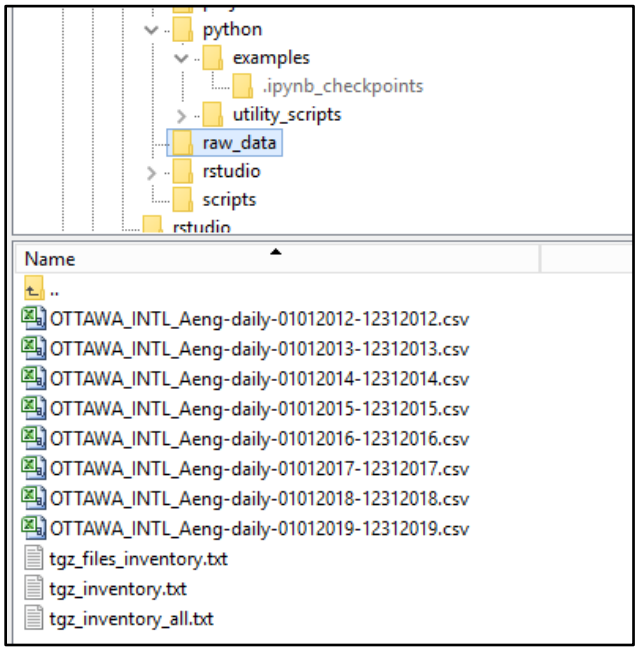
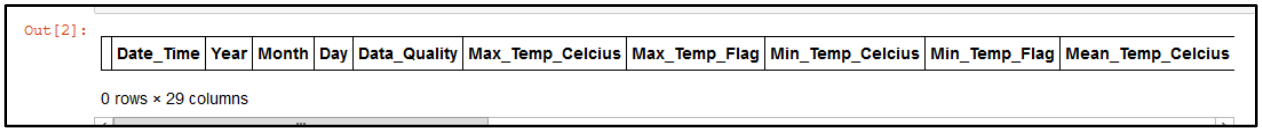
Dynamically Import Flat Files Using Your OS and Python – Part2

The unpackaged tar.gz files contained a number of .csv files that we want to import into our Python environment as a data frame.



Create a template data frame **OTTAWA_INTL_Climate** that we can append our data as we import our source files through our loop. We will also use the list **OTTAWA_INTL_Climate_COL_NAMES** created here to assign the column names of our imported data. We can see after our code output the structure of the table with 29 columns. See code below.



```
In [2]: #Create Empty Table to Append Data
        OTTAWA_INTL_Climate_COL_NAMES = [ "Date_Time"
                                         ,"Year"
                                         ,"Month"
                                         ,"Day"
                                         ,"Data_Quality"
                                         ,"Max_Temp_Celcius"
                                         ,"Max_Temp_Flag"
                                         ,"Min_Temp_Celcius"
                                         ,"Min_Temp_Flag"
                                         ,"Mean_Temp_Celcius"
                                         ,"Mean_Temp_Flag"
                                         ,"Heat_Deg_Days_Celcius"
                                         ,"Heat_Deg_Days_Flag"
                                         ,"Cool_Deg_Days_Celcius"
                                         ,"Cool_Deg_Days_Flag"
                                         ,"Total_Rain_mm"
                                         ,"Total_Rain_Flag"
                                         ,"Total_Snow_cm"
                                         ,"Total_Snow_Flag"
                                         ,"Total_Precip_mm"
                                         ,"Total_Precip_Flag"
                                         ,"Snow_on_Grnd_cm"
                                         ,"Snow_on_Grnd_Flag"
                                         ,"Dir_of_Max_Gust_10s_deg"
                                         ,"Dir_of_Max_Gust_Flag"
                                         ,"Spd_of_Max_Gust_km_h"
                                         ,"Spd_of_Max_Gust_Flag"
                                         ,"Source_File"
                                         ,"Source_File_Period"]

        #Create Empty Data Table
        #creates a new dataframe that's empty
        OTTAWA_INTL_Climate = pd.DataFrame(columns = OTTAWA_INTL_Climate_COL_NAMES)
        OTTAWA_INTL_Climate
```

