

# Data Cleaning and Preprocessing, EDA, and Handling Missing Data in Pandas

## Objective:

The main objective of this project is to design and implement a robust data preprocessing system that addresses common challenges such as missing values, outliers, inconsistent formatting, and noise. By performing effective data preprocessing, the project aims to enhance the quality, reliability, and usefulness of the data for machine learning.

**Dataset:** [https://drive.google.com/file/d/1F3lRf32JM8ejnXq-Cbf9y7fa57zSHGz\\_/view?usp=sharing](https://drive.google.com/file/d/1F3lRf32JM8ejnXq-Cbf9y7fa57zSHGz_/view?usp=sharing)

```
In [3]: import warnings
import sys
if not sys.warnoptions:
    warnings.simplefilter("ignore")
```

```
In [5]: import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy.stats import norm
```

## Load Dataset

```
In [7]: data = pd.read_csv("Employee2.csv")
data
```

Out[7]:

	Company	Age	Salary	Place	Country	Gender
0	TCS	20.0	NaN	Chennai	India	0
1	Infosys	30.0	NaN	Mumbai	India	0
2	TCS	35.0	2300.0	Calcutta	India	0
3	Infosys	40.0	3000.0	Delhi	India	0
4	TCS	23.0	4000.0	Mumbai	India	0
...	...	...	...	...	...	...
143	TCS	33.0	9024.0	Calcutta	India	1
144	Infosys	22.0	8787.0	Calcutta	India	1
145	Infosys	44.0	4034.0	Delhi	India	1
146	TCS	33.0	5034.0	Mumbai	India	1
147	Infosys	22.0	8202.0	Cochin	India	0

148 rows × 6 columns

### Data Exploration:

Explore the data, list down the unique values in each feature and find its length. Perform the statistical analysis and renaming of the columns.

In [9]:

```
data3 = data.copy()  
data3
```

Out[9]:

	Company	Age	Salary	Place	Country	Gender
0	TCS	20.0	NaN	Chennai	India	0
1	Infosys	30.0	NaN	Mumbai	India	0
2	TCS	35.0	2300.0	Calcutta	India	0
3	Infosys	40.0	3000.0	Delhi	India	0
4	TCS	23.0	4000.0	Mumbai	India	0
...	...	...	...	...	...	...
143	TCS	33.0	9024.0	Calcutta	India	1
144	Infosys	22.0	8787.0	Calcutta	India	1
145	Infosys	44.0	4034.0	Delhi	India	1
146	TCS	33.0	5034.0	Mumbai	India	1
147	Infosys	22.0	8202.0	Cochin	India	0

148 rows × 6 columns

```
In [11]: print("Displaying first 10 rows")
print("\t")
data3.head(10)
```

Displaying first 10 rows

Out[11]:

	Company	Age	Salary	Place	Country	Gender
0	TCS	20.0	NaN	Chennai	India	0
1	Infosys	30.0	NaN	Mumbai	India	0
2	TCS	35.0	2300.0	Calcutta	India	0
3	Infosys	40.0	3000.0	Delhi	India	0
4	TCS	23.0	4000.0	Mumbai	India	0
5	Infosys	NaN	5000.0	Calcutta	India	0
6	TCS	NaN	6000.0	Chennai	India	1
7	Infosys	23.0	7000.0	Mumbai	India	1
8	TCS	34.0	8000.0	Calcutta	India	1
9	CTS	45.0	9000.0	Delhi	India	0

In [13]:

```
print("Displaying last 15 rows")
print("\t")
data3.tail(15)
```

Displaying last 15 rows

Out[13]:

	Company	Age	Salary	Place	Country	Gender	
133		NaN	22.0	8943.0	Chennai	India	0
134	Tata Consultancy Services	31.0	8345.0	Mumbai	India	0	
135		CTS	40.0	9284.0	Calcutta	India	1
136		NaN	NaN	NaN	Delhi	India	0
137		CTS	31.0	2034.0	Mumbai	India	0
138		CTS	44.0	3033.0	Cochin	India	0
139	Cognizant	22.0	2934.0		Noida	India	0
140		Infosys	44.0	4034.0	Hyderabad	India	0
141		TCS	33.0	5034.0	Calcutta	India	0
142	Infosys Pvt Lmt	22.0	8202.0	Mumbai	India	0	
143		TCS	33.0	9024.0	Calcutta	India	1
144		Infosys	22.0	8787.0	Calcutta	India	1
145		Infosys	44.0	4034.0	Delhi	India	1
146		TCS	33.0	5034.0	Mumbai	India	1
147		Infosys	22.0	8202.0	Cochin	India	0

In [15]:

```
print("The shape of the dataset is:")
data3.shape
```

The shape of the dataset is:

Out[15]: (148, 6)

In [17]:

```
print("Feature properties of the dataset is:")
print("\t")
data3.info()
```

