Data Integration

This notebook highlights the process of integrating data from four different data sources (Excel files) onto a master file. For efficiency, the data manipulation and cleanup is done with Python. After the processing is done, a master Excel file is exported.

Raw data files and table definitions can be found at https://www.kaggle.com/anikannal/solar-power-generation-data (https://www.kaggle.com/anikannal/solar-power-generation-data)

Problem Breakdown

- Customer would like to analize data on power generation from two solar power plants and their respective weather sensor logs.
- The information recides in four separate Excel workbooks (two power generation files and two weather sensor files.)
- Customer requests:
  -- Consolidate data into one workbook
  -- Manipulate data to facilitate comprehensibility
  ---- Change Codes to easy-to-read labels
  ---- Create Code dictionary for future reference
  -- Conduct basic data cleanup

Plant 1 Power Generation Data - Sample (Plant 2 file has the same structure and data formats)

	A	B	C	D	E	F	G
1	DATE_TIME	PLANT_ID	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
2	14-06-2020 14:00	4135001	wCURE6d3bPkepu2	14471.125	1410.95	5178.25	7007547.25
3	14-06-2020 14:00	4135001	YxYtjZvoooNbGkE	14466.85714	1410.528571	5160.857143	7404599.857
4	14-06-2020 14:00	4135001	iCRJl6heRkivqQ3	14436.28571	1407.542857	5267	7405122
5	14-06-2020 14:00	4135001	McdE0feGgRqW7Ca	14418.42857	1405.8	5298.571429	7387221.571
6	14-06-2020 14:00	4135001	adLQvlD726eNBSB	14416.14286	1405.585714	5471.285714	6502967.286
7	4/6/2020 13:00	4135001	adLQvlD726eNBSB	14413.42857	1405.3	4762.857143	6427029.857
8	14-06-2020 14:00	4135001	z9Y9gH1T5YWrNuG	14370.5	1401.125	5121.625	7230351.625
9	14-06-2020 14:00	4135001	sjndEbLyjtCKgGv	14357.42857	1399.842857	5215.857143	7240719.857
10	14-06-2020 14:00	4135001	3PZuoBAID5Wc2HD	14351.125	1399.225	5411	7216130
11	14-06-2020 14:00	4135001	zVJPv84UY57bAof	14329.14286	1397.085714	5221.285714	7342181.286
12	14-06-2020 14:00	4135001	ZoEaEvLYb1n2sOq	14312.71429	1395.485714	5199	7320962
13	14-06-2020 14:00	4135001	1IF53ai7Xc0U56Y	14302.71429	1394.514286	5386.285714	6412267.286
14	22-05-2020 11:30	4135001	adLQvlD726eNBSB	14300.28571	1394.285714	2760.142857	6324333.143
15	14-06-2020 14:00	4135001	VHMLBKoKgIrUVDU	14299.85714	1394.214286	5374.714286	7434940.714
16	14-06-2020 14:00	4135001	zBIq5rxdHJRwDNY	14289.57143	1393.242857	5231.571429	6563053.571
17	14-06-2020 14:00	4135001	rGa61gmuvPhdLxV	14285.71429	1392.842857	5221.285714	7335891.286
18	14-06-2020 14:00	4135001	ZnxXDlPa8U1GXgE	14273.28571	1391.642857	5264	6749607
19	30-05-2020 12:30	4135001	iCRJl6heRkivqQ3	14224.85714	1386.942857	3145.571429	7294398.571
20	14-06-2020 14:00	4135001	7JYdWkrLSPkdwr4	14204	1384.871429	5245.857143	7826097.857
21	14-06-2020 14:00	4135001	pkci93gMrogZuBj	14149.875	1379.625	5201.875	7394362.875
22	30-05-2020 12:00	4135001	wCURE6d3bPkepu2	14140.57143	1378.671429	2591	6898471

Plant 2 Weather Senson Data (Plant 1 file has the same structure and data formats)

	A	B	C	D	E	F
1	DATE_TIME	PLANT_ID	SOURCE_KEY	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
2	6/4/2020 12:30	4136001	iq8k7ZNt4Mwm3w0	32.12407938	54.41021503	1.098766043
3	6/2/2020 12:30	4136001	iq8k7ZNt4Mwm3w0	31.93959243	62.5643057	1.083226548
4	6/1/2020 12:00	4136001	iq8k7ZNt4Mwm3w0	32.09725337	53.4779175	1.079778667
5	6/4/2020 12:45	4136001	iq8k7ZNt4Mwm3w0	32.99476957	61.4893392	1.075527686
6	6/7/2020 12:30	4136001	iq8k7ZNt4Mwm3w0	33.3942382	60.12068173	1.060821653
7	6/6/2020 12:15	4136001	iq8k7ZNt4Mwm3w0	32.69831372	54.08856197	1.057177323
8	6/2/2020 12:45	4136001	iq8k7ZNt4Mwm3w0	33.18994672	66.63595276	1.056203602
9	6/5/2020 13:15	4136001	iq8k7ZNt4Mwm3w0	33.69049059	58.17828634	1.041016201
10	5/30/2020 11:45	4136001	iq8k7ZNt4Mwm3w0	32.96299803	54.69868941	1.035970958
11	6/7/2020 12:15	4136001	iq8k7ZNt4Mwm3w0	32.8434731	56.15691227	1.031367188
12	5/30/2020 12:30	4136001	iq8k7ZNt4Mwm3w0	34.25551093	56.78703055	0.996041441
13	6/8/2020 12:15	4136001	iq8k7ZNt4Mwm3w0	32.2993113	52.53112987	0.99298776
14	6/5/2020 12:30	4136001	iq8k7ZNt4Mwm3w0	33.09526663	59.62057593	0.989781671
15	5/16/2020 11:45	4136001	iq8k7ZNt4Mwm3w0	35.47820933	63.82718397	0.98961924
16	5/15/2020 11:30	4136001	iq8k7ZNt4Mwm3w0	34.65145467	56.08395293	0.987932009
17	5/30/2020 11:15	4136001	iq8k7ZNt4Mwm3w0	31.90578917	51.04826314	0.987074708
18	6/5/2020 12:00	4136001	iq8k7ZNt4Mwm3w0	31.85498457	51.42422273	0.986755757
19	5/16/2020 11:30	4136001	iq8k7ZNt4Mwm3w0	34.83090617	63.09209893	0.981038666
20	5/16/2020 12:00	4136001	iq8k7ZNt4Mwm3w0	36.23130214	66.01627883	0.97979434
21	5/15/2020 12:45	4136001	iq8k7ZNt4Mwm3w0	35.9929465	57.72948603	0.979090417
22	5/15/2020 12:15	4136001	iq8k7ZNt4Mwm3w0	36.37757097	57.58858162	0.977187623

WORKFLOW:

1) Load Dependencies

In [1]:
```
import pandas as pd
import numpy as np
import datetime as dt
```

2) Import source Excel files

In [2]:
```
def import_xlx(file):
    '''import excel file into target dataframe'''
    dir = 'D:/Springboard/Projects/SolarPower/data/raw/'
    df = pd.read_excel(dir + file, dtype={'DATE_TIME':np.str}) #import dates as string a
nd manipulate in python
    return df
```

In [3]:
```
p1 = import_xlx('Plant_1_Generation_Data.xlsx')
p2 = import_xlx('Plant_2_Generation_Data.xlsx')
w1 = import_xlx('Plant_1_Weather_Sensor_Data.xlsx')
w2 = import_xlx('Plant_2_Weather_Sensor_Data.xlsx')
```

3) Analyze structure and content of powerplant files

In [4]:
```
p1.head(3)
```

Out[4]:

	DATE_TIME	PLANT_ID	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
0	15-05-2020 00:00	4135001	1BY6WEcLGh8j5v7	0.0	0.0	0.0	6259559.0
1	15-05-2020 00:00	4135001	1IF53ai7Xc0U56Y	0.0	0.0	0.0	6183645.0
2	15-05-2020 00:00	4135001	3PZuoBAID5Wc2HD	0.0	0.0	0.0	6987759.0

Check to see if there are missing values anywhere in the dataset.

In [5]:
```
p1.isnull().sum()
```

Out[5]:
```
DATE_TIME      0
PLANT_ID       0
SOURCE_KEY     0
DC_POWER       0
AC_POWER       0
DAILY_YIELD    0
TOTAL_YIELD    0
dtype: int64
```

In [6]:
```
#check for data type consistency
p1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 68778 entries, 0 to 68777
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   DATE_TIME    68778 non-null  object 
 1   PLANT_ID     68778 non-null  int64  
 2   SOURCE_KEY   68778 non-null  object 
 3   DC_POWER     68778 non-null  float64
 4   AC_POWER     68778 non-null  float64
 5   DAILY_YIELD  68778 non-null  float64
 6   TOTAL_YIELD  68778 non-null  float64
dtypes: float64(4), int64(1), object(2)
memory usage: 3.7+ MB
```

4) Perform data cleanup

Plant ID and Source Key appear to be categorical values. Let's see how many posible values each hold.

In [7]:
```
print(p1.PLANT_ID.unique())
```

```
[4135001]
```

There is only one ID in this columns. For easier identification lets change the id to '1'.

In [8]:
```
p1['PLANT_ID'] = '1'
coder = {}
coder[4135001] = 'PLANT ID: 1'
```

Check data in 'SOURCE_KEY' column.

In [9]:
```
sources = p1.SOURCE_KEY.unique()
print(sources)
print()
print('There are '+ str(len(p1.SOURCE_KEY.unique())) + ' distinct values.')
```

```
['1BY6WEcLGh8j5v7' '1IF53ai7Xc0U56Y' '3PZuoBAID5Wc2HD' '7JYdWkrLSPkdwr4'
 'McdE0feGgRqW7Ca' 'VHMLBKoKgIrUVDU' 'WRmjgnKYAwPKWDb' 'ZnxXDlPa8U1GXgE'
 'ZoEaEvLYb1n2sOq' 'adLQvlD726eNBSB' 'bvBOhCH3iADSZry' 'iCRJl6heRkivqQ3'
 'ih0vzX44oOqAx2f' 'pkci93gMrogZuBj' 'rGa61gmuvPhdLxV' 'sjndEbLyjtCKgGv'
 'uHbuxQJl8lW7ozc' 'wCURE6d3bPkepu2' 'z9Y9gH1T5YWrNuG' 'zBIq5rxdHJRwDNY'
 'zVJPv84UY57bAof' 'YxYtjZvoooNbGkE']

There are 22 distinct values.
```

There are 22 different values for the source column. These are hard to read. We are going to map this values to something easier to read. Format will be 'P1_SRC_X' for Plant 1 SOURCE #.

We will store the key:value combination for later reference.

In [ ]:
```
```

In [10]:
```
#create dictionary to hold source key mapping
p1_source = {}
idx=1
for source in sources:
    p1_source[source] = 'P1_SRC_' + str(idx)
    idx += 1
```

In [11]:
```
#replace old codes with new values
p1['SOURCE_KEY'] = p1['SOURCE_KEY'].replace(p1_source)
```

The DATE_TIME values are been treated as strings. Perform data cleanup to standardize format and make the column a DATETIME type.

Separate DATES and TIMES into their own columns for easier manipulation, after clean-up bring back together into DATE_TIME column. ** This is needed because some date entries in the file have inconsistent formats. Removing the time components from them allow python logic to correctly identify and format the dates.

In [12]:
```
p1['DATE'] = p1['DATE_TIME'].str.slice(0,10)
p1['TIME'] = p1['DATE_TIME'].str.slice(11,16)
```

In [13]:
```
p1.head(3)
```

Out[13]:

	DATE_TIME	PLANT_ID	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	DATE	TIM
0	15-05-2020 00:00	1	P1_SRC_1	0.0	0.0	0.0	6259559.0	15-05-2020	00:(
1	15-05-2020 00:00	1	P1_SRC_2	0.0	0.0	0.0	6183645.0	15-05-2020	00:(
2	15-05-2020 00:00	1	P1_SRC_3	0.0	0.0	0.0	6987759.0	15-05-2020	00:(

In [14]:
```
#Convert the Date column from string to date type
p1['DATE'] = pd.to_datetime(p1['DATE'])
p1.drop('DATE_TIME', axis = 1, inplace=True)
```

In [15]:
```
# add seconds for time standardization
p1['TIME'] = p1['TIME'] + ':00'
# convert str to time
p1['TIME'] = pd.to_datetime(p1['TIME'], format= '%H:%M:%S').dt.time
p1['DATE_TIME'] = ''

#Combine date + time into DATE_TIME column
for idx in range (len(p1)):
    p1.iloc[idx,8] = pd.datetime.combine(p1.iloc[idx,6],p1.iloc[idx,7])

# Reorder columns
p1.drop(['DATE','TIME'], axis=1, inplace=True)
p1 = p1[['DATE_TIME', 'PLANT_ID', 'SOURCE_KEY', 'DC_POWER', 'AC_POWER', 'DAILY_YIELD',
'TOTAL_YIELD']]
```

```
<ipython-input-15-588f2891d22f>:9: FutureWarning: The pandas.datetime class is deprecat
ed and will be removed from pandas in a future version. Import from datetime module ins
tead.
  p1.iloc[idx,8] = pd.datetime.combine(p1.iloc[idx,6],p1.iloc[idx,7])
```

In [16]:
```
# Vizualize current status of dataset
p1.sample(5)
```

Out[16]:

	DATE_TIME	PLANT_ID	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
45176	2020-06-06 19:00:00	1	P1_SRC_3	0.000000	0.000000	6496.000000	7158982.000
54772	2020-11-06 08:15:00	1	P1_SRC_8	2913.857143	285.814286	258.428571	6724468.429
35861	2020-02-06 07:00:00	1	P1_SRC_21	1669.857143	163.400000	59.142857	7247825.143
14118	2020-05-22 11:00:00	1	P1_SRC_8	7082.857143	692.971429	2251.142857	6574115.143
38471	2020-03-06 13:00:00	1	P1_SRC_15	11547.428570	1127.485714	4329.142857	7253659.143

Power plant 1 dataset is ready.

Repeat steps above to process Power plant 2 file.

5) Analyze structure and content

In [17]:
```
p2.head(3)
```

Out[17]:

	DATE_TIME	PLANT_ID	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
0	2020-05-15 00:00:00	4136001	4UPUqMRk7TRMgml	0.0	0.0	9425.000000	2.429011e+06
1	2020-05-15 00:00:00	4136001	81aHJ1q11NBPMrL	0.0	0.0	0.000000	1.215279e+09
2	2020-05-15 00:00:00	4136001	9kRcWv60rDACzjR	0.0	0.0	3075.333333	2.247720e+09

In [18]:
```
p2.isnull().sum()
```

Out[18]:
```
DATE_TIME      0
PLANT_ID       0
SOURCE_KEY     0
DC_POWER       0
AC_POWER       0
DAILY_YIELD    0
TOTAL_YIELD    0
dtype: int64
```

In [19]:
```
p2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 67698 entries, 0 to 67697
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   DATE_TIME    67698 non-null  object 
 1   PLANT_ID     67698 non-null  int64  
 2   SOURCE_KEY   67698 non-null  object 
 3   DC_POWER     67698 non-null  float64
 4   AC_POWER     67698 non-null  float64
 5   DAILY_YIELD  67698 non-null  float64
 6   TOTAL_YIELD  67698 non-null  float64
dtypes: float64(4), int64(1), object(2)
memory usage: 3.6+ MB
```

6) Perform data cleanup

In [20]:
```
print(p2.PLANT_ID.unique())
```

```
[4136001]
```

In [21]:
```
p2['PLANT_ID'] = '2'
coder[4136001] = 'PLANT ID: 2'
```

In [22]:
```
sources = p2.SOURCE_KEY.unique()
print(sources)
print('There are '+ str(len(p2.SOURCE_KEY.unique())) + ' values.')
```

```
['4UPUqMRk7TRMgml' '81aHJ1q11NBPMrL' '9kRcWv60rDACzjR' 'Et9kgGMDl729KT4'
 'IQ2d7wF4YD8zU1Q' 'LYwnQax7tkwH5Cb' 'LlT2YUhhzqhg5Sw' 'Mx2yZCDsyf6DPfv'
 'NgDl19wMapZy17u' 'PeE6FRyGXUgsRhN' 'Qf4GUc1pJu5T6c6' 'Quc1TzYxW2pYoWX'
 'V94E5Ben1TlhnDV' 'WcxssY2VbP4hApt' 'mqwcsP2rE7J0TFp' 'oZ35aAeoifZaQzV'
 'oZZkBaNadn6DNKz' 'q49J1IKaHRwDQnt' 'rrq4fwE8jgrTyWY' 'vOuJvMaM2sgwLmb'
 'xMbIugepa2P7lBB' 'xoJJ8DcxJEcupym']
There are 22 values.
```

In [23]:
```
#create dictionary to hold source key mapping
p2_source = {}
idx=1
for source in sources:
    p2_source[source] = 'P2_SRC_' + str(idx)
    idx += 1
```

In [24]:
```
p2['SOURCE_KEY'] = p2['SOURCE_KEY'].replace(p2_source)
```

In [25]:
```
p2['DATE'] = p2['DATE_TIME'].str.slice(0,10)
p2['TIME'] = p2['DATE_TIME'].str.slice(11,16)
```