The next step is to loop through each file that has the extension **.csv** in the location **/home/dag_analytics_serviceacct/dag/raw_data**, import the file and create a data frame, then append that data frame to our master data frame **OTTAWA_INTL_Climate** and then remove the imported file from our source directory. Remember in Part1 of this demonstration that we have already archived the files if we ever need to reprocess or restore. We will also add 2 new columns to the data frame that will indicate the file name of the source data, and we will extract the date range of the data based off of the name of the source file.

```
In [12]: count=0
         for filename in os.listdir(directory+'/raw_data/'):  #look for files in this directory
             if filename.endswith(".csv"):                    #loop through all files that are csv
                 count=count+1                                #keeps track of loop iteration
                 print("File",count,"-",directory,'/raw_data/',filename,sep='')
                 OTTAWA_INTL_Climate_IMP = pd.read_csv( directory+"/raw_data/"+filename, #read in csv file
                           sep=",",
                           delimiter=',',        #data is comma separated
                           skiprows=range(0,27), #data starts at row 29 in the file
                           header=None,
                           names = OTTAWA_INTL_Climate_COL_NAMES)                 #indicate the column names created in teh prvious s
                 df_OTTAWA_INTL_Climate_IMP = pd.DataFrame(OTTAWA_INTL_Climate_IMP) #create data frame with imported data

                 #create column Source_File that indicated the filename for the data
                 df_OTTAWA_INTL_Climate_IMP['Source_File'] = filename

                 #create a column that will extract the period of the files base don the file name
                 df_OTTAWA_INTL_Climate_IMP['Source_File_Period'] = filename[23:40]

                 #append data to OTTAWA_INTL_Climate
                 OTTAWA_INTL_Climate = OTTAWA_INTL_Climate.append(df_OTTAWA_INTL_Climate_IMP)
                 os.system('rm ' + directory+"/raw_data/"+filename)    #remove file once it has been processed
                 print("File",count,"-","Done")                #display that process has completed for the file

         File1-/home/dag_analytics_serviceacct/dag/raw_data/OTTAWA_INTL_Aeng-daily-01012016-12312016.csv
         File 1 - Done
         File2-/home/dag_analytics_serviceacct/dag/raw_data/OTTAWA_INTL_Aeng-daily-01012013-12312013.csv
         File 2 - Done
         File3-/home/dag_analytics_serviceacct/dag/raw_data/OTTAWA_INTL_Aeng-daily-01012014-12312014.csv
         File 3 - Done
         File4-/home/dag_analytics_serviceacct/dag/raw_data/OTTAWA_INTL_Aeng-daily-01012012-12312012.csv
         File 4 - Done
         File5-/home/dag_analytics_serviceacct/dag/raw_data/OTTAWA_INTL_Aeng-daily-01012015-12312015.csv
         File 5 - Done
```

Now that we have our source files processed and into a data frame let's check out the default data types that have been assigned to the columns.

```
In [4]: #display the column types of the imported data
        OTTAWA_INTL_Climate.dtypes
```

```
Out[4]: Date_Time                 object
        Year                      object
        Month                     object
        Day                       object
        Data_Quality             float64
        Max_Temp_Celcius         float64
        Max_Temp_Flag             object
        Min_Temp_Celcius         float64
        Min_Temp_Flag             object
        Mean_Temp_Celcius        float64
        Mean_Temp_Flag            object
        Heat_Deg_Days_Celcius    float64
        Heat_Deg_Days_Flag        object
        Cool_Deg_Days_Celcius    float64
        Cool_Deg_Days_Flag        object
        Total_Rain_mm            float64
        Total_Rain_Flag           object
        Total_Snow_cm            float64
        Total_Snow_Flag           object
        Total_Precip_mm          float64
        Total_Precip_Flag         object
        Snow_on_Grnd_cm          float64
        Snow_on_Grnd_Flag         object
        Dir_of_Max_Gust_10s_deg  float64
        Dir_of_Max_Gust_Flag      object
        Spd_of_Max_Gust_km_h      object
        Spd_of_Max_Gust_Flag      object
        Source_File               object
        Source_File_Period        object
        dtype: object
```

In this case we would definitely want to change our Date_Time column to a date-time format so that we can process dates properly on this data frame.

```
In [5]: #based on the information in the previous step we will want top convert some data types based on our data requirements
        OTTAWA_INTL_Climate['Date_Time'] = pd.to_datetime(OTTAWA_INTL_Climate['Date_Time'], errors='coerce')
        OTTAWA_INTL_Climate['Year' ] = OTTAWA_INTL_Climate['Year' ].astype(int)
        OTTAWA_INTL_Climate['Month'] = OTTAWA_INTL_Climate['Month'].astype(int)
        OTTAWA_INTL_Climate['Day'  ] = OTTAWA_INTL_Climate['Day'  ].astype(int)
```