Feature properties of the dataset is:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 148 entries, 0 to 147
Data columns (total 6 columns):
 #   Column   Non-Null Count   Dtype  
 ---  --       --           --      
 0   Company  140 non-null    object  
 1   Age      130 non-null    float64 
 2   Salary   124 non-null    float64 
 3   Place    134 non-null    object  
 4   Country  148 non-null    object  
 5   Gender   148 non-null    int64   
dtypes: float64(2), int64(1), object(3)
memory usage: 7.1+ KB
```

```
In [19]: print("Feature names of the dataset is:")
data3.columns
```

Feature names of the dataset is:

```
Out[19]: Index(['Company', 'Age', 'Salary', 'Place', 'Country', 'Gender'], dtype='object')
```

```
In [21]: print("Statistical Analysis of the dataset")
print("\t")
data3.describe()
```

Statistical Analysis of the dataset

Out[21]:

	Age	Salary	Gender
<b>count</b>	130.000000	124.000000	148.000000
<b>mean</b>	30.484615	5312.467742	0.222973
<b>std</b>	11.096640	2573.764683	0.417654
<b>min</b>	0.000000	1089.000000	0.000000
<b>25%</b>	22.000000	3030.000000	0.000000
<b>50%</b>	32.500000	5000.000000	0.000000
<b>75%</b>	37.750000	8000.000000	0.000000
<b>max</b>	54.000000	9876.000000	1.000000

In [23]:

```
print("Unique Value Analysis of the Dataset")
print("\t")
for column in data3.columns:
    unique_values = data3[column].unique() # Displays unique values in a column
    count_unique_values = len(unique_values) # Displays number of unique values in a column
    print(f"Column Name: {column}")
    print(f"Unique Values: {unique_values}")
    print(f"Number of Unique Values: {count_unique_values}")
    print("\n")
```

### Unique Value Analysis of the Dataset

Column Name: Company

Unique Values: ['TCS' 'Infosys' 'CTS' nan 'Tata Consultancy Services' 'Congnizant'  
'Infosys Pvt Lmt']

Number of Unique Values: 7

Column Name: Age

Unique Values: [20. 30. 35. 40. 23. nan 34. 45. 18. 22. 32. 37. 50. 21. 46. 36. 26. 41.  
24. 25. 43. 19. 38. 51. 31. 44. 33. 17. 0. 54.]

Number of Unique Values: 30

Column Name: Salary

Unique Values: [ nan 2300. 3000. 4000. 5000. 6000. 7000. 8000. 9000. 1089. 1234. 3030.  
3045. 3184. 4824. 5835. 7084. 8943. 8345. 9284. 9876. 2034. 7654. 2934.  
4034. 5034. 8202. 9024. 4345. 6544. 6543. 3234. 4324. 5435. 5555. 8787.  
3454. 5654. 5009. 5098. 3033.]

Number of Unique Values: 41

Column Name: Place

Unique Values: ['Chennai' 'Mumbai' 'Calcutta' 'Delhi' 'Podicherry' 'Cochin' nan 'Noida'  
'Hyderabad' 'Bhopal' 'Nagpur' 'Pune']

Number of Unique Values: 12

Column Name: Country

Unique Values: ['India']

Number of Unique Values: 1

Column Name: Gender

Unique Values: [0 1]

Number of Unique Values: 2

```
In [25]: print("Renamed Features of the Dataset")
print("\t")
```

```
data3.columns = data3.columns.str.upper()  
data3
```

Renamed Features of the Dataset

Out[25]:

	COMPANY	AGE	SALARY	PLACE	COUNTRY	GENDER
0	TCS	20.0	NaN	Chennai	India	0
1	Infosys	30.0	NaN	Mumbai	India	0
2	TCS	35.0	2300.0	Calcutta	India	0
3	Infosys	40.0	3000.0	Delhi	India	0
4	TCS	23.0	4000.0	Mumbai	India	0
...	...	...	...	...	...	...
143	TCS	33.0	9024.0	Calcutta	India	1
144	Infosys	22.0	8787.0	Calcutta	India	1
145	Infosys	44.0	4034.0	Delhi	India	1
146	TCS	33.0	5034.0	Mumbai	India	1
147	Infosys	22.0	8202.0	Cochin	India	0

148 rows × 6 columns

In [27]:

```
print("Rename Place to City")  
print("\t")  
data3 = data3.rename(columns={'PLACE': 'CITY'})  
data3
```

Rename Place to City

Out[27]:

	COMPANY	AGE	SALARY	CITY	COUNTRY	GENDER
0	TCS	20.0	NaN	Chennai	India	0
1	Infosys	30.0	NaN	Mumbai	India	0
2	TCS	35.0	2300.0	Calcutta	India	0
3	Infosys	40.0	3000.0	Delhi	India	0
4	TCS	23.0	4000.0	Mumbai	India	0
...	...	...	...	...	...	...
143	TCS	33.0	9024.0	Calcutta	India	1
144	Infosys	22.0	8787.0	Calcutta	India	1
145	Infosys	44.0	4034.0	Delhi	India	1
146	TCS	33.0	5034.0	Mumbai	India	1
147	Infosys	22.0	8202.0	Cochin	India	0

148 rows × 6 columns

### Data Cleaning:

Find the missing and inappropriate values, treat them appropriately.

Remove all duplicate rows.

Find the outliers.

Replace the value 0 in age as NaN.

Treat the null values in all columns using any measures (removing/ replace the values with mean/median/mode).

In [29]:

```
print("Missing/Null values in each column:")
print('\t')
missing_values = data3.isnull().sum()
missing_values
```

Missing/Null values in each column:

```
Out[29]: COMPANY      8  
          AGE        18  
          SALARY     24  
          CITY       14  
          COUNTRY    0  
          GENDER      0  
          dtype: int64
```

```
In [31]: print("Duplicate values in the dataset:")  
print('\t')  
data3.duplicated().sum()
```

Duplicate values in the dataset:

```
Out[31]: 4
```

```
In [33]: data3.shape
```

```
Out[33]: (148, 6)
```

```
In [35]: data3.drop_duplicates(inplace=True)
```

```
In [37]: data3.shape
```

```
Out[37]: (144, 6)
```

```
In [39]: print("Replacing 0 with NaN values in the age column")  
print('\t')  
data3['AGE'] = data3['AGE'].replace(0,np.nan)  
data3
```

Replacing 0 with NaN values in the age column

Out[39]:

	COMPANY	AGE	SALARY	CITY	COUNTRY	GENDER
0	TCS	20.0	NaN	Chennai	India	0
1	Infosys	30.0	NaN	Mumbai	India	0
2	TCS	35.0	2300.0	Calcutta	India	0
3	Infosys	40.0	3000.0	Delhi	India	0
4	TCS	23.0	4000.0	Mumbai	India	0
...	...	...	...	...	...	...
142	Infosys Pvt Lmt	22.0	8202.0	Mumbai	India	0
143	TCS	33.0	9024.0	Calcutta	India	1
145	Infosys	44.0	4034.0	Delhi	India	1
146	TCS	33.0	5034.0	Mumbai	India	1
147	Infosys	22.0	8202.0	Cochin	India	0

144 rows × 6 columns

In [41]:

```
print("Replacing missing values in the numerical columns with median and mode")
print('\t')
data3['AGE'] = data3['AGE'].fillna(data3['AGE'].median())
data3['SALARY'] = data3['SALARY'].fillna(data3['SALARY'].median())
data3
```

Replacing missing values in the numerical columns with median and mode

Out[41]:

	COMPANY	AGE	SALARY	CITY	COUNTRY	GENDER
0	TCS	20.0	5000.0	Chennai	India	0
1	Infosys	30.0	5000.0	Mumbai	India	0
2	TCS	35.0	2300.0	Calcutta	India	0
3	Infosys	40.0	3000.0	Delhi	India	0
4	TCS	23.0	4000.0	Mumbai	India	0
...	...	...	...	...	...	...
142	Infosys Pvt Lmt	22.0	8202.0	Mumbai	India	0
143	TCS	33.0	9024.0	Calcutta	India	1
145	Infosys	44.0	4034.0	Delhi	India	1
146	TCS	33.0	5034.0	Mumbai	India	1
147	Infosys	22.0	8202.0	Cochin	India	0

144 rows × 6 columns

In [43]:

```
print("Replacing missing values in the categorical columns with mode")
print('\t')
data3['COMPANY'] = data3['COMPANY'].fillna(data3['COMPANY'].mode()[0])
data3['CITY'] = data3['CITY'].fillna(data3['CITY'].mode()[0])
data3['GENDER'] = data3['GENDER'].fillna(data3['GENDER'].mode()[0])
data3
```

Replacing missing values in the categorical columns with mode

Out[43]:

	COMPANY	AGE	SALARY	CITY	COUNTRY	GENDER
0	TCS	20.0	5000.0	Chennai	India	0
1	Infosys	30.0	5000.0	Mumbai	India	0
2	TCS	35.0	2300.0	Calcutta	India	0
3	Infosys	40.0	3000.0	Delhi	India	0
4	TCS	23.0	4000.0	Mumbai	India	0
...	...	...	...	...	...	...
142	Infosys Pvt Lmt	22.0	8202.0	Mumbai	India	0
143	TCS	33.0	9024.0	Calcutta	India	1
145	Infosys	44.0	4034.0	Delhi	India	1
146	TCS	33.0	5034.0	Mumbai	India	1
147	Infosys	22.0	8202.0	Cochin	India	0

144 rows × 6 columns

In [45]:

```
print("Boxplot Graph to visualize outliers in AGE and SALARY columns")
print('\t')
plt.figure(figsize=(10, 6))

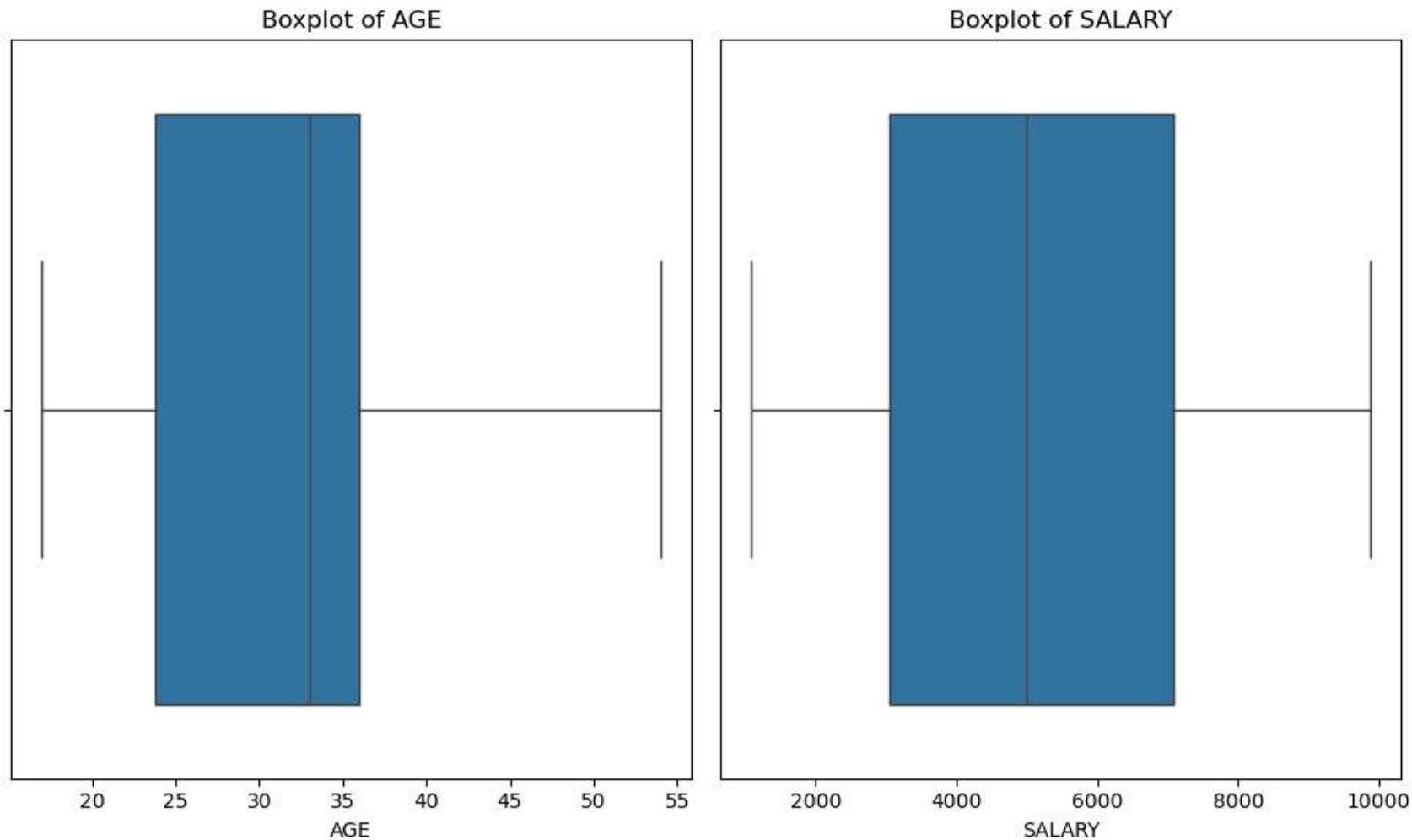
# Plot for Age
plt.subplot(1, 2, 1)
sns.boxplot(x=data3['AGE'])
plt.title('Boxplot of AGE')
plt.xlabel('AGE')

# Plot for Salary
plt.subplot(1, 2, 2)
sns.boxplot(x=data3['SALARY'])
plt.title('Boxplot of SALARY')
plt.xlabel('SALARY')

# Show the plots
```

```
plt.tight_layout()  
plt.show()
```

Boxplot Graph to visualize outliers in AGE and SALARY columns



```
In [47]: print("Using IQR Method to determine outliers in the AGE column")  
print('\t')  
Q1_age = data3['AGE'].quantile(0.25)  
Q3_age = data3['AGE'].quantile(0.75)  
  
