt.hist(bikes[bikes.dur<16000].dur, log=True, bins=1000) #plot with log scale
```

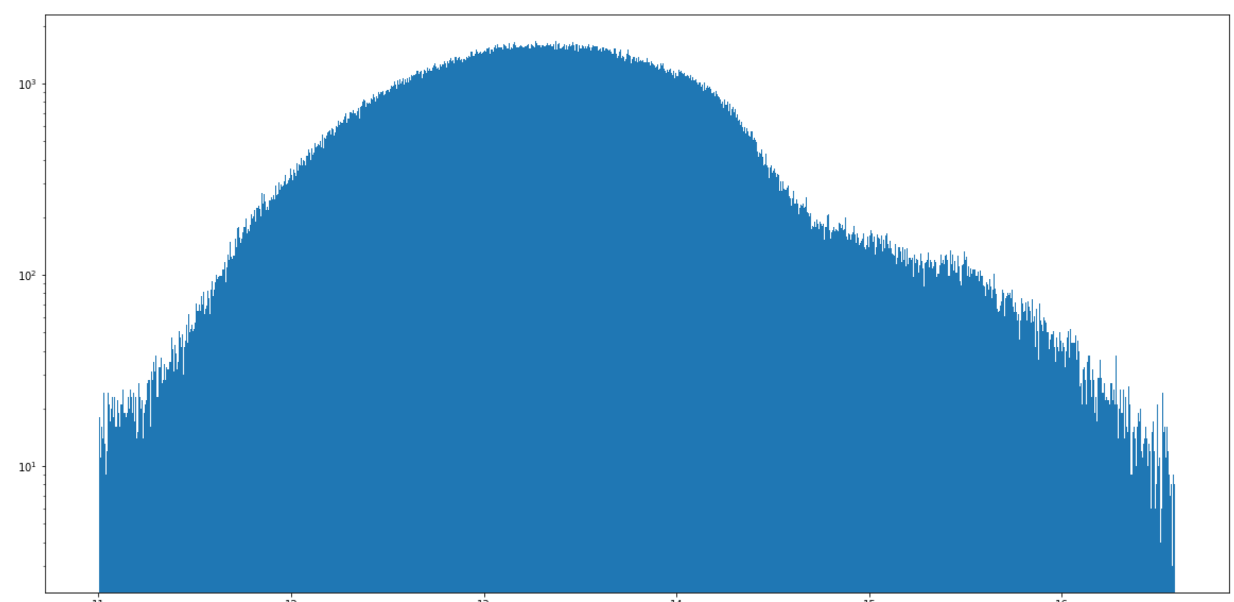


In [8]:
```
short = bikes[bikes.dur<16000]
```

In [9]:
```
# describes magnitude better with log closer to Gaussian Dist.  Better representa
_=plt.hist(np.log1p(short.dur), log=True, bins=1000)
```

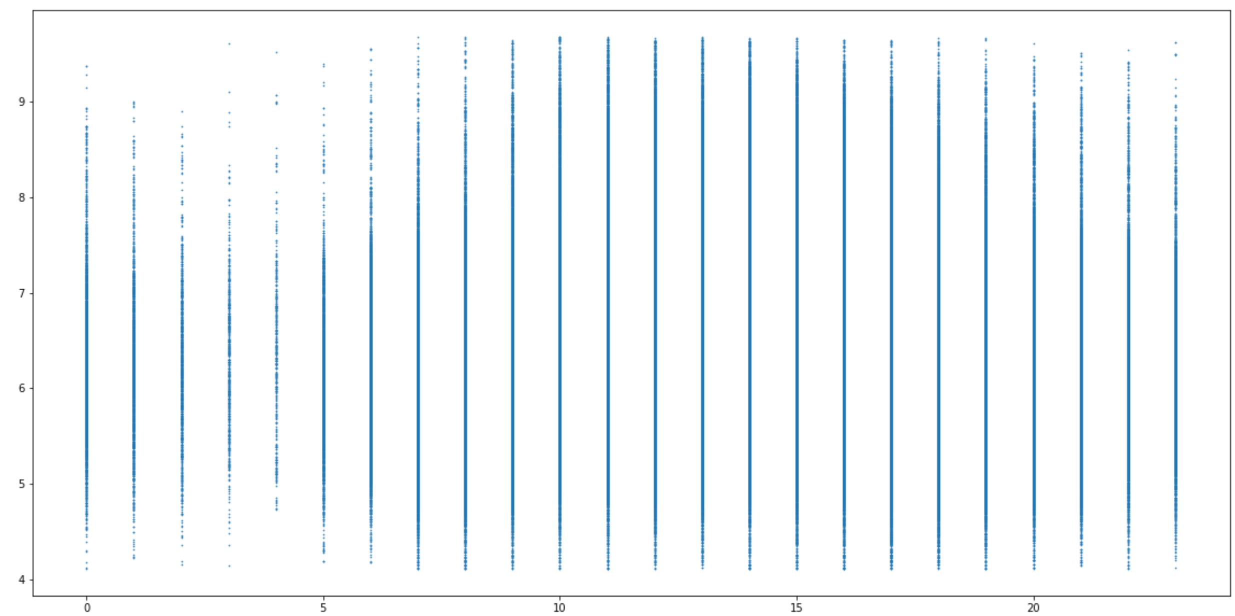


In [10]:
```
# plot w/ milliseconds (3 magnituteds higher).  More smoothness in distribution (
_=plt.hist(np.log1p(short['Duration (ms)']), log=True, bins=1000)
```



In [11]:
```python
# tries compares hour of the day of check out with duration.
plt.scatter(short.start.dt.hour, np.log1p(short.dur), s=.4)
```

Out[11]:
```
<matplotlib.collections.PathCollection at 0x1cc36129308>
```



In [12]:
```python
#log1p is 1 + x.  used to distort data for smaller numbers. not with large number
np.log1p(0), np.log(0), np.log(1+0)
```

