Data Exploration

```
[1] import numpy as np
    import pandas as pd

    #load the dataset
    df = pd.read_csv('/content/Employee.csv')

    #display first 5 rows
    df.head()
```

	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

Next steps: Generate code with df | View recommended plots | New interactive sheet

```
[2] #shape of the dataframe
    df.shape
```

```
(148, 6)
```

```
[3] #column names
    df.columns
```

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

```
[4] #count of unique values
    df.nunique()
```

```
#count of unique values
df.nunique()
```

	0
Company	6
Age	29
Salary	40
Place	11
Country	1
Gender	2

dtype: int64

```
#unique values in 'Company' column
df['Company'].unique()
```

```
array(['TCS', 'Infosys', 'CTS', nan, 'Tata Consultancy Services',
       'Congnizant', 'Infosys Pvt Lmt'], dtype=object)
```

```
#unique values in 'Age' column
df['Age'].unique()
```

```
array([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.])
```

```
#unique values in 'Salary' column
df['Salary'].unique()
```

```
array([  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.])
```

```
        21., 46., 36., 26., 41., 24., 25., 43., 19., 38., 51., 31., 44.,
        33., 17.,  0., 54.])
```

```
#unique values in 'Salary' column
df['Salary'].unique()
```

```
array([  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.])
```

[8]
```
#unique values in 'Place' column
df['Place'].unique()
```

```
array(['Chennai', 'Mumbai', 'Calcutta', 'Delhi', 'Podicherry', 'Cochin',
       nan, 'Noida', 'Hyderabad', 'Bhopal', 'Nagpur', 'Pune'],
      dtype=object)
```

[9]
```
#unique values in 'Country' column
df['Country'].unique()
```

```
array(['India'], dtype=object)
```

[10]
```
#unique values in 'Gender' column
df['Gender'].unique()
```

```
array([0, 1])
```

[11]
```
df.describe().T
```

	count	mean	std	min	25%	50%	75%	max
Age	130.0	30.484615	11.096640	0.0	22.0	32.5	37.75	54.0
Salary	124.0	5312.467742	2573.764683	1089.0	3030.0	5000.0	8000.00	9876.0
Gender	148.0	0.222973	0.417654	0.0	0.0	0.0	0.00	1.0

[12] `df.describe(include='all')`

	Company	Age	Salary	Place	Country	Gender
count	140	130.000000	124.000000	134	148	148.000000
unique	6	NaN	NaN	11	1	NaN
top	TCS	NaN	NaN	Mumbai	India	NaN
freq	53	NaN	NaN	37	148	NaN
mean	NaN	30.484615	5312.467742	NaN	NaN	0.222973
std	NaN	11.096640	2573.764683	NaN	NaN	0.417654
min	NaN	0.000000	1089.000000	NaN	NaN	0.000000
25%	NaN	22.000000	3030.000000	NaN	NaN	0.000000
50%	NaN	32.500000	5000.000000	NaN	NaN	0.000000
75%	NaN	37.750000	8000.000000	NaN	NaN	0.000000
max	NaN	54.000000	9876.000000	NaN	NaN	1.000000

[13] `df.info()`

```
<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
```

[14] `#checking null values`

```
#checking null values
df.isnull().sum()
```

	0
Company	8
Age	18
Salary	24
Place	14
Country	0
Gender	0

dtype: int64

```
[15] #renaming the columns
df.rename(columns={'Company': 'company', 'Age': 'age', 'Salary': 'salary', 'Place': 'place', 'Country': 'country', 'Gender': 'gender'}, inplace=True)

df.columns
```

```
Index(['company', 'age', 'salary', 'place', 'country', 'gender'], dtype='object')
```

Data Cleaning

```
[16] #missing values
df.isna().sum()
```

	0
company	8
age	18
salary	24
place	14

Data Cleaning

```
#missing values
df.isna().sum()
```

	0
company	8
age	18
salary	24
place	14
country	0
gender	0

dtype: int64

```
[17] #unique values in 'company' column
df['company'].unique()
```

```
array(['TCS', 'Infosys', 'CTS', nan, 'Tata Consultancy Services',
       'Congnizant', 'Infosys Pvt Lmt'], dtype=object)
```

Before replacing null values in 'company' column, in 'company' column it seems that 'TCS' in different names like 'Tata Consultancy Services' and 'CTS' which is also mispelled, so renaming this 'Tata Consultancy Services' and 'CTS' to 'TCS'. Then there is 'Infosys' and 'Infosys Pvt Lmt', so renaming this 'Infosys Pvt Lmt' to 'Infosys'

```
[18] #replacing values in 'company' column
df['company'] = df['company'].replace({
    'Tata Consultancy Services' : 'TCS',
    'CTS' : 'TCS',
    'Infosys Pvt Lmt' : 'Infosys'
})
```

Before replacing null values in 'company' column, in 'company' column it seems that 'TCS' in different names like 'Tata Consultancy Services' and 'CTS' which is also mispelled, so renaming this 'Tata Consultancy Services' and 'CTS' to 'TCS'. Then there is 'Infosys' and 'Infosys Pvt Lmt', so renaming this 'Infosys Pvt Lmt' to 'Infosys'

```
#replacing values in 'company' column
df['company'] = df['company'].replace({
    'Tata Consultancy Services' : 'TCS',
    'CTS' : 'TCS',
    'Infosys Pvt Lmt' : 'Infosys'
})
```

```
[19] #count of null values in 'company' column
     df['company'].isna().sum()
```

```
8
```

```
[20] #replacing null values using mode in 'company' column
     mode_value = df['company'].mode()[0]
     df['company'] = df['company'].fillna(mode_value)
```

```
[21] #checking null values after replacing it with mode
     df['company'].isna().sum()
```

```
0
```

```
[22] #unique values in 'age' column
     df['age'].unique()
```

```
array([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.])
```

```
[23] #replacing value '0' in age as Nan
     df['age'] = df['age'].replace(0, np.nan)
```

[23]
```
#replacing value '0' in age as Nan
df['age'] = df['age'].replace(0, np.nan)
```

[24]
```
#checking null vallues in 'place' column
df['place'].isna().sum()
```

```
14
```

[25]
```
#replacing null values using mode in 'place' column
mode_value_place = df['place'].mode()[0]
df['place'] = df['place'].fillna(mode_value_place)
```

[26]
```
#checking null values after replacing it with mode
df['place'].isna().sum()
```

```
0
```

[27]
```
#checking for other missing values
df.isnull().sum()
```

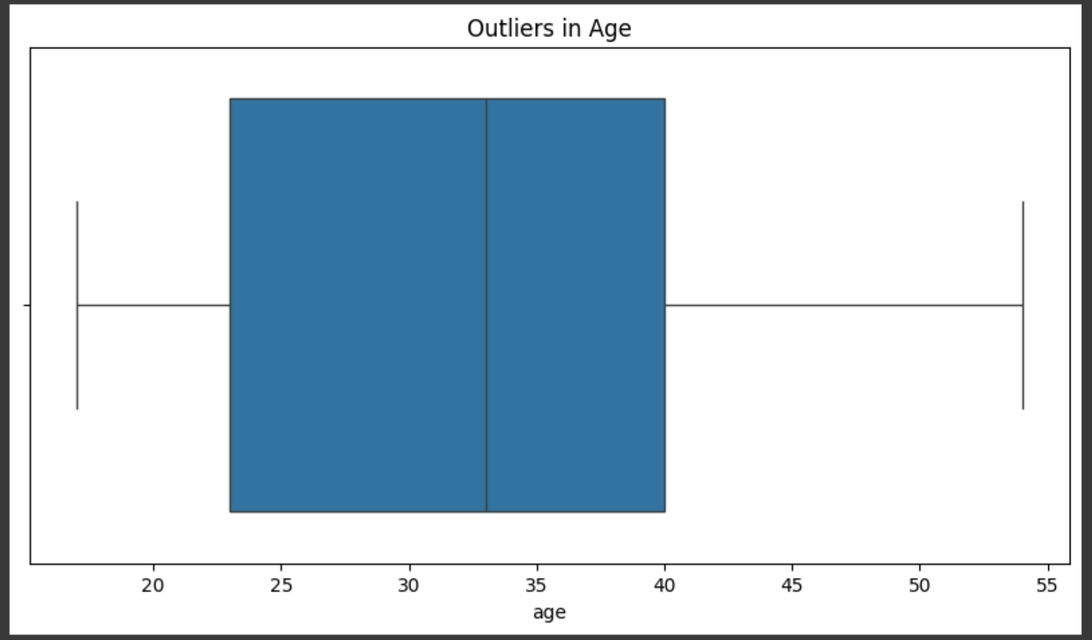
	0
company	0
age	24
salary	24
place	0
country	0
gender	0