IQR_age = Q3_age - Q1_age
```

```
# Calculate the Lower and upper bounds for outliers in 'age'
lower_bound_age = Q1_age - 1.5 * IQR_age
upper_bound_age = Q3_age + 1.5 * IQR_age

# Identify outliers in the 'age' column
age_outliers = data3[(data3['AGE'] < lower_bound_age) | (data3['AGE'] > upper_bound_age)]

print("Using IQR Method to determine outliers in the SALARY column")
print('\t')
Q1_salary = data3['SALARY'].quantile(0.25)
Q3_salary = data3['SALARY'].quantile(0.75)
IQR_salary = Q3_salary - Q1_salary

# Calculate the Lower and upper bounds for outliers in 'salary'
lower_bound_salary = Q1_salary - 1.5 * IQR_salary
upper_bound_salary = Q3_salary + 1.5 * IQR_salary

# Identify outliers in the 'salary' column
salary_outliers = data3[(data3['SALARY'] < lower_bound_salary) | (data3['SALARY'] > upper_bound_salary)]

# 3. Display the outliers in Age and Salary columns
print("Outliers in Age:")
print(age_outliers)
print("\n")
print("Outliers in Salary:")
print(salary_outliers)
```

Using IQR Method to determine outliers in the AGE column

Using IQR Method to determine outliers in the SALARY column

Outliers in Age:  
Empty DataFrame  
Columns: [COMPANY, AGE, SALARY, CITY, COUNTRY, GENDER]  
Index: []

Outliers in Salary:  
Empty DataFrame  
Columns: [COMPANY, AGE, SALARY, CITY, COUNTRY, GENDER]  
Index: []

The above findings indicate that there are no outliers in AGE and SALARY column.

### Data Analysis:

Filter the data with age >40 and salary<5000.

Plot the chart with age and salary.

Count the number of people from each place and represent it visually.

```
In [49]: print("Filtered data of employees with age > 40 and Salary < 5000")
print('\t')
filtered_data = data3[(data3['AGE'] > 40) & (data3['SALARY'] < 5000)]
filtered_data
```

Filtered data of employees with age > 40 and Salary < 5000

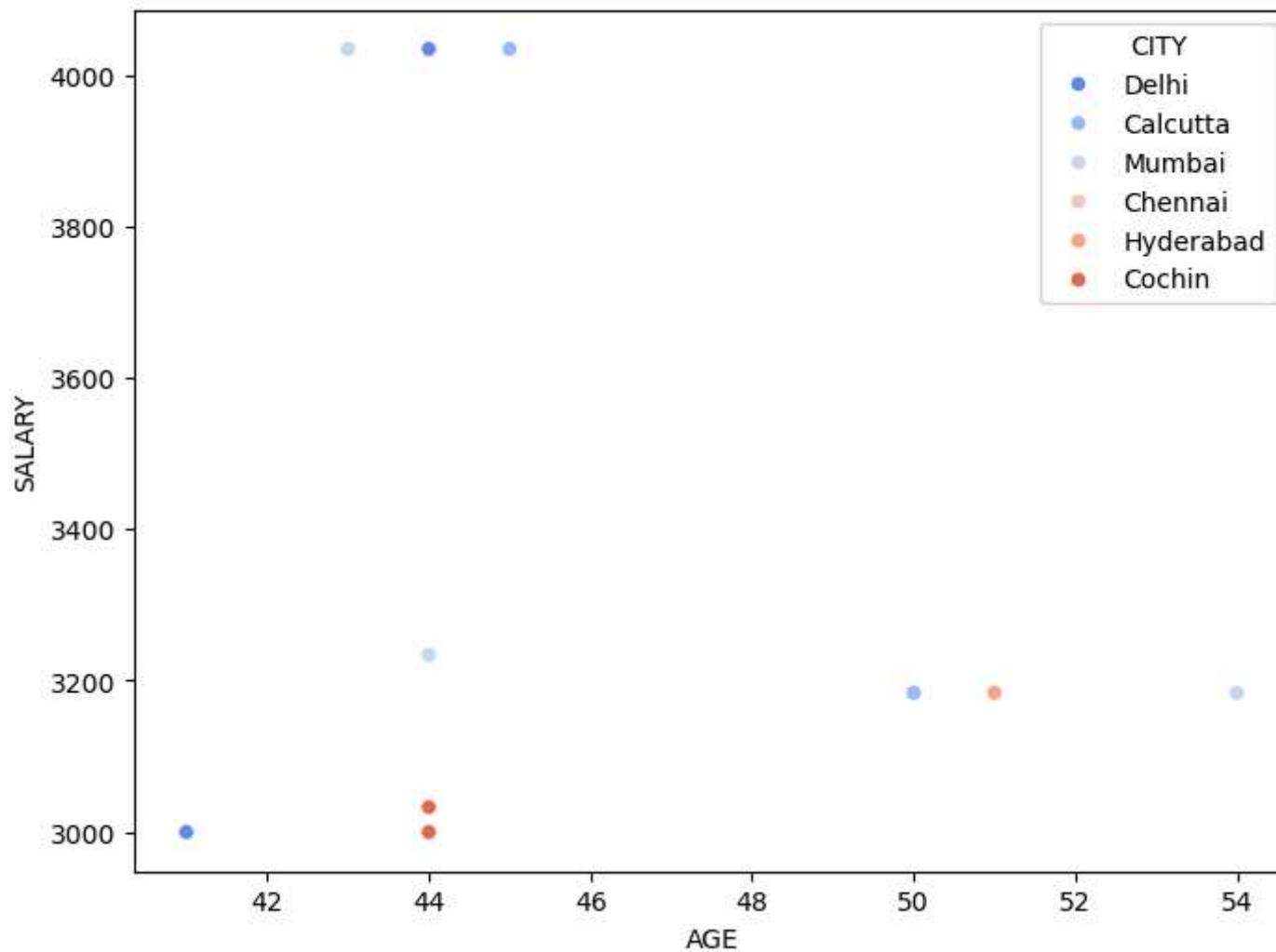
Out[49]:

	COMPANY	AGE	SALARY	CITY	COUNTRY	GENDER
21	Infosys	50.0	3184.0	Delhi	India	0
32	Infosys	45.0	4034.0	Calcutta	India	0
39	Infosys	41.0	3000.0	Mumbai	India	0
50	Infosys	41.0	3000.0	Chennai	India	0
57	Infosys	51.0	3184.0	Hyderabad	India	0
68	Infosys	43.0	4034.0	Mumbai	India	0
75	Infosys	44.0	3000.0	Cochin	India	0
86	Infosys	41.0	3000.0	Delhi	India	0
93	Infosys	54.0	3184.0	Mumbai	India	0
104	Infosys	44.0	4034.0	Delhi	India	0
122	Infosys	44.0	3234.0	Mumbai	India	0
129	Infosys	50.0	3184.0	Calcutta	India	0
138	CTS	44.0	3033.0	Cochin	India	0
140	Infosys	44.0	4034.0	Hyderabad	India	0
145	Infosys	44.0	4034.0	Delhi	India	1

In [51]:

```
plt.figure(figsize=(8, 6))
sns.scatterplot(data=filtered_data, x='AGE', y='SALARY', hue='CITY', palette='coolwarm')
plt.title('Scatter Plot of Filtered Data')
plt.xlabel('AGE')
plt.ylabel('SALARY')
plt.legend(title='CITY')
plt.show()
```

Scatter Plot of Filtered Data

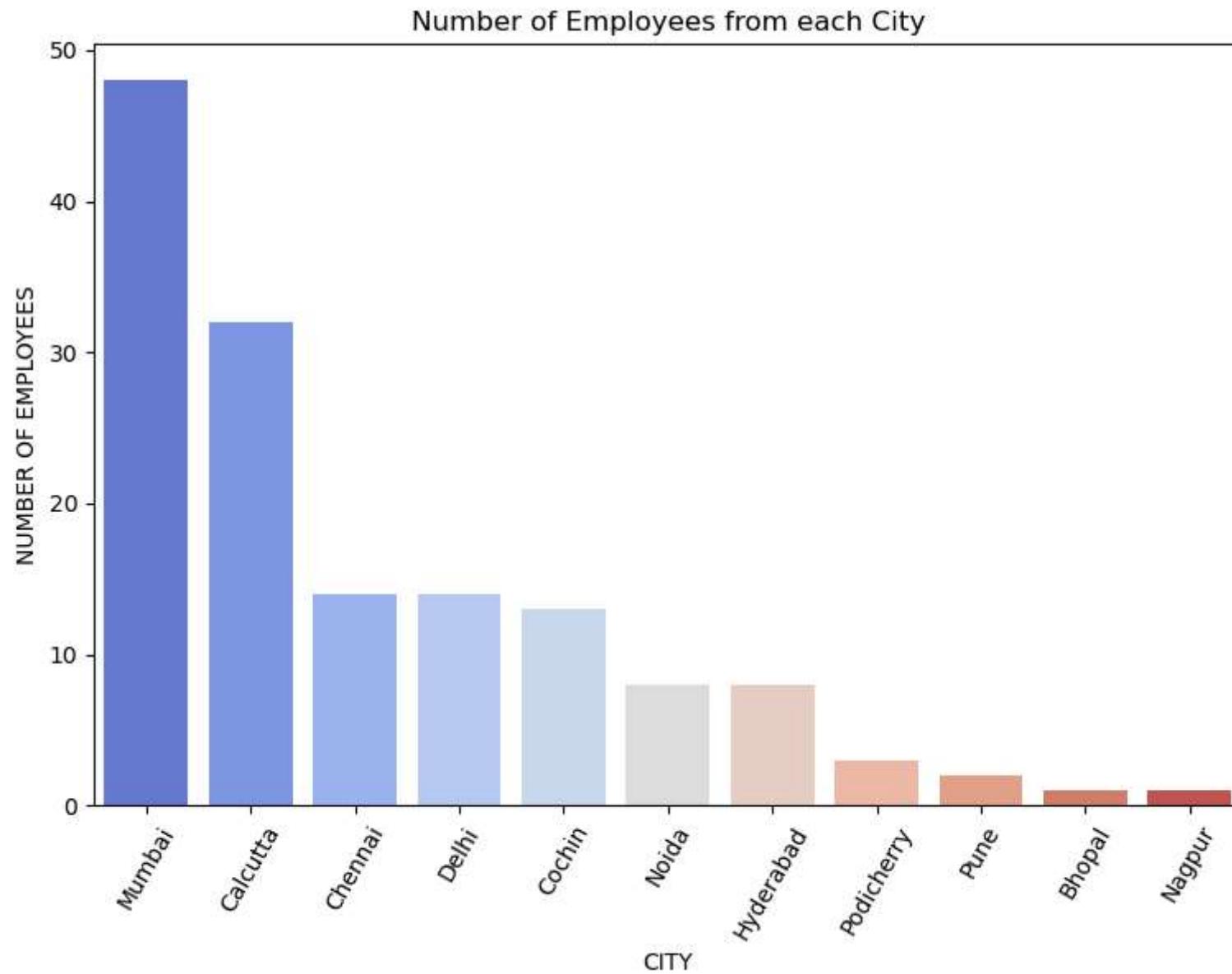


