ASSIGNMENT 2 - EDA AND PREPROCESSING

The dataset "Employee.csv" contains employee-related data. The primary goal of this project is to design and implement a comprehensive data preprocessing system that addresses common challenges such as missing values, outliers, inconsistent formatting, and noise. By performing effective preprocessing, your task is to analyze the salary per employee and improve the overall quality, reliability, and usability of the data for further analysis and machine learning applications

SOURCE

Dataset: https://drive.google.com/file/d/1F3lRf32JM8ejnXq-Cbf9y7fa57zSHGz_/view?usp=sharing

IMPORTING MODULES

In [68]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

import warnings
import sys
if not sys.warnoptions:
    warnings.simplefilter("ignore")
```

LOAD DATASET

In [70]:

```
# LOAD THE DATASET
data = pd.read_csv("Employee.csv")
data
```

Out[70]:

	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

1. DISPLAY FIRST & LAST ROWS

In [72]:

```
# DISPLAY FIRST FEW ROES TO UNDERSTAND THE STRUCTURE OF THE DATA
print("First Few Rows: ")
data.head(10)
```

```
First Few Rows:
```

Out[72]:

	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 [74]:

```
# DISPLAY LAST FEW ROWS
print("Last Few Rows: ")
data.tail(10)
```

```
Last Few Rows:
```

Out[74]:

	Company	Age	Salary	Place	Country	Gender
138	CTS	44.0	3033.0	Cochin	India	0
139	Congnizant	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

2. MAKE COPY OF ORIGINAL DATASET

In [76]:

```
# CREATE COPY OF ORIGINAL DATASET
data_copy = data.copy()
data_copy
```

Out[76]:

	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

3. SHAPE OF THE DATA

In [78]:

```
# SHAPE OF THE DATASET
print("Shape of the data:")
data.shape
```

```
Shape of the data:
```

Out[78]:

```
(148, 6)
```

4. DATATYPE OF EACH COLUMN

In [80]:

```
# DISPLAY DATA TYPE OF EACH COLUMN
print("Dataset Info:")
data.info()
```

```
Dataset 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
```

5. STATISTICAL SUMMARY OF DATA

In [82]:

```
# DISPLAY STATISTICSL SUMMARY
print("Statistical Summary:")
data.describe()
```

```
Statistical Summary:
```

Out[82]:

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

6. DISPLAY ALL COLUMN NAMES

In [84]:

```
# DISPLAY PARTICULAR COLUMN
print("Columns of the dataset:")
data.columns
```

```
Columns of the dataset:
```

Out[84]:

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

7. UNIQUE VALUE IN EACH COLUMN AND ITS LENGTH

In [86]:

```
for column in data.columns:
    unique_values = data[column].unique()  # Get unique values in the column
    unique_count = len(unique_values)  # Get the count of unique values
    print(f"COLUMN: {column}")
    print(f"UNIQUE VALUES: {unique_values}")
    print(f"COUNT OF UNIQUE VALUES: {unique_count}")
    print("\n")
```

```
COLUMN: Company
UNIQUE VALUES: ['TCS' 'Infosys' 'CTS' nan 'Tata Consultancy Services' 'Congnizant'
 'Infosys Pvt Lmt']
COUNT OF UNIQUE VALUES: 7


COLUMN: 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.]
COUNT OF UNIQUE VALUES: 30


COLUMN: Salary
UNIQUE VALUES: [  nan 2300. 3000. 4000. 5000. 6000. 7000. 8000. 9000. 1089. 1234. 30
30.
 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.]
COUNT OF UNIQUE VALUES: 41


COLUMN: Place
UNIQUE VALUES: ['Chennai' 'Mumbai' 'Calcutta' 'Delhi' 'Podicherry' 'Cochin' nan 'Noi
da'
 'Hyderabad' 'Bhopal' 'Nagpur' 'Pune']
COUNT OF UNIQUE VALUES: 12


COLUMN: Country
UNIQUE VALUES: ['India']
COUNT OF UNIQUE VALUES: 1


COLUMN: Gender
UNIQUE VALUES: [0 1]
COUNT OF UNIQUE VALUES: 2
```

8. RENAMING COLUMN NAMES

In [88]:

```
# Rename columns to lowercase and replace spaces with underscores for consistency
data.columns = data.columns.str.replace(' ', '_').str.lower()
```

In [90]:
```
data
```

Out[90]:

	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 [92]:
```
# Rename the 'Country' column to 'Country_Name'
data = data.rename(columns={'country': 'country_name'})
data
```

Out[92]:

	company	age	salary	place	country_name	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

1. NULL / MISSING VALUES IN EACH COLUMN

In [94]:

```
# DISPLAY NULL VALUES IN EACH COLUMN
print("Null values in each column:")
print(data.isnull().sum())
```

```
Null values in each column:
company          8
age             18
salary          24
place           14
country_name     0
gender           0
dtype: int64
```

In [30]:

```
data
```

Out[30]:

	company	age	salary	place	country_name	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

1.1 HANDLING MISSING VALUES

In [96]:

```
# Replace the value 0 in the 'age' column with NaN
data['age'] = data['age'].replace(0, np.nan)
data
```

Out[96]:

	company	age	salary	place	country_name	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 [98]:

```python
# For numerical columns (Age, Salary), fill missing values with the median
data['age'] = data['age'].fillna(data['age'].median())
data['salary'] = data['salary'].fillna(data['salary'].median())
data
```

Out[98]:

	company	age	salary	place	country_name	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
...	...	...	...	...	...	...
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 [100…

```
# For categorical columns (Company, Place, Gender), fill missing values with the mo
data['company'] = data['company'].fillna(data['company'].mode()[0])
data['place'] = data['place'].fillna(data['place'].mode()[0])
data['gender'] = data['gender'].fillna(data['gender'].mode()[0])
data
```

Out[100…

	company	age	salary	place	country_name	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
...	...	...	...	...	...	...
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

2. DUPLICATE VALUES

In [102…

```
# FINDING THE TOTAL NO OF DUPLICATES
data.duplicated().sum()
```

Out[102…

```
4
```

In [104…

```
data.shape
```

Out[104…

```
(148, 6)
```

In [106…

```
# TO REMOVE DUPLICATES
data.drop_duplicates(inplace=True)
```

In [108…

```
data.shape
```

Out[108…

```
(144, 6)
```

2. FINDING OUTLIERS

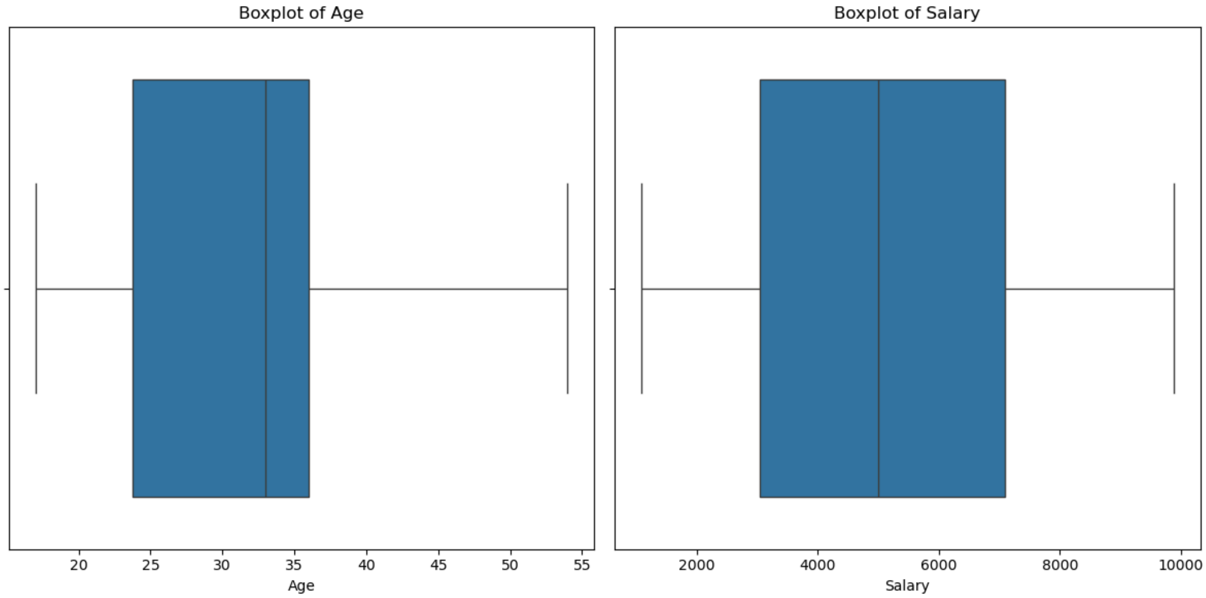
In [110…

```
# Create box plots to visualize the outliers for 'age' and 'salary'
plt.figure(figsize=(12, 6))

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

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

# Show the plots
plt.tight_layout()
plt.show()
```



In [112…

```
# 1.  Calculate Q1, Q3, and IQR for the 'age' column
Q1_age = data['age'].quantile(0.25)
Q3_age = data['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 = data[(data['age'] < lower_bound_age) | (data['age'] > upper_bound_ag