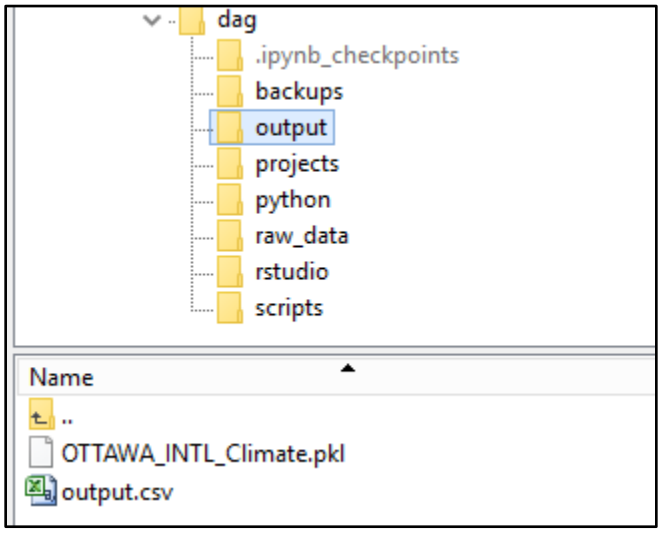
Now that we have changed our columns to our desired types, let's do another check to ensure that they have been converted properly.

```
In [6]: #RE-display the column types after changes
        OTTAWA_INTL_Climate.dtypes

Out[6]: Date_Time                  datetime64[ns]
        Year                               int64
        Month                              int64
        Day                                int64
        Data_Quality                     float64
        Max_Temp_Celcius                 float64
        Max_Temp_Flag                     object
        Min_Temp_Celcius                 float64
        Min_Temp_Flag                     object
        Mean_Temp_Celcius                float64
        Mean_Temp_Flag                    object
        Heat_Deg_Days_Celcius            float64
        Heat_Deg_Days_Flag                object
        Cool_Deg_Days_Celcius            float64
        Cool_Deg_Days_Flag                object
        Total_Rain_mm                    float64
        Total_Rain_Flag                   object
        Total_Snow_cm                    float64
        Total_Snow_Flag                   object
        Total_Precip_mm                  float64
        Total_Precip_Flag                 object
        Snow_on_Grnd_cm                  float64
        Snow_on_Grnd_Flag                 object
        Dir_of_Max_Gust_10s_deg          float64
        Dir_of_Max_Gust_Flag              object
        Spd_of_Max_Gust_km_h              object
        Spd_of_Max_Gust_Flag              object
        Source_File                       object
        Source_File_Period                object
        dtype: object
```

In the final 2 steps we will output the final data frame to a .csv output, and, we will also Pickle the data frame for when we want to restore it for future projects, or to continue to use the data frame for further data processing or to append more source file data in the future.

```
In [7]: #export ActiveAccountDetails to csv to check
        OTTAWA_INTL_Climate.to_csv('/home/dag_analytics_serviceacct/dag/output/output.csv',
                                   index=False, header=True, columns=(OTTAWA_INTL_Climate_COL_NAMES))

In [8]: #pickle files and send to HDFS to store them for another session
        OTTAWA_INTL_Climate.to_pickle('/home/dag_analytics_serviceacct/dag/output/OTTAWA_INTL_Climate.pkl')
```

We can see our pickled file and our csv output in …/dag/output



Looking at the csv output we can see the row data that has been imported and the 2 information columns at the end, indicating the source file from which that row was generated and the time frame of that row based on the file name.

	X	Y	Z	AA	AB	AC
1	Dir_of_Max_Gust_10s_deg	Dir_of_Max_Gust_Flag	Spd_of_Max_Gust_km_h	Spd_of_Max_Gust_Flag	Source_File	Source_File_Period
364	29		56		OTTAWA_INTL_Aeng-daily-01012016-12312016.csv	01012016-12312016
365	9		41		OTTAWA_INTL_Aeng-daily-01012016-12312016.csv	01012016-12312016
366			<31		OTTAWA_INTL_Aeng-daily-01012013-12312013.csv	01012013-12312013
367	22		54		OTTAWA_INTL_Aeng-daily-01012013-12312013.csv	01012013-12312013
368	28		54		OTTAWA_INTL_Aeng-daily-01012013-12312013.csv	01012013-12312013