Analysis of Electric Vehicle Charging Stations

Team Primary Roles

- Grant: Created notebook at the beginning of the project (he is no longer part of the project)
- Kevin: Code writing (specifically heat maps, heatmaps layers and tables) and notebooks set up
- Pham: Code Writing (specifically defining functions of peak demands, choropleth maps, matrix heatmaps)
- Joselyne: Writing portion, defining and explaining results, notebook cleaning and section distribution

All team members participated in all activities but the main activities assigned to each were described above.

Abstract

> Electric vehicles' (EV) charging demand for electricity as well as the cost and fees of such varies over location and time. During peak-demand times fees tend to be higher, which results in a high rate for EV users. In order to determine a plan that is cost-effective, it is helpful to determine the distribution in space and time of the EV charging events based on historical data. This will be accomplished by understanding the peak-demand charging energy in the stations across the state as well as the peak-demand charging time throughout different periods e.g., hourly, daily, weekly.

Introduction

> The Colorado Energy Office (CEO) focuses on reducing greenhouse gas emissions and consumer energy costs by advancing clean energy, energy efficiency and zero emission vehicles to benefit all Coloradans. In an effort to preserve and help Colorado's communities and natural environment, the CEO is working to enable a path to transition to 100% clean electricity generation by the year 2040. A specific method that the CEO is implementing to accomplish such a goal is promoting the use of EV's in the state. In order to help with this goal of transitioning to 100% clean energy, this project will use python code (see here) and data collected over time across EV charging stations to determine the energy demand across EV's charging stations throughout the state of Colorado. The impact of having a notebook like this is that it will give the CEO visual representations to understand the energy usage patterns over various periods of time (hourly, daily, weekly) and by different spaces (counties, addresses stations). The final goal is to produce a notebook that can be used later on with different and/or more recent data too compare new energy usage and energy transaction patterns with previous patterns and visually understand the new data.

Methods

> In this project, we are using the Excel file titled FY20EV.xlsx which contains all of the data for the EV project. This is a numerical file that contains stations addresses, transaction duration, transaction start and end time, charging energy (kWh), amongst other information regarding the EV. This file's size is 7,601,682 bytes.

> In this notebook, the data in the file are being utilized to find the peak demand of energy and transactions in certain categories such as weekdays vs weekends.

Dependencies

```
from google.colab import drive
import matplotlib.pyplot as plt
import numpy as np
import seaborn as sns
import os      # operating system routines
import pandas as pd
import folium
from folium import plugins
from folium.plugins import HeatMap
from folium.plugins import HeatMapWithTime
from branca.element import Figure
import json


pd.__version__ #check version/verify import
```

```
'1.1.5'
```

```
drive.mount('/content/drive/')
```

```
Mounted at /content/drive/
```

Note on accessing Data: The data is hosted in a shared folder, to which everyone on this project has access. To work with the data affiliated with this notebook, right click the shared folder in Google Drive, then select 'Add Shortcut to Drive'. This will create a shortcut to the data folder located in your drive. As long as it isn't moved, the below block should run. If you move it, change the path in the third line below.

```
cwd = os.getcwd()
pathCEO = cwd + '/drive/My Drive/'                        # always the same
pathFournier = 'Colab Notebooks/Math Clinic/2020fa/CEO/' # Path for Fournier
if os.path.exists(pathCEO + pathFournier) :
   pathCEO += pathFournier
pathCEO += 'CEO_data/'                                    # change this one if you move
os.listdir(pathCEO)
```

```
['CEO Database_2019.accdb',
 'FY20EV.xlsx',
 'lm_cnty',
 'CountiesinColorado.geojson',
 'Counties in Colorado.geojson']
```

```
io = pathCEO              # absolute path tested above
io += 'FY20EV.xlsx'       # file whose path starts from the same directory as the note
# df = pd.read_excel(io)  # omit sheet_name argument if only 1 sheet
```

Data Frames & Definitions

In order to efficiently use the data given to us, we must delete useless columns from the data file. We used the geojson library in order to obtain the geographic data structures of Colorado for the heat maps that we will be utilizing later.

> This cell takes about 15 seconds to run (more than average cell)

```
df = pd.read_excel(io) #Creates a data frame to read excel data
remove = ['Address 2','Country','Currency','End SOC','Ended By','Fee','Model Number',
          'Org Name','Plug In Event Id','Plug Type','Port Number','Port Type','Start S
          'Transaction Date (Pacific Time)','User ID', 'Driver Postal Code'] #Choosen
df = df.drop(columns=remove) # remove useless columns
colList = df.columns.to_list()
#Import Geographical Colorado Data
geojson= pathCEO +'Counties in Colorado.geojson'
with open(geojson) as f:
  Colorado_geo = json.load(f)
```


In our data, we had some addresses that did not have an assigned county for what we decided to manually input their corresponding county.

```
# Fill in missing counties
df.loc[df['Address 1'] == 'S Railroad St','County'] = 'Larimer County'          #106-9
df.loc[df['Address 1'] == '781 E Industrial Blvd','County'] = 'Pueblo County'   #94 -
df.loc[df['Address 1'] == '501 S Galena St','County'] = 'Denver County'         #74 -
df.loc[df['Address 1'] == '3425 south Shields','County'] = 'Larimer County'     #61 -
df.loc[df['Address 1'] == '1309 East 3rd Ave','County'] = 'Denver County'       #14 -
```

Data frames to group data by start date and start hour as well as to devide the week by days, and days vs weekends as follows.

- Days of the week:

  > Monday = 0 ~ Sunday = 6

- Weekdays vs. Weekend:

  > Weekdays (0 - 4) = 1 and Weekends (5 - 6) = 0

```
df['Start Date'] = pd.to_datetime(df['Start Date'])
df['Start Hour'] = pd.DatetimeIndex(df['Start Date']).hour
df['MondaytoSunday'] = pd.to_datetime(df['Start Date']).dt.weekday.astype(int) #This l
df['1-WD/0-WK'] = [1 if x <= 4 else 0 for x in df['MondaytoSunday']] #This line separa
df['Charging Length'] =  pd.to_datetime(df['Charging Time (hh:mm:ss)'], format='%H:%M:
df['Month'] = pd.DatetimeIndex(df['Start Date']).month
```

Definitions to create plots of peak demand from temporal perspective. Details of code are given in the form of comments within the code.

```
def Plot_peak_demand_per_day(Weekday,day):
    Weekday = df.loc[df['MondaytoSunday'] == Weekday ]  # Pick day (0-6)
    peak = Weekday['Start Hour'].value_counts()[:24] #make plot of Start Time for each
    hour = np.arange(24)
    plot = peak.loc[hour].plot(kind='bar')
    plt.xlabel("Start Hour")
    plt.ylabel("Number of transactions")
    plt.title(day)
    return plot
def Plot_Peak_Demand_by_transaction(Weekday, title):
    Weekday = df.loc[df['1-WD/0-WK'] == Weekday ]  # Pick  Weekday = 1 or Weekend = 0
    plt.title(title)
    return plot
def Plot_Peak_Demand_by_Energy_Weekday():
```

```
    pvt = pd.pivot_table(data = df, columns="County", index=['Start Hour'], aggfunc =
    pvt['WD'] = df['1-WD/0-WK']
    pvt_weekday = pvt.loc[pvt['WD'] == 1]
    plot_weekday = pvt_weekday.plot( subplots=False,use_index=True, logy = True, leger
    return plot_weekday
def Plot_Peak_Demand_by_Energy_Weekend():
    pvt = pd.pivot_table(data = df, columns="County", index=['Start Hour'], aggfunc =
    pvt['WD'] = df['1-WD/0-WK']
    pvt_weekend = pvt.loc[pvt['WD'] == 0]
    plot_weekend = pvt_weekend.plot( subplots = False, use_index= True, logy = True, l
    return plot_weekend
def per_station(Weekday,Station):
    Station = df.loc[df['Address 1'] == Station]
    Weekday = df.loc[df['1-WD/0-WK'] == Weekday ]  # Pick  Weekday = 1 or Weekend = 0
    peak = Weekday['Start Hour'].value_counts()[:24] #make plot of Start Time for eacl
    hour = np.arange(24)
    plot = peak.loc[hour].plot(kind='bar')
    plt.xlabel("Start Hour")
    plt.ylabel("Number of transactions")
    #plt.title(Station)  # change
    return plot
def peak_time(Weekday): # around 8AM
    Weekday = df.loc[df['1-WD/0-WK'] == Weekday ]  # Pick  Weekday = 1 or Weekend = 0
    t= Weekday['Start Hour'].tolist()
    peak_time = max(t,key =t.count)
    return peak_time
```

Function to obtain max demand of enery or max number of transactions by hours by time interval.

```
def MAX_DEMAND_BY_HOUR_BY_TIME_INTERVAL(Weekday,Timeinterval,Demand): # 3 Options: (1-
  Weekday = df.loc[df['1-WD/0-WK'] == Weekday ] # Pick 1 for Weekday , Pick 0 for Weel

  # Time interval: Pick 0(For the whole day) or Pick 1(0-6); 2(6-12); 3(12-18) ;4(18-2
  if Timeinterval == 0:
    Weekday_time = Weekday
  elif Timeinterval == 1:
    Weekday_time = Weekday[Weekday['Start Hour'] <= 5]
  elif Timeinterval == 2:
    Weekday_time = Weekday[(Weekday['Start Hour'] > 5) & (Weekday['Start Hour'] <= 11)
  elif Timeinterval == 3:
    Weekday_time = Weekday[(Weekday['Start Hour'] > 11) & (Weekday['Start Hour'] <= 17
  elif Timeinterval == 4:
    Weekday_time = Weekday[Weekday['Start Hour'] > 17]
  else:
    return ("Wrong Input for Time Interval")

  # Create Using data to merge
  grouped = Weekday_time[['Address 1', 'County','Longitude','Latitude','1-WD/0-WK']]
```



```
  grouped = grouped.drop_duplicates(subset = ['Address 1'], ignore_index = True) # dro
  Using_data = grouped.sort_values('Address 1',ignore_index=True ) # sort aphabetical

  #Create pivot table for peak energy  on Weekday or on Weekend
  pvt_e = pd.pivot_table(data = Weekday_time, columns="Address 1", index=['Start Hour'
  # Create pivot table for peak number of transactions on Weekday of Weekend
  pvt_trans = Weekday_time.groupby(["Address 1","Start Hour"])["Start Hour"].count().u

  # Pick Demand measured by number of transactions or demand measured by energy
  if Demand == 'Peak Number of Transactions':
    pvt_W = pvt_trans
  elif Demand == "Energy (kWh)":
    pvt_W = pvt_e
  else:
    return ("Wrong Input for Demand")

  #Create dataframe for the peak energy at the peak time
  peak_time = pd.DataFrame(pvt_W.idxmax().values.tolist(),columns = ['Start Hour']) #
  max_demand_at_peak_time = pd.DataFrame(pvt_W.max().values.tolist(), columns = [Demar
  address = pd.DataFrame(pvt_W.columns.to_list(), columns =['Address 1']) # Get Addres
  frames = [address,peak_time,max_demand_at_peak_time]
  Master_data = pd.concat(frames,axis = 1) # get single data frame contain address pea
  MAX_DEMAND = Master_data.merge(Using_data, left_on='Address 1', right_on='Address 1'

  return MAX_DEMAND
```

This cell creates our **pvt table** which shows the sum of the energy by hour in every address for which we have data.

```
pvt = pd.pivot_table(data = df, columns="Address 1", index=['Start Hour'], aggfunc = '
```

The code from the following 13 cells creates heat maps on different points of information to show the total energy used in different time groups. There will be two final heat maps that are interactive. The user can zoom in and out of the map as desired to see the physical distribution of our data points. The data represented on these maps will also adjust according to the changes being done to the window scale. The more the user zooms in, the more segregated and clearly defined the data points (the colors) will be. For a larger and broader visualization of the information in the maps we can simply zoom out as desired.

> The colors in the map are:
>
> - red (higher energy demand)
> - orange
> - yellow

- green
- light blue
- royal blue (least energy demand).

With these heat maps there is also the option to toggle different time groups on the layer control on the top right corner of the map, as well as choosing the type of view of the map.

> The four time options are:
>
> 1. 00:00-06:00
> 2. 06:00-12:00
> 3. 12:00-18:00
> 4. 18:00-00:00
> 5. And a fifth option for all time groups.

