Instructions

The assignment is at the bottom!

This cell automatically downloads Capital Bikeshare data

And here we read in the data

In [2]:

```
!pip install seaborn
```

```
Requirement already satisfied: seaborn in c:\users\cneub\desktop\mlnn-masamitsu\venv
\lib\site-packages (0.11.2)
Requirement already satisfied: matplotlib>=2.2 in c:\users\cneub\desktop\mlnn-masamit
su\venv\lib\site-packages (from seaborn) (3.5.1)
Requirement already satisfied: pandas>=0.23 in c:\users\cneub\desktop\mlnn-masamitsu
\venv\lib\site-packages (from seaborn) (1.4.0)
Requirement already satisfied: scipy>=1.0 in c:\users\cneub\desktop\mlnn-masamitsu\ve
nv\lib\site-packages (from seaborn) (1.7.3)
Requirement already satisfied: numpy>=1.15 in c:\users\cneub\desktop\mlnn-masamitsu\v
env\lib\site-packages (from seaborn) (1.22.1)
Requirement already satisfied: pyparsing>=2.2.1 in c:\users\cneub\desktop\mlnn-masami
tsu\venv\lib\site-packages (from matplotlib>=2.2->seaborn) (3.0.7)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\cneub\desktop\mlnn-masam
itsu\venv\lib\site-packages (from matplotlib>=2.2->seaborn) (4.29.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\cneub\desktop\mlnn-masam
itsu\venv\lib\site-packages (from matplotlib>=2.2->seaborn) (1.3.2)
Requirement already satisfied: cycler>=0.10 in c:\users\cneub\desktop\mlnn-masamitsu
\venv\lib\site-packages (from matplotlib>=2.2->seaborn) (0.11.0)
Requirement already satisfied: packaging>=20.0 in c:\users\cneub\desktop\mlnn-masamit
su\venv\lib\site-packages (from matplotlib>=2.2->seaborn) (21.3)
Requirement already satisfied: python-dateutil>=2.7 in c:\users\cneub\desktop\mlnn-ma
samitsu\venv\lib\site-packages (from matplotlib>=2.2->seaborn) (2.8.2)
Requirement already satisfied: pillow>=6.2.0 in c:\users\cneub\desktop\mlnn-masamitsu
\venv\lib\site-packages (from matplotlib>=2.2->seaborn) (9.0.0)
Requirement already satisfied: pytz>=2020.1 in c:\users\cneub\desktop\mlnn-masamitsu
\venv\lib\site-packages (from pandas>=0.23->seaborn) (2021.3)
Requirement already satisfied: six>=1.5 in c:\users\cneub\desktop\mlnn-masamitsu\venv
\lib\site-packages (from python-dateutil>=2.7->matplotlib>=2.2->seaborn) (1.16.0)
WARNING: You are using pip version 21.1.2; however, version 22.0.3 is available.
You should consider upgrading via the 'C:\Users\cneub\Desktop\mlnn-masamitsu\venv\Scr
ipts\python.exe -m pip install --upgrade pip' command.
```

In [3]:

```
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = 20, 10
import pandas as pd
import numpy as np
bikes = pd.read_csv('../data/bikeshare.csv.gz')
bikes['start'] = pd.to_datetime(bikes['Start date'], infer_datetime_format=True)
bikes['end'] = pd.to_datetime(bikes['End date'], infer_datetime_format=True)
```

```
bikes["dur"] = (bikes['Duration (ms)']/1000).astype(int)
bikes.head()
```

Out[3]:

	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 [4]:

```
bikes.dur.mean()
```

Out[4]:

```
992.8716543657755
```

In [5]:

```
bikes.dur.std()
```

Out[5]:

```
2073.9809135296764
```

In [6]:

```
bikes[bikes.dur>16000].shape
```

Out[6]:

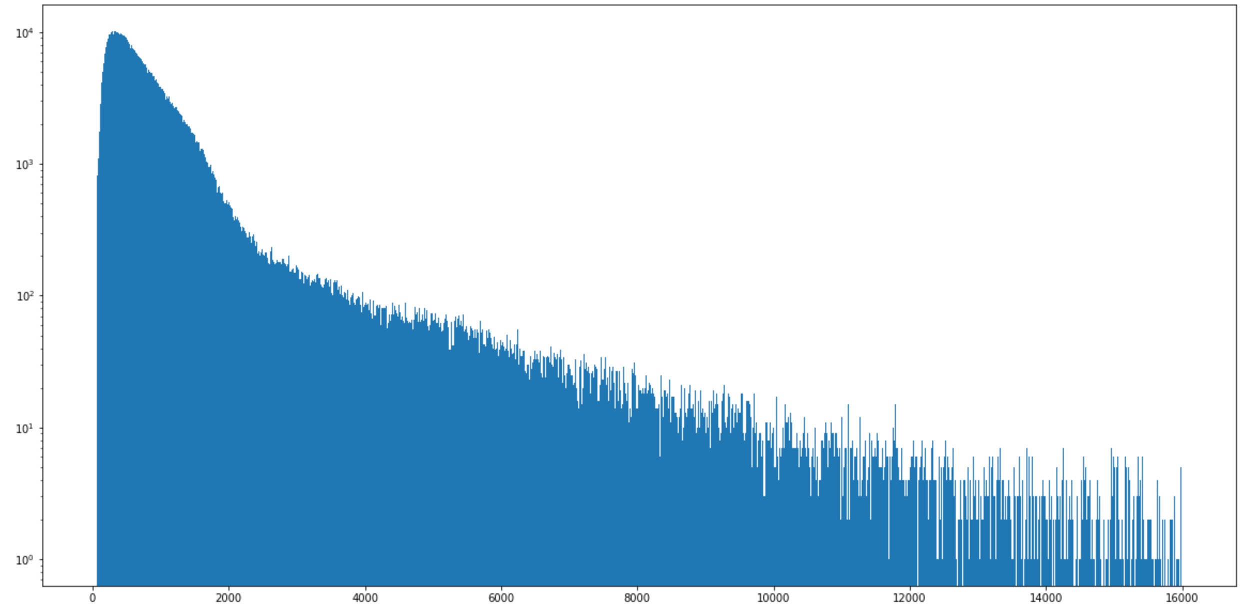
```
(973, 12)
```

In [7]:

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

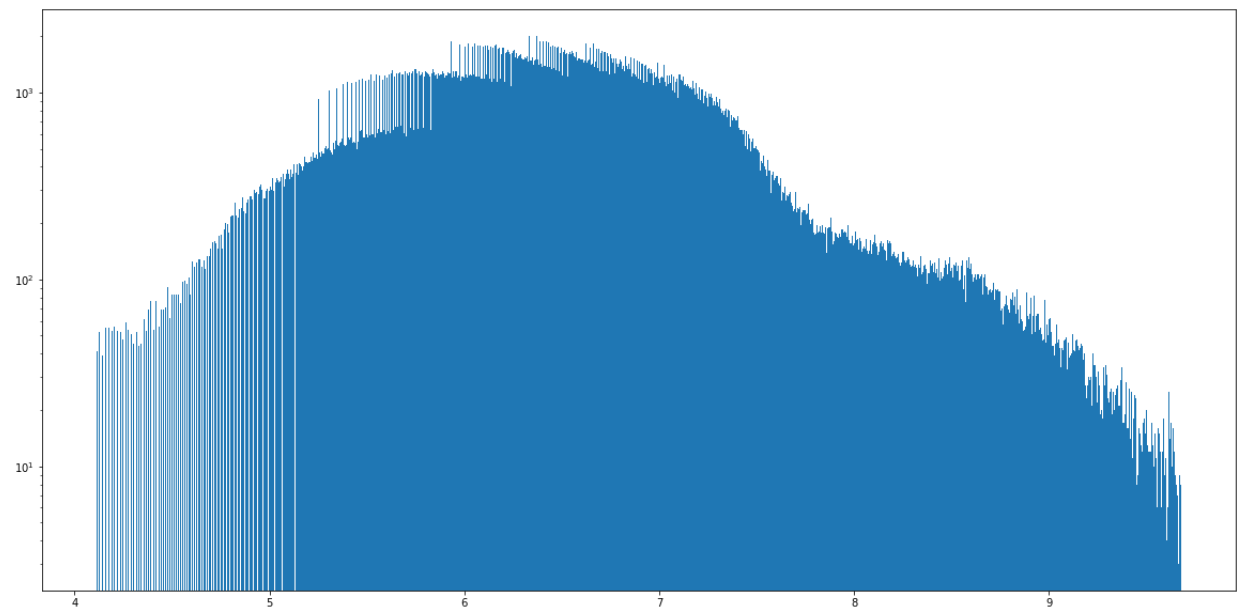
In [8]:

```
_=plt.hist(bikes[bikes.dur<16000].dur, log=True, bins=1000)
```



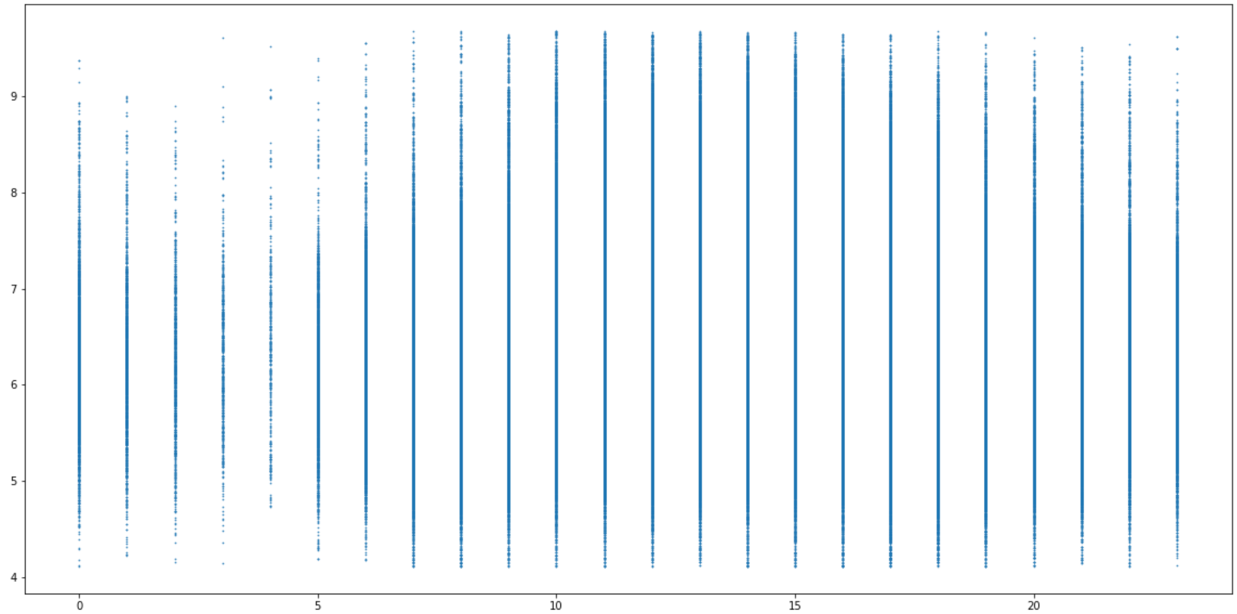
In [9]:
```
short = bikes[bikes.dur<16000]
```