dtype: int64

[28]
```
#checking the outliers of 'age' and 'salary' before replacing null values
import matplotlib.pyplot as plt
import seaborn as sns
```

```
[28] #checking the outliers of 'age' and 'salary' before replacing null values
     import matplotlib.pyplot as plt
     import seaborn as sns

     #visualizing outliers in 'age' column using boxplots
     plt.figure(figsize=(10, 5))
     sns.boxplot(x=df['age'])
     plt.title('Outliers in Age')
     plt.show()
```

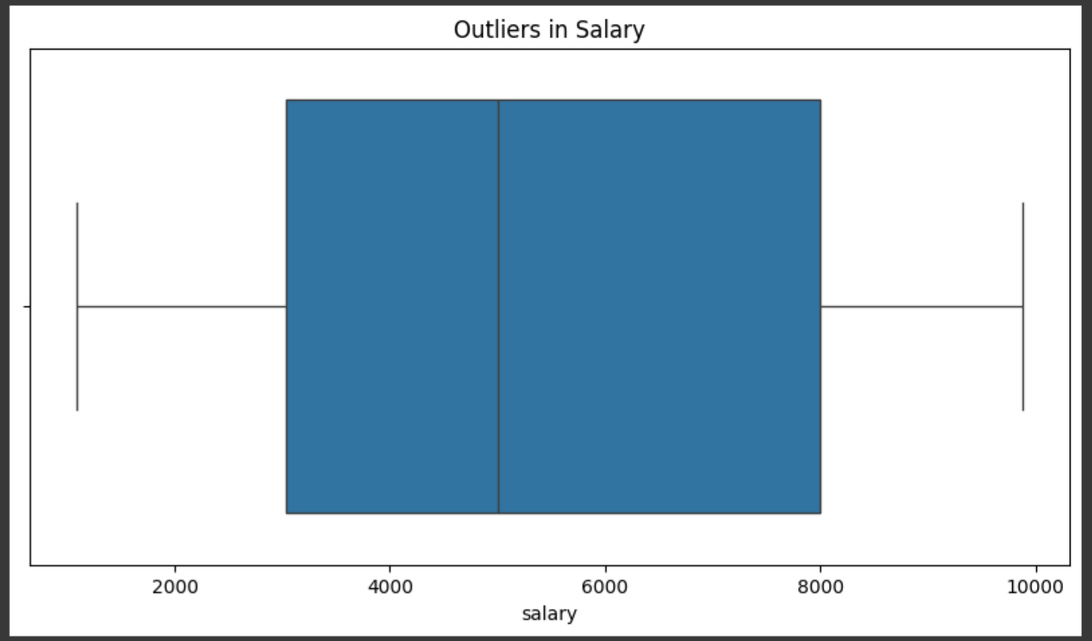


It seems that there is no any outliers in 'age' column

```
[29] #visualizing outliers in 'Salary' column using boxplots
```

It seems that there is no any outliers in 'age' column

```
#visualizing outliers in 'Salary' column using boxplots
plt.figure(figsize=(10, 5))
sns.boxplot(x=df['salary'])
plt.title('Outliers in Salary')
plt.show()
```



It seems that there is no any outliers in 'salary' column

```
[30] #replacing null values in 'age' and 'salary' using median
```

[30]
```
#replacing null values in 'age' and 'salary' using median
df.fillna({
    'age' : df['age'].median(),
    'salary' : df['salary'].median()
}, inplace=True)
```

```
#checking null values
df.isnull().sum()
```

	0
company	0
age	0
salary	0
place	0
country	0
gender	0

dtype: int64

[32]
```
#checking duplicate rows
df.duplicated().sum()
```

```
5
```

[33]
```
#removing duplicate rows
df.drop_duplicates(inplace=True)
```

[34]
```
#checking duplicates after removal
df.duplicated().sum()
```

```
0
```

Data Analysis

```
#filtering data with 'age' > 40 and salary < 5000
filtered_df = df[(df['age'] > 40) & (df['salary'] < 5000)]

filtered_df
```

	company	age	salary	place	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	TCS	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