> The four map types are:
>
> 1. Open street map
> 2. Stamen Terrain
> 3. Stamen Toner
> 4. cartodb positron

Note on heat maps: Time layers on the maps shouldn't be stacked. One layer of data doesn't depend on the other. Stacking time layer doesn't give any information. Overlaped layers only overlap the colors in the spectrum but aren't adding the data in such layer.

```
 #Define the Colorado map.
 def generateBaseMap(default_location = [39.113014, -105.7821], default_zoom_start=7):
    base_map = folium.Map(location=default_location, control_scale=True, zoom_start=de
    return base_map
base_map = generateBaseMap()
```

The code that creates the layers of the heat map seems repetitive, the reason for such is because writing the code any other way results in a malfunctioning map (values and information displayed in the maps are incorrect). We've tried to create a loop and other actions to reduce the redundancy of it but it all resulted in invalid and incorrect data.

```
#Create base_map1 for weekdays.
base_map1 = folium.Map(location=[39.113014, -105.7821], control_scale=True, zoom_start
#Create the heat map data for weekdays 00:00-06:00 and add it base_map1
max_e_by_hr1 = MAX_DEMAND_BY_HOUR_BY_TIME_INTERVAL(1,1,"Energy (kWh)")
heat_data_by_max_energy1 = max_e_by_hr1[['Latitude','Longitude', 'Energy (kWh)']]
heatmap_data_by_energy1 = heat_data_by_max_energy1.values.tolist()
HeatMap(heatmap_data_by_energy1, name = 'Weekday 00:00-06:00').add_to(base_map1)
```



```
    <folium.plugins.heat_map.HeatMap at 0x7f17ba5265c0>
```

```
#Create the heat map data for weekdays 06:00-12:00 and add it base_map1
max_e_by_hr2 = MAX_DEMAND_BY_HOUR_BY_TIME_INTERVAL(1,2,"Energy (kWh)")
heat_data_by_max_energy2 = max_e_by_hr2[['Latitude','Longitude', 'Energy (kWh)']]
heatmap_data_by_energy2 = heat_data_by_max_energy2.values.tolist()
HeatMap(heatmap_data_by_energy2,  name = 'Weekday 06:00-12:00').add_to(base_map1)
```

```
    <folium.plugins.heat_map.HeatMap at 0x7f17ba51eeb8>
```

```
#Create the heat map data for weekdays 12:00-18:00 and add it base_map1
max_e_by_hr3 = MAX_DEMAND_BY_HOUR_BY_TIME_INTERVAL(1,3,"Energy (kWh)")
heat_data_by_max_energy3 = max_e_by_hr3[['Latitude','Longitude', 'Energy (kWh)']]
heatmap_data_by_energy3 = heat_data_by_max_energy3.values.tolist()
HeatMap(heatmap_data_by_energy3, name = 'Weekday 12:00-18:00').add_to(base_map1)
```

```
    <folium.plugins.heat_map.HeatMap at 0x7f17ba51acc0>
```

```
#Create the heat map data for weekdays 18:00-24:00 and add it base_map1
max_e_by_hr4 = MAX_DEMAND_BY_HOUR_BY_TIME_INTERVAL(1,4,"Energy (kWh)")
heat_data_by_max_energy4 = max_e_by_hr4[['Latitude','Longitude', 'Energy (kWh)']]
heatmap_data_by_energy4 = heat_data_by_max_energy4.values.tolist()
HeatMap(heatmap_data_by_energy4, name = 'Weekday 18:00-24:00').add_to(base_map1)
```

```
    <folium.plugins.heat_map.HeatMap at 0x7f17b7e969e8>
```

```
#Create the heat map data for weekdays (all time groups) and add it base_map2
max_e_by_hrWD = MAX_DEMAND_BY_HOUR_BY_TIME_INTERVAL(1,0,"Energy (kWh)")
heat_data_by_max_energyWD = max_e_by_hrWD[['Latitude','Longitude', 'Energy (kWh)']]
heatmap_data_by_energyWD = heat_data_by_max_energyWD.values.tolist()
HeatMap(heatmap_data_by_energyWD, name = 'Weekday ALL').add_to(base_map1)
```

```
    <folium.plugins.heat_map.HeatMap at 0x7f17ba5269b0>
```

```
#Create base_map2 for weekends
base_map2 = folium.Map(location=[39.113014, -105.7821], control_scale=True, zoom_start
```

```
#Create the heat map data for weekends 00:00-06:00 and add it base_map2
max_e_by_hr5 = MAX_DEMAND_BY_HOUR_BY_TIME_INTERVAL(0,1,"Energy (kWh)")
heat_data_by_max_energy5 = max_e_by_hr5[['Latitude','Longitude', 'Energy (kWh)']]
# Heatmap
heatmap_data_by_energy5 = heat_data_by_max_energy5.values.tolist()
HeatMap(heatmap_data_by_energy5, name = 'Weekend 00:00-06:00').add_to(base_map2)
```

```
    <folium.plugins.heat_map.HeatMap at 0x7f17b7c33b70>
```

```
#Create the heat map data for weekends 06:00-12:00 and add it base_map2
max_e_by_hr6 = MAX_DEMAND_BY_HOUR_BY_TIME_INTERVAL(0,2,"Energy (kWh)")
heat_data_by_max_energy6 = max_e_by_hr6[['Latitude','Longitude', 'Energy (kWh)']]
heatmap_data_by_energy6 = heat_data_by_max_energy6.values.tolist()
HeatMap(heatmap_data_by_energy6, name = 'Weekend 06:00-012:00').add_to(base_map2)
```

```
<folium.plugins.heat_map.HeatMap at 0x7f17b7c33198>
```

```
#Create the heat map data for weekends 12:00-18:00 and add it base_map2
max_e_by_hr7 = MAX_DEMAND_BY_HOUR_BY_TIME_INTERVAL(0,3,"Energy (kWh)")
heat_data_by_max_energy7 = max_e_by_hr7[['Latitude','Longitude', 'Energy (kWh)']]
heatmap_data_by_energy7 = heat_data_by_max_energy7.values.tolist()
HeatMap(heatmap_data_by_energy7, name = 'Weekend 12:00-18:00').add_to(base_map2)
```

```
<folium.plugins.heat_map.HeatMap at 0x7f17b7e96c18>
```

```
#Create the heat map data for weekends 18:00-24:00 and add it base_map2
max_e_by_hr8 = MAX_DEMAND_BY_HOUR_BY_TIME_INTERVAL(0,4,"Energy (kWh)")
heat_data_by_max_energy8 = max_e_by_hr8[['Latitude','Longitude', 'Energy (kWh)']]
heatmap_data_by_energy8 = heat_data_by_max_energy8.values.tolist()
HeatMap(heatmap_data_by_energy8, name = 'Weekend 18:00-24:00').add_to(base_map2)
```

```
<folium.plugins.heat_map.HeatMap at 0x7f17b728fc88>
```

```
#Create the heat map data for weekends (all time groups) and add it base_map2
max_e_by_hrWE = MAX_DEMAND_BY_HOUR_BY_TIME_INTERVAL(0,0,"Energy (kWh)")
heat_data_by_max_energyWE = max_e_by_hrWE[['Latitude','Longitude', 'Energy (kWh)']]
heatmap_data_by_energyWE = heat_data_by_max_energyWE.values.tolist()
HeatMap(heatmap_data_by_energyWE, name = 'Weekend ALL').add_to(base_map2)
```

```
<folium.plugins.heat_map.HeatMap at 0x7f17b728f748>
```

```
#Add tile layers for easier viewing
folium.TileLayer('openstreetmap').add_to(base_map1)
folium.TileLayer('Stamen Terrain').add_to(base_map1)
folium.TileLayer('Stamen Toner').add_to(base_map1)
folium.TileLayer('cartodbpositron').add_to(base_map1)
#Add layer control to toggle layers and different time groups
folium.LayerControl().add_to(base_map1)
```

```
<folium.map.LayerControl at 0x7f17b728fb00>
```

```
#Add tile layers for easier viewing
folium.TileLayer('openstreetmap').add_to(base_map2)
folium.TileLayer('Stamen Terrain').add_to(base_map2)
folium.TileLayer('Stamen Toner').add_to(base_map2)
folium.TileLayer('cartodbpositron').add_to(base_map2)
#Add layer control to toggle layers and different time groups
folium.LayerControl().add_to(base_map2)
```



```
#Show Weekend map
```

```
<folium.map.LayerControl at 0x7f17b62e46a0>
```

The following three cells have the neccesary code to create our choropleth maps (See here) on different points of information.