```
In [53]: print("Count of employees from each City")
print('\t')
city_counts = data3['CITY'].value_counts()
city_counts
```

Count of employees from each City

```
Out[53]: CITY
Mumbai      48
Calcutta    32
Chennai     14
Delhi       14
Cochin      13
Noida        8
Hyderabad   8
Podicherry  3
Pune         2
Bhopal      1
Nagpur      1
Name: count, dtype: int64
```

```
In [55]: plt.figure(figsize = (9, 6))
sns.barplot(x = city_counts.index, y = city_counts.values, palette='coolwarm')
plt.title('Number of Employees from each City')
plt.xlabel('CITY')
plt.ylabel('NUMBER OF EMPLOYEES')
plt.xticks(rotation = 60)
plt.show()
```



#### Data Encoding:

Convert categorical variables into numerical representations using techniques such as one-hot encoding, label encoding, making them suitable for analysis by machine learning algorithms.

```
In [57]: print("Output after Label Encoding")
print('\t')
from sklearn.preprocessing import LabelEncoder

# Identify categorical variables
categorical_columns = ['COMPANY', 'CITY', 'COUNTRY', 'GENDER']

# Initialize LabelEncoder for 'gender' column (ordinal data)
label_encoder = LabelEncoder()

# Apply Label encoding for 'CITY'
data3['CITY'] = label_encoder.fit_transform(data3['CITY'])

# Apply Label encoding for 'COUNTRY'
data3['COUNTRY'] = label_encoder.fit_transform(data3['COUNTRY'])

# Display the first few rows of the encoded dataframe
print(data3.head(15))
```

Output after Label Encoding

	COMPANY	AGE	SALARY	CITY	COUNTRY	GENDER
0	TCS	20.0	5000.0	2	0	0
1	Infosys	30.0	5000.0	6	0	0
2	TCS	35.0	2300.0	1	0	0
3	Infosys	40.0	3000.0	4	0	0
4	TCS	23.0	4000.0	6	0	0
5	Infosys	33.0	5000.0	1	0	0
6	TCS	33.0	6000.0	2	0	1
7	Infosys	23.0	7000.0	6	0	1
8	TCS	34.0	8000.0	1	0	1
9	CTS	45.0	9000.0	4	0	0
10	CTS	23.0	5000.0	6	0	0
11	CTS	34.0	1089.0	1	0	0
12	CTS	45.0	5000.0	2	0	0
13	CTS	18.0	1234.0	6	0	0
14	Infosys	40.0	3000.0	1	0	0

```
In [59]: print("Output after OneHot Encoding")
print('\t')
from sklearn.preprocessing import OneHotEncoder
```

```
# Identify categorical variables
categorical_columns = ['COMPANY', 'CITY', 'COUNTRY', 'GENDER']

# Apply One-Hot Encoding using pd.get_dummies for 'company', 'place', 'country_name' columns (nominal data)
data_encoded = pd.get_dummies(data3, columns=['COMPANY', 'CITY', 'COUNTRY', 'GENDER'], drop_first=True)