```
C:\Users\samvt\anaconda3\lib\site-packages\ipykernel_launcher.py:2: RuntimeWarn
ing: divide by zero encountered in log
```

Out[12]:
```
(0.0, -inf, 0.0)
```

In [13]:
```python
# creating new column 'log_dur' to maintain column for future predictions
bikes['log_dur'] = np.round(np.log1p(bikes.dur), 1)
```

In [14]:
```python
bikes['log_dur_ms'] = np.round(np.log1p(bikes['Duration (ms)']), 1)
```

In [15]:
```python
bikes.head()
```

Out[15]:

	Duration (ms)	Start date	End date	Start station number	Start station	End station number	End station	Bike number	Member Type	s
0	301295	3/31/2016 23:59	4/1/2016 0:04	31280	11th & S St NW	31506	1st & Rhode Island Ave NW	W00022	Registered	2(03 23:5!
1	557887	3/31/2016 23:59	4/1/2016 0:08	31275	New Hampshire Ave & 24th St NW	31114	18th St & Wyoming Ave NW	W01294	Registered	2(03 23:5!
2	555944	3/31/2016 23:59	4/1/2016 0:08	31101	14th & V St NW	31221	18th & M St NW	W01416	Registered	2(03 23:5!
3	766916	3/31/2016 23:57	4/1/2016 0:09	31226	34th St & Wisconsin Ave NW	31214	17th & Corcoran St NW	W01090	Registered	2(03 23:5
4	139656	3/31/2016 23:57	3/31/2016 23:59	31011	23rd & Crystal Dr	31009	27th & Crystal Dr	W21934	Registered	2(03 23:5

In [16]:
```python
monday = bikes[bikes.start.dt.dayofweek==1] # Monday should actually be 0, not 1
```

In [17]:
```python
dur_hour = monday.groupby(['log_dur', monday.start.dt.hour]).count()
```

In [18]:
```
dur_hour
```
Out[18]:

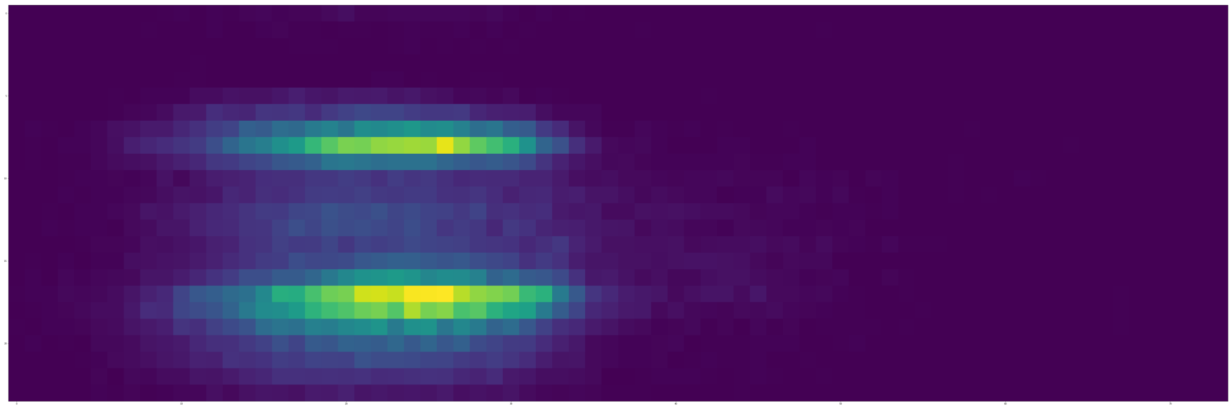
log_dur	start	Duration (ms)	Start date	End date	Start station number	Start station	End station number	End station	Bike number	Member Type	start
4.1	**7**	1	1	1	1	1	1	1	1	1	1
	9	2	2	2	2	2	2	2	2	2	2
	11	1	1	1	1	1	1	1	1	1	1
	14	2	2	2	2	2	2	2	2	2	2
	16	2	2	2	2	2	2	2	2	2	2
...	**...**	...	...	...	...	...	...	...	...	...	...
11.2	**21**	2	2	2	2	2	2	2	2	2	2
11.3	**14**	1	1	1	1	1	1	1	1	1	1
	17	1	1	1	1	1	1	1	1	1	1

In [19]:
```
duration_hour = dur_hour.start.unstack().T.fillna(0) #un tabulates on the start c
duration_hour
```
Out[19]:

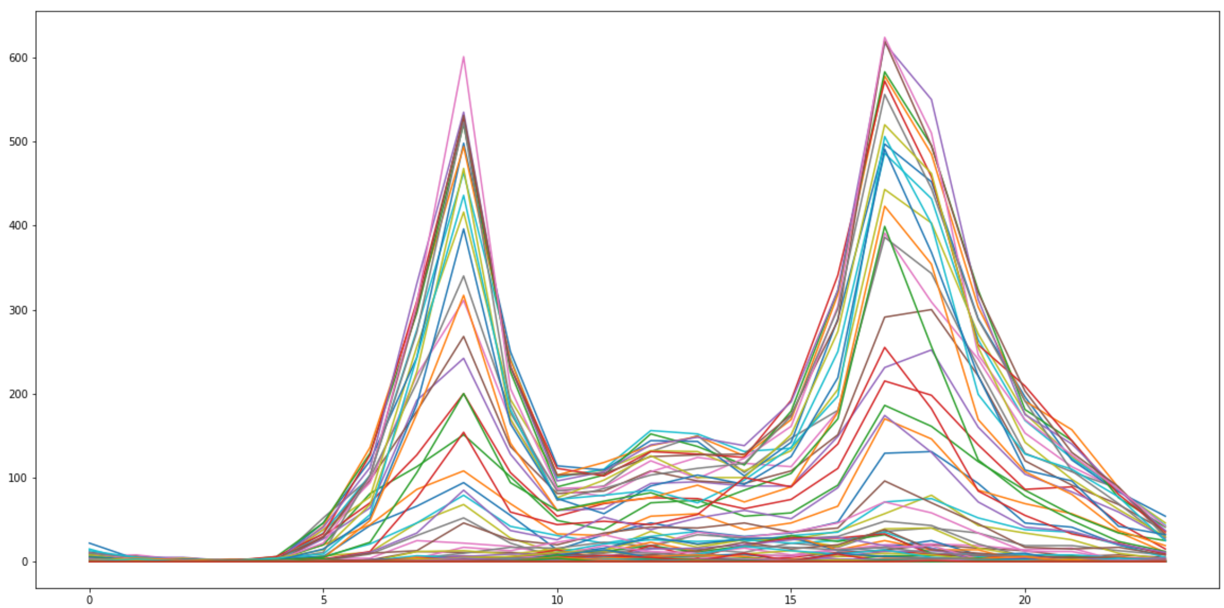
log_dur start	4.1	4.2	4.3	4.4	4.5	4.6	4.7	4.8	4.9	5.0	...	10.5	10.6	10.7	10.8	10.9
0	0.0	0.0	0.0	0.0	0.0	1.0	1.0	1.0	2.0	3.0	...	0.0	0.0	0.0	0.0	0.0
1	0.0	0.0	1.0	1.0	0.0	0.0	0.0	0.0	3.0	1.0	...	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	2.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	...	0.0	0.0	0.0	1.0	0.0
4	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	1.0	...	0.0	0.0	0.0	0.0	0.0
5	0.0	0.0	1.0	0.0	0.0	1.0	4.0	1.0	7.0	6.0	...	0.0	0.0	0.0	0.0	0.0
6	0.0	0.0	0.0	2.0	1.0	2.0	4.0	9.0	11.0	21.0	...	0.0	0.0	0.0	1.0	0.0
7	1.0	5.0	4.0	1.0	5.0	12.0	25.0	31.0	46.0	46.0	...	0.0	1.0	1.0	0.0	0.0
8	0.0	3.0	2.0	6.0	7.0	11.0	22.0	52.0	68.0	79.0	...	4.0	2.0	1.0	0.0	0.0