Just as before, our maps are intereactive, the user can zoom in and out as desired but the data showed will only pertain the state of Colorado.

We have decided to use choropleth maps in addition to heat maps because we are dealing with counties and addresses which and we aggregated the data into these well-known geographic units in which the data takes place.

Define Colorado data for choropleth maps

```
# Choropleth
# Define data for choropleth on Weekday
def Colorado_data_Weekday(Weekday,Demand): # Pick 1 for Weekday 0 for Weekkend , Pick
  max_e_by_county_weekday= MAX_DEMAND_BY_HOUR_BY_TIME_INTERVAL(Weekday,0,Demand)

#Some string mofidication for county to match with GEOJSON propeties.
  county=[]
  for i in max_e_by_county_weekday['County']:
    county.append(i[:-7])
  max_e_by_county_weekday['county']= county
  max_e_by_county_weekday['county']= max_e_by_county_weekday['county'].str.upper() # I
  Colorado_data = max_e_by_county_weekday[['county','Start Hour', Demand]]
  return Colorado_data
```

Define Choropleth Maps

```
# Choropleth code
# Build Choropleth for peak energy on weekday and weekend
def Choropleth_Peak_Energy_Weekday(Weekday):
  base_map = generateBaseMap()
  folium.Choropleth(
    geo_data=Colorado_geo, # Geographical data
    name='choropleth',
    data=Colorado_data_Weekday(Weekday,'Energy (kWh)'),
    columns=['county', 'Energy (kWh)'],
    key_on = 'feature.properties.county',  #   Problem is right here, key_on = ????
    fill_color='YlOrRd',
    nan_fill_color = 'White',
    fill_opacity=0.8,
    line_opacity=0.2,
    bins = [0,300,600,900,1200,1500],
```

```
      legend_name='Total Energy(kWh) at Peak Time'
  ).add_to(base_map)
   return base_map

# Build Choropleth for peak number of transaction on weekday and weekend
def Choropleth_Peak_Transaction_Weekday(Weekday):
  base_map = generateBaseMap()
  folium.Choropleth(
    geo_data=Colorado_geo, # Geographical data
    name='choropleth',
    data=Colorado_data_Weekday(Weekday,'Peak Number of Transactions'),
    columns=['county','Peak Number of Transactions'],
    key_on = 'feature.properties.county',  #   Problem is right here, key_on = ????
    fill_color='YlOrRd',
    nan_fill_color = 'White',
    fill_opacity=0.8,
    line_opacity=0.2,
    bins = [0,20,40,60,80,100,120],
    legend_name='Number of Transactions'
  ).add_to(base_map)
  return base_map
```

Define Choropleth Peak Time

```
# Build Choropleth for peak time on weekday and weekend
def Choropleth_Peak_Time_Weekday(Weekday,Demand):
  base_map = generateBaseMap()
  folium.Choropleth(
    geo_data=Colorado_geo, # Geographical data
    name='choropleth',
    data=Colorado_data_Weekday(Weekday,Demand),
    columns=['county', 'Start Hour'],
    key_on = 'feature.properties.county',  #   Problem is right here, key_on = ????
    fill_color='YlGnBu',
    nan_fill_color = 'White',
    fill_opacity=0.8,
    line_opacity=0.2,
    legend_name='Peak Time'
  ).add_to(base_map)
  return base_map
```

Code to define the Matrix Heatmap Function See here.

Matrix heatmaps visualise our data through maximum energy variations in colouring. When applied to a tabular format, our heatmaps are useful for cross-examining multivariate data, through placing

Hours variable in the rows and County, Address, or Weekdays in the columns and colouring the cells

```
#Define Matrix Heatmap function:
def Matrix_Heatmap(Columns):            # Pick "County" or "MondaytoSunday" or "Addres
  pvt_matrix = pd.pivot_table(data = df, columns=Columns, index=['Start Hour'], aggfur
  Z = pvt_matrix.to_numpy().T
  Z = np.diag(1/(np.max(Z,axis = 1)))@Z # normalize rows below value 1
  yl = np.array([x.replace(' County', '') if isinstance(x, str) else
                 ['Mon','Tues','Wednes','Thurs','Fri','Satur','Sun'][x] + 'day'
                 for x in pvt_matrix.columns])
  yt = [x + .5 for x in range(Z.shape[0])]
  figheight = max(Z.shape[0]//3, 4)
  fig, ax = plt.subplots(2, 1, figsize=(10,figheight))
  ax[0].pcolor(Z)
  ax[0].set_title('Original row order')
  ax[0].set_xticks(range(0,24,2))
  ax[0].set_yticks(yt)
  ax[0].set_yticklabels(yl, fontsize=8)
  index_array = np.argsort(             # get the row indexes of the sort ...
   np.argmax(Z, axis=1))                # ... of the column indexes of the column-max
  ax[1].pcolor(Z[index_array,:])        # order the same Z rows so that ridge is revea
  ax[1].set_title('Ridge-sorted, normalized rows for ' + Columns)
  ax[1].set_xlabel('Start Hour')
  ax[1].set_xticks(range(0,24,2))
  ax[1].set_ylabel(Columns)
  ax[1].set_yticks(yt)
  ax[1].set_yticklabels(yl[index_array], fontsize=8)
  fig.tight_layout()
  plot = plt.show()
  return plot, pvt_matrix
```

Results & Discussion

> The following are the graphs, tables and maps obtained from the code above. We will describe each of them as well as what they tells us in terms of the EV project.

Graphs

Boxplots

Following boxplots represent each county's energy usage by start hour. Please click on any boxplot for it to be increased on a bigger scale. (Cell takes about 15 sec to run, more than average)

These graphs have been done with the purpose to calculate the uncertainty and variability of our results.