In [10]:
```
_=plt.hist(np.log1p(short.dur), log=True, bins=1000)
```



In [11]:
```
plt.scatter(short.start.dt.hour, np.log1p(short.dur), s=.4)
```

Out[11]:
```
<matplotlib.collections.PathCollection at 0x24233435810>
```



In [12]:
```
np.log1p(0), np.log(0)
```

```
C:\Users\cneub\AppData\Local\Temp\ipykernel_17696\1076539907.py:1: RuntimeWarning: di
vide by zero encountered in log
  np.log1p(0), np.log(0)
```

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

In [13]:
```
bikes['log_dur'] = np.round(np.log1p(bikes.dur), 1)
```

In [14]:
```
monday = bikes[bikes.start.dt.dayofweek==1]
```

In [15]:
```
dur_hour = monday.groupby(['log_dur', monday.start.dt.hour]).count()
```

In [16]:
```
dur_hour
```

Out[16]:

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
	19	1	1	1	1	1	1	1	1	1	1
11.4	**18**	1	1	1	1	1	1	1	1	1	1

1184 rows × 12 columns

In [17]:

```
duration_hour = dur_hour.start.unstack().T.fillna(0)
duration_hour
```

Out[17]:

log_dur	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	11.(
start																	
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	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	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	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	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	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	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	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	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	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	0.(
10	0.0	0.0	1.0	3.0	5.0	7.0	8.0	5.0	10.0	31.0	...	0.0	0.0	0.0	0.0	0.0	0.(
11	1.0	0.0	2.0	5.0	4.0	7.0	7.0	10.0	13.0	22.0	...	1.0	0.0	0.0	0.0	0.0	0.(
12	0.0	0.0	4.0	2.0	7.0	6.0	12.0	16.0	36.0	30.0	...	0.0	1.0	0.0	0.0	0.0	0.(
13	0.0	2.0	6.0	3.0	5.0	6.0	4.0	15.0	20.0	36.0	...	0.0	0.0	0.0	0.0	0.0	0.(
14	2.0	0.0	1.0	1.0	3.0	8.0	9.0	11.0	26.0	24.0	...	0.0	0.0	0.0	0.0	0.0	0.(
15	0.0	3.0	0.0	5.0	1.0	7.0	6.0	22.0	26.0	31.0	...	0.0	0.0	0.0	0.0	0.0	0.(
16	2.0	6.0	1.0	11.0	6.0	10.0	14.0	17.0	36.0	35.0	...	0.0	0.0	0.0	0.0	2.0	0.(
17	3.0	7.0	7.0	13.0	12.0	14.0	20.0	36.0	57.0	71.0	...	0.0	0.0	0.0	3.0	1.0	1.(
18	0.0	4.0	7.0	9.0	13.0	20.0	21.0	40.0	79.0	75.0	...	0.0	0.0	2.0	4.0	1.0	0.(
19	3.0	0.0	7.0	7.0	9.0	16.0	19.0	34.0	43.0	52.0	...	0.0	1.0	2.0	3.0	0.0	1.(
20	0.0	7.0	2.0	4.0	2.0	13.0	14.0	19.0	34.0	38.0	...	0.0	1.0	1.0	1.0	1.0	1.(
21	1.0	2.0	1.0	2.0	3.0	6.0	16.0	19.0	26.0	35.0	...	1.0	2.0	0.0	1.0	0.0	0.(
22	1.0	0.0	2.0	2.0	1.0	8.0	1.0	13.0	10.0	20.0	...	1.0	0.0	1.0	0.0	0.0	0.(
23	0.0	0.0	1.0	0.0	2.0	5.0	4.0	8.0	3.0	5.0	...	0.0	0.0	1.0	1.0	0.0	0.(

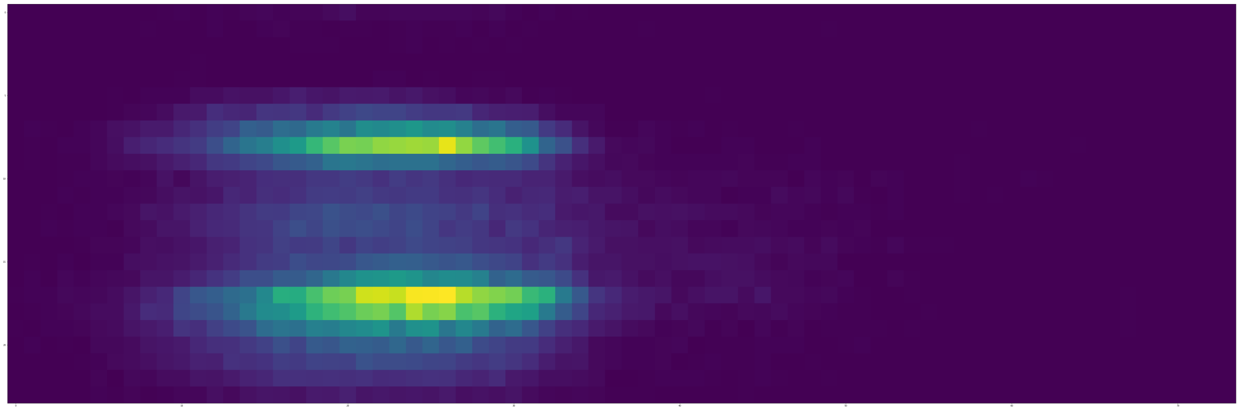
24 rows × 74 columns

In [18]:

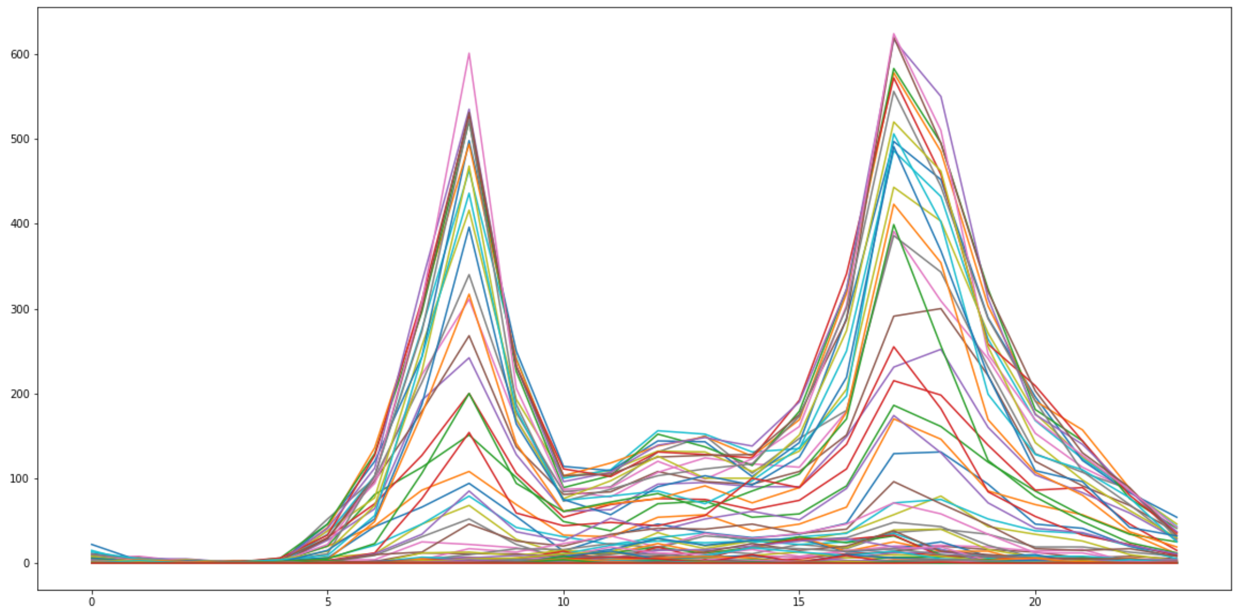
```
plt.figure(figsize=(100,100))
plt.imshow(duration_hour)
```

Out[18]:

```
<matplotlib.image.AxesImage at 0x2422c010370>
```



In [19]:
```
_=plt.plot(duration_hour)
```



In [20]:
```
bikes['Member Type'].value_counts()
```

Out[20]:
```
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 [21]:
```
np.round(.65, 1)
```

Out[21]:
```
0.6
```

In [22]:
```
37//6, (37//6)/10, 37/60
```

Out[22]:
```
(6, 0.6, 0.6166666666666667)
```

In [23]:
```
bikes['hour_of_day'] = (bikes.start.dt.hour + (bikes.start.dt.minute//6)/10)
```

In [24]:

```
bikes
```

Out[24]:

	Duration (ms)	Start date	End date	Start station number	Start station	End station number	End station	Bike number	Memb Typ
0	301295	3/31/2016 23:59	4/1/2016 0:04	31280	11th & S St NW	31506	1st & Rhode Island Ave NW	W00022	Registere
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	Registere
2	555944	3/31/2016 23:59	4/1/2016 0:08	31101	14th & V St NW	31221	18th & M St NW	W01416	Registere
3	766916	3/31/2016 23:57	4/1/2016 0:09	31226	34th St & Wisconsin Ave NW	31214	17th & Corcoran St NW	W01090	Registere
4	139656	3/31/2016 23:57	3/31/2016 23:59	31011	23rd & Crystal Dr	31009	27th & Crystal Dr	W21934	Registere
...	...	...	...	...	...	...	...	...	
552394	782042	1/1/2016 0:16	1/1/2016 0:29	31266	11th & M St NW	31278	18th & R St NW	W22090	Registere
552395	213976	1/1/2016 0:15	1/1/2016 0:19	31506	1st & Rhode Island Ave NW	31509	New Jersey Ave & R St NW	W01294	Registere
552396	715013	1/1/2016 0:13	1/1/2016 0:25	31222	New York Ave & 15th St NW	31214	17th & Corcoran St NW	W21427	Registere
552397	448007	1/1/2016 0:10	1/1/2016 0:17	32039	Old Georgetown Rd & Southwick St	32002	Bethesda Ave & Arlington Rd	W22202	Registere
552398	166066	1/1/2016 0:06	1/1/2016 0:09	31102	11th & Kenyon St NW	31105	14th & Harvard St NW	W01346	Registere

552399 rows × 14 columns

In [25]:

```
bikes['roundhour_of_day'] = (bikes.start.dt.hour ) # keep the hour handy as well
```

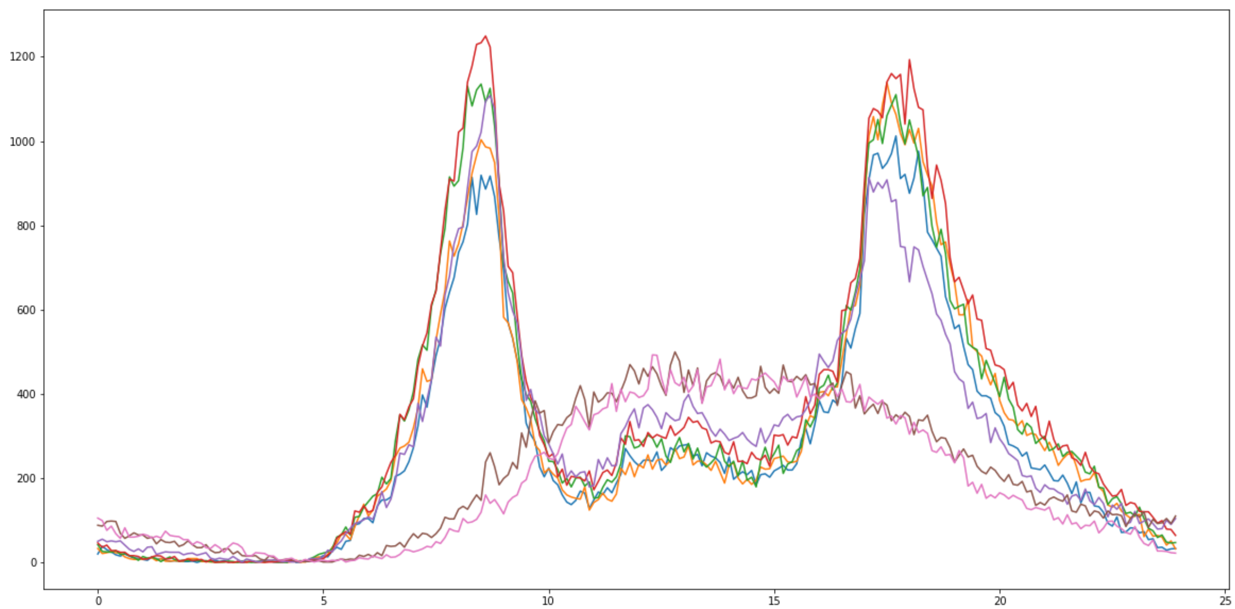
Aggregate to get a count per hour/minute of the day across all trips

In [26]:

```
reg_bikes = bikes[bikes['Member Type']=='Registered']
hours = reg_bikes.groupby([reg_bikes.hour_of_day, reg_bikes.start.dt.dayofweek]).agg(
hours['hour'] = hours.index
day_hour_count = hours.dur.unstack()
plt.figure(figsize=(20,10))
plt.plot(day_hour_count.index, day_hour_count[0])
plt.plot(day_hour_count.index, day_hour_count[1])
plt.plot(day_hour_count.index, day_hour_count[2])
plt.plot(day_hour_count.index, day_hour_count[3])
plt.plot(day_hour_count.index, day_hour_count[4])
plt.plot(day_hour_count.index, day_hour_count[5])
plt.plot(day_hour_count.index, day_hour_count[6])
```

Out[26]:

```
[<matplotlib.lines.Line2D at 0x242308bf130>]
```



In [27]:

```
day_hour_count
```

Out[27]:

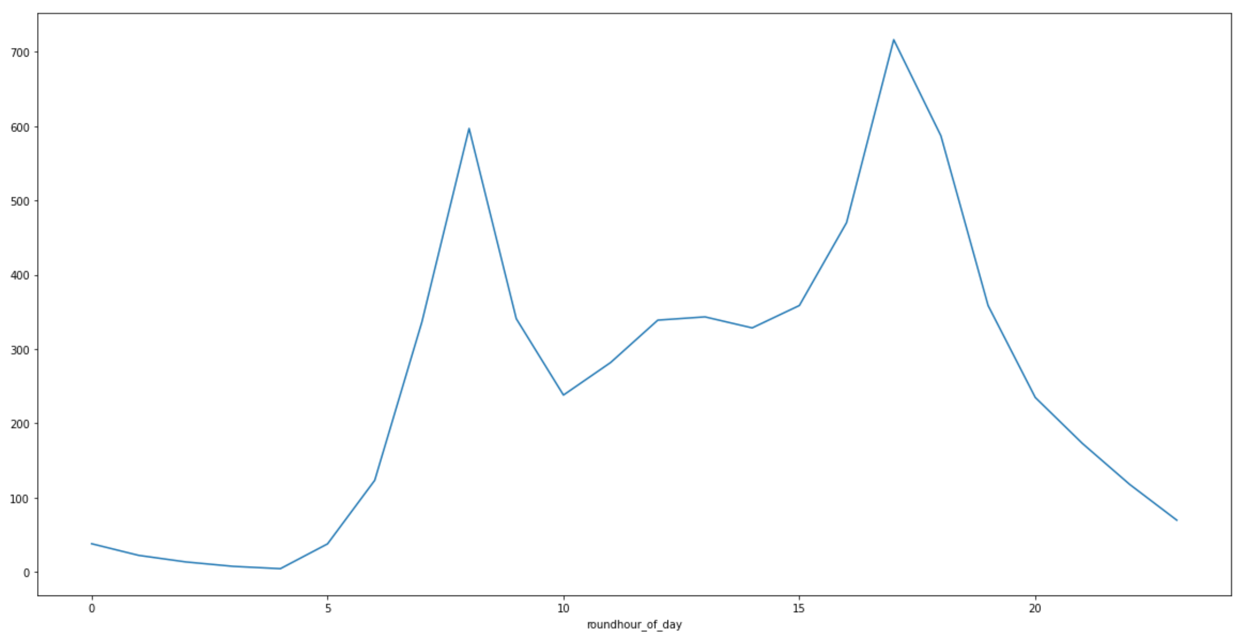
start	0	1	2	3	4	5	6
hour_of_day							
0.0	21.0	34.0	43.0	47.0	51.0	89.0	106.0
0.1	39.0	22.0	27.0	37.0	56.0	87.0	100.0
0.2	31.0	24.0	26.0	42.0	50.0	98.0	77.0
0.3	26.0	27.0	25.0	29.0	52.0	99.0	87.0
0.4	19.0	24.0	29.0	29.0	50.0	98.0	69.0
...	...	...	...	...	...	...	...
23.5	36.0	65.0	60.0	94.0	80.0	93.0	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 [28]:

```
hoursn = bikes.groupby('roundhour_of_day').agg('count')
hoursn['hour'] = hoursn.index
(hoursn.start/90).plot() # 90 days in a quarter
```

Out[28]: <AxesSubplot:xlabel='roundhour_of_day'>



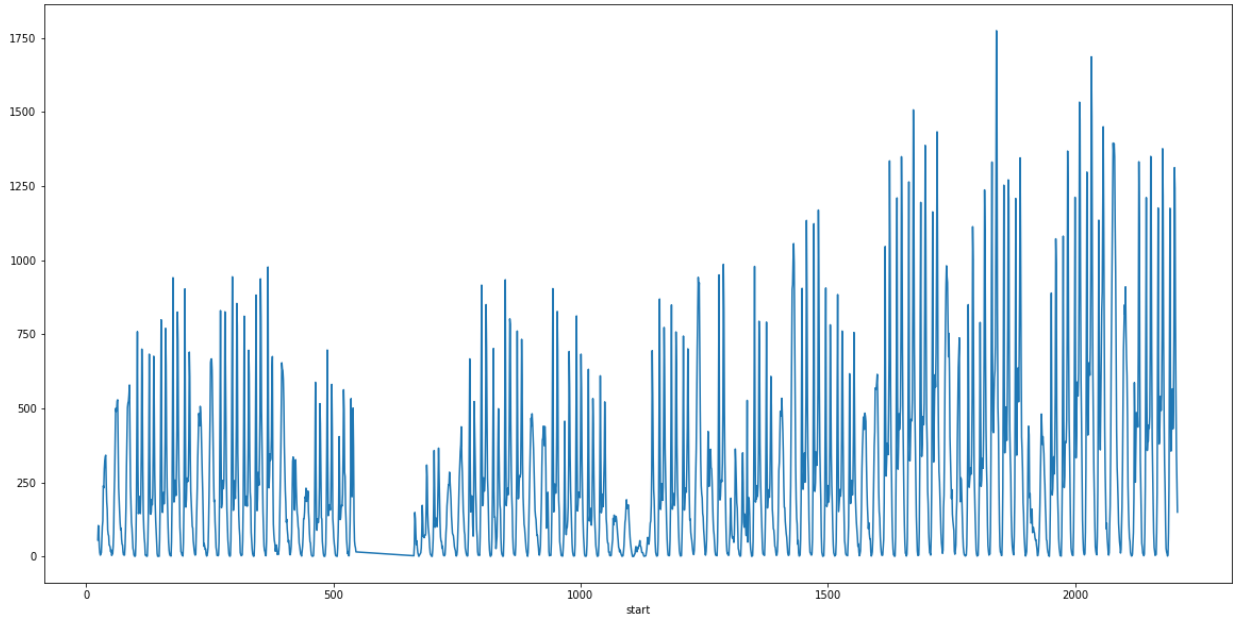
In [29]:

```
hour_count = bikes.groupby(bikes.start.dt.dayofyear*24 + bikes.start.dt.hour).count()
```

In [30]:

```
plt.figure(figsize=(20,10))
hour_count.start.plot()
```

Out[30]: <AxesSubplot:xlabel='start'>



In [31]:

```
day_count = bikes.groupby(bikes.start.dt.dayofyear).count()
```

In [32]:

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

In [33]:

```
day_hour.start.unstack()
```

Out[33]:

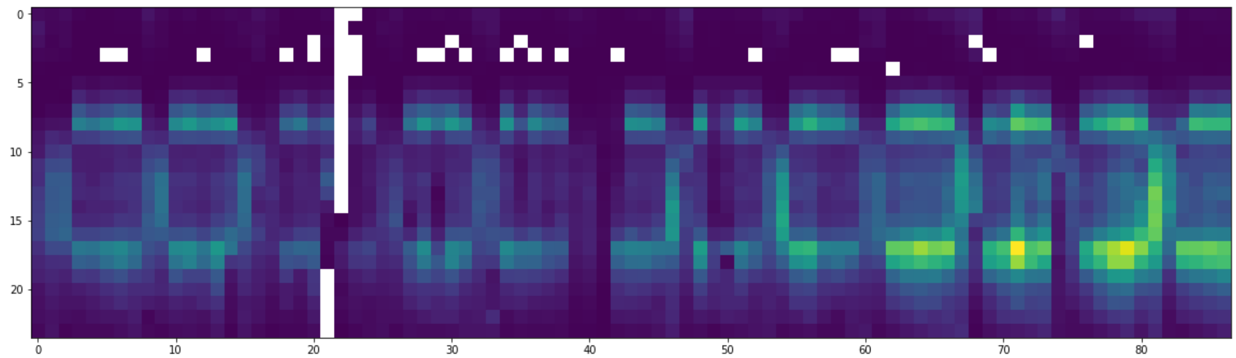
start	0	1	2	3	4	5	6	7	8	9	...	14	15	16	17
start															
1	56.0	105.0	74.0	32.0	13.0	5.0	10.0	14.0	54.0	101.0	...	324.0	338.0	342.0	247.0
2	37.0	31.0	17.0	23.0	4.0	7.0	10.0	34.0	80.0	203.0	...	495.0	525.0	529.0	392.0
3	59.0	42.0	39.0	15.0	6.0	9.0	5.0	33.0	87.0	168.0	...	524.0	546.0	579.0	398.0
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	700.0
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	676.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
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	611.0
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	1332.0
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	1350.0
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	1376.0
91	28.0	16.0	10.0	2.0	8.0	80.0	240.0	750.0	1175.0	589.0	...	431.0	504.0	746.0	1312.0

87 rows × 24 columns

In [34]:

```python
plt.figure(figsize=(20,10))
plt.imshow(day_hour.start.unstack().T)
```

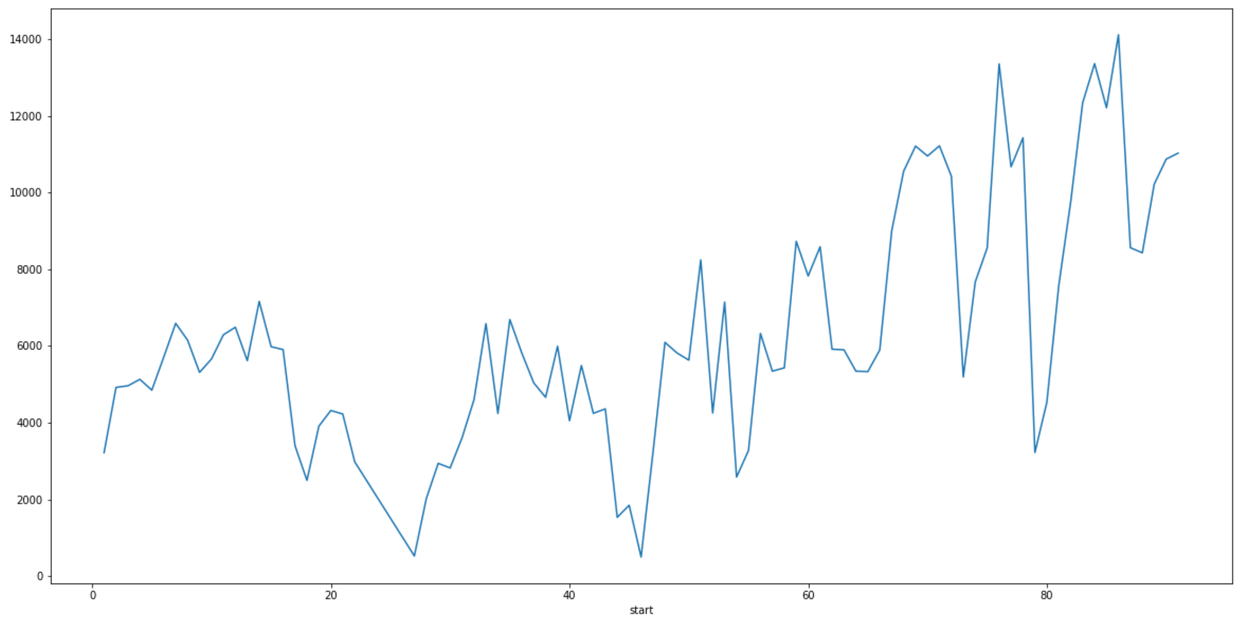
Out[34]: <matplotlib.image.AxesImage at 0x2422a70eaa0>



In [35]:

```python
day_count.start.plot()
```

Out[35]: <AxesSubplot:xlabel='start'>



In [36]:

```python
bikes.start.dt.dayofyear
```

Out[36]:

```
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 [37]:

```python
bikes[bikes.start=="2016-01-10"].shape
```

Out[37]:
```
(1, 15)
```

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

In [38]:
```
monday = day_hour_count[[0]].copy()
```

In [39]:
```
monday["hour_of_day"] = monday.index
monday.index.name = None
monday.rename(columns = {0: "usage", "hour_of_day":"hour_of_day"}, inplace = True)
```

In [40]:
```
monday
```

Out[40]:

start	usage	hour_of_day
0.0	21.0	0.0
0.1	39.0	0.1
0.2	31.0	0.2
0.3	26.0	0.3
0.4	19.0	0.4
...	...	...
23.5	36.0	23.5
23.6	37.0	23.6
23.7	30.0	23.7
23.8	33.0	23.8
23.9	34.0	23.9

240 rows × 2 columns

In [41]:
```
saturday = day_hour_count[[5]].copy()
```

In [42]:
```
saturday["hour_of_day"] = saturday.index
saturday.index.name = None
saturday.rename(columns = {5: "usage", "hour_of_day":"hour_of_day"}, inplace = True)
```

In [43]:
```
saturday
```

Out[43]:

start	usage	hour_of_day
0.0	89.0	0.0
0.1	87.0	0.1
0.2	98.0	0.2
0.3	99.0	0.3
0.4	98.0	0.4
...	...	...
23.5	93.0	23.5
23.6	95.0	23.6
23.7	105.0	23.7
23.8	93.0	23.8
23.9	111.0	23.9

240 rows × 2 columns

2a. Create 3 models fit to `monday.hour_of_day` with varying polynomial degrees (choose from `n=1,2,3,5,10,15`). (Repeat for `saturday` below)

Plot all the results for each polynomial.

Monday - Polynomials

In [44]:
```
from sklearn.preprocessing import PolynomialFeatures
from sklearn import linear_model
xmon = monday.hour_of_day.values.reshape(-1, 1)
ymon = monday.usage.values.reshape(-1, 1)
np.isnan(xmon).any(), np.isnan(ymon).any()
```

Out[44]:
```
(False, True)
```

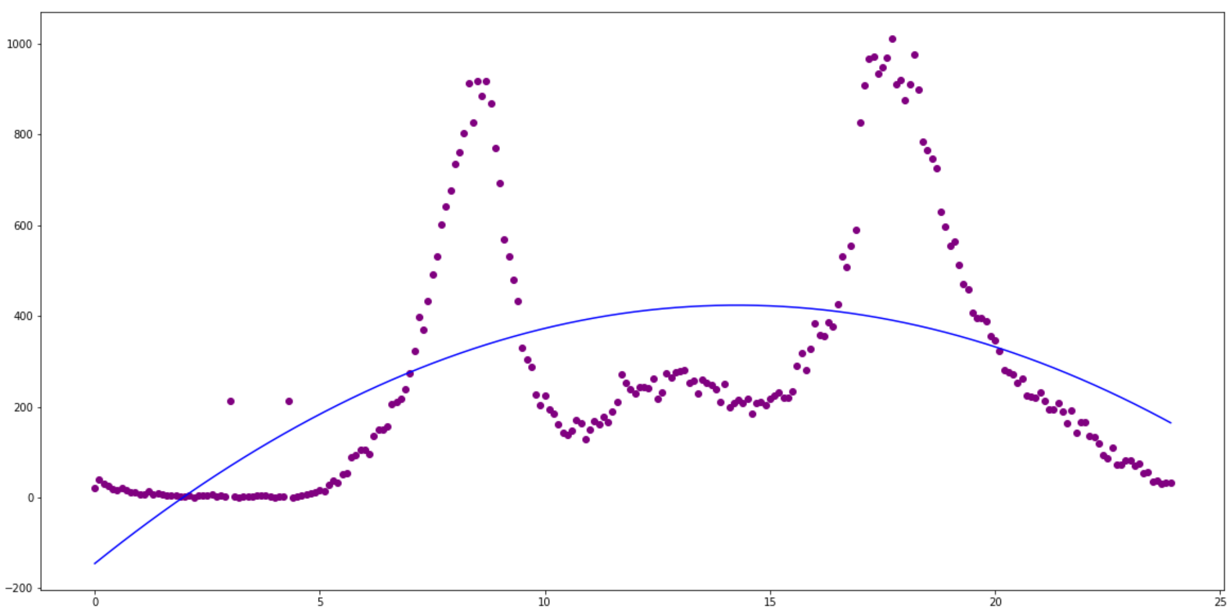
In [45]:
```
ymon[np.isnan(ymon)] = np.median(ymon[~np.isnan(ymon)])
```

First, I started with 2, 3, 5, and 10 degree polynomials. The fits are not excellent, but the 10 degree polynomial seems to fit Monday the best.

