Experiment 7

Name: Hatim Sawai
Uid: 2021300108
Batch: C

Load the dataset

In [ ]:
```
import pandas as pd
import numpy as np
# read csv file
df = pd.read_csv("employee.csv", sep=",")
df.head()
```

Out[ ]:

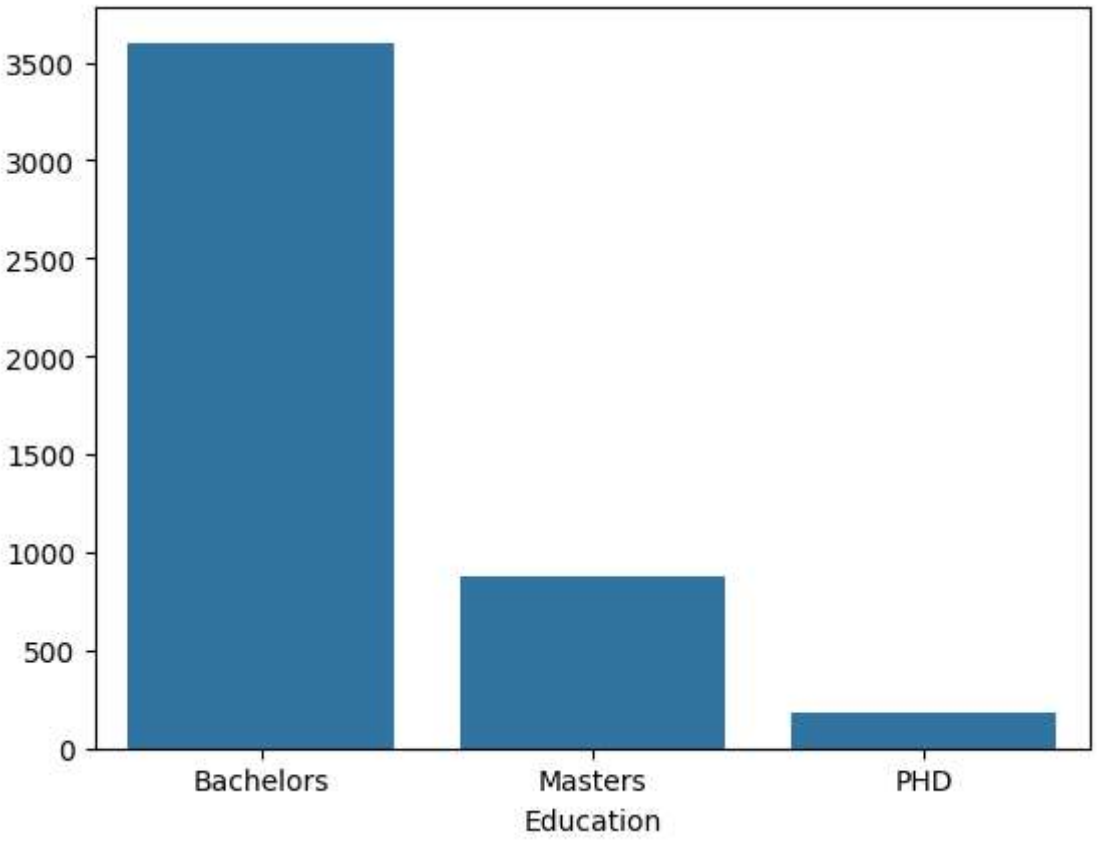
	Education	JoiningYear	City	PaymentTier	Age	Gender	EverBenched	Experience
0	Bachelors	2017	Bangalore	3	34	Male	No	
1	Bachelors	2013	Pune	1	28	Female	No	
2	Bachelors	2014	New Delhi	3	38	Female	No	
3	Masters	2016	Bangalore	3	27	Male	No	
4	Masters	2017	Pune	3	24	Male	Yes	

Research Questions

Q.1) What is the distribution of educational qualifications among employees?
ans:
1) Bachelor's - 77.39%
2) Master's - 18.76%
3) PHD - 3.84%

In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns
education_counts = df['Education'].value_counts()
sns.barplot(x=education_counts.index, y=education_counts.values)
```

Out[ ]:
```
<Axes: xlabel='Education'>
```



Q.2) How does the length of service (Joining Year) vary across different cities?

ans:

1) Pune - 2015

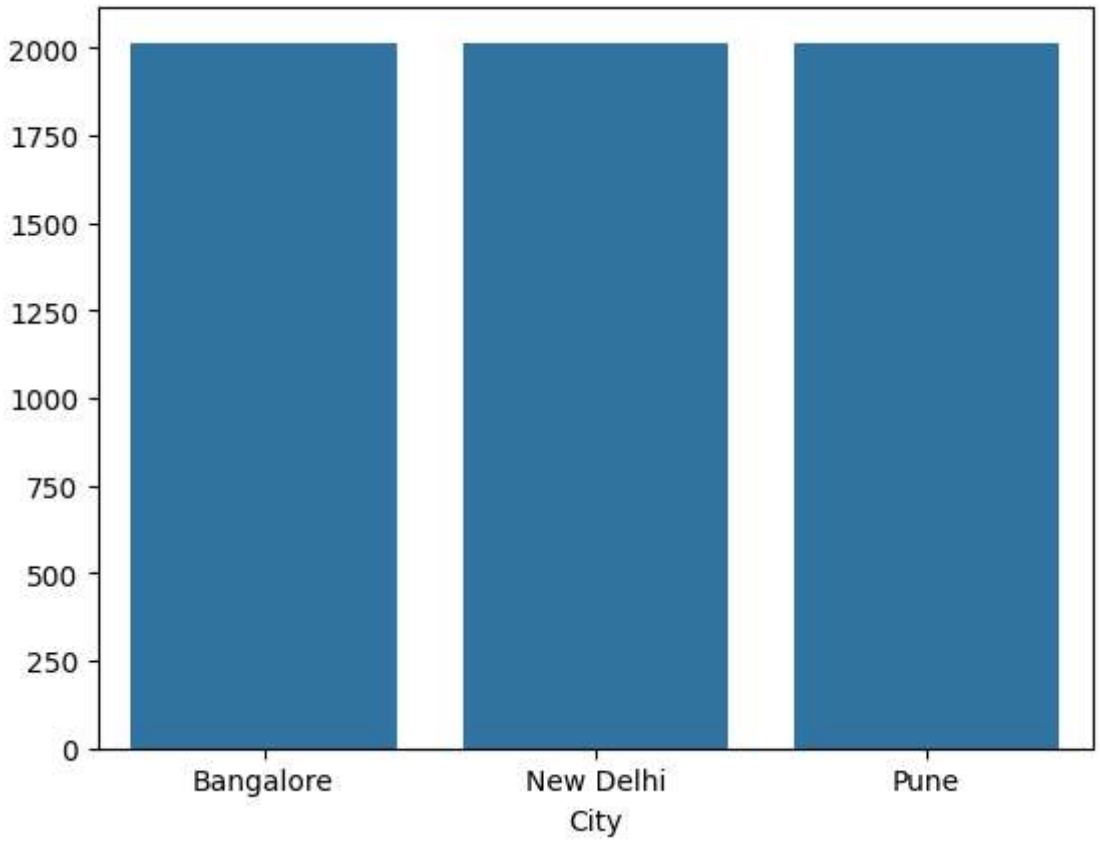
2) Bangalore - 2014

3) Delhi - 2015

In [ ]:
```
# Calculate the average joining year for each city
city_avg_joining_year = df.groupby('City')['JoiningYear'].mean()
print(city_avg_joining_year.values.tolist())
sns.barplot(x=city_avg_joining_year.index, y=city_avg_joining_year.values)
```

```
[2014.8595152603232, 2015.522039757995, 2015.0015772870663]
```

Out[ ]: <Axes: xlabel='City'>

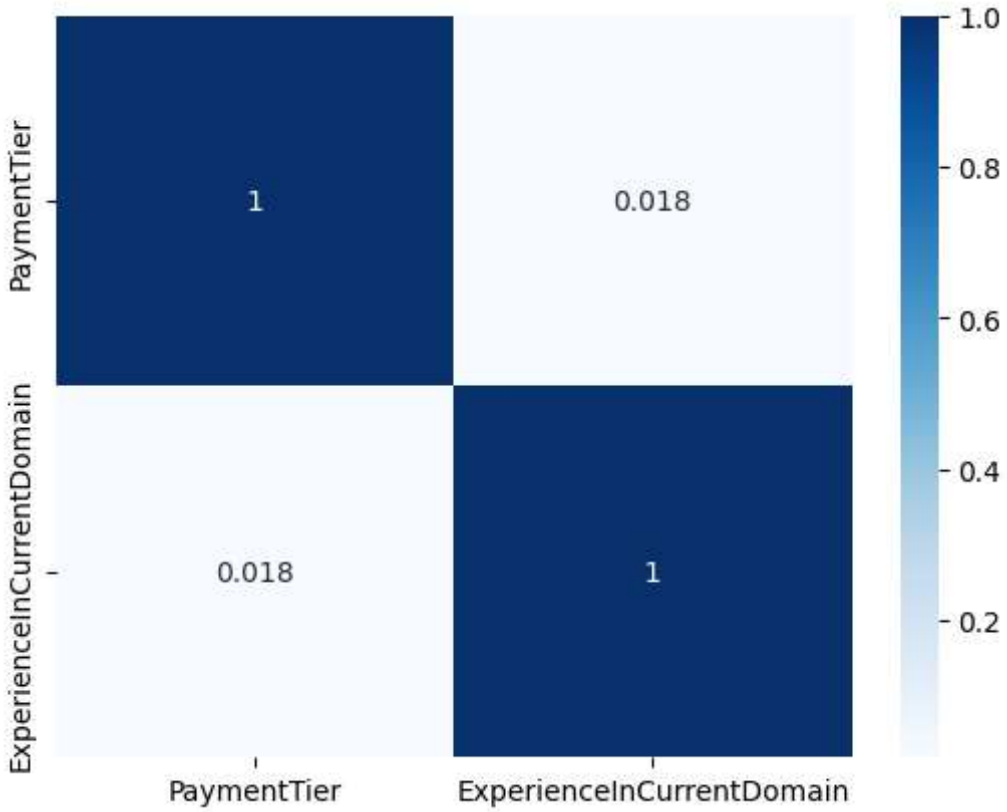


Q.3) Is there a correlation between Payment Tier and Experience in Current Domain?
ans: Yes, there is a correlation between Payment Tier and Experience in Current Domain i.e 0.018

In [ ]:
```
# Calculate Pearson's correlation coefficient
correlation = df['PaymentTier'].corr(df['ExperienceInCurrentDomain'])
print(correlation)
sns.heatmap(df[['PaymentTier', 'ExperienceInCurrentDomain']].corr(), annot=True, cm
```

```
0.018314321387224046
```

Out[ ]: <Axes: >



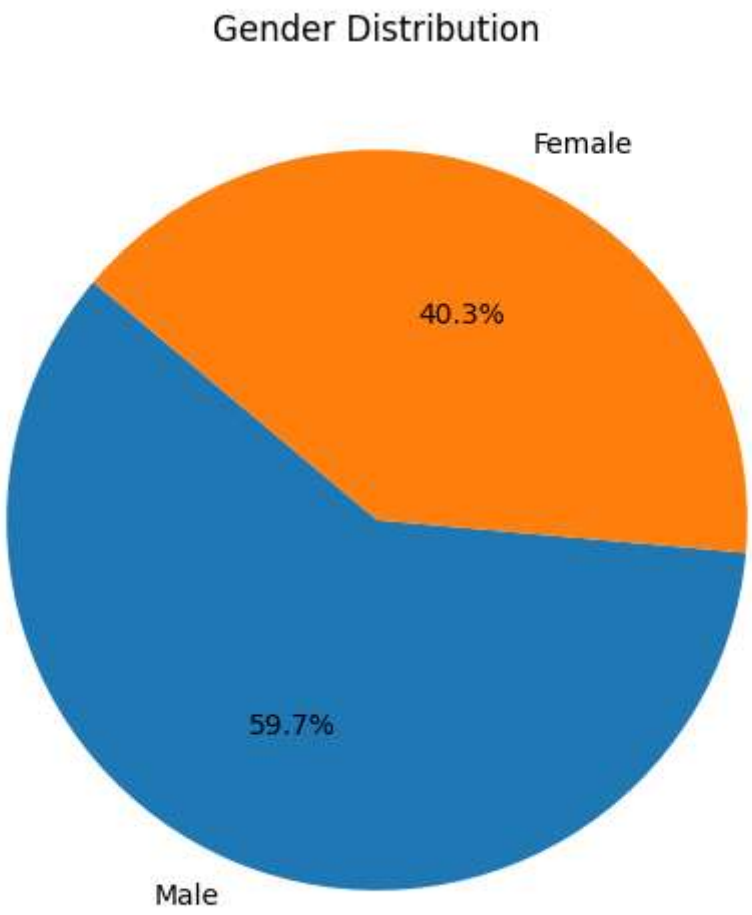
Q.4) What is the gender distribution within the workforce?

ans:

1) Female: 40.3%

2) Male: 59.7%

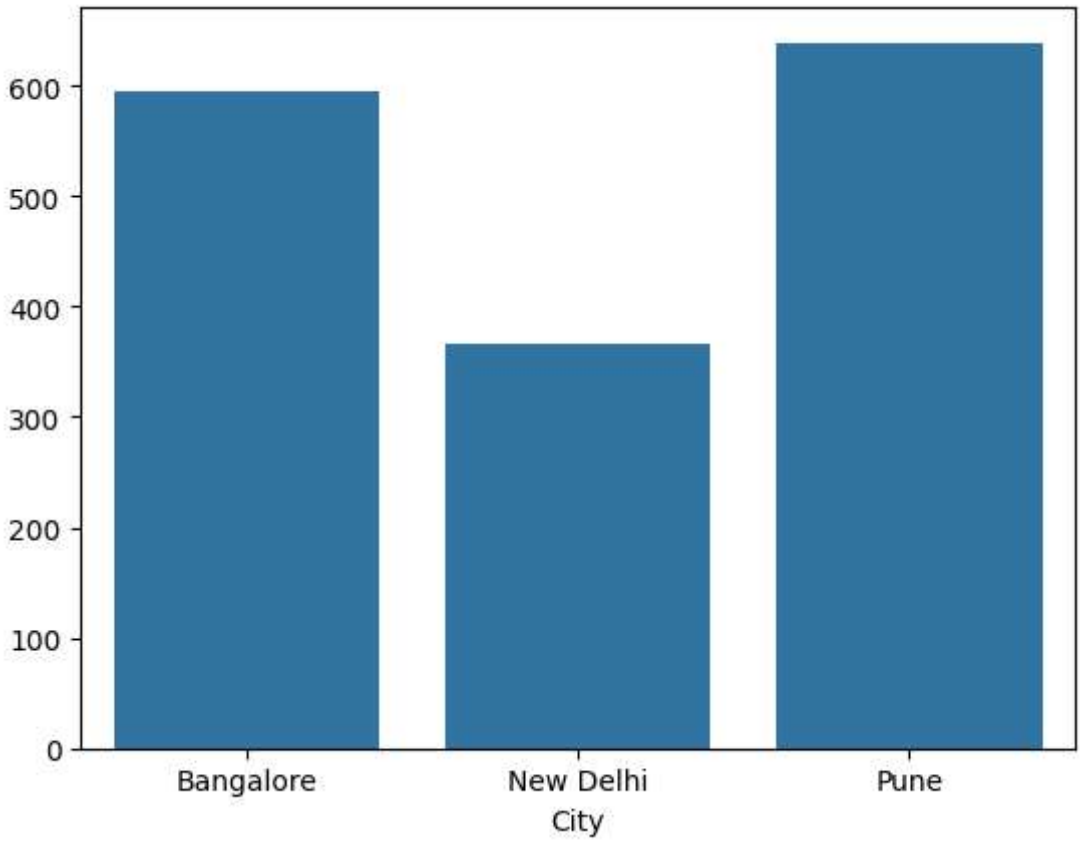
In [ ]:
```
# Count the frequency of each gender
gender_counts = df['Gender'].value_counts()
plt.figure(figsize=(6, 6))
plt.pie(gender_counts, labels=gender_counts.index, autopct='%1.1f%%', startangle=14
plt.title('Gender Distribution')
plt.show()
```



Q.5) Are there any patterns in leave-taking behavior among employees?
ans: Yes, there are patterns in leave-taking behavior among employees.

In [ ]:
```
# show pattern in leave-taking behavior among employees
# show leaves taken per city
city_leave_counts = df.groupby('City')['LeaveOrNot'].sum()
sns.barplot(x=city_leave_counts.index, y=city_leave_counts.values)
```

Out[ ]:
```
<Axes: xlabel='City'>
```



Pre-Processing of Data

In [ ]:
```
print(df.Education.unique())
print(df.City.unique())
print(df.JoiningYear.unique())
```

```
['Bachelors' 'Masters' 'PHD']
['Bangalore' 'Pune' 'New Delhi']
[2017 2013 2014 2016 2015 2012 2018]
```

In [ ]:
```
df_string = df.select_dtypes(include=['object'])
df_string = df_string.apply(lambda x: pd.factorize(x)[0])
df[df_string.columns] = df_string
df.head()
```

Out[ ]:

	Education	JoiningYear	City	PaymentTier	Age	Gender	EverBenched	ExperienceInCur
0	0	2017	0	3	34	0	0	
1	0	2013	1	1	28	1	0	
2	0	2014	2	3	38	1	0	
3	1	2016	0	3	27	0	0	
4	1	2017	1	3	24	0	1	

In [ ]:
```
df.describe()
```

Out[ ]:

	Education	JoiningYear	City	PaymentTier	Age	Gender	EverE
count	4653.00000	4653.000000	4653.000000	4653.000000	4653.000000	4653.000000	4653
mean	0.26456	2015.062970	0.769826	2.698259	29.393295	0.402966	0
std	0.52112	1.863377	0.821372	0.561435	4.826087	0.490547	0
min	0.00000	2012.000000	0.000000	1.000000	22.000000	0.000000	0
25%	0.00000	2013.000000	0.000000	3.000000	26.000000	0.000000	0
50%	0.00000	2015.000000	1.000000	3.000000	28.000000	0.000000	0
75%	0.00000	2017.000000	1.000000	3.000000	32.000000	1.000000	0
max	2.00000	2018.000000	2.000000	3.000000	41.000000	1.000000	1

In [ ]:
```
df.to_csv("employee-processed.csv", index=False)
```

In [ ]:
```
new_df = pd.read_csv("employee-processed.csv", sep=",")
new_df.head()
```

Out[ ]:

	Education	JoiningYear	City	PaymentTier	Age	Gender	EverBenched	ExperienceInCur
0	0	2017	0	3	34	0	0	
1	0	2013	1	1	28	1	0	
2	0	2014	2	3	38	1	0	
3	1	2016	0	3	27	0	0	
4	1	2017	1	3	24	0	1	

Comparision of different models

In [ ]:
```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, confusion_matrix
import os
```

In [ ]:
```
data = pd.read_csv("employee-processed.csv", sep=",")
data = pd.get_dummies(data, drop_first=True)
data = data.dropna()
```

In [ ]:
```
X = data.drop("LeaveOrNot", axis=1)  # Features
y = data["LeaveOrNot"]  # Target variable
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_sta
```

In [ ]:
```
naive_bayes = GaussianNB()
decision_tree = DecisionTreeClassifier()
random_forest = RandomForestClassifier()
```

In [ ]:
```
naive_bayes.fit(X_train, y_train)
decision_tree.fit(X_train, y_train)
random_forest.fit(X_train, y_train)
```

Out[ ]:
```
▾ RandomForestClassifier

RandomForestClassifier()
```

In [ ]:
```
def evaluate_model(model, X_test, y_test):
    y_pred = model.predict(X_test)
    acc = accuracy_score(y_test, y_pred)
    cm = confusion_matrix(y_test, y_pred)
    return acc, cm

nb_accuracy, nb_confusion_matrix = evaluate_model(naive_bayes, X_test, y_test)
dt_accuracy, dt_confusion_matrix = evaluate_model(decision_tree, X_test, y_test)
rf_accuracy, rf_confusion_matrix = evaluate_model(random_forest, X_test, y_test)
```

In [ ]:
```
print("Naive Bayes Accuracy:", nb_accuracy)
print("Naive Bayes Confusion Matrix:\n", nb_confusion_matrix)
print("Decision Tree Accuracy:", dt_accuracy)
print("Decision Tree Confusion Matrix:\n", dt_confusion_matrix)
print("Random Forest Accuracy:", rf_accuracy)
print("Random Forest Confusion Matrix:\n", rf_confusion_matrix)
```

```
Naive Bayes Accuracy: 0.6659505907626209
Naive Bayes Confusion Matrix:
 [[482 128]
 [183 138]]
Decision Tree Accuracy: 0.832438238453276
Decision Tree Confusion Matrix:
 [[541  69]
 [ 87 234]]
Random Forest Accuracy: 0.8506981740064447
Random Forest Confusion Matrix:
 [[559  51]
 [ 88 233]]
```