> These graphs were separated into counties in order to see the variability in energy usage as a function of county and hours of a day. The bigger the boxes whiting the plot, the more variability there is in the results and the smalle they the boxes the less variability. We can see that there are some counties like Mesa County, El Paso County, Garfield County, etc. that have a significantly larger uncertainty compared to counties like Baca county, Moffat County, Kiowa County, etc. whose uncertainty is not as high. Analyzing the uncertainty in energy usage by county gives us a more accurate representation and understanding of our following maps.

```
df.groupby('County').boxplot(column='Energy (kWh)',by='Start Hour', figsize=(80,80));
```

Colormap

Following plot shows energy demand in kWh across all individual stations as well. We have decided to use a colormap in order to show critical values from which the other values deviate. As we can see in the following, There are a few addresses that have a much lower energy demand compared to the rest of the data.

```
pvt.plot( subplots=False,use_index=True,colormap='coolwarm',figsize=(15,10),linestyle=
plt.legend(bbox_to_anchor=(1.05, 1), borderaxespad=0., ncol=3);
```

Tables

We've decided to have the following tables included because they serves as a check point for the maps that follow. While the other maps visually show the results, these tables have the **exact** values of our results in case they are of interest to the viewer.

This table shows the sum of the energy per hour in every station.

- 0 - 12:00 AM
- 1 - 1:00 AM
- ...
- 22 - 10:00 PM
- 23 - 11:00 PM

At the bottom of the table we have the sum/total energy per station for the entire day. At the right end of the table we have the total energy across all stations for each hour.

```
pvt
```


This table represents the peak number of transactions throughout the day in all stations. We were able to figure out that the max number of transactions occur on the weekdays for all day.

```
MAX_DEMAND_BY_HOUR_BY_TIME_INTERVAL(1,0,'Peak Number of Transactions')
```

Heat Maps

The following heat maps measure the max energy by hour by time interval.

> As mentioned before, the following heat maps are interactive which allows the user to zoom in and out.
>
> > The colors in the maps are:
> >
> > - red (higher energy demand)
> > - orange
> > - yellow
> > - green
> > - light blue
> > - royal blue (least energy demand)

The user can also switch the time intervals and the map type by clicking the **Layer symbol in the top right corner**

REMINDER: Layers should NEVER be stacked together. Each individual layer should be viewed by itself. When layers are stacked they don't give any information at all. In fact, it misleads to incorrect information.

Peak Demand Energy on WEEKDAYS

```
base_map1
```

Peak Demand Energy on WEEKENDS

```
base_map2
```

Discussion: When arranging both maps on a scale of 50km:50mi we are able to see some differences in peak demand energy amongst weekdays and weekends.

1. During the Weekdays the overall energy is higher across the state compared to the weekend. We can tell this by noticing that the red 'spots' during the weekedays are more intense that those from the weekend.

a. Select the "WEEKDAY ALL" and "WEEKEND ALL" options on the maps and deselect all other toggles

2. If we only compare the two maps from 00:00-06:00, we realize that at this time the energy demand is higher on the weekdays in places like Fort Collins and Colorado Springs while it is higher on the weekends in Summit County.

a. Select the "WEEKDAY 00:00-06:00" and "WEEKEND 00:00-06:00" options on the maps and deselect all other toggles

3. From 06:00-12:00, the energy increases significantly more on the weekdays than on the weekends across the state. We also see the appearance of energy being used in Durango County while in the weekends the energy used is minimal. Overall, there is more demand across the state on the weekdays during this time slot.

a. Select the "WEEKDAY 06:00-12:00" and "WEEKEND 06:00-12:00" options on the maps and deselect all other toggles

4. From 12:00-18:00, the energy decreased on the weekdays compared to the 06:00-12:00 time slot. In the weekends, we also notice a decrease in energy with the exception of Fort Collins as the energy increased in this County.

a. Select the "WEEKDAY 12:00-18:00" and "WEEKEND 12:00-18:00" options on the maps and deselect all other toggles

5. Finally, from 12:00-24:00, the energy demand decreases across all state throughout the weekdays and the weekends as it was expected.

a. Select the "WEEKDAY 12:00-24:00" and "WEEKEND 12:00-24:00" options on the maps and deselect all other toggles

As expected, there was more energy used in the weekdays than on the weekends across the state (with a few exceptions).

Choropleth Maps

The following Choropleth maps have a legend on the top right corner. Note that each legend represents something different on each maps

Peak Demand Energy on WEEKDAYS

Following map represents the total energy at the peak time on weekedays by county. The maximun energy reached at the peak time was between 1,200-1,500 kWh in places like Fort Collins.

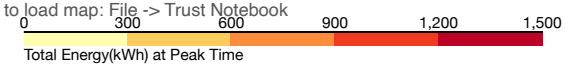
```
Choropleth_Peak_Energy_Weekday(1)
```

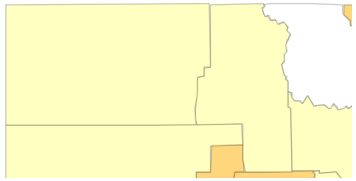
Total Energy at peak time on WEEKENDS

Following map represents the total energy at the peak time on weekends by county. What this map could possibly tell us is that the EV users might be using a non-EV during the weekend. This might be the reason why on the peak time the amount of energy doesn't exceed 600 kWh compared to the 1,500 kWh on the weekdays.

```
Choropleth_Peak_Energy_Weekday(0)
```

Make this Notebook Trusted to load map: File -> Trust Notebook



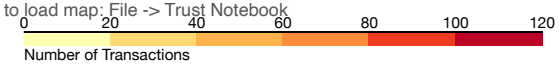


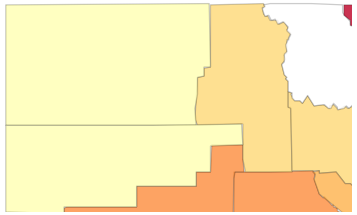
Peak number of transactions on WEEKDAYS

The following maps shows the number of transactions by county. This map as well as the following doesn't measure the duration nor the amount of energy during each transaction. It rather informs us of the peak number of times that a user plugged in their EV.

```
Choropleth_Peak_Transaction_Weekday(1)
```


Make this Notebook Trusted to load map: File -> Trust Notebook



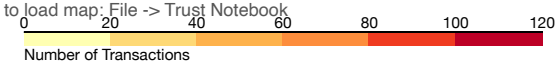


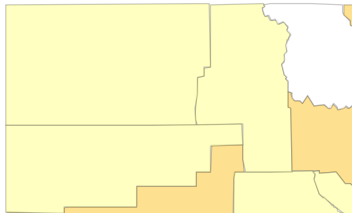
Peak number of transactions on WEEKENDS

Comparing the following map with the previous we can see that there is a fewer demand of transactions over the weekend than over the weekdays.

```
Choropleth_Peak_Transaction_Weekday(0)
```

Make this Notebook Trusted to load map: File -> Trust Notebook





Peak Time of Energy on WEEKDAYS

The following two maps tells the peak time, in other words, it tells us when the most energy is being used. The legend should be read as follows:

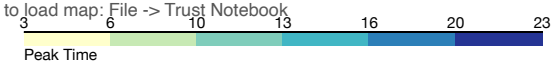
- 0- 12:00 a.m.
- 6- 6:00 a.m.
- 10- 10:00 a.m.
- 13- 1:00 p.m.
- 16- 4:00 p.m.
- 20- 8:00 p.m.
- 23- 11:00 p.m.