In [46]:

```
# 2 Degree Polynomial
poly = PolynomialFeatures(degree = 2)
xmon_2 = poly.fit_transform(xmon.reshape(-1, 1))

# Linear
linear = linear_model.LinearRegression()
linear.fit(xmon_2, ymon)

xmon_2 = np.squeeze(np.asarray(xmon_2))
linear.coef_ = np.squeeze(np.asarray(linear.coef_))

plt.scatter(xmon, ymon, c = "purple")
plt.plot(xmon, np.dot(xmon_2, linear.coef_) + linear.intercept_, c = "blue")
```

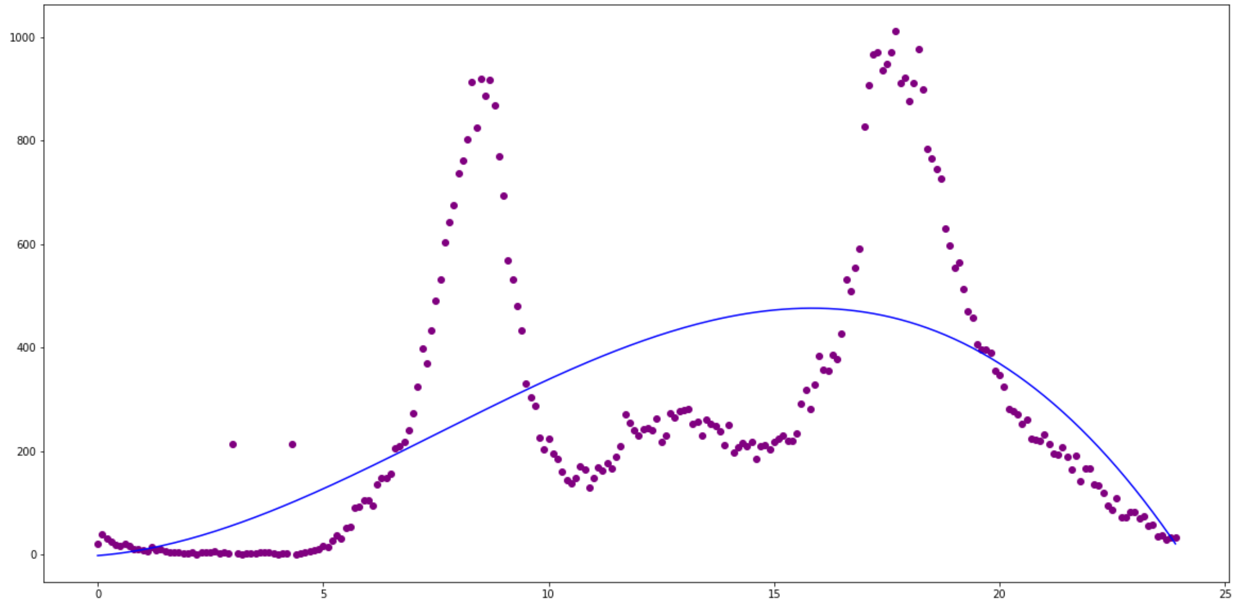
Out[46]: [<matplotlib.lines.Line2D at 0x24233b12530>]



In [47]:

```
# 3 Degree Polynomial
poly = PolynomialFeatures(degree = 3)
xmon_3 = poly.fit_transform(xmon.reshape(-1, 1))

# Linear
linear = linear_model.LinearRegression()
linear.fit(xmon_3, ymon)

xmon_3 = np.squeeze(np.asarray(xmon_3))
linear.coef_ = np.squeeze(np.asarray(linear.coef_))

plt.scatter(xmon, ymon, c = "purple")
plt.plot(xmon, np.dot(xmon_3, linear.coef_) + linear.intercept_, c = "blue")
```

Out[47]: [<matplotlib.lines.Line2D at 0x24233b6df60>]

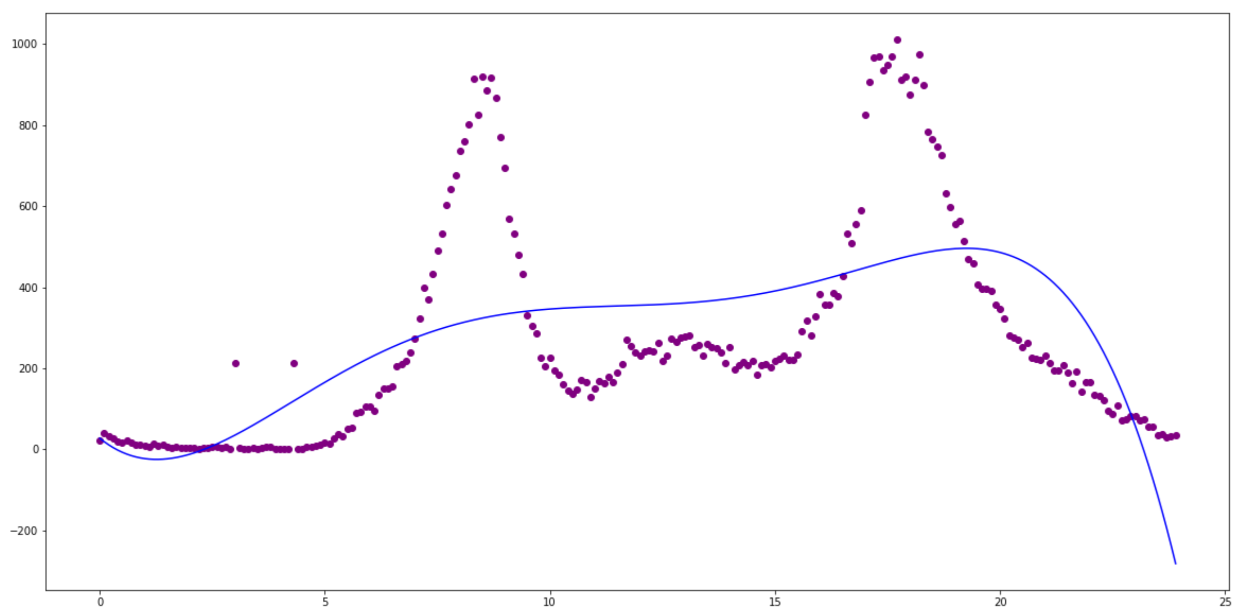


In [48]:

```
# 5 Degree Polynomial
poly = PolynomialFeatures(degree = 5)
xmon_5 = poly.fit_transform(xmon.reshape(-1, 1))

# Linear
linear = linear_model.LinearRegression()
linear.fit(xmon_5, ymon)

xmon_5 = np.squeeze(np.asarray(xmon_5))
linear.coef_ = np.squeeze(np.asarray(linear.coef_))

plt.scatter(xmon, ymon, c = "purple")
plt.plot(xmon, np.dot(xmon_5, linear.coef_) + linear.intercept_, c = "blue")
```

Out[48]:

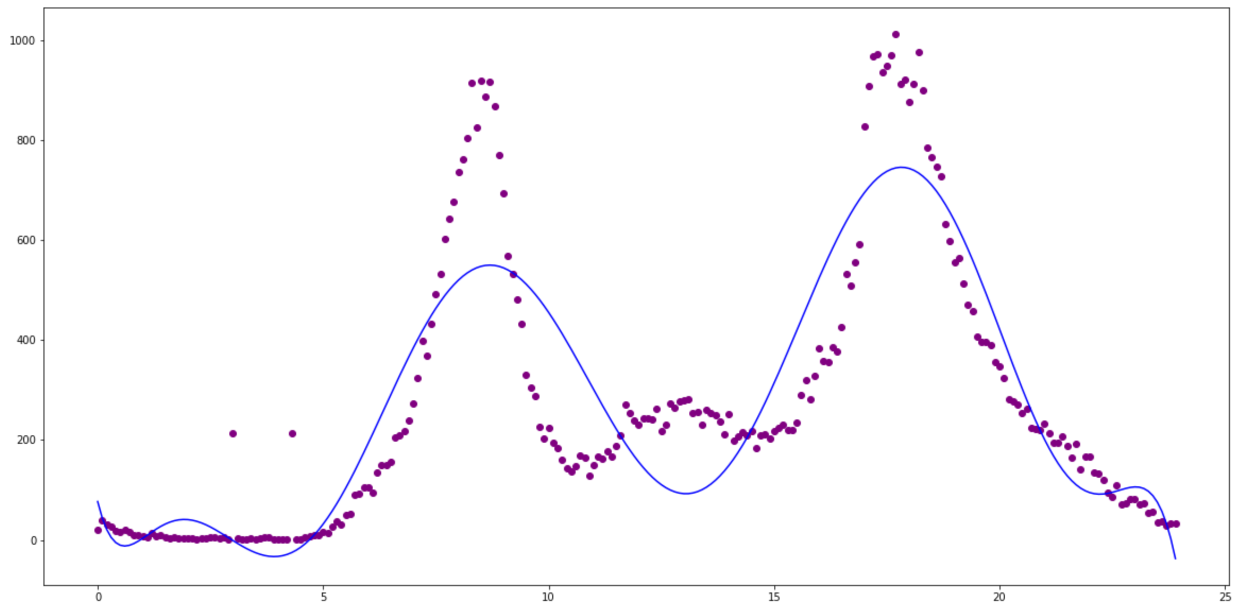
```
[<matplotlib.lines.Line2D at 0x24233bb7220>]
```



As we can see above, a lower degree of polynomial does not fit the Monday data well. A 10 degree polynomial is much better, but still not particularly ideal.

In [49]:
```
# 10 Degree Polynomial
poly = PolynomialFeatures(degree = 10)
xmon_10 = poly.fit_transform(xmon.reshape(-1, 1))

# Linear
linear = linear_model.LinearRegression()
linear.fit(xmon_10, ymon)

xmon_10 = np.squeeze(np.asarray(xmon_10))
linear.coef_ = np.squeeze(np.asarray(linear.coef_))

plt.scatter(xmon, ymon, c = "purple")
plt.plot(xmon, np.dot(xmon_10, linear.coef_) + linear.intercept_, c = "blue")
```

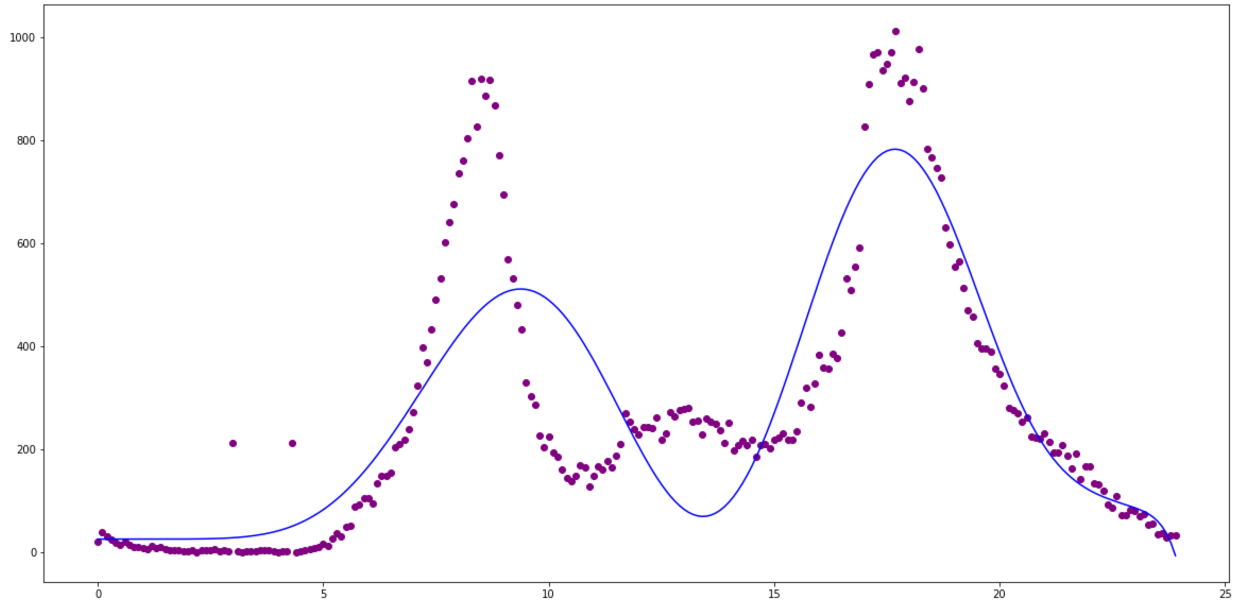
Out[49]: [<matplotlib.lines.Line2D at 0x24233c29f60>]



Out of curiosity, I tried to fit this data with a 15 degree polynomial. The fit does not appear to improve between 10-15 degrees, and I worry about overfitting in the 15 degree model.