In [26]:
```
p2['DATE'] = pd.to_datetime(p2['DATE'])
p2.drop('DATE_TIME', axis = 1, inplace=True)
```

In [27]:
```
p2['TIME'] = p2['TIME'] + ':00'
p2['TIME'] = pd.to_datetime(p2['TIME'], format= '%H:%M:%S').dt.time
p2['DATE_TIME'] = ''
for idx in range (len(p2)):
    p2.iloc[idx,8] = pd.datetime.combine(p2.iloc[idx,6],p2.iloc[idx,7])

p2.drop(['DATE','TIME'], axis=1, inplace=True)
p2 = p2[['DATE_TIME', 'PLANT_ID', 'SOURCE_KEY', 'DC_POWER', 'AC_POWER', 'DAILY_YIELD',
'TOTAL_YIELD']]
```

```
<ipython-input-27-4fbbd9230d85>:5: FutureWarning: The pandas.datetime class is deprecat
ed and will be removed from pandas in a future version. Import from datetime module ins
tead.
  p2.iloc[idx,8] = pd.datetime.combine(p2.iloc[idx,6],p2.iloc[idx,7])
```

In [28]:
```
p2.sample(5)
```

Out[28]:

	DATE_TIME	PLANT_ID	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
47837	2020-06-08 14:15:00	2	P2_SRC_6	0.000000	0.000000	1330.000000	1.795072e+09
32534	2020-06-01 07:45:00	2	P2_SRC_15	202.053333	198.093333	222.666667	5.937129e+08
24500	2020-05-28 06:00:00	2	P2_SRC_20	15.120000	14.593333	1.266667	2.305880e+06
61875	2020-06-15 05:45:00	2	P2_SRC_8	0.000000	0.000000	0.000000	2.670741e+06
14387	2020-05-22 09:45:00	2	P2_SRC_2	0.000000	0.000000	1605.000000	1.215316e+09

Both Power Plant Generation files are ready.

--

Process Weather Sensor files

6) Analyze structure and content

In [29]:
```
w1.head(3)
```

Out[29]:

	DATE_TIME	PLANT_ID	SOURCE_KEY	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIAT
0	2020-05-15 00:00:00	4135001	HmiyD2TTLFNqkNe	25.184316	22.857507	
1	2020-05-15 00:15:00	4135001	HmiyD2TTLFNqkNe	25.084589	22.761668	
2	2020-05-15 00:30:00	4135001	HmiyD2TTLFNqkNe	24.935753	22.592306	

In [30]:
```
w1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3182 entries, 0 to 3181
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   DATE_TIME            3182 non-null   object 
 1   PLANT_ID             3182 non-null   int64  
 2   SOURCE_KEY           3182 non-null   object 
 3   AMBIENT_TEMPERATURE  3182 non-null   float64
 4   MODULE_TEMPERATURE   3182 non-null   float64
 5   IRRADIATION          3182 non-null   float64
dtypes: float64(3), int64(1), object(2)
memory usage: 149.3+ KB
```

7) Perform data cleanup

Code for readeability

In [31]:
```
w1.PLANT_ID.unique()
```

Out[31]:
```
array([4135001], dtype=int64)
```

In [32]:
```
w1['PLANT_ID'] = '1'
```

In [33]:
```
w1['SOURCE_KEY'].unique()
```

Out[33]:
```
array(['HmiyD2TTLFNqkNe'], dtype=object)
```

In [34]:
```
w1['SOURCE_KEY'] = 'WXS1'
coder['HmiyD2TTLFNqkNe'] = 'WXS1'
```

Process Data/Time data

In [35]:
```
w1['DATE'] = w1['DATE_TIME'].str.slice(0,10)
w1['DATE'] = pd.to_datetime(w1['DATE'])
w1['TIME'] = w1['DATE_TIME'].str.slice(11,16)
w1['TIME'] = w1['TIME'] + ':00'
w1['TIME'] = pd.to_datetime(w1['TIME'], format= '%H:%M:%S').dt.time
w1.drop('DATE_TIME', axis=1, inplace=True)
w1['DATE_TIME'] = ''
```

In [36]:
```
for idx in range (len(w1)):
    w1.iloc[idx,7] = pd.datetime.combine(w1.iloc[idx,5],w1.iloc[idx,6])
w1.drop(['DATE','TIME'], axis=1, inplace=True)
w1= w1[['DATE_TIME', 'PLANT_ID', 'SOURCE_KEY', 'AMBIENT_TEMPERATURE', 'MODULE_TEMPERATUR
E', 'IRRADIATION']]
w1.head(3)
```

```
<ipython-input-36-f041425cb444>:2: FutureWarning: The pandas.datetime class is deprecat
ed and will be removed from pandas in a future version. Import from datetime module ins
tead.
  w1.iloc[idx,7] = pd.datetime.combine(w1.iloc[idx,5],w1.iloc[idx,6])
```

Out[36]:

	DATE_TIME	PLANT_ID	SOURCE_KEY	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
0	2020-05-15 00:00:00	1	WXS1	25.184316	22.857507	0.0
1	2020-05-15 00:15:00	1	WXS1	25.084589	22.761668	0.0
2	2020-05-15 00:30:00	1	WXS1	24.935753	22.592306	0.0

Work second weather file

8) Analyze structure and content

In [37]:
```
w2.head(3)
```

Out[37]:

	DATE_TIME	PLANT_ID	SOURCE_KEY	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATI
0	2020-05-15 00:00:00	4136001	iq8k7ZNt4Mwm3w0	27.004764	25.060789	
1	2020-05-15 00:15:00	4136001	iq8k7ZNt4Mwm3w0	26.880811	24.421869	
2	2020-05-15 00:30:00	4136001	iq8k7ZNt4Mwm3w0	26.682055	24.427290	

In [38]:
```
w2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3259 entries, 0 to 3258
Data columns (total 6 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   DATE_TIME            3259 non-null   object
 1   PLANT_ID             3259 non-null   int64
 2   SOURCE_KEY           3259 non-null   object
 3   AMBIENT_TEMPERATURE  3259 non-null   float64
 4   MODULE_TEMPERATURE   3259 non-null   float64
 5   IRRADIATION          3259 non-null   float64
dtypes: float64(3), int64(1), object(2)
memory usage: 152.9+ KB
```

9) Perform data cleanup

Code for redeability

In [39]:
```
w2.PLANT_ID.unique()
```

Out[39]:
```
array([4136001], dtype=int64)
```

In [40]:
```
w2.SOURCE_KEY.unique()
```

Out[40]:
```
array(['iq8k7ZNt4Mwm3w0'], dtype=object)
```

In [41]:
```
w2['PLANT_ID'] = '2'
w2['SOURCE_KEY'] = 'WXS2'
coder['iq8k7ZNt4Mwm3w0'] = 'WXS2'
```

Process Date/Time data

In [42]:
```
w2['DATE'] = w2['DATE_TIME'].str.slice(0,10)
w2['DATE'] = pd.to_datetime(w2['DATE'])
w2['TIME'] = w2['DATE_TIME'].str.slice(11,16)
w2['TIME'] = w2['TIME'] + ':00'
w2['TIME'] = pd.to_datetime(w2['TIME'], format= '%H:%M:%S').dt.time
w2.drop('DATE_TIME', axis=1, inplace=True)
w2['DATE_TIME'] = ''
```

In [43]:
```
for idx in range (len(w2)):
    w2.iloc[idx,7] = pd.datetime.combine(w2.iloc[idx,5], w2.iloc[idx,6])
w2.drop(['DATE','TIME'], axis=1, inplace=True)
w2= w2[['DATE_TIME', 'PLANT_ID', 'SOURCE_KEY', 'AMBIENT_TEMPERATURE', 'MODULE_TEMPERATUR
E', 'IRRADIATION']]
w2.head(3)
```

```
<ipython-input-43-399aca825677>:2: FutureWarning: The pandas.datetime class is deprecat
ed and will be removed from pandas in a future version. Import from datetime module ins
tead.
  w2.iloc[idx,7] = pd.datetime.combine(w2.iloc[idx,5], w2.iloc[idx,6])
```

Out[43]:

	DATE_TIME	PLANT_ID	SOURCE_KEY	AMBIENT_TEMPERATURE	MODULE_TEMPERATURE	IRRADIATION
0	2020-05-15 00:00:00	2	WXS2	27.004764	25.060789	0.0
1	2020-05-15 00:15:00	2	WXS2	26.880811	24.421869	0.0
2	2020-05-15 00:30:00	2	WXS2	26.682055	24.427290	0.0

Merge datasets into Master File

10) Prepare dataset for merge

In [44]:
```
# join powerplant datasets
df_power = pd.concat([p1,p2], ignore_index=True)
# join wx sensor datasets
df_wx = pd.concat([w1,w2], ignore_index=True)
```

In [45]:
```
# rename commun column name to avoid conflict
df_power.rename(columns={'SOURCE_KEY':'PLANT_SOURCE'}, inplace=True)
```

In [46]:
```
# rename columns for redeability
df_wx.rename(columns={'SOURCE_KEY':'WX_SENSOR', 'AMBIENT_TEMPERATURE': 'AMBIENT_TEMP',
'MODULE_TEMPERATURE':'MODULE_TEMP'}, inplace=True)
```

11) Merge Datasets

In [47]:
```
df = pd.merge(df_power, df_wx, how='left', left_on=['DATE_TIME', 'PLANT_ID'], right_on=[
'DATE_TIME', 'PLANT_ID'])
```

12) Conduct final cleanup on Master File

Limit Temperature and Irradiation values level of precision (rounding.)

In [48]:
```
df['AMBIENT_TEMP'] = round(df['AMBIENT_TEMP'],4)
df['MODULE_TEMP'] = round(df['MODULE_TEMP'],4)
df['IRRADIATION'] = round(df['IRRADIATION'],6)
df['DC_POWER'] = round(df['DC_POWER'],4)
df['AC_POWER'] = round(df['AC_POWER'],4)
df['DAILY_YIELD'] = round(df['DAILY_YIELD'],4)
df['TOTAL_YIELD'] = round(df['TOTAL_YIELD'],4)
df
```

Out[48]:

	DATE_TIME	PLANT_ID	PLANT_SOURCE	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	W
0	2020-05-15 00:00:00	1	P1_SRC_1	0.0	0.0	0.0	6259559.0	
1	2020-05-15 00:00:00	1	P1_SRC_2	0.0	0.0	0.0	6183645.0	
2	2020-05-15 00:00:00	1	P1_SRC_3	0.0	0.0	0.0	6987759.0	
3	2020-05-15 00:00:00	1	P1_SRC_4	0.0	0.0	0.0	7602960.0	
4	2020-05-15 00:00:00	1	P1_SRC_5	0.0	0.0	0.0	7158964.0	
...	...	...	...	...	...	...	...	
136471	2020-06-17 23:45:00	2	P2_SRC_18	0.0	0.0	4157.0	520758.0	
136472	2020-06-17 23:45:00	2	P2_SRC_19	0.0	0.0	3931.0	121131356.0	
136473	2020-06-17 23:45:00	2	P2_SRC_20	0.0	0.0	4322.0	2427691.0	
136474	2020-06-17 23:45:00	2	P2_SRC_21	0.0	0.0	4218.0	106896394.0	
136475	2020-06-17 23:45:00	2	P2_SRC_22	0.0	0.0	4316.0	209335741.0	

136476 rows × 11 columns

Export Master FIle

13) Create Lookup table of New and old codes for export

In [49]:
```
code = {**p1_source, **p2_source}
code = {**code, **coder}
```

In [50]:
```
code_df = pd.DataFrame.from_dict(code, orient='index', columns=['New_Code'])
code_df['Old_Code'] = code_df.index
code_df.reset_index(drop=True, inplace=True)
code_df.head()
```

Out[50]:

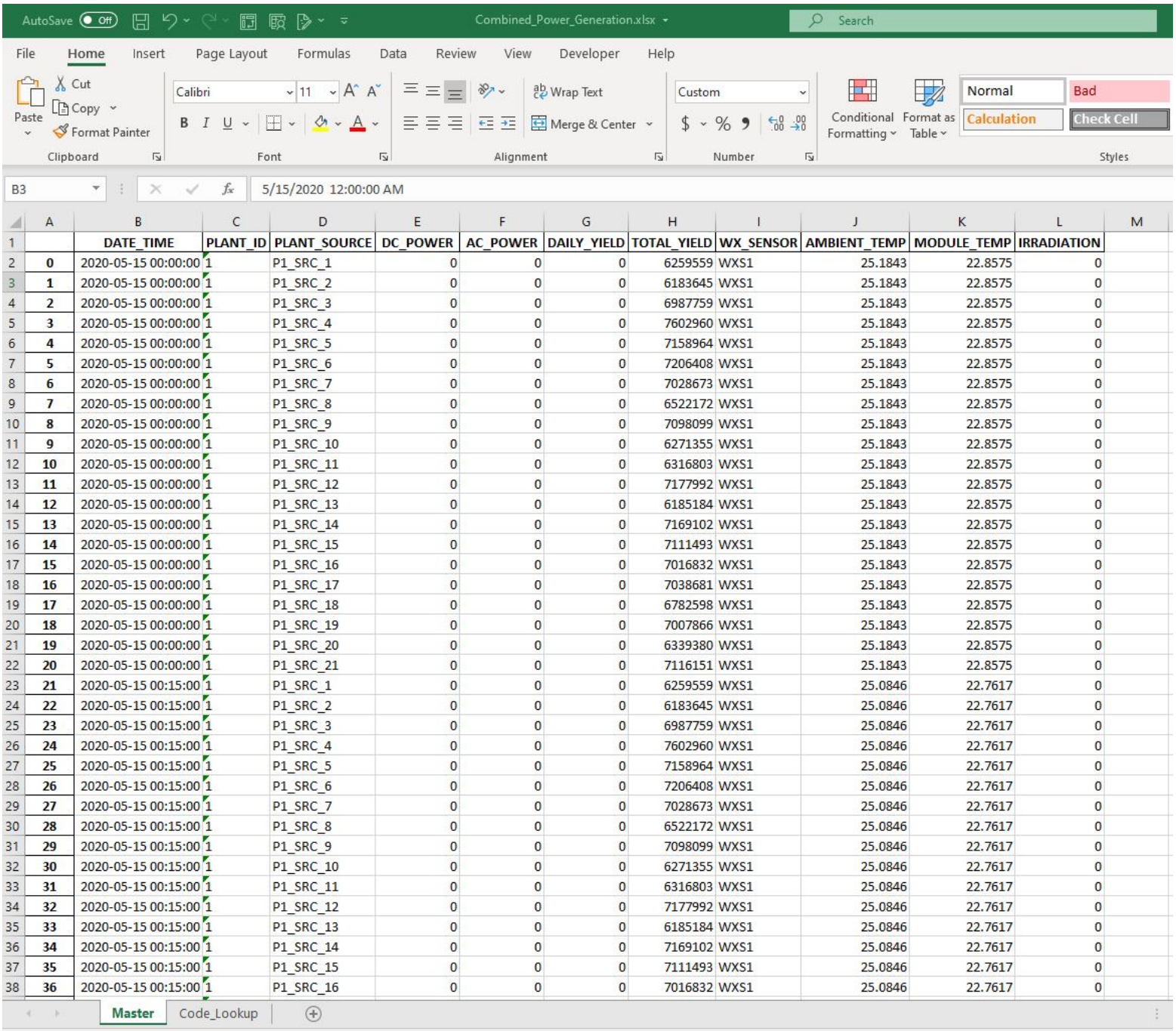
	New_Code	Old_Code
0	P1_SRC_1	1BY6WEcLGh8j5v7
1	P1_SRC_2	1IF53ai7Xc0U56Y
2	P1_SRC_3	3PZuoBAID5Wc2HD
3	P1_SRC_4	7JYdWkrLSPkdwr4
4	P1_SRC_5	McdE0feGgRqW7Ca

14) Export merged master file to Excel Workbook

In [51]:
```
file = 'D:/Springboard/Projects/SolarPower/data/final/Combined_Power_Generation.xlsx'
with pd.ExcelWriter(file, engine='openpyxl') as writer:
    df.to_excel(writer, sheet_name='Master')
    code_df.to_excel(writer, sheet_name='Code_Lookup')
```