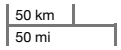
We can tell by looking at the map below that there are places like Colorado springs whose peak time is between 8:00-11:00 pm but overall, the average peak time on the weekdays is around 8:00 AM - 15:00 PM.

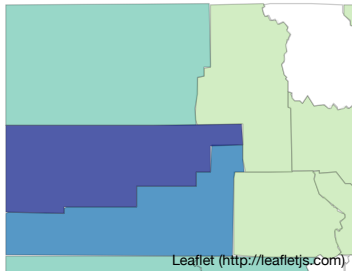
Note on the following maps on. Peak Time: The maps show when the user plugs in their EV but it does NOT describe any type of duration.

```
Choropleth_Peak_Time_Weekday(1,'Energy (kWh)')
```


Make this Notebook Trusted to load map: File -> Trust Notebook



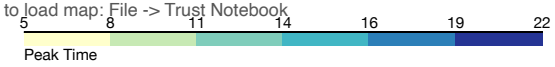


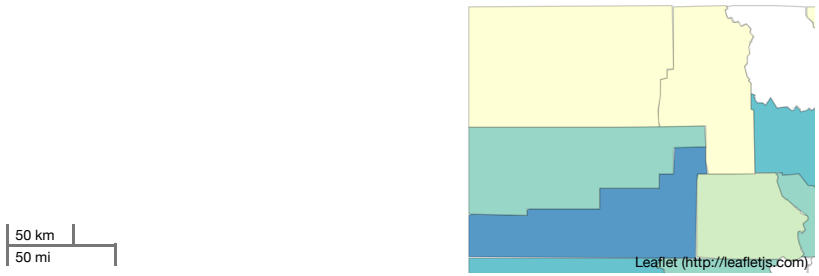


Peak Time of Energy on WEEKENDS

```
Choropleth_Peak_Time_Weekday(0,'Energy (kWh)')
```

Make this Notebook Trusted to load map: File -> Trust Notebook



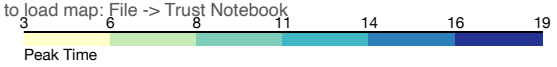


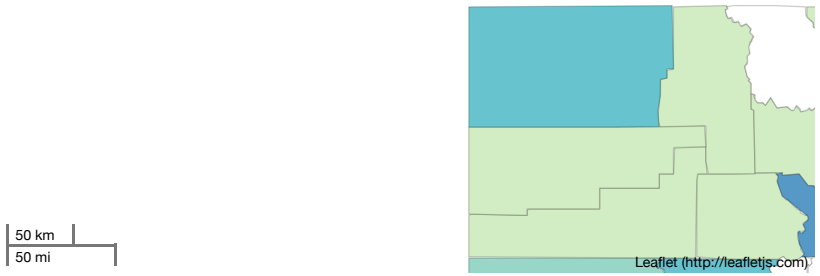
Peak time by number of transaction on WEEKDAYS

Similar as before, thess maps measure the peak time but in terms of transactions. What these maps tells us is the hour in which the most number of transactions took place for any given county. What this means is that if the most transactions at one county was 6 transactions from 3-6 a.m. that is the peak in that specific place, while some other county's peak time by transactions might be 34 at a later time in the day.

```
Choropleth_Peak_Time_Weekday(1,'Peak Number of Transactions')
```


Make this Notebook Trusted to load map: File -> Trust Notebook

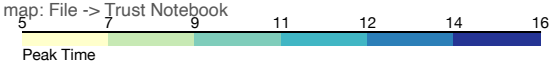


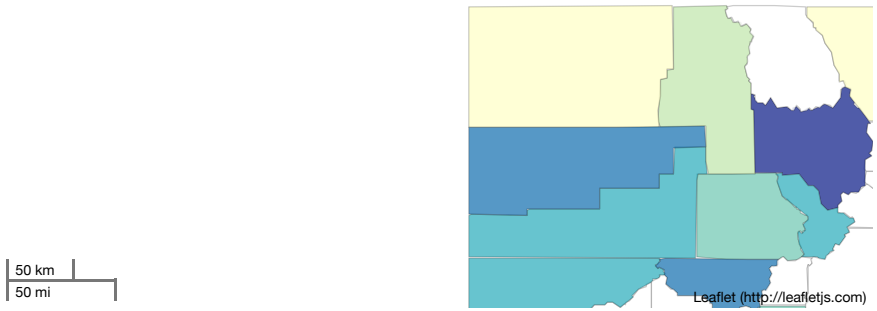


Peak time by number of transaction on WEEKENDS

```
Choropleth_Peak_Time_Weekday(0,'Peak Number of Transactions') # Peak time by number of
```

Make this Notebook Trusted to load map: File -> Trust Notebook





Matrix HeatMap

The color distribution for the following matrix heatmaps is:

- yellow - maximum energy at peak hour
- green
- light blue
- royal blue (least energy demand)

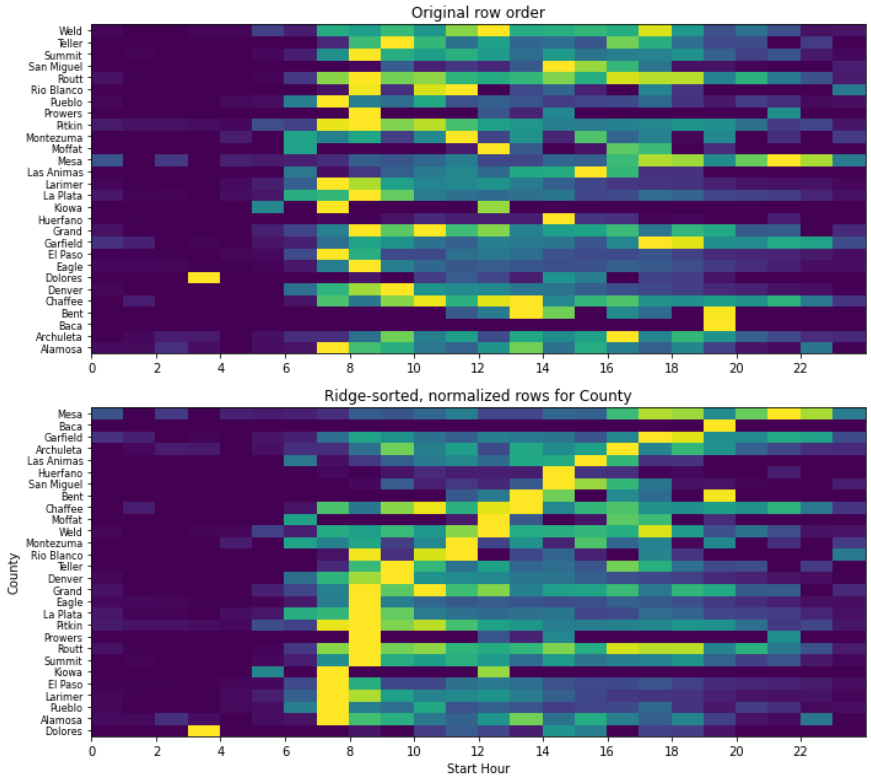
Peak Demand Energy by County

The following two matrix heatmaps show the peak energy at the peak time for each county in our data.

The counties in the first map are in the original order given to us in the data. They are later normalized in the second map to show us the peak demand time from 0 (midnight) to 23 (11:00 p.m.).

Some of the most significant points to discuss is that the time in which most counties show their peak energy is from 7:00-8:00 a.m. and from 8:00-9:00 a.m. We can see that the Dolores County is an outlier where it's peak time is from 3:00-4:00 a.m. but this might be because there are only a few EV users (possibly only one) and they charge their vehicle at this time.

```
pt = Matrix_Heatmap('County') # 28 County
```



Peak Demand Energy by Address

Similarly as before, we see two matrix maps arranged by address in which the first map is ordered as given in the data and the second one ordered by time.

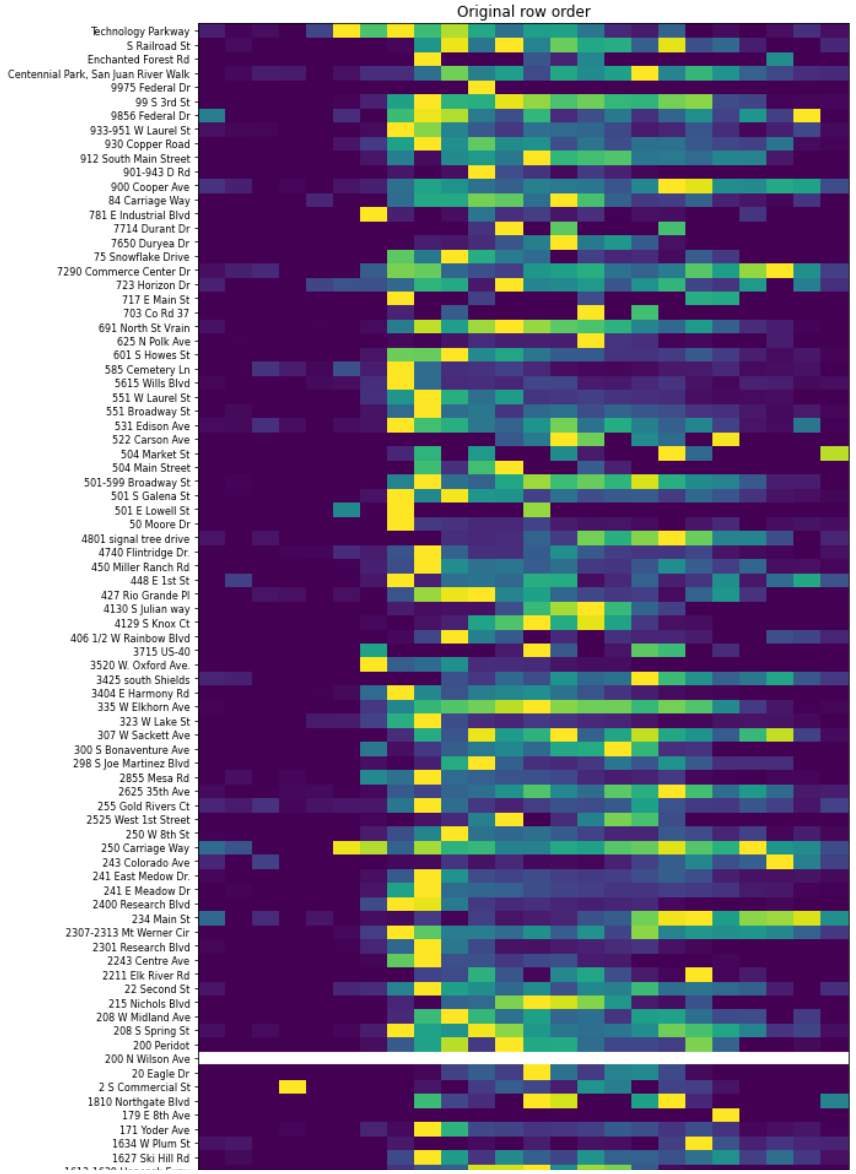
There is one outlier (200 N Wilson Ave), this could be because this station has never been used.

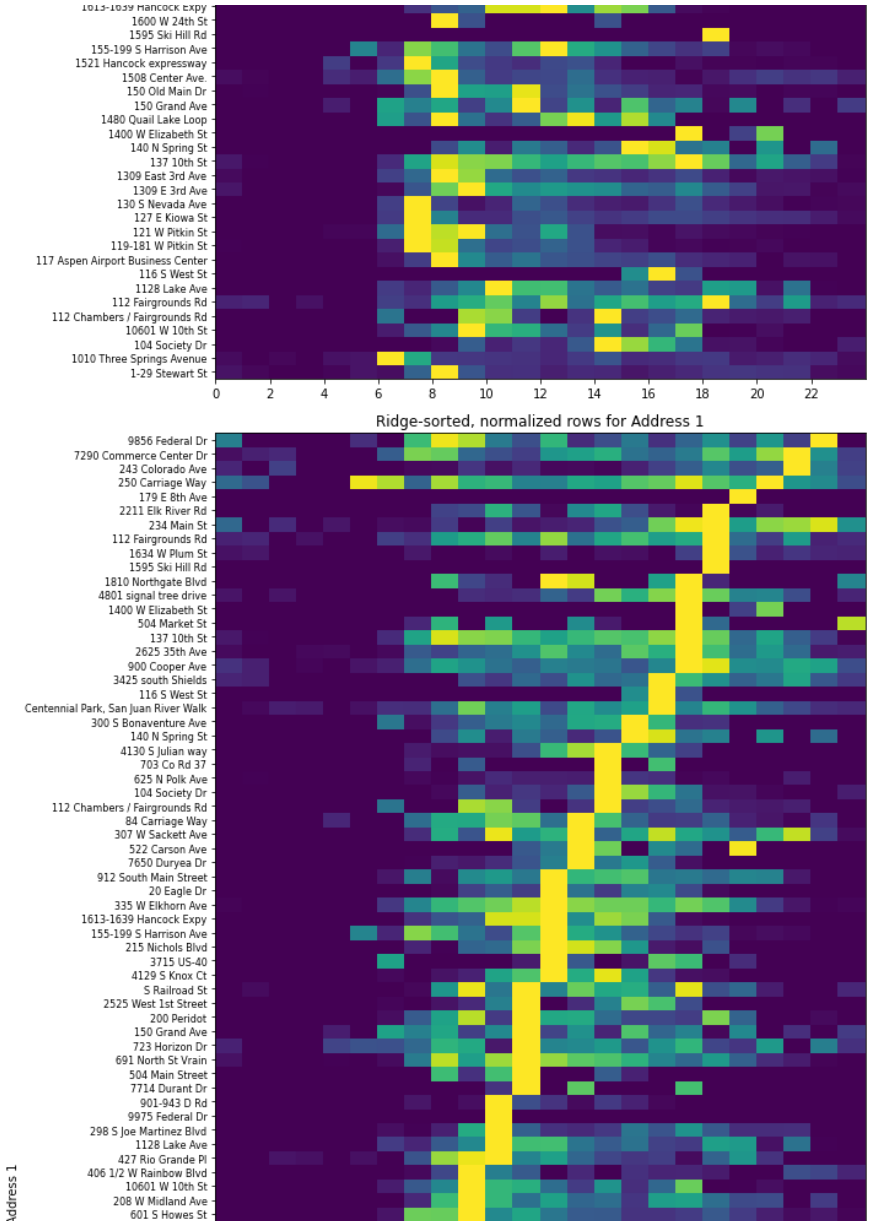
We see similar results in the maps by address than those by county. We also see that the peak energy in the majority of addresses is between 7:00-8:00 a.m. and 8:00-9:00 a.m.

```
pt = Matrix_Heatmap('Address 1') # 108 Address 1
```



```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: RuntimeWarning: 
  """
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: RuntimeWarning: 
  """
```

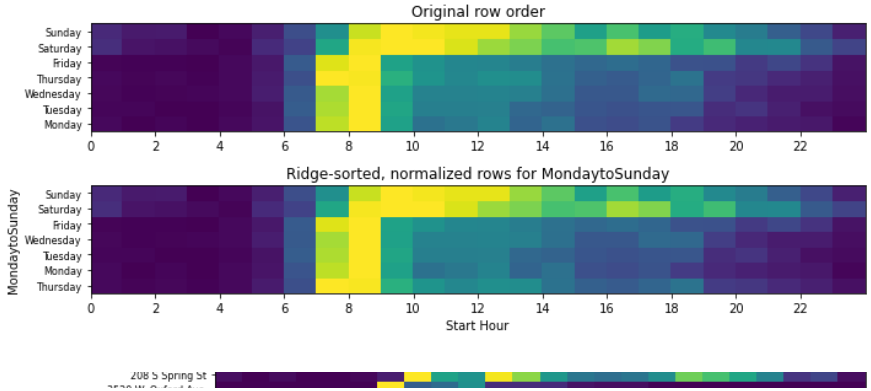




Peak demand energy by Day

Finally, the following two matrix maps are order by Monday through Sunday. As expected, the peak energy across all stations is from 7:00-9:00 a.m. But we notice that on the weekends (Sat-Sun) the peak energy time is from 8:00-11:00/12:00 p.m.


```
Matrix_Heatmap('MondaytoSunday');
```



Discussion: The matrix Heatmaps are highly revealing. We can see the distribution of the peak demand across stations, counties, and all 7 days of the week.

As we can see from matrix heatmaps, peak hours are around 8:00-10:00 AM all cross different categories on the weekdays. However, on the weekends, peak hours spreads out in the middle of the day, from 9:00 AM to 14:00 PM

Conclusions

> Our conclusions are still in progress. With the information that we have been able to gather so far we see a significant difference in the energy demand when comparing the weekdays and the weekends. We are also able to figure which counties have a higher number of peak energy demand in the state and which have a lower demand. Some counties have higher demands than others but most follow the same pattern (higher on weekdays than weekends). In the following maps that we are working on, we expect to be able to see the peak demand of **transactions** in all stations and see when these peak demands are taking place (weekday or weekend, which specific day of the week, etc).

Appendix

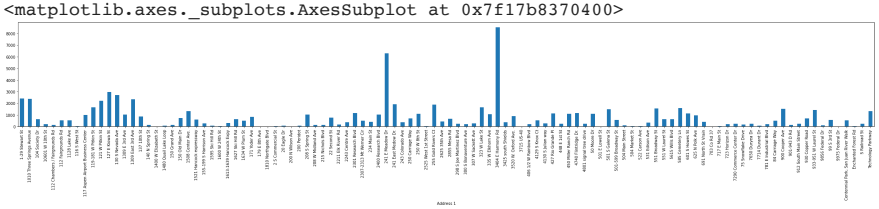
The following code section is a place in which we wrote code that helped us understand our code and make sense out of the maps that we presented before. Many of the tables below just helped us make sure that our maps presented correct and accurate information. The reader is welcome to look over the following but it is not necesserary to understand the past results, reason why it is not explained in detail.

```
pvt = pd.pivot_table(data = df, columns="Address 1", index=['Start Hour'], aggfunc = '
pvt2 = pd.pivot_table(data = df, columns="Start Hour", index=['Address 1'], aggfunc =
pvt3 = pd.pivot_table(data = df, columns="Address 1", index=['Start Hour'], aggfunc =
pvt4 = pd.pivot_table(data = df, columns="Start Hour", index=['Address 1'], aggfunc =
```

```
maxaddress = pvt3.max()
maxhours = pvt4.max()
```

```
maxaddress.plot(figsize = (40,5), kind = 'bar' )
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f17b8370400>
```



```
pvt3 = pd.pivot_table(data = df, columns="1-WD/0-WK", index=['City'], aggfunc = 'max',
pvt3.plot(kind = 'bar', figsize = (20,10))
pvt3
```

```
pvt4 = pd.pivot_table(data = df, columns="1-WD/0-WK", index=['Start Hour'], aggfunc =
pvt4.plot(figsize = (20,10))
pvt4
```



```
# Create a pivot table for number of transactions
pvt_trans = df.groupby(["Address 1","Start Hour"])["Start Hour"].count().unstack('Addr
pvt_trans
```

```
pvt4 = pd.pivot_table(data = df, columns='County', index=['Start Hour'], aggfunc = 'su
pvt4
```