In [50]:
```
# 15 Degree Polynomial
poly = PolynomialFeatures(degree = 15)
xmon_15 = poly.fit_transform(xmon.reshape(-1, 1))

# Linear
linear = linear_model.LinearRegression()
linear.fit(xmon_15, ymon)

xmon_15 = np.squeeze(np.asarray(xmon_15))
linear.coef_ = np.squeeze(np.asarray(linear.coef_))

plt.scatter(xmon, ymon, c = "purple")
plt.plot(xmon, np.dot(xmon_15, linear.coef_) + linear.intercept_, c = "blue")
```

Out[50]: [<matplotlib.lines.Line2D at 0x24234000af0>]



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

In [51]:

```
xsat = saturday.hour_of_day.values.reshape(-1, 1)
ysat = saturday.usage.values.reshape(-1, 1)
np.isnan(xsat).any(), np.isnan(ysat).any()
```

Out[51]:

```
(False, False)
```

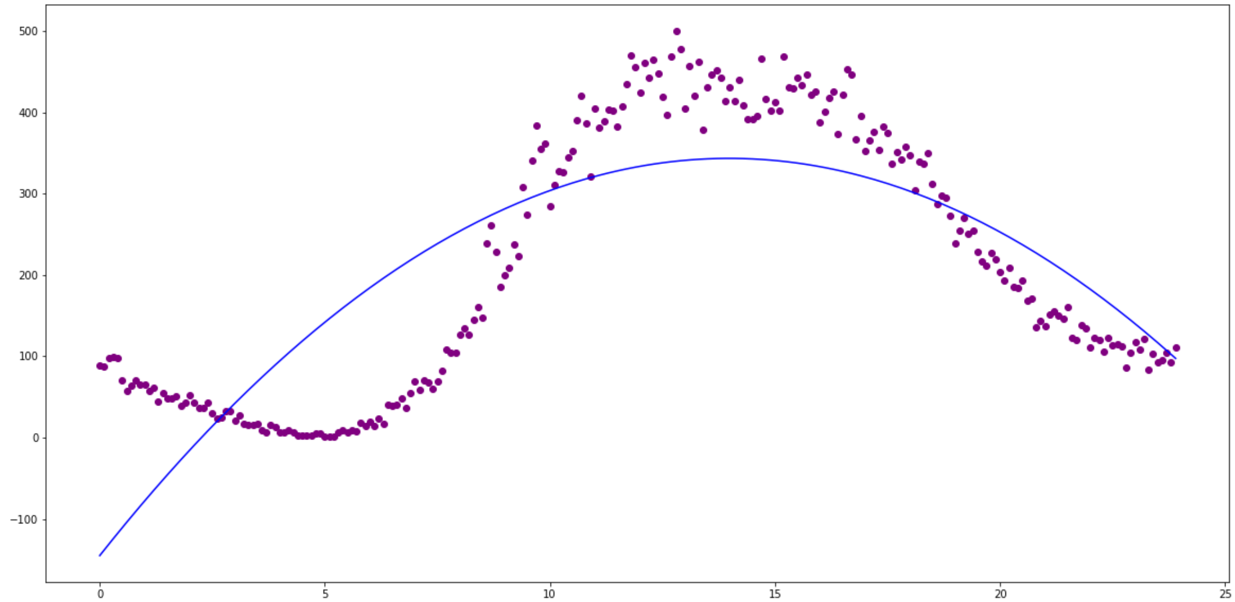
First, I decided to start with 2 degrees in my polynomial to see how well the model fit.

In [52]:

```
# 2 Degree Polynomial
poly = PolynomialFeatures(degree = 2)
xsat_2 = poly.fit_transform(xsat.reshape(-1, 1))

# Linear
linear = linear_model.LinearRegression()
linear.fit(xsat_2, ysat)
linear.coef_, linear.intercept_

xsat_2 = np.squeeze(np.asarray(xsat_2))
linear.coef_ = np.squeeze(np.asarray(linear.coef_))

plt.scatter(xsat, ysat, c = "purple")
plt.plot(xsat, np.dot(xsat_2, linear.coef_) + linear.intercept_, c = "blue")
```

Out[52]:

```
[<matplotlib.lines.Line2D at 0x24234042620>]
```



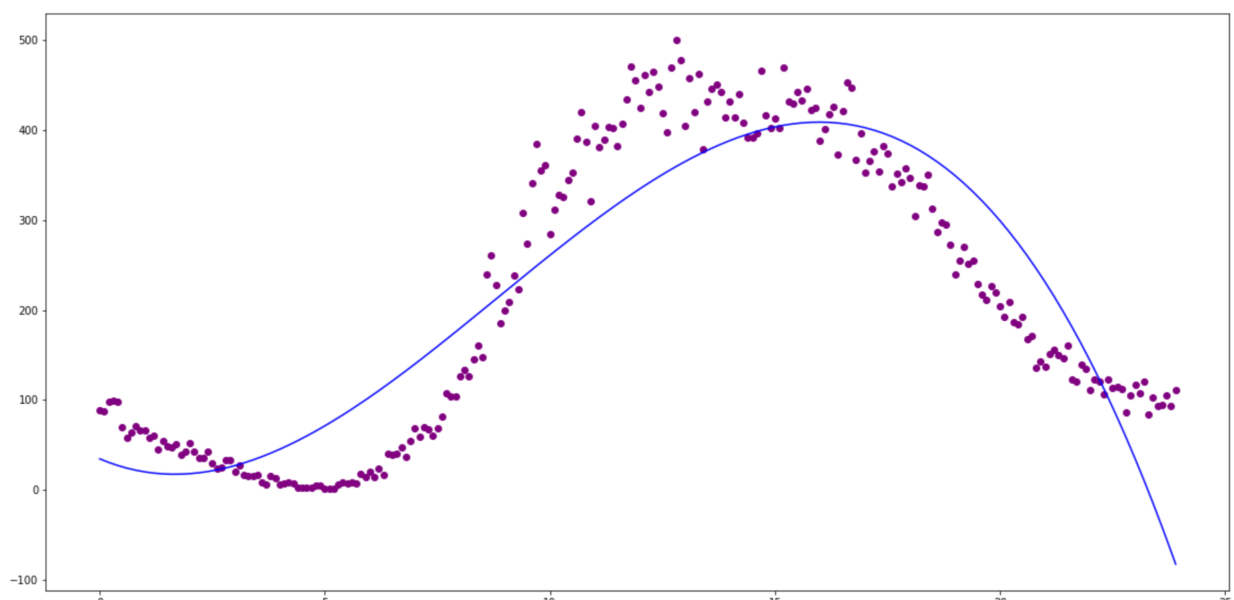
Clearly, we should increae the degree of our polynomial model (so I tested out 3-5 below).

In [53]:

```
# 3 Degree Polynomial
poly = PolynomialFeatures(degree = 3)
xsat_3 = poly.fit_transform(xsat.reshape(-1, 1))

# Linear
linear = linear_model.LinearRegression()
linear.fit(xsat_3, ysat)
linear.coef_, linear.intercept_

xsat_3 = np.squeeze(np.asarray(xsat_3))
linear.coef_ = np.squeeze(np.asarray(linear.coef_))

plt.scatter(xsat, ysat, c = "purple")
plt.plot(xsat, np.dot(xsat_3, linear.coef_) + linear.intercept_, c = "blue")
```

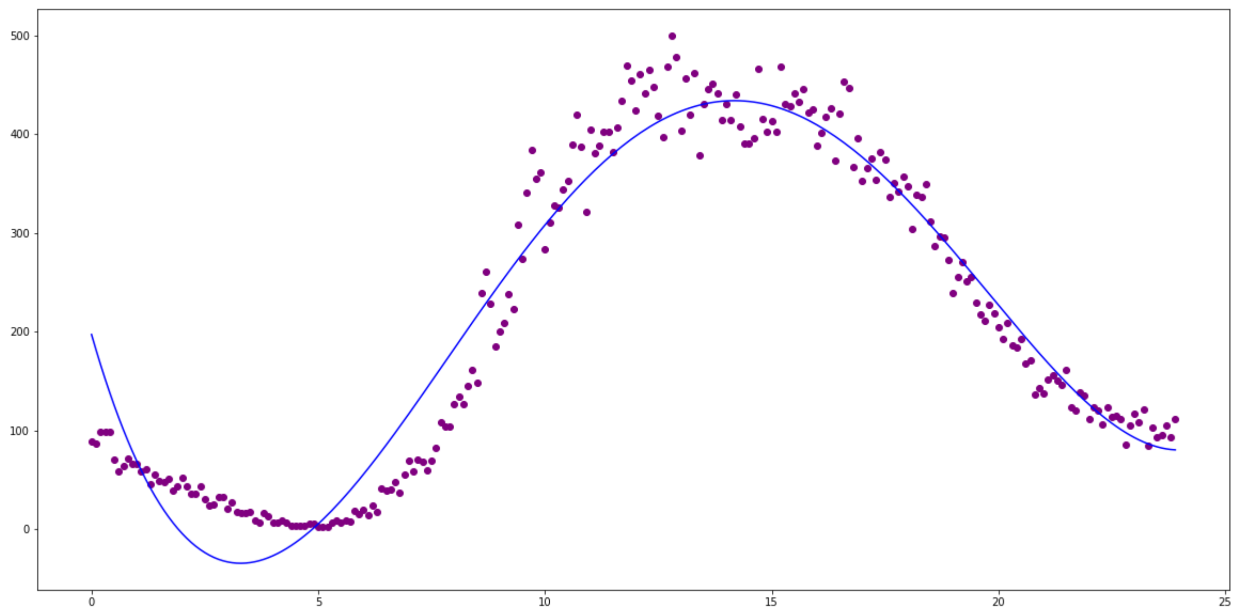
Out[53]:

```
[<matplotlib.lines.Line2D at 0x242340bd570>]
```