# Display the first few rows of the encoded dataframe
print(data_encoded.head())
```

Output after OneHot Encoding

	AGE	SALARY	COMPANY_Cognizant	COMPANY_Infosys	COMPANY_Infosys	Pvt Lmt	\
0	20.0	5000.0	False	False		False	
1	30.0	5000.0	False	True		False	
2	35.0	2300.0	False	False		False	
3	40.0	3000.0	False	True		False	
4	23.0	4000.0	False	False		False	

	COMPANY_TCS	COMPANY_Tata Consultancy Services	CITY_1	CITY_2	CITY_3	\	
0	True		False	False	True	False	
1	False		False	False	False	False	
2	True		False	True	False	False	
3	False		False	False	False	False	
4	True		False	False	False	False	

	CITY_4	CITY_5	CITY_6	CITY_7	CITY_8	CITY_9	CITY_10	GENDER_1
0	False	False						
1	False	False	True	False	False	False	False	False
2	False	False						
3	True	False	False	False	False	False	False	False
4	False	False	True	False	False	False	False	False

### Feature Scaling:

After the process of encoding, perform the scaling of the features using StandardScaler and MinMaxScaler.

```
In [61]: print("Output after executing StandardScaler and MinMaxScaler")
print('\t')
from sklearn.preprocessing import StandardScaler, MinMaxScaler

# Initialize StandardScaler and MinMaxScaler
standard_scaler = StandardScaler()
```

```
minmax_scaler = MinMaxScaler()

# Select only the numerical columns i.e., 'AGE', 'SALARY'
numerical_columns = ['AGE', 'SALARY']

# Apply StandardScaler to numerical columns
data_standard_scaled = data3.copy()
data_standard_scaled[numerical_columns] = standard_scaler.fit_transform(data_standard_scaled[numerical_columns])

# Apply MinMaxScaler to numerical columns
data_minmax_scaled = data3.copy()
data_minmax_scaled[numerical_columns] = minmax_scaler.fit_transform(data_minmax_scaled[numerical_columns])

# Display the first few rows of the scaled data
print("Data after Standard Scaling:")
print(data_standard_scaled.head(20))

print("\nData after Min-Max Scaling:")
print(data_minmax_scaled.head(20))
```

Output after executing StandardScaler and MinMaxScaler

Data after Standard Scaling:

	COMPANY	AGE	SALARY	CITY	COUNTRY	GENDER
0	TCS	-1.484676	-0.100827	2	0	0
1	Infosys	-0.267174	-0.100827	6	0	0
2	TCS	0.341577	-1.243735	1	0	0
3	Infosys	0.950328	-0.947426	4	0	0
4	TCS	-1.119426	-0.524127	6	0	0
5	Infosys	0.098077	-0.100827	1	0	0
6	TCS	0.098077	0.322472	2	0	1
7	Infosys	-1.119426	0.745771	6	0	1
8	TCS	0.219827	1.169070	1	0	1
9	CTS	1.559079	1.592369	4	0	0
10	CTS	-1.119426	-0.100827	6	0	0
11	CTS	0.219827	-1.756350	1	0	0
12	CTS	1.559079	-0.100827	2	0	0
13	CTS	-1.728177	-1.694972	6	0	0
14	Infosys	0.950328	-0.947426	1	0	0
15	TCS	-1.119426	-0.947426	4	0	0
16	Infosys	-1.119426	-0.934727	9	0	0
17	TCS	0.219827	-0.100827	3	0	0
18	TCS	-1.241176	-0.100827	2	0	0
19	Infosys	-0.023674	-0.100827	6	0	0

Data after Min-Max Scaling:

	COMPANY	AGE	SALARY	CITY	COUNTRY	GENDER
0	TCS	0.081081	0.445089	2	0	0
1	Infosys	0.351351	0.445089	6	0	0
2	TCS	0.486486	0.137817	1	0	0
3	Infosys	0.621622	0.217480	4	0	0
4	TCS	0.162162	0.331285	6	0	0
5	Infosys	0.432432	0.445089	1	0	0
6	TCS	0.432432	0.558894	2	0	1
7	Infosys	0.162162	0.672698	6	0	1
8	TCS	0.459459	0.786503	1	0	1
9	CTS	0.756757	0.900307	4	0	0
10	CTS	0.162162	0.445089	6	0	0
11	CTS	0.459459	0.000000	1	0	0
12	CTS	0.756757	0.445089	2	0	0
13	CTS	0.027027	0.016502	6	0	0
14	Infosys	0.621622	0.217480	1	0	0

15	TCS	0.162162	0.217480	4	0	0
16	Infosys	0.162162	0.220895	9	0	0
17	TCS	0.459459	0.445089	3	0	0
18	TCS	0.135135	0.445089	2	0	0
19	Infosys	0.405405	0.445089	6	0	0

In [ ]: