In [229]:

```
import pandas as pd
import numpy as np
```

In [230]:

```
emv = pd.read_csv("D:\Data Engineer books\CORE TOPICS\Data sets\EmployeeDataToClean.csv"
```

With out OPP's Data Cleaning

In [231]:

```
# sample Data Checking
data.head(4)
```

Out[231]:

	EmployeeID	NationalIDNumber	LoginID	Title	PhoneNumber	BirthDate	MaritalStatus	G
0	1	14417807	adventure-works\guy1	Gustavo Achong	9.257522e+09	21-02-1986 00:00	M	
1	2	253022876	adventure-works\kevin0	Catherine Abel	9.235868e+09	12-03-1991 00:00	S	
2	3	509647174	NaN	Kim Abercrombie	9.416422e+09	21-09-1978 00:00	M	
3	4	112457891	NaN	Humberto Acevedo	8.800042e+09	01-11-1978 00:00	S	

In [232]:

```
# Volume Check
emv.shape
```

Out[232]:

```
(50, 14)
```

In [233]:

```
# column volme chcking
emv.shape[1]
```

Out[233]:

```
14
```

In [234]:

```
# row valume check
emv.shape[0]
```

Out[234]:

```
50
```

In [235]:

```
# chcking Data types
emv.dtypes
```

Out[235]:

```
EmployeeID            int64
NationalIDNumber      int64
LoginID              object
Title                object
PhoneNumber         float64
BirthDate            object
MaritalStatus        object
Gender               object
HireDate             object
Dept                 object
Salary              float64
Job Grade            object
CurrentFlag           int64
rowguid              object
dtype: object
```

In [236]:

```
# getting columns list
list(emv.columns)
```

Out[236]:

```
['EmployeeID',
 'NationalIDNumber',
 'LoginID',
 'Title',
 'PhoneNumber',
 'BirthDate',
 'MaritalStatus',
 'Gender',
 'HireDate',
 'Dept',
 'Salary',
 'Job Grade',
 'CurrentFlag',
 'rowguid']
```

In [237]:

```
# getting the null values
emv.isna().sum()
```

Out[237]:

```
EmployeeID           0
NationalIDNumber     0
LoginID             37
Title                0
PhoneNumber          7
BirthDate            0
MaritalStatus        0
Gender               0
HireDate             0
Dept                 0
Salary               6
Job Grade            0
CurrentFlag          0
rowguid              0
dtype: int64
```

In [238]:

```
# checking the null value persentage
(emv.isna().sum()/emv.shape[0])*100
```

Out[238]:

```
EmployeeID           0.0
NationalIDNumber     0.0
LoginID             74.0
Title                0.0
PhoneNumber         14.0
BirthDate            0.0
MaritalStatus        0.0
Gender               0.0
HireDate             0.0
Dept                 0.0
Salary              12.0
Job Grade            0.0
CurrentFlag          0.0
rowguid              0.0
dtype: float64
```

In [239]:

```
# droping the columns
emv = emv.drop(["LoginID"],axis = 1)
```

In [240]:

```
# checking columns list
list(emv.columns)
```

Out[240]:

```
['EmployeeID',
 'NationalIDNumber',
 'Title',
 'PhoneNumber',
 'BirthDate',
 'MaritalStatus',
 'Gender',
 'HireDate',
 'Dept',
 'Salary',
 'Job Grade',
 'CurrentFlag',
 'rowguid']
```

In [241]:

```
# filling Missing values with mean in salary Column
emv["Salary"] = emv["Salary"].fillna(emv["Salary"].median())
```

In [242]:

```
# checking null values
emv.isna().sum()
```

Out[242]:

```
EmployeeID          0
NationalIDNumber    0
Title               0
PhoneNumber         7
BirthDate           0
MaritalStatus       0
```

```
Gender              0
HireDate            0
Dept                0
Salary              0
Job Grade           0
CurrentFlag         0
rowguid             0
dtype: int64
```

In [243]:

```
emv['PhoneNumber'] = emv['PhoneNumber'].fillna(0000000000)
```

In [244]:

```
try:
    emv['PhoneNumber'] = emv['PhoneNumber'].astype('int')
except Exception as e:
    print(e)
```

In [245]:

```
# Masking the phone number
def phno_masked(a):
    a = str(a)
    if a == '0':
        return "XXXXXXXXXX"
    result = a[0:3] + "XXXXX" + a[-2:]
    return result
```

In [246]:

```
emv['PhoneNumber'] = emv['PhoneNumber'].astype('object')
```

In [247]:

```
emv["PhoneNumber"] = emv['PhoneNumber'].apply(phno_masked)
```

In [248]:

```
emv["PhoneNumber"].head(10)
```

Out[248]:

```
0    925XXXXX51
1    923XXXXX60
2    941XXXXX59
3    880XXXXX25
4    XXXXXXXXXX
5    XXXXXXXXXX
6    XXXXXXXXXX
7    XXXXXXXXXX
8    918XXXXX76
9    941XXXXX79
Name: PhoneNumber, dtype: object
```

In [249]:

```
# chaning the gender
try:
    emv["Gender"] = emv["Gender"].apply(lambda x: "Male" if x== "M" else "Female")
except Exception as e:
    print(e)
```

In [250]:

```
emv["Gender"].head(2)
```

Out[250]:
```
0    Male
1    Male
Name: Gender, dtype: object
```

In [251]:
```
# changing the Marital Status
try:
    emv["MaritalStatus"] = emv["MaritalStatus"].apply(lambda x: "Married" if x== "M" els
except Exception as e:
    print(e)
```

In [252]:
```
emv["MaritalStatus"].head(2)
```

Out[252]:
```
0       Married
1    Un-Married
Name: MaritalStatus, dtype: object
```

In [253]:
```
# chaning the date data type
emv["BirthDate"] = pd.to_datetime(emv["BirthDate"],format = "%d-%m-%Y %H:%M")
emv["HireDate"] = pd.to_datetime(emv["HireDate"],format = "%d-%m-%Y %H:%M")
```

In [254]:
```
# Getting Unique values
emv["Job Grade"].unique()
```

Out[254]:
```
array(['Admin', 'Management', 'Operations'], dtype=object)
```

In [255]:
```
# Getting Unique values count
emv["Job Grade"].nunique()
```

Out[255]:
```
3
```

With OPP's Data Cleaning

Writing the class

In [256]:
```
try:
    class DataCleaning:
        def __init__(self,data):
            self.data = data

        def volume_check(self):
            return self.data.shape

        def sample_data(self,n):
            return self.data.head(n)

        def row_count(self):
            print("The Number rows are given data:",self.data.shape[0])
```

```
    def coulmns_count(self):
        print("The Number columns are given data:",self.data.shape[1])

    def columns_list(self):
        return list(self.data.columns)

    def numaric_column_list(self):
        return list(self.data.select_dtypes(include= np.number).columns)

    def categroyical_columns_list(self):
        return list(self.data.select_dtypes(exclude=np.number).columns)

    def data_types(self):
        return self.data.dtypes

    def drop_duplicates(self):
        self.data = self.data.drop_duplicates()

    def drop_columns(self,col_name):
        self.data = self.data.drop(col_name,axis = 1)

    def drop_null(self,col_name):
        self.data = self.data.dropna(subset = [col_name])

    def missing_values(self):
        return self.data.isna().sum()

    def Getting_unique_count(self,col_name):
        return self.data[col_name].nunique()

    def getting_unique(self,col_name):
        return self.data[col_name].unique()

    def change_Gender(self,col_name):
        self.data[col_name] = self.data[col_name].apply(lambda x: "Male" if x== "M"

    def change_Marrital_status(self,col_name):
        self.data[col_name] = self.data[col_name].apply(lambda x: "Married" if x== "

    def change_date_data_type(self,col_name):
        self.data[col_name] = pd.to_datetime(self.data[col_name],format = "%d-%m-%Y

    def Phone_masking(self,col_name):
        self.data[col_name] = self.data[col_name].fillna(0000000000).astype('int64')


    def missing_values_presentage(self):
        return(self.data.isna().sum()/self.data.shape[0])*100

    def impute_missing_value_mean(self,col_name):
        self.data[col_name] = round(self.data[col_name].fillna(self.data[col_name].m

    def impute_missing_values_mode(self,col_name):
        self.data[col_name] = self.data[col_name].fillna(self.data[col_name].mode()[

    def impute_missing_values_median(self,col_name):
        self.data[col_name] = round(self.data[col_name].fillna(self.data[col_name].m

    def impute_missing_values_other(self,col_name):
```

```
            self.data[col_name] = self.data[col_name].fillna("Not Avaialable")

except Exception as e:
    print(e)
```

In [257]:

```
# Loading agian same data set,
try:
    emv1 = pd.read_csv("D:\Data Engineer books\CORE TOPICS\Data sets\EmployeeDataToClean
except Exception as e:
    print(e)
```

In [258]:

```
try:
    df = DataCleaning(emv1)
except Exception as e:
    print(e)
```

In [259]:

```
# getting sample data with OPP's
df.sample_data(5)
```

Out[259]:

	EmployeeID	NationalIDNumber	LoginID	Title	PhoneNumber	BirthDate	MaritalStatus	G
0	1	14417807	adventure-works\guy1	Gustavo Achong	9.257522e+09	21-02-1986 00:00	M	
1	2	253022876	adventure-works\kevin0	Catherine Abel	9.235868e+09	12-03-1991 00:00	S	
2	3	509647174	NaN	Kim Abercrombie	9.416422e+09	21-09-1978 00:00	M	
3	4	112457891	NaN	Humberto Acevedo	8.800042e+09	01-11-1978 00:00	S	
4	5	480168528	NaN	Pilar Ackerman	NaN	07-06-1963 00:00	M	

In [260]:

```
# getting the data types with OPP's
df.data_types()
```

Out[260]:

```
EmployeeID            int64
NationalIDNumber      int64
LoginID              object
Title                object
PhoneNumber         float64
BirthDate            object
MaritalStatus        object
```

```
Gender               object
HireDate             object
Dept                 object
Salary              float64
Job Grade            object
CurrentFlag           int64
rowguid              object
dtype: object
```

In [261]:

```
# getting the volume with OPP's
df.volume_check()
```

Out[261]:

```
(50, 14)
```

In [262]:

```
# getting the columns count with OPP's
df.coulmns_count()
```

```
The Number columns are given data: 14
```

In [263]:

```
# getting the rows count with OPP's
df.row_count()
```

```
The Number rows are given data: 50
```

In [264]:

```
# getting columns list with OPP's
df.columns_list()
```

Out[264]:

```
['EmployeeID',
 'NationalIDNumber',
 'LoginID',
 'Title',
 'PhoneNumber',
 'BirthDate',
 'MaritalStatus',
 'Gender',
 'HireDate',
 'Dept',
 'Salary',
 'Job Grade',
 'CurrentFlag',
 'rowguid']
```

In [265]:

```
# getting the missing values count with OPP's
df.missing_values()
```

Out[265]:

```
EmployeeID           0
NationalIDNumber     0
LoginID             37
Title                0
PhoneNumber          7
BirthDate            0
MaritalStatus        0
Gender               0
HireDate             0
```

```
Dept                0
Salary              6
Job Grade           0
CurrentFlag         0
rowguid             0
dtype: int64
```

In [266]:

```
# getting the missing values Persentage with OPP's
df.missing_values_presentage()
```

Out[266]:

```
EmployeeID          0.0
NationalIDNumber    0.0
LoginID            74.0
Title               0.0
PhoneNumber        14.0
BirthDate           0.0
MaritalStatus       0.0
Gender              0.0
HireDate            0.0
Dept                0.0
Salary             12.0
Job Grade           0.0
CurrentFlag         0.0
rowguid             0.0
dtype: float64
```

In [267]:

```
# drop the columns with OPP's
df.drop_columns("LoginID")
```

In [268]:

```
df.columns_list()
```

Out[268]:

```
['EmployeeID',
 'NationalIDNumber',
 'Title',
 'PhoneNumber',
 'BirthDate',
 'MaritalStatus',
 'Gender',
 'HireDate',
 'Dept',
 'Salary',
 'Job Grade',
 'CurrentFlag',
 'rowguid']
```

In [269]:

```
# filling salary with OPP's
df.impute_missing_values_median("Salary")
```

In [270]:

```
# Changning the Birthdate data type
df.change_date_data_type("BirthDate")
```

In [271]:

```
# Changning the Hiredate data type
df.change_date_data_type("HireDate")
```

This OPP's DataCleaning Class we can apply any data set not for a one data set

Inheritance

1. Take a copy to complete the class and add ur method (not recommended)

2. We can't modify the existing class (Exis,t Clas,s possibly that some team, some team member would have implemented) (not recommended)

3. We can use existing classes,s and u are allowed to use your own code (Recommended) ----> Inheritance

In [275]:

```
def AdvDataCleaning(DataCleaning):
    def missing_values_fill_zero(self,col_name):
        self.data = self.data[col_name].fillna(0).astype('int')
```

In [276]:

```
df1 = AdvDataCleaning(df)
```

In [ ]:

```

```