Next steps: Generate code with filtered_df | View recommended plots | New interactive sheet

```
[36] #plotting the chart with 'age' and 'salary'
     plt.figure(figsize=(8, 5))
```

```
#plotting the chart with 'age' and 'salary'
plt.figure(figsize=(8, 5))
sns.scatterplot(x=df['age'], y=df['salary'])
plt.title('Age v/s Salary')
plt.xlabel('Age')
plt.ylabel('Salary')
plt.show()
```


```
[37] #count of number of people by place
df['place'].value_counts()
```

```
            count
place
```

```
#count of number of people by place
df['place'].value_counts()
```

place	count
Mumbai	48
Calcutta	31
Chennai	14
Delhi	14
Cochin	13
Noida	8
Hyderabad	8
Podicherry	3
Pune	2
Bhopal	1
Nagpur	1

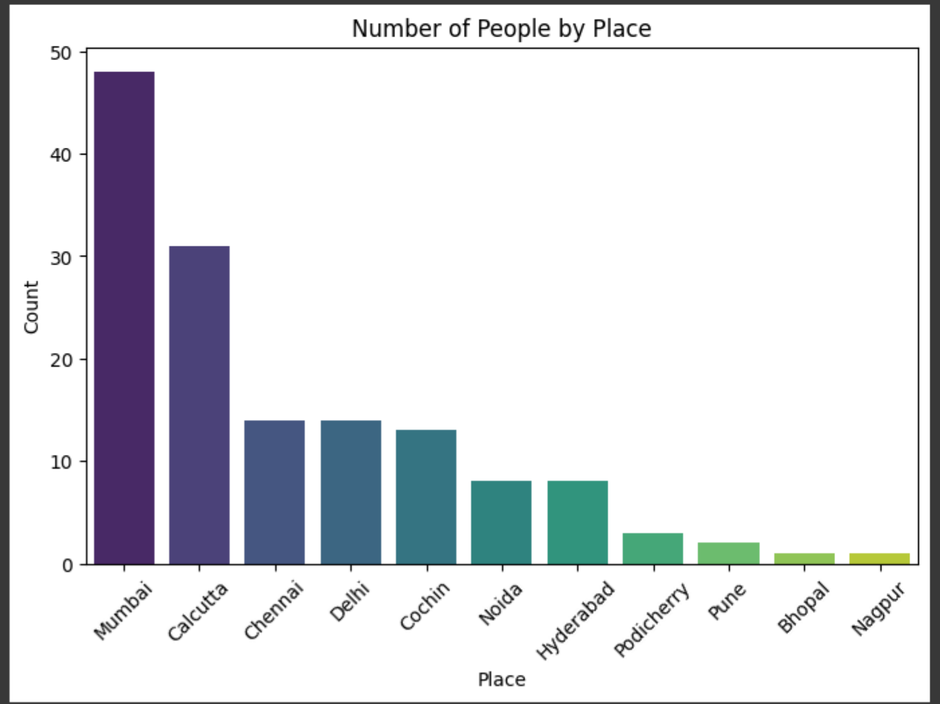
dtype: int64

```
[38] #visualization of number of people by place
place_counts = df['place'].value_counts()

plt.figure(figsize=(8,5))
sns.barplot(x=place_counts.index, y=place_counts.values, palette='viridis', hue=place_counts.index)
plt.title('Number of People by Place')
plt.xlabel('Place')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```

```python
#visualization of number of people by place
place_counts = df['place'].value_counts()

plt.figure(figsize=(8,5))
sns.barplot(x=place_counts.index, y=place_counts.values, palette='viridis', hue=place_counts.index)
plt.title('Number of People by Place')
plt.xlabel('Place')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.show()
```



Data Encoding

