Name	**Sujal Dingankar**
UID no.	**2024301005**
Experiment No.	**6**

AIM:	**Feature Engineering (Data Cleaning)**
Program 1	
PROBLEM STATEMENT :	Perform required data cleaning and feature engineering. 1. Handle missing values from following data set csv file. Use all applicable methods for a feature and check changes in Mean, Median, variance and standard deviation, before selecting the best one. 2. Scale the data. Use all applicable methods for a feature and check changes in Mean, Median, variance and standard deviation, before selecting the best one. 3. Handle outlier using IQR. 4. Fix Structural inaccuracies if any. And remove duplicates if needed. 5. Draw 3 graphs based on your expertise and derive insights from the same

PROGRAM:	import pandas as pd
	import numpy as np
	import matplotlib.pyplot as plt
	import seaborn as sns
	from sklearn.impute import SimpleImputer
	from sklearn.preprocessing import StandardScaler, MinMaxScaler, RobustScaler
	
	data = pd.read_csv('data_science_job.csv')
	
	def compare_imputation(data, col):
	  orig_stats = data[col].describe()
	  print(f"Original stats for {col}:\n{orig_stats}\n")
	  med_data = data.copy()
	  med_data[col] = med_data[col].fillna(med_data[col].median())
	  med_stats = med_data[col].describe()

```
  print(f"Median Imputation stats for {col}:\n{med_stats}\n")
  mean_data = data.copy()
  mean_data[col] = mean_data[col].fillna(mean_data[col].mean())
  mean_stats = mean_data[col].describe()
  print(f"Mean Imputation stats for {col}:\n{mean_stats}\n")
  mode_data = data.copy()
  mode_val = mode_data[col].mode()[0]
  mode_data[col] = mode_data[col].fillna(mode_val)
  mode_stats = mode_data[col].describe()
  print(f"Mode Imputation stats for {col}:\n{mode_stats}\n")
  rand_data = data.copy()
  non_nulls = rand_data[col].dropna().to_numpy()
  rand_data[col] = rand_data[col].apply(lambda x:
  np.random.choice(non_nulls) if pd.isnull(x) else x)
  rand_stats = rand_data[col].describe()
  print(f"Random Imputation stats for {col}:\n{rand_stats}\n")
  print("**********************************************")

for col in data.select_dtypes(include=['number']).columns:
  if data[col].isnull().sum() > 0:
    compare_imputation(data, col)
    string_columns = ['gender', 'relevent_experience', 'enrolled_university',
    'education_level', 'major_discipline', 'company_type']
for string_col in string_columns:
  if string_col in data.columns and data[string_col].isnull().sum() > 0:
    mode_val = data[string_col].mode()[0]
    data[string_col] = data[string_col].fillna(mode_val)
    print(f"Replaced missing values in '{string_col}' with mode:
{mode_val}\n")



df_numeric = data.select_dtypes(include=['float64', 'int64'])
# 2. Scale Data
scaler_standard = StandardScaler()
scaler_minmax = MinMaxScaler()
scaler_robust = RobustScaler()
# Apply scaling to the numerical columns
df_standard_scaled =
pd.DataFrame(scaler_standard.fit_transform(df_numeric),
columns=df_numeric.columns)
```

```
df_minmax_scaled =
pd.DataFrame(scaler_minmax.fit_transform(df_numeric),
columns=df_numeric.columns)
df_robust_scaled = pd.DataFrame(scaler_robust.fit_transform(df_numeric),
columns=df_numeric.columns)
# Compare scaling effects on Mean, Median, Variance, Standard Deviation
print("\nStandard Scaled Data Statistics:")
display(df_standard_scaled.describe())
print("\nMinMax Scaled Data Statistics:")
display(df_minmax_scaled.describe())
print("\nRobust Scaled Data Statistics:")
display(df_robust_scaled.describe())



# 3. Handle Outliers using IQR
Q1 = df_numeric.quantile(0.25)
Q3 = df_numeric.quantile(0.75)
IQR = Q3 - Q1

df_no_outliers = df_numeric[~((df_numeric < (Q1 - 1.5 * IQR)) |
(df_numeric > (Q3 + 1.5 * IQR))).any(axis=1)]
print(f"Shape before outlier removal: {df_numeric.shape}")
print(f"Shape after outlier removal: {df_no_outliers.shape}")

# 4. Fix Structural Inaccuracies and Remove Duplicates
data.drop_duplicates(inplace=True)

#5. Drawing Graphs
# 1. Histogram
plt.hist(data['experience'], bins=30, color='green', edgecolor='black')
plt.title('Experience')
plt.xlabel('Years')
plt.ylabel('Frequency')
plt.show()

# 2. Boxplot to visualize outliers
plt.figure(figsize=(12, 6))
sns.boxplot(df_no_outliers['experience'])
plt.title('Boxplot after Outlier Removal')
plt.show()
```

```
# 3. Bar Chart
value_counts = data['training_hours'].value_counts()
plt.bar(value_counts.index, value_counts.values)
plt.title('training_hours')
plt.xlabel('training_hours')
plt.ylabel('Frequency')
plt.xticks(rotation=45, ha='right')
```