In [54]:
```
# 4 Degree Polynomial
poly = PolynomialFeatures(degree = 4)
xsat_4 = poly.fit_transform(xsat.reshape(-1, 1))

# Linear
linear = linear_model.LinearRegression()
linear.fit(xsat_4, ysat)

xsat_4 = np.squeeze(np.asarray(xsat_4))
linear.coef_ = np.squeeze(np.asarray(linear.coef_))

plt.scatter(xsat, ysat, c = "purple")
plt.plot(xsat, np.dot(xsat_4, linear.coef_) + linear.intercept_, c = "blue")
```

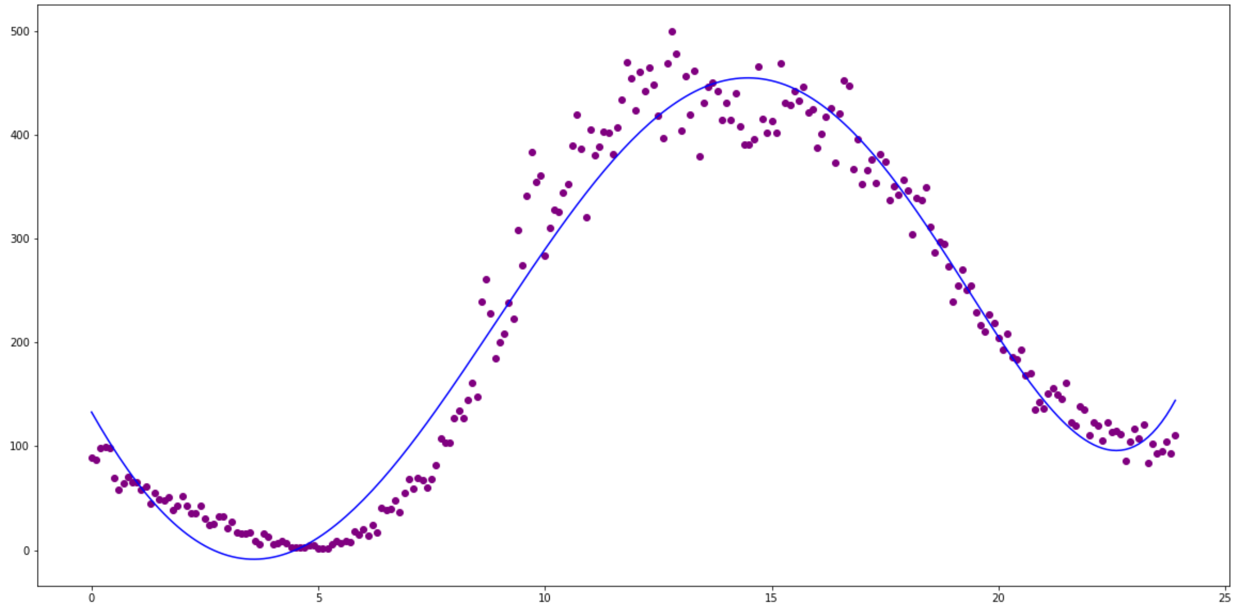
Out[54]:
```
[<matplotlib.lines.Line2D at 0x2423905caf0>]
```



I think a 5 degree polynomial fit the data best. I tried 10, but the model becomes overfit.

In [59]:
```
# 5 Degree Polynomial
poly = PolynomialFeatures(degree = 5)
xsat_5 = poly.fit_transform(xsat.reshape(-1, 1))

# Linear
linear = linear_model.LinearRegression()
linear.fit(xsat_5, ysat)
linear.coef_, linear.intercept_

xsat_5 = np.squeeze(np.asarray(xsat_5))
linear.coef_ = np.squeeze(np.asarray(linear.coef_))

plt.scatter(xsat, ysat, c = "purple")
plt.plot(xsat, np.dot(xsat_5, linear.coef_) + linear.intercept_, c = "blue")
```

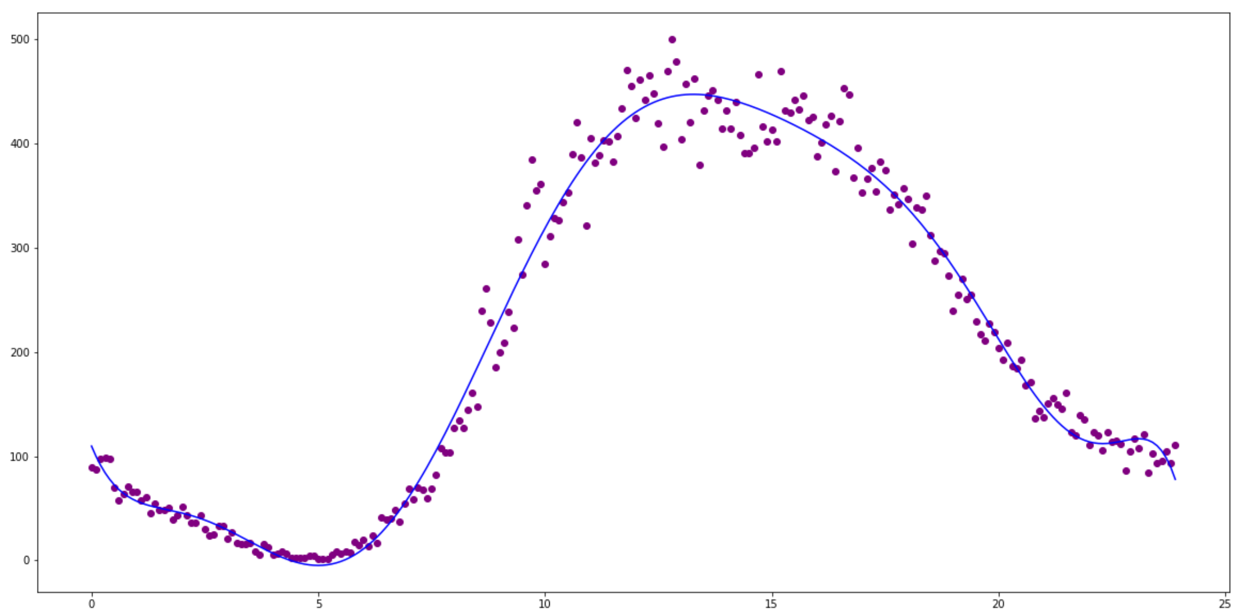
Out[59]:
```
[<matplotlib.lines.Line2D at 0x24233383c40>]
```



In [60]:

```
# 10 Degree Polynomial
poly = PolynomialFeatures(degree = 10)
xsat_10 = poly.fit_transform(xsat.reshape(-1, 1))

# Linear
linear = linear_model.LinearRegression()
linear.fit(xsat_10, ysat)
linear.coef_, linear.intercept_

xsat_10 = np.squeeze(np.asarray(xsat_10))
linear.coef_ = np.squeeze(np.asarray(linear.coef_))

plt.scatter(xsat, ysat, c = "purple")
plt.plot(xsat, np.dot(xsat_10, linear.coef_) + linear.intercept_, c = "blue")
```

Out[60]: [<matplotlib.lines.Line2D at 0x242332b4e20>]



In the above plot, we can see that the model is overfit.

3. Create 3 new models fit to `hour_of_day` with different Ridge Regression α (alpha) Ridge Coefficient values using the `monday` and `saturday` datasets.

Monday - Ridge

There is very little difference between the linear model and the Ridge model for Monday's data. Even with alpha at 50+, the fit is nearly identical. I believe adding the polynomial values would be more useful. Even though the alpha value doesn't seem to impact the fit, I graphed three models with alphas 0.5, 10, and 50.

In [73]:

```
xmon = monday.hour_of_day.values.reshape(-1, 1)
ymon = monday.usage.values.reshape(-1, 1)
np.isnan(xmon).any(), np.isnan(ymon).any()
```

Out[73]:

```
(False, False)
```

In [74]:

```
# Linear
linear = linear_model.LinearRegression()
linear.fit(xmon,ymon)

linear.coef_, linear.intercept_
```

Out[74]:

```
(array([12.97857602]), 119.68101659751045)
```

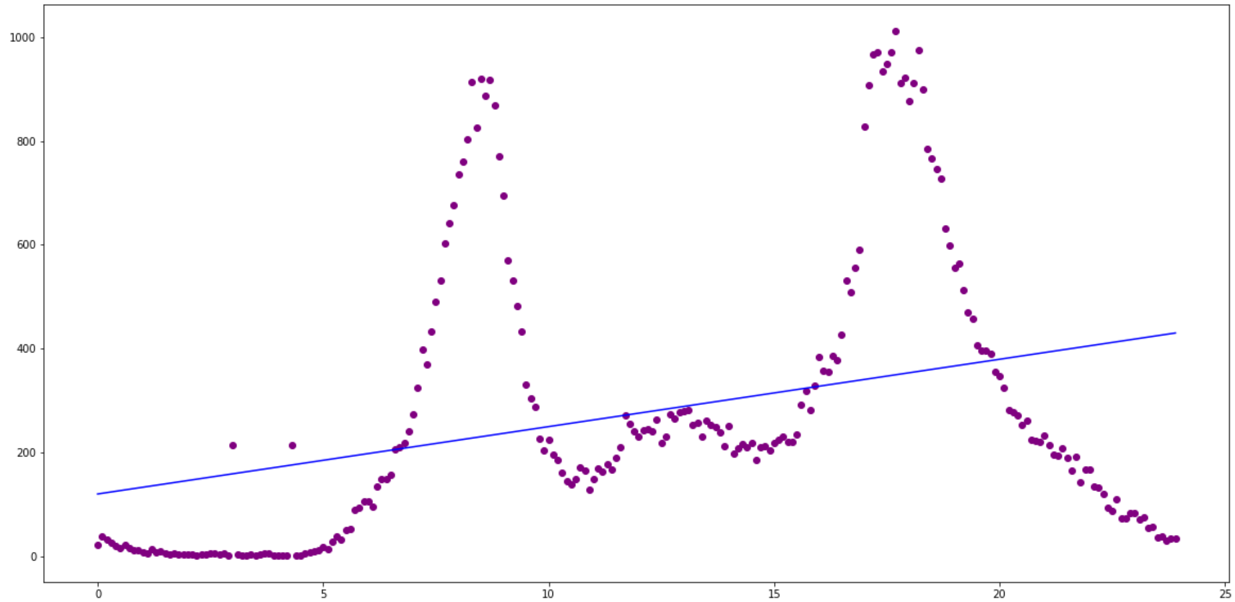
In [78]:

```
# Ridge (alpha = 0.5)
ridge = linear_model.Ridge(alpha = 0.5)
ridge.fit(xmon, ymon)

print(ridge.coef_, ridge.intercept_)

plt.scatter(xmon, ymon, c = "purple")
plt.plot(xmon, xmon*ridge.coef_ + ridge.intercept_, c = "blue")
```

```
[12.97801273] 119.68774793191145
```

Out[78]:

```
[<matplotlib.lines.Line2D at 0x24232ea2c20>]
```



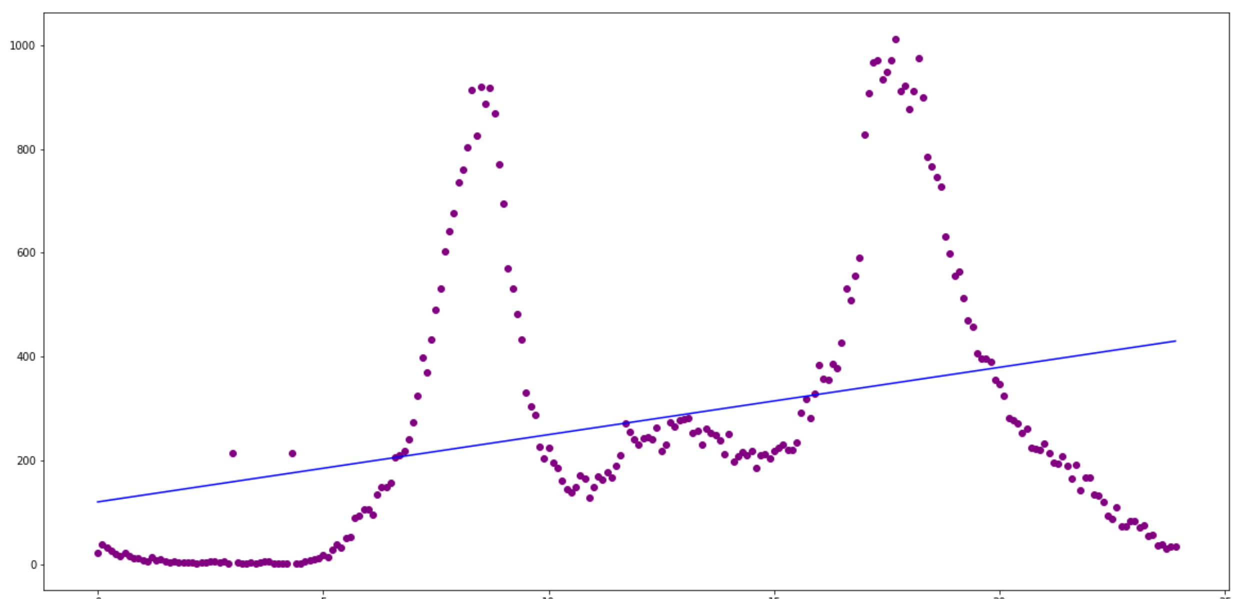
In [79]:

```
# Ridge (alpha = 10)
ridge = linear_model.Ridge(alpha = 10)
ridge.fit(xmon, ymon)

print(ridge.coef_, ridge.intercept_)

plt.scatter(xmon, ymon, c = "purple")
plt.plot(xmon, xmon*ridge.coef_ + ridge.intercept_, c = "blue")
```

```
[12.96731947] 119.8155323596246
```

Out[79]:

```
[<matplotlib.lines.Line2D at 0x24232f51de0>]
```



In [80]:

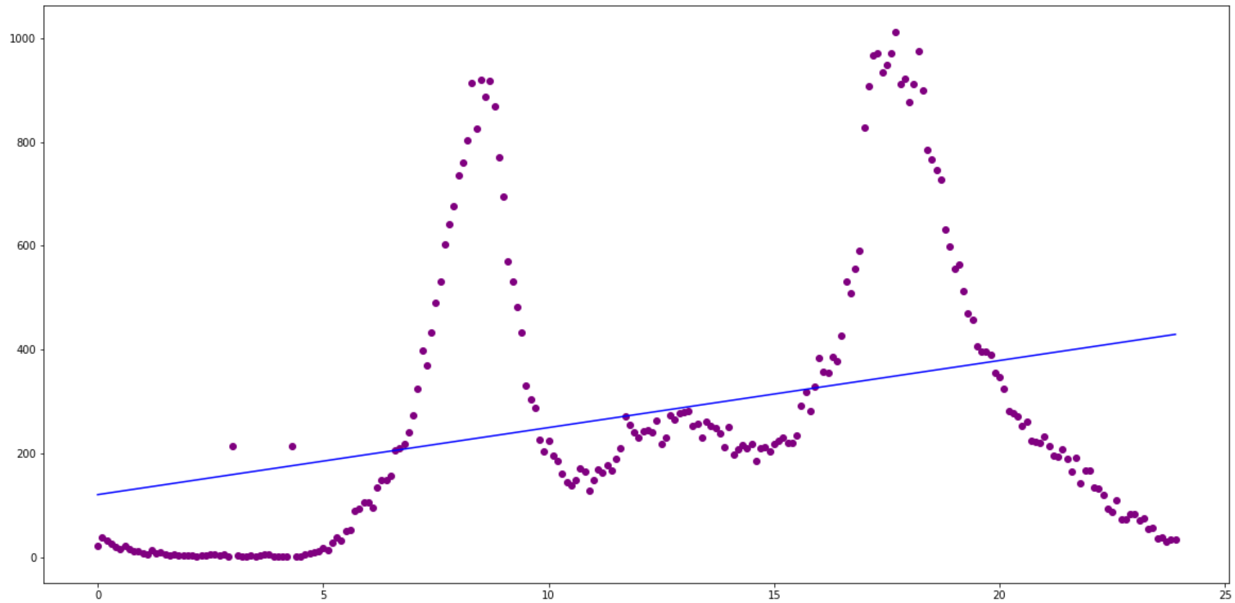
```
# Ridge (alpha = 50)
ridge = linear_model.Ridge(alpha = 50)
ridge.fit(xmon, ymon)

print(ridge.coef_, ridge.intercept_)
```

```
plt.scatter(xmon, ymon, c = "purple")
plt.plot(xmon, xmon*ridge.coef_ + ridge.intercept_, c = "blue")
```

```
[12.92248786] 120.35127011702878
```

Out[80]:

```
[<matplotlib.lines.Line2D at 0x24232f68f70>]
```



Saturday - Ridge

In [67]:

```
xsat = saturday.hour_of_day.values.reshape(-1, 1)
ysat = saturday.usage.values.reshape(-1, 1)
np.isnan(xsat).any(), np.isnan(ysat).any()
```

Out[67]:

```
(False, False)
```

In [68]:

```
# Linear
linear = linear_model.LinearRegression()
linear.fit(xsat, ysat)

linear.coef_, linear.intercept_
```

Out[68]:

```
(array([[10.13721158]]), array([91.97282158]))
```

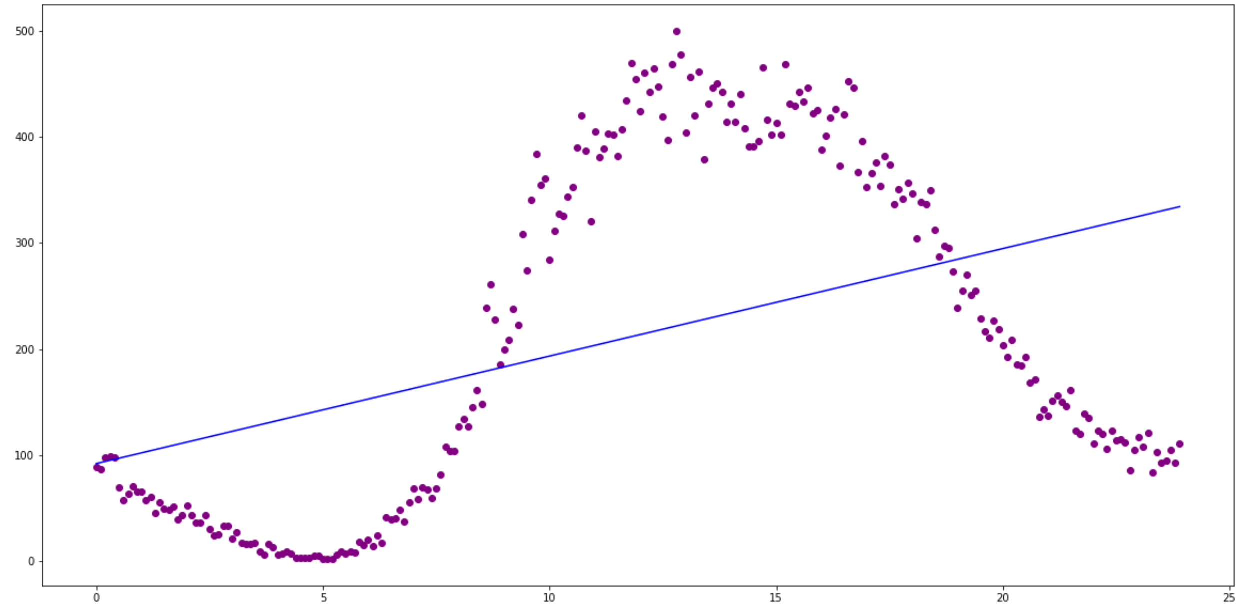
In [81]:

```
# Ridge (alpha = 0.5)
ridge = linear_model.Ridge(alpha = 0.5)
ridge.fit(xsat, ysat)

print(ridge.coef_, ridge.intercept_)

plt.scatter(xsat, ysat, c = "purple")
plt.plot(xsat, xsat*ridge.coef_ + ridge.intercept_, c = "blue")
```

```
[[10.13677161]] [91.97807924]
```

Out[81]:

```
[<matplotlib.lines.Line2D at 0x24232e1c190>]
```



In [ ]:

```
# Ridge (alpha = 10)
ridge = linear_model.Ridge(alpha = 10)
ridge.fit(xsat, ysat)

print(ridge.coef_, ridge.intercept_)

plt.scatter(xsat, ysat, c = "purple")
plt.plot(xsat, xsat*ridge.coef_ + ridge.intercept_, c = "blue")
```

In [ ]:

```
# Ridge (alpha = 50)
ridge = linear_model.Ridge(alpha = 50)
ridge.fit(xsat, ysat)

print(ridge.coef_, ridge.intercept_)

plt.scatter(xsat, ysat, c = "purple")
plt.plot(xsat, xsat*ridge.coef_ + ridge.intercept_, c = "blue")
```