```
[39] #endocing using One-Hot/Get-dummies
     df_one_hot = df.copy()
     df_one_hot = pd.get_dummies(df, columns=['company', 'place', 'country'], prefix=['company', 'place', 'country'], drop_first=True)

     df_one_hot
```

	age	salary	gender	company_Infosys	company_TCS	place_Calcutta	place_Chennai	place_Cochin	place_Delhi	place_Hyderabad	place_Mumbai	place_Nagpur	place_Noida	place_Podicherry	place_Pune
0	20.0	5000.0	0	False	True	False	True	False	False	False	False	False	False	False	False
1	30.0	5000.0	0	True	False	False	False	False	False	False	True	False	False	False	False
2	35.0	2300.0	0	False	True	True	False	False	False	False	False	False	False	False	False
3	40.0	3000.0	0	True	False	False	False	False	True	False	False	False	False	False	False
4	23.0	4000.0	0	False	True	False	False	False	False	False	True	False	False	False	False
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
142	22.0	8202.0	0	True	False	False	False	False	False	False	True	False	False	False	False
143	33.0	9024.0	1	False	True	True	False	False	False	False	False	False	False	False	False
145	44.0	4034.0	1	True	False	False	False	False	True	False	False	False	False	False	False
146	33.0	5034.0	1	False	True	False	False	False	False	False	True	False	False	False	False
147	22.0	8202.0	0	True	False	False	False	True	False	False	False	False	False	False	False

143 rows × 15 columns

Next steps: Generate code with df_one_hot | View recommended plots | New interactive sheet

```
[40] #encoding using label encoder
     from sklearn.preprocessing import LabelEncoder
     df_le = df.copy()
     le = LabelEncoder()
     df_le['company'] = le.fit_transform(df_le['company'])
     df_le['place'] = le.fit_transform(df_le['place'])
```

143 rows × 15 columns

Next steps: Generate code with df_one_hot | View recommended plots | New interactive sheet

```
#encoding using label encoder
from sklearn.preprocessing import LabelEncoder
df_le = df.copy()
le = LabelEncoder()
df_le['company'] = le.fit_transform(df_le['company'])
df_le['place'] = le.fit_transform(df_le['place'])
df_le['country'] = le.fit_transform(df_le['country'])

df_le
```

	company	age	salary	place	country	gender
0	2	20.0	5000.0	2	0	0
1	1	30.0	5000.0	6	0	0
2	2	35.0	2300.0	1	0	0
3	1	40.0	3000.0	4	0	0
4	2	23.0	4000.0	6	0	0
...	...	...	...	...	...	...
142	1	22.0	8202.0	6	0	0
143	2	33.0	9024.0	1	0	1
145	1	44.0	4034.0	4	0	1
146	2	33.0	5034.0	6	0	1
147	1	22.0	8202.0	3	0	0

143 rows × 6 columns

Next steps: Generate code with df_le | View recommended plots | New interactive sheet

Feature Scaling

```
[41] #before scaling splitting into features (X) and target (y)
     #using df_one_hot here

     X = df_one_hot.drop(columns=['gender'])
     y = df_one_hot['gender']
```

```
[42] X.head()
```

	age	salary	company_Infosys	company_TCS	place_Calcutta	place_Chennai	place_Cochin	place_Delhi	place_Hyderabad	place_Mumbai	place_Nagpur	place_Noida	place_Podicherry	place_Pune
0	20.0	5000.0	False	True	False	True	False	False	False	False	False	False	False	False
1	30.0	5000.0	True	False	False	False	False	False	False	True	False	False	False	False
2	35.0	2300.0	False	True	True	False	False	False	False	False	False	False	False	False
3	40.0	3000.0	True	False	False	False	False	True	False	False	False	False	False	False
4	23.0	4000.0	False	True	False	False	False	False	False	True	False	False	False	False

Next steps: Generate code with X | View recommended plots | New interactive sheet

```
y.head()
```

	gender
0	0
1	0
2	0
3	0
4	0

dtype: int64

[44]
```
#splitting data into train and test

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)
```

```
#feature scaling using standardization(StandardScaler)

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

[46]
```
#feature scaling using normaliztion(MinMaScaler)
#using df_le here
#splitting into features and target

X = df_le.drop(columns=['gender'])
y = df_le['gender']

X.head()
```

	company	age	salary	place	country
0	2	20.0	5000.0	2	0
1	1	30.0	5000.0	6	0
2	2	35.0	2300.0	1	0
3	1	40.0	3000.0	4	0
4	2	23.0	4000.0	6	0

Next steps: Generate code with X | View recommended plots | New interactive sheet

Next steps: Generate code with X | View recommended plots | New interactive sheet

[47] y.head()

	gender
0	0
1	0
2	0
3	0
4	0

dtype: int64

[48]
```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, random_state=42)

X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

[ ] Start coding or generate with AI.