# 2. Calculate Q1, Q3, and IQR for the 'salary' column
Q1_salary = data['salary'].quantile(0.25)
Q3_salary = data['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 = data[(data['salary'] < lower_bound_salary) | (data['salary'] > up

# 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)
```

```
Outliers in Age:
Empty DataFrame
Columns: [company, age, salary, place, country_name, gender]
Index: []


Outliers in Salary:
Empty DataFrame
Columns: [company, age, salary, place, country_name, gender]
Index: []
```

Based on the results of the Interquartile Range (IQR) method for detecting outliers in the age and salary columns, no outliers were identified in the dataset, indicating that the values for both features fall within the expected range for the majority of the data.

DATA ANALYSIS

1. FILTER THE DATA WHERE AGE > 40 AND SALARY < 50000

In [114…

```
filtered_data = data[(data['age'] > 40) & (data['salary'] < 5000)]
filtered_data
```

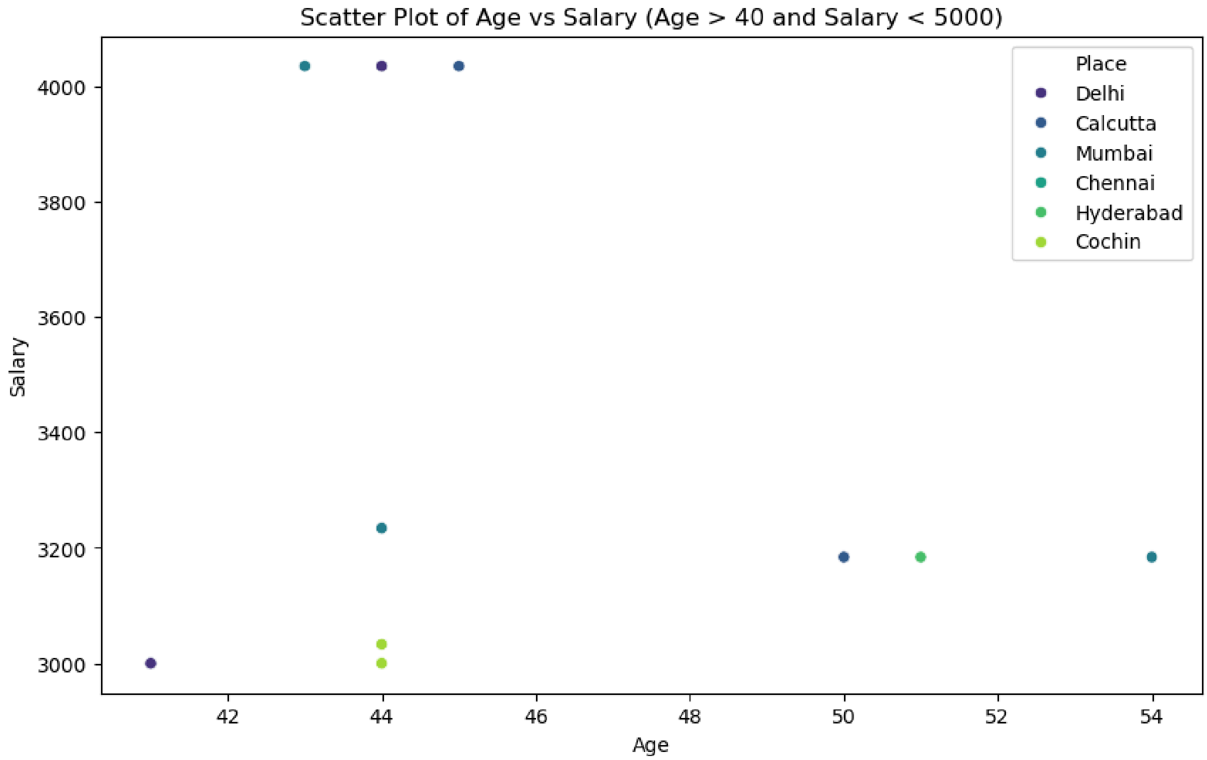
Out[114…

	company	age	salary	place	country_name	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

2. SCATTER PLOT TO VISUALIZE THE RELATIONSHIP BETWEEN AGE AND SALARY

In [116…

```
plt.figure(figsize=(10, 6))
sns.scatterplot(data=filtered_data, x='age', y='salary', hue='place', palette='viri
plt.title('Scatter Plot of Age vs Salary (Age > 40 and Salary < 5000)')
plt.xlabel('Age')
plt.ylabel('Salary')
plt.legend(title='Place')
plt.show()
```



3. COUNT THE NUMBER OF PEOPLE FROM EACH PLACE

In [118…

```
place_counts = data['place'].value_counts()
place_counts
```

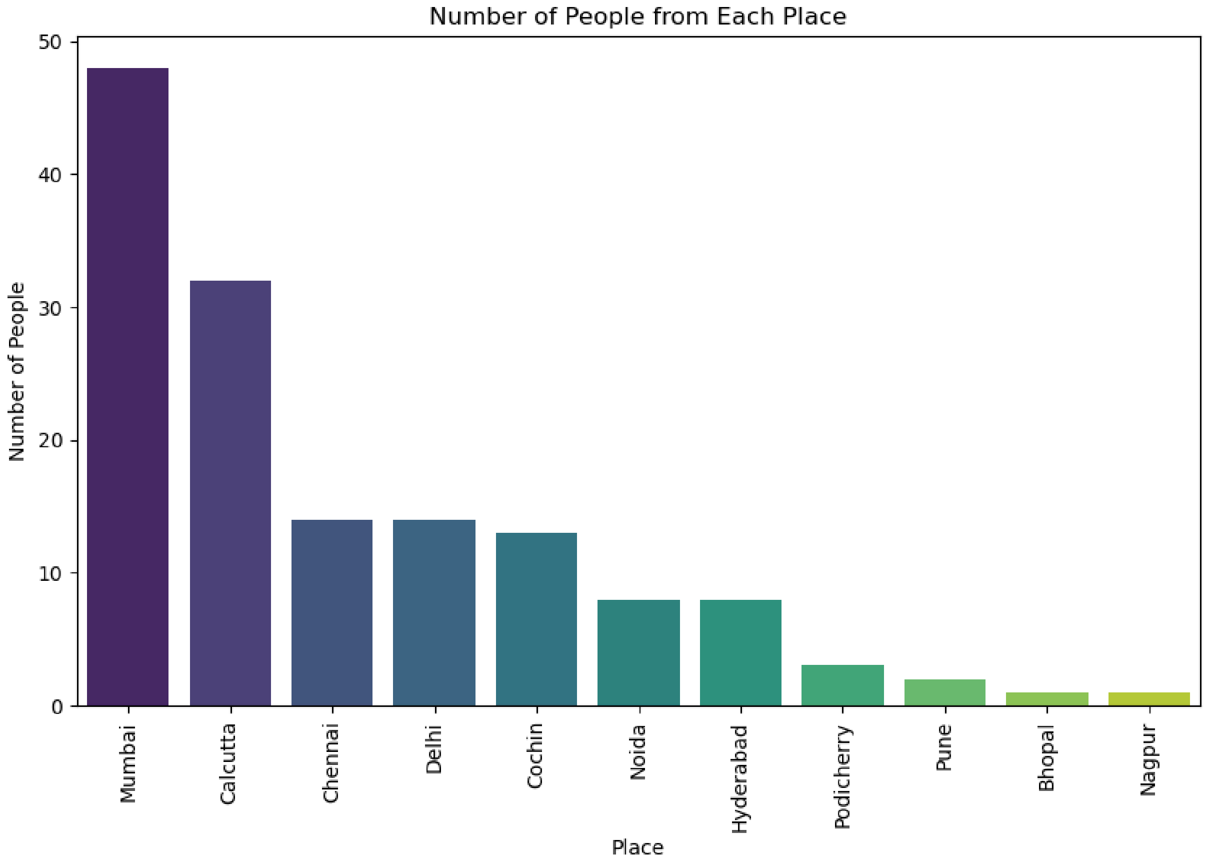
Out[118…

```
place
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
```

4. REPRESENT THE COUNT OF PEOPLE FROM EACH PLACE VISUALLY

In [120…

```
plt.figure(figsize=(10, 6))
sns.barplot(x=place_counts.index, y=place_counts.values, palette='viridis')
plt.title('Number of People from Each Place')
plt.xlabel('Place')
plt.ylabel('Number of People')
plt.xticks(rotation=90)
plt.show()
```



In [ ]:

DATA ENCODING

CONVERT CATEGORICAL VARIABLE INTO NUMERICAL USING DATA ENCODING

LABEL ENCODING

In [124…

```
from sklearn.preprocessing import LabelEncoder

# Identify categorical variables
categorical_columns = ['company', 'place', 'country_name', 'gender']

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

```
# Apply label encoding for 'gender'
data['gender'] = label_encoder.fit_transform(data['gender'])

# Display the first few rows of the encoded dataframe
print(data.head(20))
```

```
   company   age  salary       place country_name  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
5  Infosys  33.0  5000.0    Calcutta        India       0
6      TCS  33.0  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
10     CTS  23.0  5000.0      Mumbai        India       0
11     CTS  34.0  1089.0    Calcutta        India       0
12     CTS  45.0  5000.0     Chennai        India       0
13     CTS  18.0  1234.0      Mumbai        India       0
14 Infosys  40.0  3000.0    Calcutta        India       0
15     TCS  23.0  3000.0       Delhi        India       0
16 Infosys  23.0  3030.0  Podicherry        India       0
17     TCS  34.0  5000.0      Cochin        India       0
18     TCS  22.0  5000.0     Chennai        India       0
19 Infosys  32.0  5000.0      Mumbai        India       0
```

ONE HOT ENCODING

In [126…

```
from sklearn.preprocessing import OneHotEncoder

# Identify categorical variables
categorical_columns = ['company', 'place', 'country_name', 'gender']

# Apply One-Hot Encoding using pd.get_dummies for 'company', 'place', 'country' col
data_encoded = pd.get_dummies(data, columns=['company', 'place', 'country_name'], d

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

```
    age  salary  gender  company_Congnizant  company_Infosys  \
0  20.0  5000.0       0               False            False   
1  30.0  5000.0       0               False             True   
2  35.0  2300.0       0               False            False   
3  40.0  3000.0       0               False             True   
4  23.0  4000.0       0               False            False   

   company_Infosys Pvt Lmt  company_TCS  company_Tata Consultancy Services  \
0                    False         True                              False   
1                    False        False                              False   
2                    False         True                              False   
3                    False        False                              False   
4                    False         True                              False   

   place_Calcutta  place_Chennai  place_Cochin  place_Delhi  place_Hyderabad  \
0           False           True         False        False            False   
1           False          False         False        False            False   
2            True          False         False        False            False   
3           False          False         False         True            False   
4           False          False         False        False            False   

   place_Mumbai  place_Nagpur  place_Noida  place_Podicherry  place_Pune  
0         False         False        False             False       False  
1          True         False        False             False       False  
2         False         False        False             False       False  
3         False         False        False             False       False  
4          True         False        False             False       False  
```

FEATURE SCALING

APPLY STANDARD SCALER AND MINMAX SCALER FOR FEATURE SCALING TO NORMALIZE NUMERICAL DATA

In [136…

```python
from sklearn.preprocessing import StandardScaler, MinMaxScaler

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

# Select only the numerical columns (e.g., 'age', 'salary')
numerical_columns = ['age', 'salary']  # Replace with your actual numerical columns

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

# Apply MinMaxScaler to numerical columns
data_minmax_scaled = data.copy()
```

```
data_minmax_scaled[numerical_columns] = minmax_scaler.fit_transform(data_minmax_sca

# Display the first few rows of the scaled data
print("Data after Standard Scaling:")
print(data_standard_scaled.head(10))

print("\nData after Min-Max Scaling:")
print(data_minmax_scaled.head(10))
```

```
Data after Standard Scaling:
   company       age    salary     place country_name  gender
0      TCS -1.484676 -0.100827   Chennai        India       0
1  Infosys -0.267174 -0.100827    Mumbai        India       0
2      TCS  0.341577 -1.243735  Calcutta        India       0
3  Infosys  0.950328 -0.947426     Delhi        India       0
4      TCS -1.119426 -0.524127    Mumbai        India       0
5  Infosys  0.098077 -0.100827  Calcutta        India       0
6      TCS  0.098077  0.322472   Chennai        India       1
7  Infosys -1.119426  0.745771    Mumbai        India       1
8      TCS  0.219827  1.169070  Calcutta        India       1
9      CTS  1.559079  1.592369     Delhi        India       0

Data after Min-Max Scaling:
   company       age    salary     place country_name  gender
0      TCS  0.081081  0.445089   Chennai        India       0
1  Infosys  0.351351  0.445089    Mumbai        India       0
2      TCS  0.486486  0.137817  Calcutta        India       0
3  Infosys  0.621622  0.217480     Delhi        India       0
4      TCS  0.162162  0.331285    Mumbai        India       0
5  Infosys  0.432432  0.445089  Calcutta        India       0
6      TCS  0.432432  0.558894   Chennai        India       1
7  Infosys  0.162162  0.672698    Mumbai        India       1
8      TCS  0.459459  0.786503  Calcutta        India       1
9      CTS  0.756757  0.900307     Delhi        India       0
```