Cleaning and analysing employee data

Below is a snippet from a table that contains information about employees that work at Company XYZ:

employee_name employee_id date_joined age yrs_of_experience Andy 123456 2015-02-15 45 24 Beth 789456 NaN 36 15 Cindy 654123 2017-05-16 34 14 Dale 963852 2018-01-15 25 4

Company XYZ recently migrated database systems causing some of the date_joined records to be NULL. It has been said that NULL records for date_joined fields indicates the employees joined prior to 2010.

While investigating, you find out there are multiple employees with the same name and duplicate records for some employees.

Write code to find the number of employees that joined each month. Can group all of the null values as Dec 1 2009.

In [1]:
```
import pandas as pd #allows for data manipulation and analysis
import numpy as np #enables numerical computing in python
from datetime import datetime #The datetime module supplies classes for ma
from dateutil.parser import parse #offers a generic date/time string parse
```

In [2]:
```
#import data
#no need to place apostrophes for number
data = {'employee_name':['Andy', 'Beth', 'Cindy', 'Dale'],
        'employee_id':[123456, 789456,654123,963852],
        'date_joined':['2015-02-15',np.nan,'2017-05-16','2018-01-15'],
        'age':[45,36,34,25],
        'yrs_of_experience':[24,15,14,4]
        }
df = pd.DataFrame(data, columns = ['employee_name','employee_id','date_joi
df
```

Out[2]:

	employee_name	employee_id	date_joined	age	yrs_of_experience
0	Andy	123456	2015-02-15	45	24
1	Beth	789456	NaN	36	15
2	Cindy	654123	2017-05-16	34	14
3	Dale	963852	2018-01-15	25	4

In [3]:
```
#replace all the null values as Dec 1 2009
df["date_joined"].fillna("2009-12-01", inplace = True)
```

In [9]:
```
#parse month-year(YYYY-mmm) from date_joined column

df['date_joined_month']=df['date_joined'].str[:7]
df
```

Out[9]:

	employee_name	employee_id	date_joined	age	yrs_of_experience	date_joined_month
0	Andy	123456	2015-02-15	45	24	2015-02
1	Beth	789456	2009-12-01	36	15	2009-12
2	Cindy	654123	2017-05-16	34	14	2017-05
3	Dale	963852	2018-01-15	25	4	2018-01

In [10]:
```
#grouping employees and perform count over each month
employees_per_month = df.groupby('date_joined_month')['date_joined_month']
print(employees_per_month)
```

```
date_joined_month
2009-12    1
2015-02    1
2017-05    1
2018-01    1
Name: date_joined_month, dtype: int64
```

Another approach below (use of pivots)

In [11]:
```
#strip dataframe to contain just date_joined_month and employee id

df=df[['date_joined_month', 'employee_id']]
```

In [15]:
```
#pivot df on date_joined_month by the unique number of employees ids
df_pivot = df.groupby(['date_joined_month']).employee_id.nunique().reset_i


#rename columns for clear pivot presentations
df_pivot.columns = ['month_joined','num_of_employees']
df_pivot
```

Out[15]:

	month_joined	num_of_employees
0	2009-12	1
1	2015-02	1
2	2017-05	1
3	2018-01	1

In [ ]:
```
```