02. Cleaning the Source Data Set

[28]
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib
from matplotlib import pyplot as plt
from pandas_profiling import ProfileReport
from pathlib import Path
```

[29]
```
df = pd.read_csv("data/01.a.Detail_Incident.csv", parse_dates=
['Open_Time', 'Reopen_Time', 'Resolved_Time','Close_Time', ])
```

[30]
```
df.dtypes
```

```
CI_Name_aff                        object
CI_Type_aff                        object
CI_Subtype_aff                     object
Service_Component_WBS_aff          object
Incident_ID                        object
Status                             object
Impact                              int64
Urgency                             int64
Priority                            int64
Category                           object
KM_number                          object
Alert_Status                       object
Count_Reassignments               float64
Open_Time                  datetime64[ns]
Reopen_Time                datetime64[ns]
Resolved_Time              datetime64[ns]
Close_Time                 datetime64[ns]
Handle_Time_Hours                 float64
Closure_Code                       object
Count_Related_Interactions        float64
Related_Interaction                object
Count_Related_Incidents           float64
Count_Related_Changes             float64
Related_Change                     object
CI_Name_CBy                        object
CI_Type_CBy                        object
CI_Subtype_CBy                     object
ServiceComp_WBS_CBy                object
dtype: object
```

Drop Records where Resolved_Time is Missing

[31]
```
df.iloc[:,13:17].isnull().sum()
```

```
Open_Time            0
Reopen_Time      44322
Resolved_Time     1780
Close_Time           0
dtype: int64
```

[32]
```
df = df.dropna(subset=['Resolved_Time'])
```

[33]
```
df.iloc[:,13:17].isnull().sum()
```

```
Open_Time            0
Reopen_Time      42607
Resolved_Time        0
Close_Time           0
dtype: int64
```

Limit timeframe of all records

greater than 1 october 2013

less than 31 march 2014

[34]
```
df = df[df['Open_Time'] >= pd.to_datetime('10-01-2013')]
```

[35]
```
df.iloc[:,13:17].describe()
```

	Open_Time	Reopen_Time	Resolved_Time	Close_Time
count	43709	2038	43709	43709
unique	43455	2036	43496	43500
top	2014-01-22 15:46:06	2013-11-12 10:36:33	2013-11-22 16:34:33	2014-02-27 15:04:32

freq	Open_Time	Reopen_Time	Resolved_Time	Close_Time
first	2013-10-01 07:33:21	2013-10-01 11:43:47	2013-10-01 08:18:27	2013-10-01 08:18:30
last	2014-03-31 17:24:49	2014-03-31 16:21:15	2014-03-31 22:47:29	2014-03-31 22:47:32

Deal with Status of 'work in progress'

[36]
```
df.Status.value_counts()
```

```
Closed              43700
Work in progress        9
Name: Status, dtype: int64
```

[37]
```
df = df[ df['Status'] == 'Closed' ]
```

[38]
```
df.Status.value_counts()
```

```
Closed    43700
Name: Status, dtype: int64
```

Remove non-incident records

[39]
```
print(df.Category.value_counts())
```

```
incident                  35208
request for information    8482
complaint                     9
request for change            1
Name: Category, dtype: int64
```

[40]
```
df = df[ df['Category'] == 'incident' ]
```

[41]
```
print(df.Category.value_counts())
print(df.Status.value_counts())
```

```
print(df.Alert_Status.value_counts())
```

```
incident    35208
Name: Category, dtype: int64
Closed    35208
Name: Status, dtype: int64
closed    35208
Name: Alert_Status, dtype: int64
```

Deal with Reopen_Time Missing Values

[42]
```
df.Reopen_Time.isnull().sum()
```

```
33782
```

[43]
```
df['ReopenedFlag'] = ~ df.Reopen_Time.isnull()
```

[44]
```
df['ReopenedFlag'] = df['ReopenedFlag'].astype(int)
```

[45]
```
df['ReopenedFlag'].value_counts()
```

```
0    33782
1     1426
Name: ReopenedFlag, dtype: int64
```

Set Missing to Zero for `Count_Related_Changes`, `Count_Related_Incidents`, and `Count_Related_Interactions`

[46]
```
print(df['Count_Related_Changes'].isnull().sum())
print(df['Count_Related_Incidents'].isnull().sum())
print(df['Count_Related_Interactions'].isnull().sum())
```

```
34732
34164
111
```

[47]
```
df['Count_Related_Changes'] =
df['Count_Related_Changes'].fillna(0)
df['Count_Related_Incidents'] =
df['Count_Related_Incidents'].fillna(0)
df['Count_Related_Interactions'] =
df['Count_Related_Interactions'].fillna(0)
```

[48]
```
print(df['Count_Related_Changes'].isnull().sum())
print(df['Count_Related_Incidents'].isnull().sum())
print(df['Count_Related_Interactions'].isnull().sum())
```

```
0
0
0
```

Set Missing to "Not Applicable" for `Related_Change`

[49]
```
df['Related_Change'].value_counts().sum()
```

```
476
```

[50]
```
df['Related_Change'] = df['Related_Change'].fillna("Not
Applicable")
```

[51]
```
df['Related_Change'].value_counts()
```

```
Not Applicable    34732
C00003013           110
C00014762            78
#MULTIVALUE          18
C00001012            10
C00012714            10
C00000713             9
C00009165             7
C00009722             7
C00017302             5
C00008750             5
C00014221             5
C00006833             4
C00004344             3
C00015613             3
C00009821             3
```

C00000829	3
C00001807	3
C00001026	3
C00006448	2
C00012545	2
C00011501	2
C00013454	2
C00012116	2
C00002389	2
C00014458	2
C00003404	2
C00002268	2
C00016781	2
C00000527	2
C00007098	2
C00001250	2
C00016192	2
C00001507	2
C00001549	2
C00005866	2
C00004739	2
C00008442	2
C00013072	2
C00008726	2
C00008222	2
C00004294	2
C00007015	2
C00005261	2
C00011591	1
C00001137	1
C00016571	1
C00012062	1
C00013379	1
C00015705	1
C00007202	1
C00010941	1
C00004044	1
C00006401	1
C00006599	1
C00001730	1
C00004090	1
C00000360	1
C00015923	1
C00004994	1
C00007161	1
C00006745	1
C00001831	1
C00009025	1
C00010379	1
C00008467	1
C00007055	1
C00004385	1
C00017230	1
C00001062	1
C00006823	1
C00013606	1
C00006824	1
C00008356	1

C00015758	1
C00002378	1
C00014707	1
C00008486	1
C00005050	1
C00016689	1
C00010182	1
C00000385	1
C00015776	1
C00004490	1
C00015609	1
C00008700	1
C00009448	1
C00009947	1
C00014475	1
C00009567	1
C00011182	1
C00013064	1
C00014075	1
C00014624	1
C00000589	1
C00000600	1
C00007747	1
C00003040	1
C00009563	1
C00005456	1
C00007132	1
C00014360	1
C00010785	1
C00013595	1
C00016295	1
C00014661	1
C00018294	1
C00014375	1
C00014122	1
C00004950	1
C00014622	1
C00018435	1
C00004493	1
C00016153	1
C00011170	1
C00012038	1
C00004854	1
C00008054	1
C00000122	1
C00018267	1
C00015544	1
C00015025	1
C00010344	1
C00018403	1
C00011406	1
C00015140	1
C00011858	1
C00014296	1
C00001455	1
C00002178	1
C00017553	1
C00013740	1

```
C00009966             1
C00001667             1
C00014876             1
C00014981             1
C00007983             1
C00005369             1
C00004384             1
C00017136             1
C00018421             1
C00017031             1
C00017321             1
C00008787             1
C00006302             1
C00004614             1
C00015047             1
C00010749             1
C00010740             1
C00010259             1
C00013104             1
C00013982             1
C00009069             1
C00016233             1
C00011366             1
C00004679             1
C00007092             1
C00000596             1
C00013273             1
C00013125             1
C00005110             1
C00004549             1
C00007263             1
C00001215             1
C00017594             1
C00000633             1
C00005847             1
C00012923             1
C00005815             1
C00013867             1
C00003624             1
C00002337             1
C00018549             1
C00010314             1
C00017161             1
C00005858             1
C00007572             1
C00002375             1
C00007099             1
C00000050             1
C00003468             1
C00002007             1
C00006422             1
C00015040             1
Name: Related_Change, dtype: int64
```

Drop columns

- with constant values,
- longer needed (`Reopen_Time`)

[52]
```
df = df.drop(['Category', 'Status', 'Alert_Status',
'Reopen_Time'], axis='columns')
```

[53]
```
df.columns
```

```
Index(['CI_Name_aff', 'CI_Type_aff', 'CI_Subtype_aff',
       'Service_Component_WBS_aff', 'Incident_ID', 'Impact', 'Urgency',
       'Priority', 'KM_number', 'Count_Reassignments', 'Open_Time',
       'Resolved_Time', 'Close_Time', 'Handle_Time_Hours',
'Closure_Code',
       'Count_Related_Interactions', 'Related_Interaction',
       'Count_Related_Incidents', 'Count_Related_Changes',
'Related_Change',
       'CI_Name_CBy', 'CI_Type_CBy', 'CI_Subtype_CBy',
'ServiceComp_WBS_CBy',
       'ReopenedFlag'],
      dtype='object')
```

END and OUTPUT

[54]
```
with open("data/02.a.Detail_Incident.csv",'w') as f:
    df.to_csv(f, index=False)
```

[55]
```
df.reset_index(drop=True, inplace=True)
profile = ProfileReport(df, title="Profile of BPIC 2014
Detail_Incident Data after Secondary Cleaning", html={'style':
{'full_width': True}})
```

[56]
```
profile.to_file(Path(str("reports/02.b.Detail_Incident_Profile.ht
ml")))
```

[ ]