9	2.0	3.0	2.0	4.0	3.0	11.0	18.0	22.0	28.0	42.0	...	1.0	1.0	0.0	0.0	0.0

In [20]:
```
# 15:00 of lecture
plt.figure(figsize=(100,100))
plt.imshow(duration_hour)
```

Out[20]:
```
<matplotlib.image.AxesImage at 0x1cc363079c8>
```



In [21]:
```
# plot shows peaks before and after work day commute and possible during lunch ti
_=plt.plot(duration_hour)
```



In [22]:
```
# Casual members are typical visitors/tourists that do not need membership
bikes['Member Type'].value_counts()
```

Out[22]:
```
Registered    467432
Casual         84967
Name: Member Type, dtype: int64
```

Create a new column that represents the hour+minute of the day as a fraction (i.e. 1:30pm = 13.5)

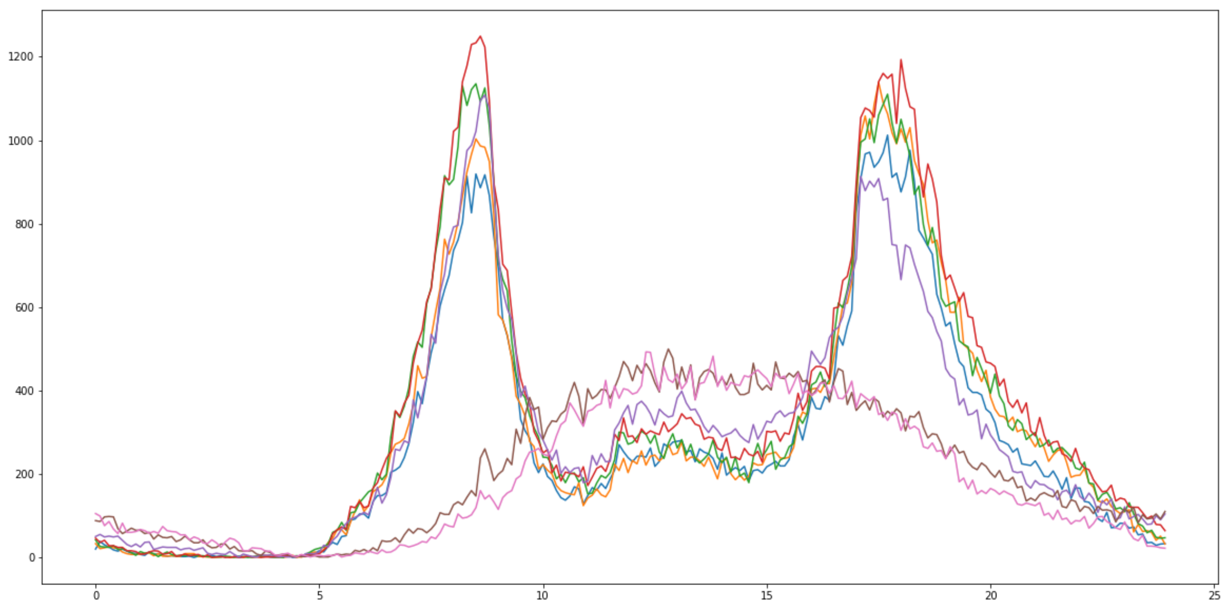
In [23]:
```
np.round(.65, 1)
```
Out[23]:
```
0.6
```
In [24]:
```
37//6, (37//6)/10, 37/60
```
Out[24]:
```
(6, 0.6, 0.6166666666666667)
```
In [25]:
```
bikes['hour_of_day'] = (bikes.start.dt.hour + (bikes.start.dt.minute//6)/10)
```
In [26]:
```
bikes['roundhour_of_day'] = (bikes.start.dt.hour ) # keep the hour handy as well
```
In [27]:
```
bikes.head()
```
Out[27]:

	Duration (ms)	Start date	End date	Start station number	Start station	End station number	End station	Bike number	Member Type	s
0	301295	3/31/2016 23:59	4/1/2016 0:04	31280	11th & S St NW	31506	1st & Rhode Island Ave NW	W00022	Registered	2(03 23:5!
1	557887	3/31/2016 23:59	4/1/2016 0:08	31275	New Hampshire Ave & 24th St NW	31114	18th St & Wyoming Ave NW	W01294	Registered	2(03 23:5!
2	555944	3/31/2016 23:59	4/1/2016 0:08	31101	14th & V St NW	31221	18th & M St NW	W01416	Registered	2(03 23:5!
3	766916	3/31/2016 23:57	4/1/2016 0:09	31226	34th St & Wisconsin Ave NW	31214	17th & Corcoran St NW	W01090	Registered	2(03 23:5
4	139656	3/31/2016 23:57	3/31/2016 23:59	31011	23rd & Crystal Dr	31009	27th & Crystal Dr	W21934	Registered	2(03 23:5

Aggregate to get a count per hour/minute of the day across all trips

In [28]:
```python
reg_bikes = bikes[bikes['Member Type']=='Registered']
hours = reg_bikes.groupby([reg_bikes.hour_of_day, reg_bikes.start.dt.dayofweek]).
hours['hour'] = hours.index
day_hour_count = hours.dur.unstack()
plt.figure(figsize=(20,10))
# pandas day of week assignment Monday = 0, Sunday = 6
# https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DatetimeIndex
plt.plot(day_hour_count.index, day_hour_count[0]) # Monday
plt.plot(day_hour_count.index, day_hour_count[1]) # Tuesday
plt.plot(day_hour_count.index, day_hour_count[2]) # Wednesday
plt.plot(day_hour_count.index, day_hour_count[3]) # Thursday
plt.plot(day_hour_count.index, day_hour_count[4]) # Friday
plt.plot(day_hour_count.index, day_hour_count[5]) # Saturday
plt.plot(day_hour_count.index, day_hour_count[6]) # Sunday