Final Master File

Master worksheet


AutoSave Off Combined_Power_Generation.xlsx Search

File Home Insert Page Layout Formulas Data Review View Developer Help

Cut Copy Format Painter Paste Clipboard

Calibri 11 Font

Wrap Text Merge & Center Alignment

Custom Number

Conditional Formatting Format as Table

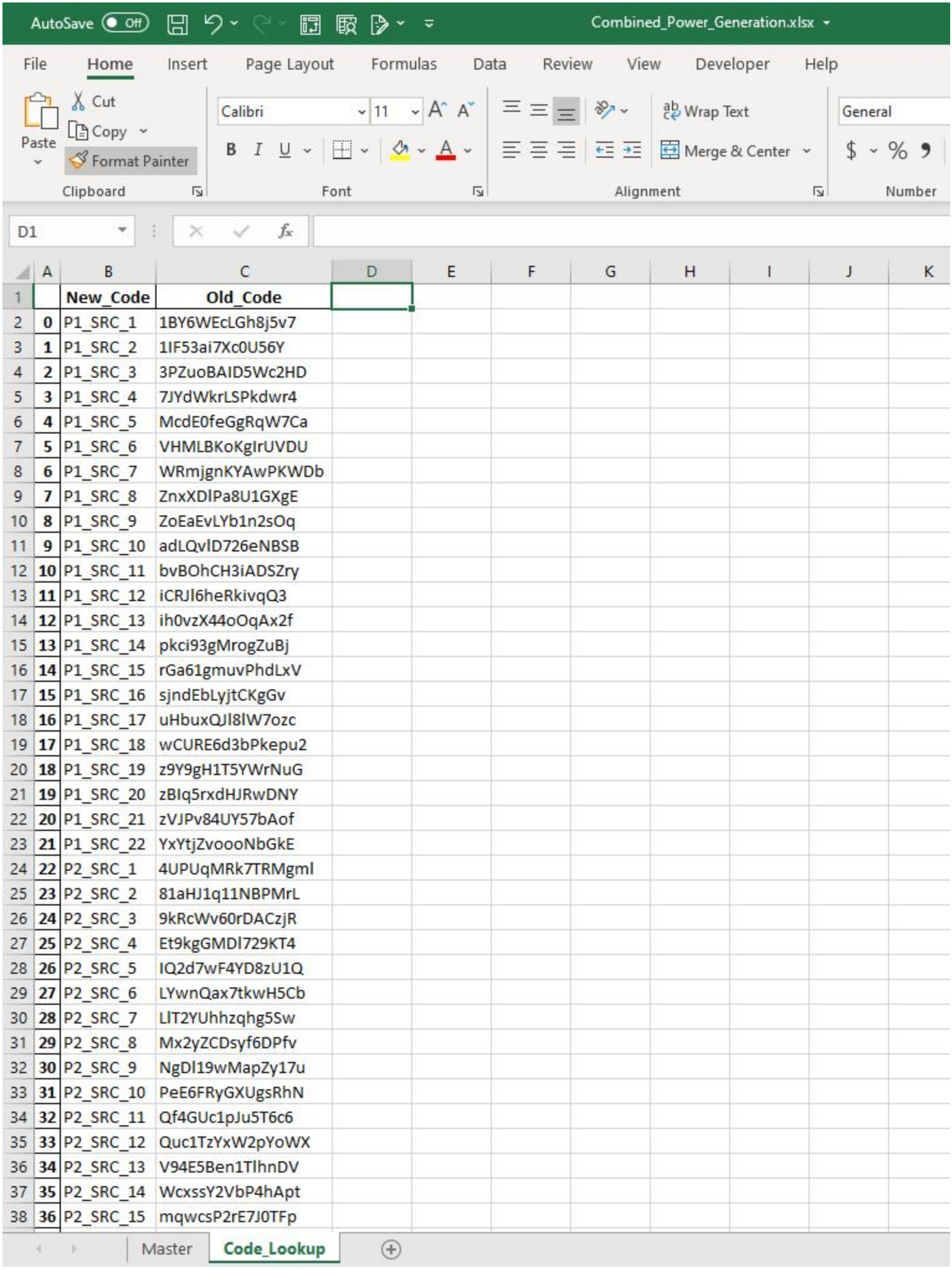
Normal Bad Calculation Check Cell Styles