RESULT:

```
Original stats for city_development_index:
count    18679.000000
mean         0.828951
std          0.123334
min          0.448000
25%          0.740000
50%          0.903000
75%          0.920000
max          0.949000
Name: city_development_index, dtype: float64

Median Imputation stats for city_development_index:
count    19158.000000
mean         0.830802
std          0.122330
min          0.448000
25%          0.743000
50%          0.903000
75%          0.920000
max          0.949000
Name: city_development_index, dtype: float64

Mean Imputation stats for city_development_index:
count    19158.000000
mean         0.828951
std          0.121783
min          0.448000
25%          0.743000
50%          0.899000
75%          0.920000
max          0.949000
Name: city_development_index, dtype: float64
```

```
Mode Imputation stats for city_development_index:
count    19158.000000
mean         0.831227
std          0.122610
min          0.448000
25%          0.743000
50%          0.910000
75%          0.920000
max          0.949000
Name: city_development_index, dtype: float64

Random Imputation stats for city_development_index:
count    19158.000000
mean         0.828800
std          0.123414
min          0.448000
25%          0.740000
50%          0.903000
75%          0.920000
max          0.949000
Name: city_development_index, dtype: float64

****************************************
```

```
Original stats for experience:
count    19093.000000
mean         9.928036
std          6.505268
min          0.000000
25%          4.000000
50%          9.000000
75%         16.000000
max         20.000000
Name: experience, dtype: float64

Median Imputation stats for experience:
count    19158.000000
mean         9.924888
std          6.494447
min          0.000000
25%          4.000000
50%          9.000000
75%         16.000000
max         20.000000
Name: experience, dtype: float64

Mean Imputation stats for experience:
count    19158.000000
mean         9.928036
std          6.494223
min          0.000000
25%          4.000000
50%          9.000000
75%         16.000000
max         20.000000
Name: experience, dtype: float64
```

```
Mode Imputation stats for experience:
count    19158.000000
mean         9.962209
std          6.520580
min          0.000000
25%          4.000000
50%          9.000000
75%         16.000000
max         20.000000
Name: experience, dtype: float64

Random Imputation stats for experience:
count    19158.000000
mean         9.926088
std          6.502971
min          0.000000
25%          4.000000
50%          9.000000
75%         16.000000
max         20.000000
Name: experience, dtype: float64

****************************************
Original stats for training_hours:
count    18392.000000
mean        65.185787
std         59.885626
min          1.000000
25%         23.000000
50%         47.000000
75%         88.000000
max        336.000000
Name: training_hours, dtype: float64
```

```
Median Imputation stats for training_hours:
count    19158.000000
mean        64.458660
std         58.784219
min          1.000000
25%         24.000000
50%         47.000000
75%         86.000000
max        336.000000
Name: training_hours, dtype: float64

Mean Imputation stats for training_hours:
count    19158.000000
mean        65.185787
std         58.676137
min          1.000000
25%         24.000000
50%         50.000000
75%         86.000000
max        336.000000
Name: training_hours, dtype: float64

Mode Imputation stats for training_hours:
count    19158.000000
mean        63.698977
std         59.126724
min          1.000000
25%         24.000000
50%         45.000000
75%         86.000000
max        336.000000
Name: training_hours, dtype: float64
```

```
Random Imputation stats for training_hours:
count    19158.000000
mean        65.138428
std         59.822879
min          1.000000
25%         23.000000
50%         47.000000
75%         88.000000
max        336.000000
Name: training_hours, dtype: float64

******************************************
Replaced missing values in 'gender' with mode: Male

Replaced missing values in 'enrolled_university' with mode: no_enrollment

Replaced missing values in 'education_level' with mode: Graduate

Replaced missing values in 'major_discipline' with mode: STEM

Replaced missing values in 'company_type' with mode: Pvt Ltd
```

Standard Scaled Data Statistics:

	enrollee_id	city_development_index	experience	training_hours	target
count	1.915800e+04	1.867900e+04	1.909300e+04	1.839200e+04	1.915800e+04
mean	-8.530370e-17	-1.734608e-16	5.210076e-18	4.742231e-17	-1.159018e-16
std	1.000026e+00	1.000027e+00	1.000026e+00	1.000027e+00	1.000026e+00
min	-1.754813e+00	-3.088847e+00	-1.526193e+00	-1.071835e+00	-5.763457e-01
25%	-8.653361e-01	-7.212366e-01	-9.112909e-01	-7.044584e-01	-5.763457e-01
50%	1.114199e-02	6.004089e-01	-1.426629e-01	-3.036836e-01	-5.763457e-01
75%	8.625578e-01	7.382493e-01	9.334161e-01	3.809734e-01	-5.763457e-01
max	1.716365e+00	9.733886e-01	1.548318e+00	4.522314e+00	1.735070e+00

MinMax Scaled Data Statistics:

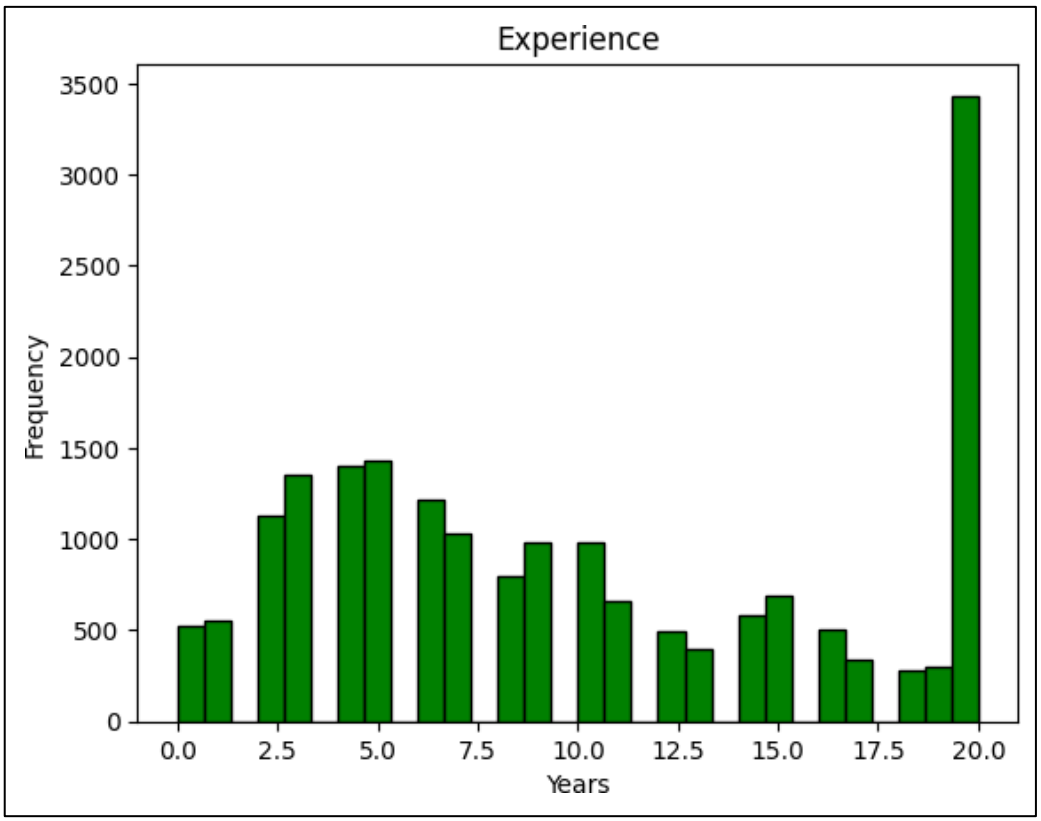
	enrollee_id	city_development_index	experience	training_hours	target
count	19158.000000	18679.000000	19093.000000	18392.000000	19158.000000
mean	0.505538	0.760381	0.496402	0.191599	0.249348
std	0.288094	0.246176	0.325263	0.178763	0.432647
min	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.256246	0.582834	0.200000	0.065672	0.000000
50%	0.508748	0.908184	0.450000	0.137313	0.000000
75%	0.754029	0.942116	0.800000	0.259701	0.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000

Robust Scaled Data Statistics:

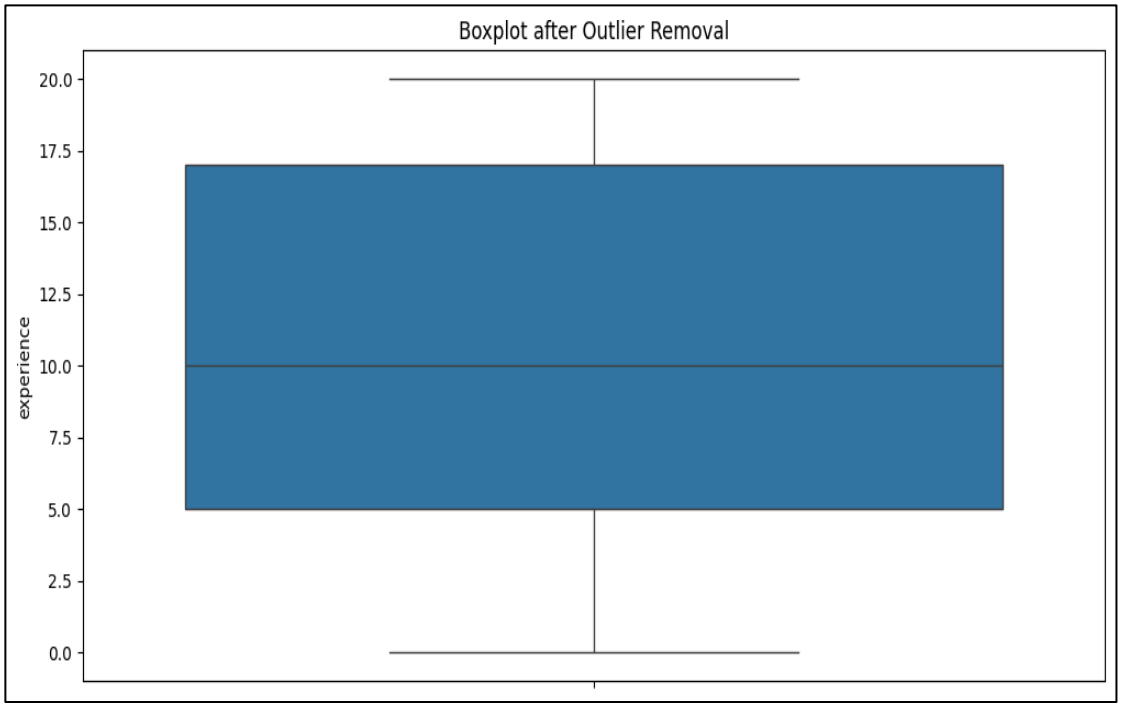
	enrollee_id	city_development_index	experience	training_hours	target
count	19158.000000	18679.000000	19093.000000	18392.000000	19158.000000
mean	-0.006448	-0.411384	0.077336	0.279781	0.249348
std	0.578754	0.685191	0.542106	0.921317	0.432647
min	-1.022028	-2.527778	-0.750000	-0.707692	0.000000
25%	-0.507252	-0.905556	-0.416667	-0.369231	0.000000
50%	0.000000	0.000000	0.000000	0.000000	0.000000
75%	0.492748	0.094444	0.583333	0.630769	0.000000
max	0.986880	0.255556	0.916667	4.446154	1.000000

Shape before outlier removal: (19158, 5)
Shape after outlier removal: (13639, 5)

Experience

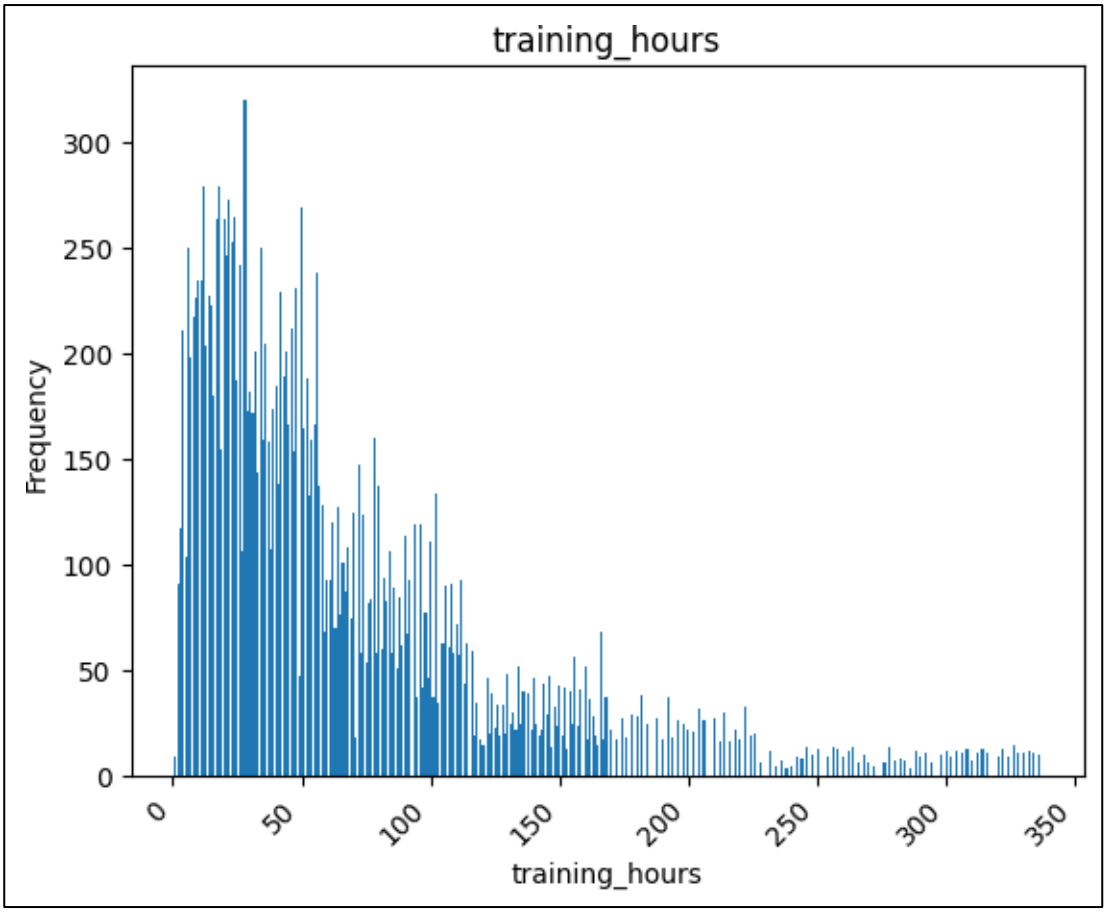


Above graph suggests that most individuals have around 20 years of experience, as it has the highest frequency in that category.



The boxplot shows that the median experience is around 10 years and the data is

symmetrically distributed between 5 and 15 years of experience.



Majority of people have training hours below 100, and frequency is decreasing with increase in hours.

CONCLUSION:	In this experiment, I was able to perform mean, median, mode imputations on data and compare them with original values. I learnt to the concept of scaling data, remove outliers and draw inferences from given data with the help of graphs.