# Saturday and Sunday (purple and brown) shows usage mainly during mid-day (no co
```

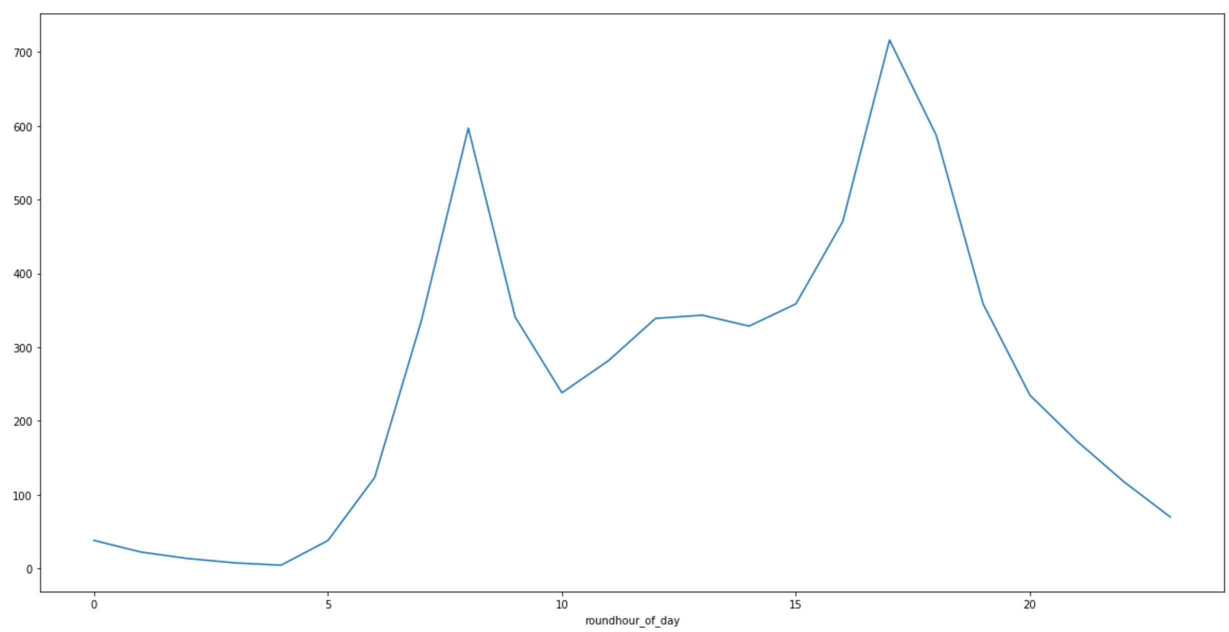
Out[28]: [<matplotlib.lines.Line2D at 0x1cc39909a48>]



In [29]:
```
day_hour_count
```
Out[29]:

start	0	1	2	3	4	5	6
hour_of_day							
0.0	21.0	34.0	43.0	47.0	51.0	89.0	106.0
0.1	39.0	22.0	27.0	37.0	56.0	87.0	100.0
0.2	31.0	24.0	26.0	42.0	50.0	98.0	77.0
0.3	26.0	27.0	25.0	29.0	52.0	99.0	87.0
0.4	19.0	24.0	29.0	29.0	50.0	98.0	69.0
...	...	...	...	...	...	...	...
23.5	36.0	65.0	60.0	94.0	80.0	93.0	28.0
23.6	37.0	61.0	66.0	100.0	81.0	95.0	28.0
23.7	30.0	42.0	49.0	80.0	101.0	105.0	27.0
23.8	33.0	52.0	47.0	79.0	91.0	93.0	24.0

In [30]:
```
# aggregates of all weekdays showing usage per hour
hoursn = bikes.groupby('roundhour_of_day').agg('count')
hoursn['hour'] = hoursn.index
(hoursn.start/90).plot() # 90 days in a quarter
```
Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x1cc399a64c8>



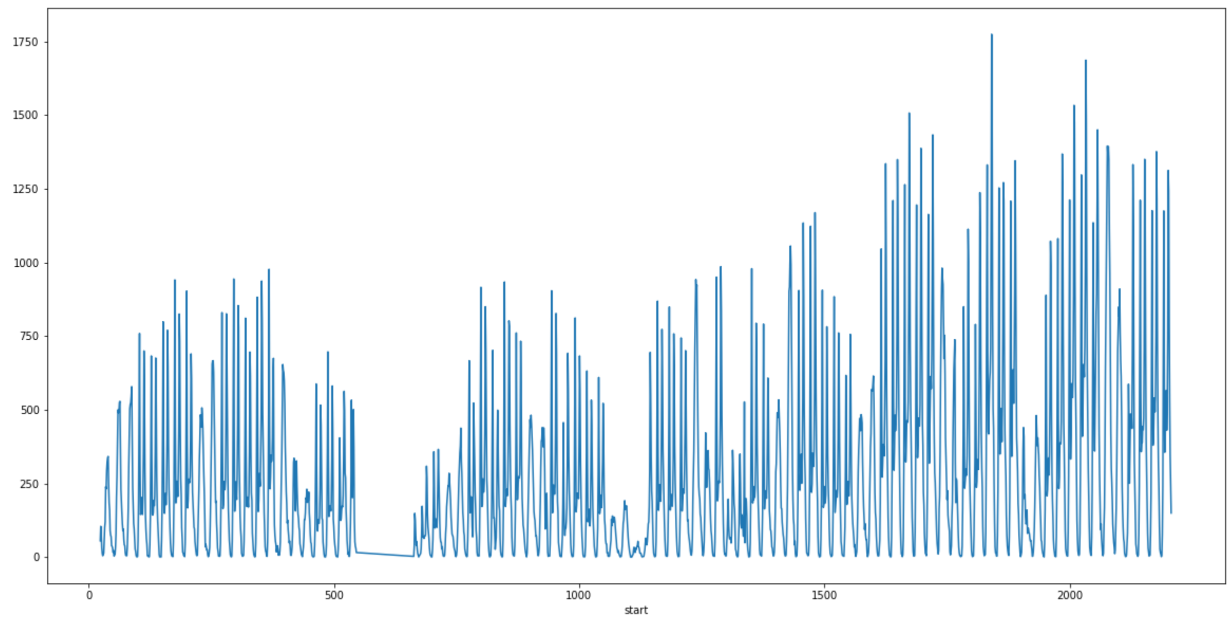
Count by days.

Dataset started March 2016.
Peaks can indicate seasonal trend, possibly increase in warmer months, specifically summer and tourist season. At day ~550-600, shows now data. Maybe corrupt data?

In [31]:
```
hour_count = bikes.groupby(bikes.start.dt.dayofyear*24 + bikes.start.dt.hour).cou
```

In [32]:
```
plt.figure(figsize=(20,10))
hour_count.start.plot()
```

Out[32]: <matplotlib.axes._subplots.AxesSubplot at 0x1cc3421cc88>



Aggregated by Day of the Year

In [33]:
```
day_count = bikes.groupby(bikes.start.dt.dayofyear).count()
```

In [34]:
```
day_hour = bikes.groupby([bikes.start.dt.dayofyear, bikes.start.dt.hour]).count()
```

In [35]:
```
day_hour.start.unstack()
```

Out[35]:

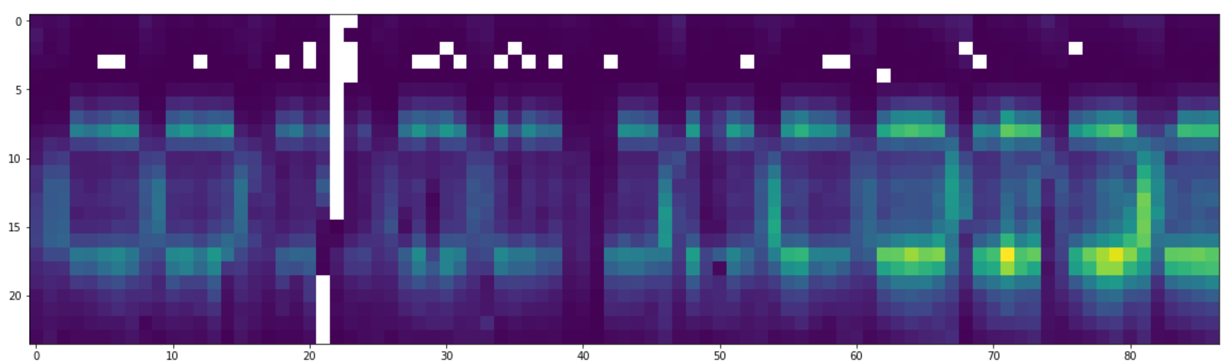
start	0	1	2	3	4	5	6	7	8	9	...	14	15	16
start														
1	56.0	105.0	74.0	32.0	13.0	5.0	10.0	14.0	54.0	101.0	...	324.0	338.0	342.0
2	37.0	31.0	17.0	23.0	4.0	7.0	10.0	34.0	80.0	203.0	...	495.0	525.0	529.0
3	59.0	42.0	39.0	15.0	6.0	9.0	5.0	33.0	87.0	168.0	...	524.0	546.0	579.0
4	20.0	6.0	2.0	1.0	3.0	58.0	192.0	468.0	759.0	321.0	...	145.0	206.0	365.0
5	5.0	5.0	3.0	1.0	2.0	42.0	131.0	363.0	683.0	329.0	...	175.0	208.0	365.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
87	113.0	82.0	50.0	34.0	12.0	24.0	94.0	166.0	297.0	509.0	...	910.0	761.0	667.0
88	15.0	7.0	2.0	3.0	8.0	42.0	81.0	197.0	587.0	464.0	...	481.0	437.0	696.0
89	31.0	11.0	9.0	3.0	8.0	79.0	240.0	727.0	1211.0	564.0	...	433.0	473.0	700.0
90	31.0	18.0	4.0	6.0	7.0	79.0	215.0	703.0	1176.0	593.0	...	493.0	545.0	749.0

Plots hour of the day (y) for 90 days.

Shows two daily peaks during rush hour on weekdays with increased mid-day activity during weekends.

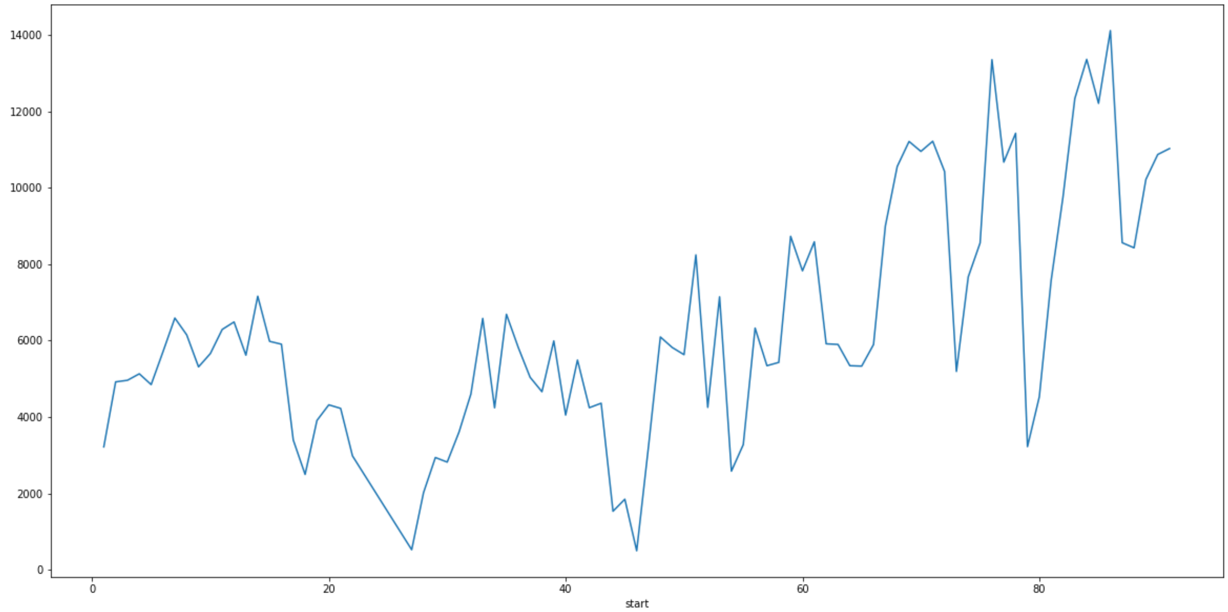
In [36]:
```
plt.figure(figsize=(20,10))
plt.imshow(day_hour.start.unstack().T)
```

Out[36]: <matplotlib.image.AxesImage at 0x1cc3421f6c8>



In [37]:
```
day_count.start.plot()
```

Out[37]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x1cc341e9888>
```