B3 5/15/2020 12:00:00 AM

	A	B	C	D	E	F	G	H	I	J	K	L	M
1		DATE_TIME	PLANT_ID	PLANT_SOURCE	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD	WX_SENSOR	AMBIENT_TEMP	MODULE_TEMP	IRRADIATION	
2	0	2020-05-15 00:00:00	1	P1_SRC_1	0	0	0	6259559	WXS1	25.1843	22.8575	0	
3	1	2020-05-15 00:00:00	1	P1_SRC_2	0	0	0	6183645	WXS1	25.1843	22.8575	0	
4	2	2020-05-15 00:00:00	1	P1_SRC_3	0	0	0	6987759	WXS1	25.1843	22.8575	0	
5	3	2020-05-15 00:00:00	1	P1_SRC_4	0	0	0	7602960	WXS1	25.1843	22.8575	0	
6	4	2020-05-15 00:00:00	1	P1_SRC_5	0	0	0	7158964	WXS1	25.1843	22.8575	0	
7	5	2020-05-15 00:00:00	1	P1_SRC_6	0	0	0	7206408	WXS1	25.1843	22.8575	0	
8	6	2020-05-15 00:00:00	1	P1_SRC_7	0	0	0	7028673	WXS1	25.1843	22.8575	0	
9	7	2020-05-15 00:00:00	1	P1_SRC_8	0	0	0	6522172	WXS1	25.1843	22.8575	0	
10	8	2020-05-15 00:00:00	1	P1_SRC_9	0	0	0	7098099	WXS1	25.1843	22.8575	0	
11	9	2020-05-15 00:00:00	1	P1_SRC_10	0	0	0	6271355	WXS1	25.1843	22.8575	0	
12	10	2020-05-15 00:00:00	1	P1_SRC_11	0	0	0	6316803	WXS1	25.1843	22.8575	0	
13	11	2020-05-15 00:00:00	1	P1_SRC_12	0	0	0	7177992	WXS1	25.1843	22.8575	0	
14	12	2020-05-15 00:00:00	1	P1_SRC_13	0	0	0	6185184	WXS1	25.1843	22.8575	0	
15	13	2020-05-15 00:00:00	1	P1_SRC_14	0	0	0	7169102	WXS1	25.1843	22.8575	0	
16	14	2020-05-15 00:00:00	1	P1_SRC_15	0	0	0	7111493	WXS1	25.1843	22.8575	0	
17	15	2020-05-15 00:00:00	1	P1_SRC_16	0	0	0	7016832	WXS1	25.1843	22.8575	0	
18	16	2020-05-15 00:00:00	1	P1_SRC_17	0	0	0	7038681	WXS1	25.1843	22.8575	0	
19	17	2020-05-15 00:00:00	1	P1_SRC_18	0	0	0	6782598	WXS1	25.1843	22.8575	0	
20	18	2020-05-15 00:00:00	1	P1_SRC_19	0	0	0	7007866	WXS1	25.1843	22.8575	0	
21	19	2020-05-15 00:00:00	1	P1_SRC_20	0	0	0	6339380	WXS1	25.1843	22.8575	0	
22	20	2020-05-15 00:00:00	1	P1_SRC_21	0	0	0	7116151	WXS1	25.1843	22.8575	0	
23	21	2020-05-15 00:15:00	1	P1_SRC_1	0	0	0	6259559	WXS1	25.0846	22.7617	0	
24	22	2020-05-15 00:15:00	1	P1_SRC_2	0	0	0	6183645	WXS1	25.0846	22.7617	0	
25	23	2020-05-15 00:15:00	1	P1_SRC_3	0	0	0	6987759	WXS1	25.0846	22.7617	0	
26	24	2020-05-15 00:15:00	1	P1_SRC_4	0	0	0	7602960	WXS1	25.0846	22.7617	0	
27	25	2020-05-15 00:15:00	1	P1_SRC_5	0	0	0	7158964	WXS1	25.0846	22.7617	0	
28	26	2020-05-15 00:15:00	1	P1_SRC_6	0	0	0	7206408	WXS1	25.0846	22.7617	0	
29	27	2020-05-15 00:15:00	1	P1_SRC_7	0	0	0	7028673	WXS1	25.0846	22.7617	0	
30	28	2020-05-15 00:15:00	1	P1_SRC_8	0	0	0	6522172	WXS1	25.0846	22.7617	0	
31	29	2020-05-15 00:15:00	1	P1_SRC_9	0	0	0	7098099	WXS1	25.0846	22.7617	0	
32	30	2020-05-15 00:15:00	1	P1_SRC_10	0	0	0	6271355	WXS1	25.0846	22.7617	0	
33	31	2020-05-15 00:15:00	1	P1_SRC_11	0	0	0	6316803	WXS1	25.0846	22.7617	0	
34	32	2020-05-15 00:15:00	1	P1_SRC_12	0	0	0	7177992	WXS1	25.0846	22.7617	0	
35	33	2020-05-15 00:15:00	1	P1_SRC_13	0	0	0	6185184	WXS1	25.0846	22.7617	0	
36	34	2020-05-15 00:15:00	1	P1_SRC_14	0	0	0	7169102	WXS1	25.0846	22.7617	0	
37	35	2020-05-15 00:15:00	1	P1_SRC_15	0	0	0	7111493	WXS1	25.0846	22.7617	0	
38	36	2020-05-15 00:15:00	1	P1_SRC_16	0	0	0	7016832	WXS1	25.0846	22.7617	0	

Master Code_Lookup

Code Lookup Worksheet



	A	New_Code	Old_Code
2	0	P1_SRC_1	1BY6WEcLGh8j5v7
3	1	P1_SRC_2	1IF53ai7Xc0U56Y
4	2	P1_SRC_3	3PZuoBAID5Wc2HD
5	3	P1_SRC_4	7JYdWkrLSPkdwr4
6	4	P1_SRC_5	McdE0feGgRqW7Ca
7	5	P1_SRC_6	VHMLBKoKgIrUVDU
8	6	P1_SRC_7	WRmjgnKYAwPKWDb
9	7	P1_SRC_8	ZnxXDlPa8U1GXgE
10	8	P1_SRC_9	ZoEaEvLYb1n2sOq
11	9	P1_SRC_10	adLQvlD726eNBSB
12	10	P1_SRC_11	bvBOhCH3iADSZry
13	11	P1_SRC_12	iCRJl6heRkivqQ3
14	12	P1_SRC_13	ih0vzX44oOqAx2f
15	13	P1_SRC_14	pkci93gMrogZuBj
16	14	P1_SRC_15	rGa61gmuvPhdLxV
17	15	P1_SRC_16	sjndEbLyjtCKgGv
18	16	P1_SRC_17	uHbuxQJl8lW7ozc
19	17	P1_SRC_18	wCURE6d3bPkepu2
20	18	P1_SRC_19	z9Y9gH1T5YWrNuG
21	19	P1_SRC_20	zBIq5rxdHJRwDNY
22	20	P1_SRC_21	zVJPv84UY57bAof
23	21	P1_SRC_22	YxYtjZvoooNbGkE
24	22	P2_SRC_1	4UPUqMRk7TRMgml
25	23	P2_SRC_2	81aHJ1q11NBPMrL
26	24	P2_SRC_3	9kRcWv60rDACzjR
27	25	P2_SRC_4	Et9kgGMDl729KT4
28	26	P2_SRC_5	IQ2d7wF4YD8zU1Q
29	27	P2_SRC_6	LYwnQax7tkwH5Cb
30	28	P2_SRC_7	LlT2YUhhzqhg5Sw
31	29	P2_SRC_8	Mx2yZCDsyf6DPfv
32	30	P2_SRC_9	NgDl19wMapZy17u
33	31	P2_SRC_10	PeE6FRyGXUgsRhN
34	32	P2_SRC_11	Qf4GUc1pJu5T6c6
35	33	P2_SRC_12	Quc1TzYxW2pYoWX
36	34	P2_SRC_13	V94E5Ben1TlhnDV
37	35	P2_SRC_14	WcxssY2VbP4hApt
38	36	P2_SRC_15	mqwcsP2rE7J0TFp