In [38]:
```
bikes.start.dt.dayofyear
```

Out[38]:
```
0         91
1         91
2         91
3         91
4         91
          ..
552394     1
552395     1
552396     1
552397     1
552398     1
Name: start, Length: 552399, dtype: int64
```

In [39]:
```
bikes[bikes.start=="2016-01-10"].shape
```

Out[39]:
```
(1, 16)
```

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 two dataframe `monday` and `saturday` that represent the data for those days. (hint: Monday is day=0)

In [40]:
```
day_hour_count
```

Out[40]:

start	0	1	2	3	4	5	6
hour_of_day							
0.0	21.0	34.0	43.0	47.0	51.0	89.0	106.0
0.1	39.0	22.0	27.0	37.0	56.0	87.0	100.0
0.2	31.0	24.0	26.0	42.0	50.0	98.0	77.0
0.3	26.0	27.0	25.0	29.0	52.0	99.0	87.0
0.4	19.0	24.0	29.0	29.0	50.0	98.0	69.0
...	...	...	...	...	...	...	...
23.5	36.0	65.0	60.0	94.0	80.0	93.0	28.0
23.6	37.0	61.0	66.0	100.0	81.0	95.0	28.0
23.7	30.0	42.0	49.0	80.0	101.0	105.0	27.0
23.8	33.0	52.0	47.0	79.0	91.0	93.0	24.0
23.9	34.0	33.0	48.0	65.0	105.0	111.0	23.0

240 rows × 7 columns

In [41]:
```
monday = day_hour_count[0]
monday
```

Out[41]:
```
hour_of_day
0.0     21.0
0.1     39.0
0.2     31.0
0.3     26.0
0.4     19.0
        ...
23.5    36.0
23.6    37.0
23.7    30.0
23.8    33.0
23.9    34.0
Name: 0, Length: 240, dtype: float64
```

In [42]:
```python
saturday = day_hour_count[5]
saturday
```

Out[42]:
```
hour_of_day
0.0      89.0
0.1      87.0
0.2      98.0
0.3      99.0
0.4      98.0
        ...
23.5     93.0
23.6     95.0
23.7    105.0
23.8     93.0
23.9    111.0
Name: 5, Length: 240, dtype: float64
```

2a. Create 3 models fit to `monday` with varying polynomial degrees. Repeat for

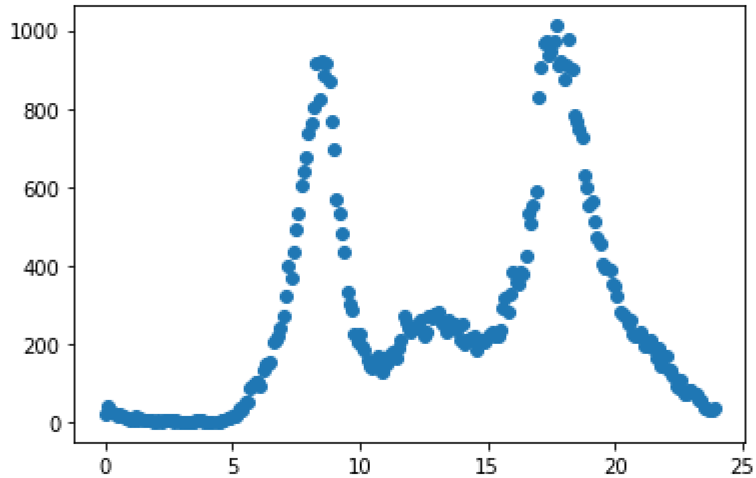
In [43]:
```python
import numpy as np
import matplotlib.pylab as plt
from sklearn import linear_model
from sklearn.preprocessing import PolynomialFeatures
%matplotlib inline
```

In [44]:
```python
monday = pd.Series.dropna(monday, axis = 0)
monday
```

Out[44]:
```
hour_of_day
0.0     21.0
0.1     39.0
0.2     31.0
0.3     26.0
0.4     19.0
        ...
23.5    36.0
23.6    37.0
23.7    30.0
23.8    33.0
23.9    34.0
Name: 0, Length: 238, dtype: float64
```

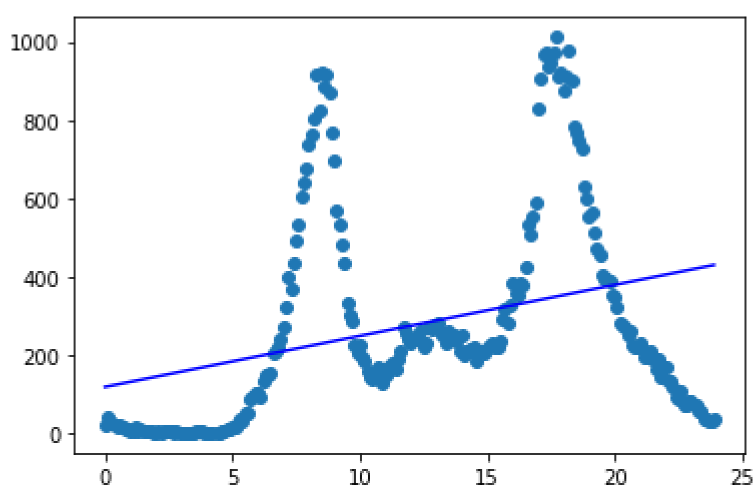
In [45]:
```
x_monday = monday.index.values.reshape(-1,1)
y_monday = monday
plt.scatter(x_monday, y_monday)
```

Out[45]: <matplotlib.collections.PathCollection at 0x1cc34f3bfc8>



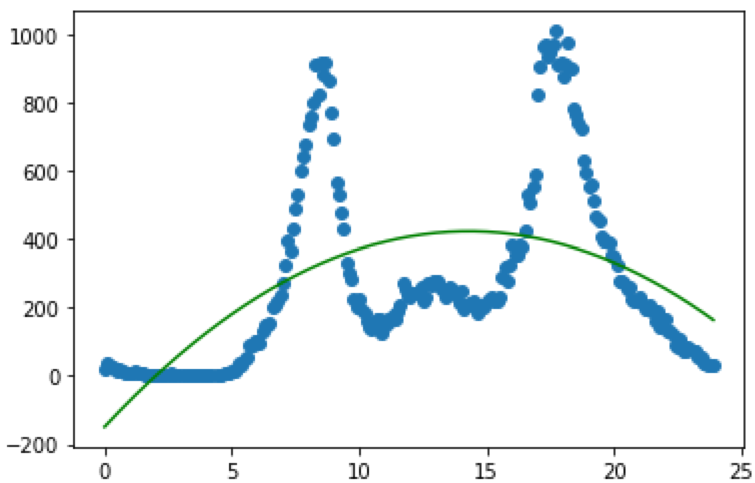
In [46]:
```
# model 1, linear model
linear = linear_model.LinearRegression()
linear.fit(x_monday, y_monday)
linear.coef_, linear.intercept_
plt.scatter(x_monday,y_monday)
plt.plot(x_monday, x_monday*linear.coef_ + linear.intercept_, c='b')
```

Out[46]: [<matplotlib.lines.Line2D at 0x1cc363a0b48>]



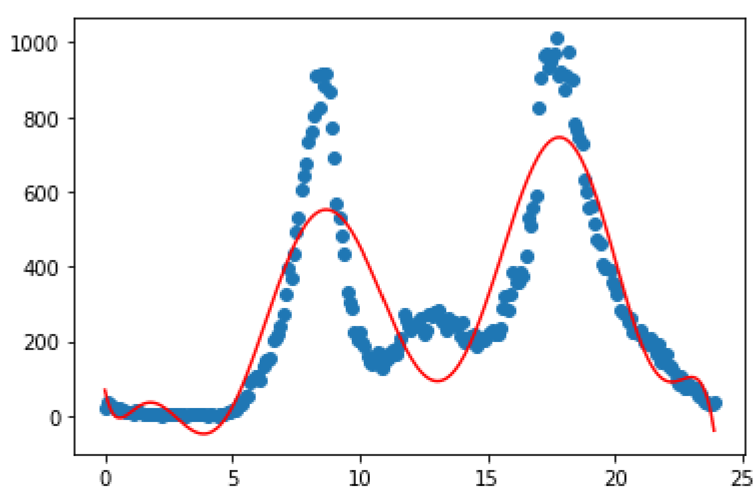
In [47]:
```python
# Model 2, x^2 polynomial model
poly = PolynomialFeatures(degree=2)
x_monday_2 = poly.fit_transform(x_monday.reshape(-1, 1))
linear = linear_model.LinearRegression()
linear.fit(x_monday_2, y_monday)
(linear.coef_, linear.intercept_)
plt.scatter(x_monday,y_monday)
plt.plot(x_monday, np.dot(x_monday_2, linear.coef_) + linear.intercept_, c='g')
```

Out[47]: [<matplotlib.lines.Line2D at 0x1cc36194508>]



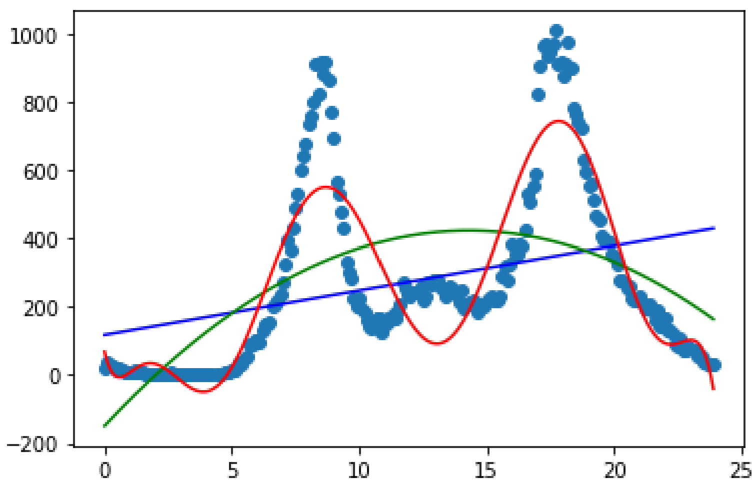
In [48]:
```python
# Model 3, x^10 polynomial model
poly = PolynomialFeatures(degree=10)
x_monday_10 = poly.fit_transform(x_monday.reshape(-1, 1))
linear = linear_model.LinearRegression()
linear.fit(x_monday_10, y_monday)
(linear.coef_, linear.intercept_)
plt.scatter(x_monday,y_monday)
plt.plot(x_monday, np.dot(x_monday_10, linear.coef_) + linear.intercept_, c='r')
```

Out[48]: [<matplotlib.lines.Line2D at 0x1cc34124fc8>]



In [49]:
```
# plotting all 3 models
plt.scatter(x_monday,y_monday)

linear = linear_model.LinearRegression()
linear.fit(x_monday, y_monday)
linear.coef_, linear.intercept_
plt.plot(x_monday, x_monday*linear.coef_ + linear.intercept_, c='b')

poly = PolynomialFeatures(degree=2)
x_monday_2 = poly.fit_transform(x_monday.reshape(-1, 1))
linear = linear_model.LinearRegression()
linear.fit(x_monday_2, y_monday)
plt.plot(x_monday, np.dot(x_monday_2, linear.coef_) + linear.intercept_, c='g')

poly = PolynomialFeatures(degree=10)
x_monday_10 = poly.fit_transform(x_monday.reshape(-1, 1))
linear = linear_model.LinearRegression()
linear.fit(x_monday_10, y_monday)
plt.plot(x_monday, np.dot(x_monday_10, linear.coef_) + linear.intercept_, c='r')
```

Out[49]:
```
[<matplotlib.lines.Line2D at 0x1cc3527c608>]
```



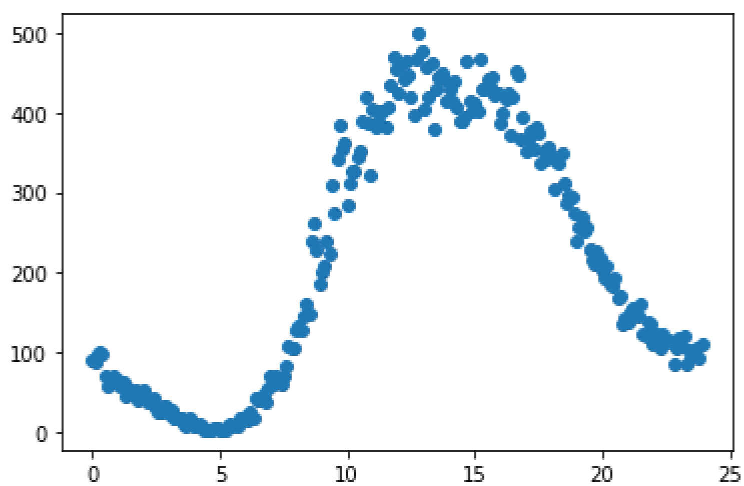
2b. Repeat 2a for saturday

In [50]:
```
saturday = pd.Series.dropna(saturday, axis = 0)
saturday
```

Out[50]:
```
hour_of_day
0.0      89.0
0.1      87.0
0.2      98.0
0.3      99.0
0.4      98.0
        ...
23.5     93.0
23.6     95.0
23.7    105.0
23.8     93.0
23.9    111.0
Name: 5, Length: 240, dtype: float64
```

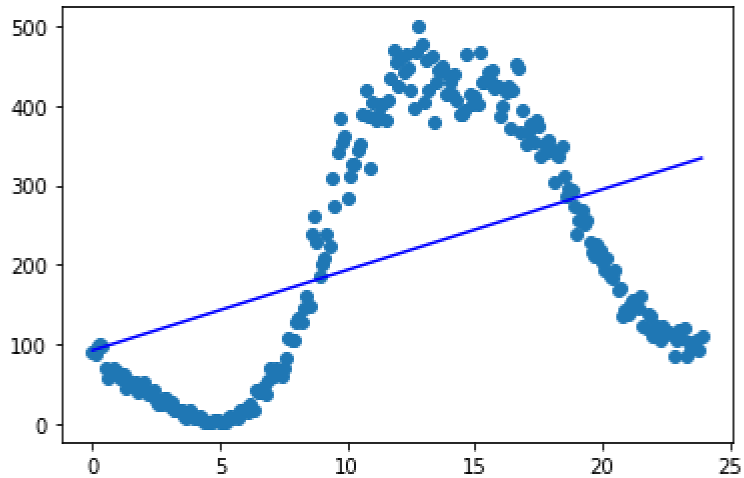
In [51]:
```
x_saturday = saturday.index.values.reshape(-1,1)
y_saturday = saturday
plt.scatter(x_saturday, y_saturday)
```

Out[51]: <matplotlib.collections.PathCollection at 0x1cc3525fe08>



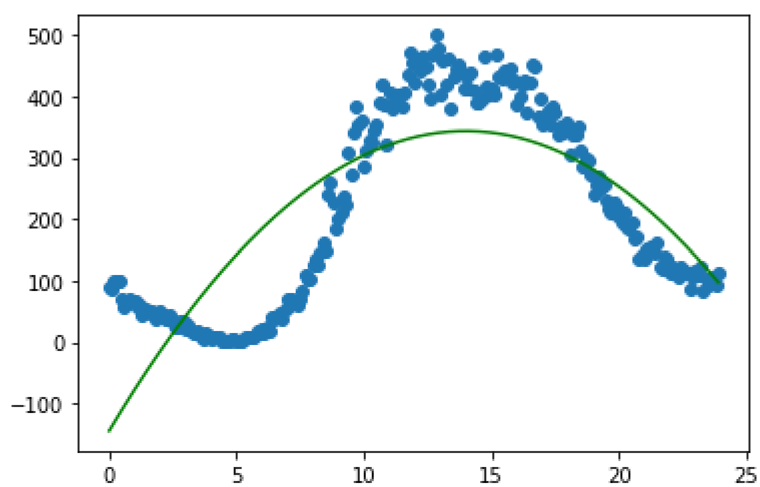
In [52]:
```
# model 1, linear model
linear = linear_model.LinearRegression()
linear.fit(x_saturday, y_saturday)
linear.coef_, linear.intercept_
plt.scatter(x_saturday,y_saturday)
plt.plot(x_saturday, x_saturday*linear.coef_ + linear.intercept_, c='b')
```

Out[52]: [<matplotlib.lines.Line2D at 0x1cc352fb048>]



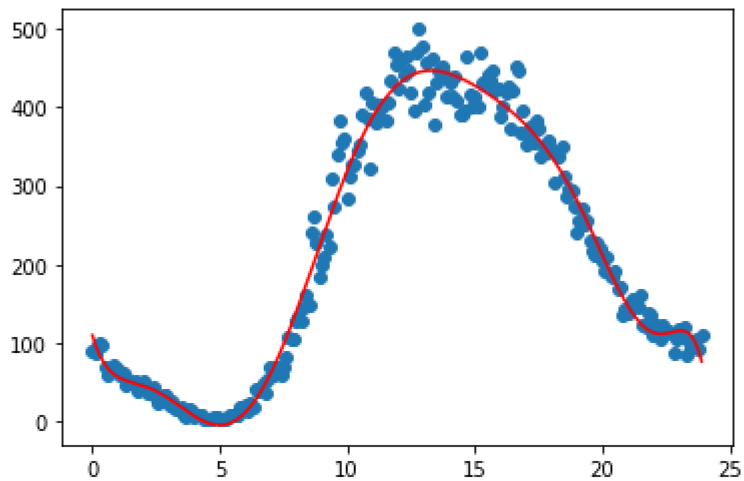
In [53]:
```
# Model 2, x^2 polynomial model
poly = PolynomialFeatures(degree=2)
x_saturday_2 = poly.fit_transform(x_saturday.reshape(-1, 1))
linear = linear_model.LinearRegression()
linear.fit(x_saturday_2, y_saturday)
(linear.coef_, linear.intercept_)
plt.scatter(x_saturday,y_saturday)
plt.plot(x_saturday, np.dot(x_saturday_2, linear.coef_) + linear.intercept_, c='g
```

Out[53]: [<matplotlib.lines.Line2D at 0x1cc351eb048>]



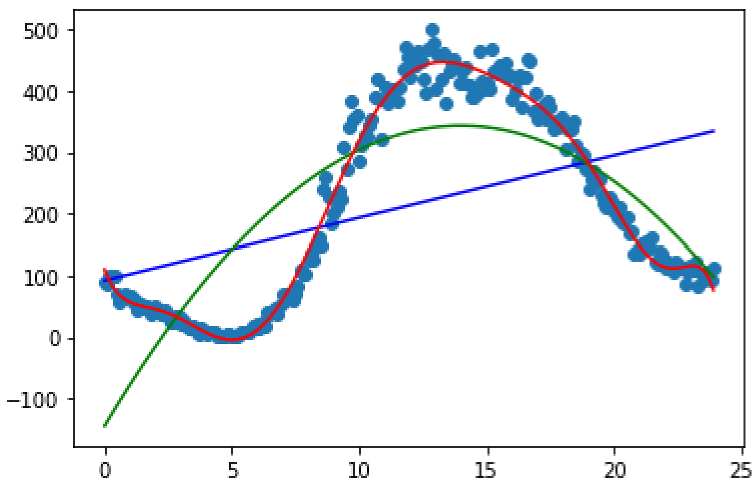
In [54]:
```
# Model 3, x^10 polynomial model
poly = PolynomialFeatures(degree=10)
x_saturday_10 = poly.fit_transform(x_saturday.reshape(-1, 1))
linear = linear_model.LinearRegression()
linear.fit(x_saturday_10, y_saturday)
(linear.coef_, linear.intercept_)
plt.scatter(x_saturday,y_saturday)
plt.plot(x_saturday, np.dot(x_saturday_10, linear.coef_) + linear.intercept_, c='
```

Out[54]:
```
[<matplotlib.lines.Line2D at 0x1cc34256f08>]
```



In [55]:
```
# plotting all 3 models
plt.scatter(x_saturday,y_saturday)

linear = linear_model.LinearRegression()
linear.fit(x_saturday, y_saturday)
linear.coef_, linear.intercept_
plt.plot(x_saturday, x_saturday*linear.coef_ + linear.intercept_, c='b')

poly = PolynomialFeatures(degree=2)
x_saturday_2 = poly.fit_transform(x_saturday.reshape(-1, 1))
linear = linear_model.LinearRegression()
linear.fit(x_saturday_2, y_saturday)
plt.plot(x_saturday, np.dot(x_saturday_2, linear.coef_) + linear.intercept_, c='g

poly = PolynomialFeatures(degree=10)
x_saturday_10 = poly.fit_transform(x_saturday.reshape(-1, 1))
linear = linear_model.LinearRegression()
linear.fit(x_saturday_10, y_saturday)
plt.plot(x_saturday, np.dot(x_saturday_10, linear.coef_) + linear.intercept_, c='
```

Out[55]:
```
[<matplotlib.lines.Line2D at 0x1cc352857c8>]
```

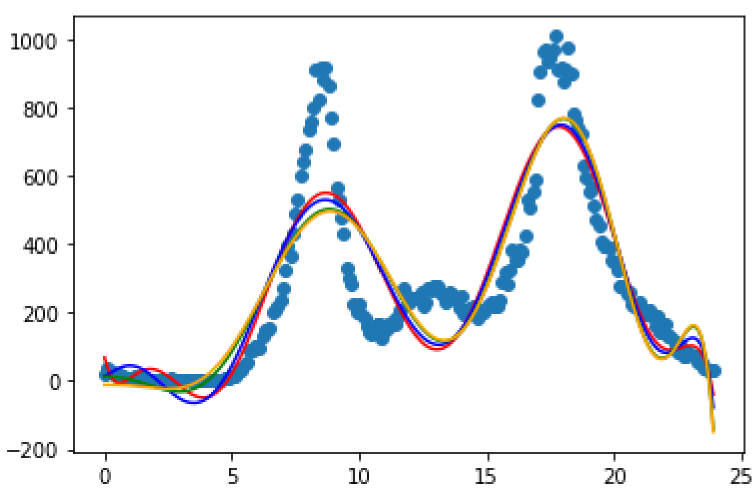


3. (for both `monday` and `saturday`) Choose one of the polynomial models and create 3 new models fit to `hour_of_day` with different Ridge Regression α (alpha) Ridge Coefficient values

In [56]:
```
#Monday

poly = PolynomialFeatures(degree=10)
x_monday_10 = poly.fit_transform(x_monday.reshape(-1, 1))
linear = linear_model.LinearRegression()
linear.fit(x_monday_10, y_monday)
(linear.coef_, linear.intercept_)

plt.scatter(x_monday,y_monday)
plt.plot(x_monday, np.dot(x_monday_10, linear.coef_) + linear.intercept_, c='r')

ridge_monday = linear_model.Ridge(alpha=5)
ridge_monday.fit(x_monday_10, y_monday)
ridge_monday.coef_, ridge_monday.intercept_
plt.plot(x_monday, np.dot(x_monday_10, ridge_monday.coef_) + ridge_monday.interce

ridge_monday = linear_model.Ridge(alpha=1000)
ridge_monday.fit(x_monday_10, y_monday)
ridge_monday.coef_, ridge_monday.intercept_
plt.plot(x_monday, np.dot(x_monday_10, ridge_monday.coef_) + ridge_monday.interce

ridge_monday = linear_model.Ridge(alpha=10000)
ridge_monday.fit(x_monday_10, y_monday)
ridge_monday.coef_, ridge_monday.intercept_
plt.plot(x_monday, np.dot(x_monday_10, ridge_monday.coef_) + ridge_monday.interce
```

```
C:\Users\samvt\anaconda3\lib\site-packages\sklearn\linear_model\_ridge.py:148:
LinAlgWarning: Ill-conditioned matrix (rcond=1.31829e-28): result may not be ac
curate.
  overwrite_a=True).T
C:\Users\samvt\anaconda3\lib\site-packages\sklearn\linear_model\_ridge.py:148:
LinAlgWarning: Ill-conditioned matrix (rcond=2.63657e-26): result may not be ac
curate.
  overwrite_a=True).T
C:\Users\samvt\anaconda3\lib\site-packages\sklearn\linear_model\_ridge.py:148:
LinAlgWarning: Ill-conditioned matrix (rcond=2.031e-25): result may not be accu
rate.
  overwrite_a=True).T
```

Out[56]:
```
[<matplotlib.lines.Line2D at 0x1cc399f8588>]
```



In [57]:
```
#Saturday

poly = PolynomialFeatures(degree=10)
x_saturday_10 = poly.fit_transform(x_saturday.reshape(-1, 1))
linear = linear_model.LinearRegression()
linear.fit(x_saturday_10, y_saturday)
(linear.coef_, linear.intercept_)

plt.scatter(x_saturday,y_saturday)
plt.plot(x_saturday, np.dot(x_saturday_10, linear.coef_) + linear.intercept_, c='

ridge_saturday = linear_model.Ridge(alpha=5)
ridge_saturday.fit(x_saturday_10, y_saturday)
ridge_saturday.coef_, ridge_saturday.intercept_
plt.plot(x_saturday, np.dot(x_saturday_10, ridge_saturday.coef_) + ridge_saturday

ridge_saturday = linear_model.Ridge(alpha=100)
ridge_saturday.fit(x_saturday_10, y_saturday)
ridge_saturday.coef_, ridge_saturday.intercept_
plt.plot(x_saturday, np.dot(x_saturday_10, ridge_saturday.coef_) + ridge_saturday

ridge_saturday = linear_model.Ridge(alpha=1000)
ridge_saturday.fit(x_saturday_10, y_saturday)
ridge_saturday.coef_, ridge_saturday.intercept_
plt.plot(x_saturday, np.dot(x_saturday_10, ridge_saturday.coef_) + ridge_saturday
```

```
C:\Users\samvt\anaconda3\lib\site-packages\sklearn\linear_model\_ridge.py:148:
LinAlgWarning: Ill-conditioned matrix (rcond=1.31596e-28): result may not be ac
curate.
  overwrite_a=True).T
C:\Users\samvt\anaconda3\lib\site-packages\sklearn\linear_model\_ridge.py:148:
LinAlgWarning: Ill-conditioned matrix (rcond=2.43788e-27): result may not be ac
curate.
  overwrite_a=True).T
C:\Users\samvt\anaconda3\lib\site-packages\sklearn\linear_model\_ridge.py:148:
LinAlgWarning: Ill-conditioned matrix (rcond=2.63192e-26): result may not be ac
curate.
  overwrite_a=True).T
```

Out[57]: [<matplotlib.lines.Line2D at 0x1cc35b4af88>]