Experiment 7

November 9, 2023

Name: Adwait Purao UID: 2021300101 Batch: B2

[ ]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

[ ]:
```
df = pd.read_csv("/content/drive/MyDrive/Colab Notebooks/Employee.csv")
```

[ ]:
```
df.head()
```

[ ]:
```
   Education  JoiningYear       City  PaymentTier  Age  Gender EverBenched  \
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

   ExperienceInCurrentDomain  LeaveOrNot
0                          0           0
1                          3           1
2                          2           0
3                          5           1
4                          2           1
```

[ ]:
```
missing_data = df.isna()
missing_counts = missing_data.sum()
print(missing_counts)
```

```
Education                    0
JoiningYear                  0
City                         0
PaymentTier                  0
Age                          0
```

```
Gender                       0
EverBenched                  0
ExperienceInCurrentDomain    0
LeaveOrNot                   0
dtype: int64
```

[ ]:

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
df['Education'] = label_encoder.fit_transform(df['Education'])
df.head()
```

[ ]:

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

   ExperienceInCurrentDomain  LeaveOrNot
0                          0           0
1                          3           1
2                          2           0
3                          5           1
4                          2           1
```

[ ]:

```
df['City'] = label_encoder.fit_transform(df['City'])
df.head()
```

[ ]:

```
   Education  JoiningYear  City  PaymentTier  Age  Gender EverBenched  \
0          0         2017     0            3   34    Male          No
1          0         2013     2            1   28  Female          No
2          0         2014     1            3   38  Female          No
3          1         2016     0            3   27    Male          No
4          1         2017     2            3   24    Male         Yes

   ExperienceInCurrentDomain  LeaveOrNot
0                          0           0
1                          3           1
2                          2           0
3                          5           1
4                          2           1
```

[ ]:

```
df['Gender'] = label_encoder.fit_transform(df['Gender'])
df.head()
```

[ ]:

```
   Education  JoiningYear  City  PaymentTier  Age  Gender EverBenched  \
0          0         2017     0            3   34       1          No
```

```
1          0         2013     2            1   28       0           No   
2          0         2014     1            3   38       0           No   
3          1         2016     0            3   27       1           No   
4          1         2017     2            3   24       1          Yes   

   ExperienceInCurrentDomain  LeaveOrNot  
0                          0           0  
1                          3           1  
2                          2           0  
3                          5           1  
4                          2           1  
```

```
[ ]: df[’EverBenched’] = label_encoder.fit_transform(df[’EverBenched’])
     df.head()
```

```
[ ]:    Education  JoiningYear  City  PaymentTier  Age  Gender  EverBenched  \
     0          0         2017     0            3   34       1            0   
     1          0         2013     2            1   28       0            0   
     2          0         2014     1            3   38       0            0   
     3          1         2016     0            3   27       1            0   
     4          1         2017     2            3   24       1            1   

        ExperienceInCurrentDomain  LeaveOrNot  
     0                          0           0  
     1                          3           1  
     2                          2           0  
     3                          5           1  
     4                          2           1  
```

```
[ ]: from sklearn.model_selection import train_test_split
```

```
[ ]: X = df.drop(’LeaveOrNot’,axis=1)
     y = df[’LeaveOrNot’]
     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4)
```

```
[ ]: from sklearn.metrics import classification_report,confusion_matrix
```

```
[ ]: from sklearn.naive_bayes import GaussianNB
```

```
[ ]: nb_classifier = GaussianNB()
     nb_classifier.fit(X_train, y_train)
     y_naive_bayes = nb_classifier.predict(X_test)
```

```
[ ]: print(”The confusion matrix for Naive Bayes is : ”)
     print(””)
     print(confusion_matrix(y_test,y_naive_bayes))
```

```
The confusion matrix for Naive Bayes is :

[[978 238]
 [374 272]]
```

[ ]:

```
print("The classification report for Naive Bayes is : ")
print("")
print(classification_report(y_test,y_naive_bayes))
```

```
The classification report for Naive Bayes is :

              precision    recall  f1-score   support

           0       0.72      0.80      0.76      1216
           1       0.53      0.42      0.47       646

    accuracy                           0.67      1862
   macro avg       0.63      0.61      0.62      1862
weighted avg       0.66      0.67      0.66      1862
```

[ ]:

```
from sklearn.tree import DecisionTreeClassifier
```

[ ]:

```
dtree = DecisionTreeClassifier()
dtree.fit(X_train,y_train)
y_decision_tree = dtree.predict(X_test)
```

[ ]:

```
print("The confusion matrix for Decision Tree is : ")
print("")
print(confusion_matrix(y_test,y_decision_tree))
```

```
The confusion matrix for Decision Tree is :

[[1060  156]
 [ 203  443]]
```

[ ]:

```
print("The classification report for Decision Tree is : ")
print("")
print(classification_report(y_test,y_decision_tree))
```

```
The classification report for Decision Tree is :

              precision    recall  f1-score   support

           0       0.84      0.87      0.86      1216
           1       0.74      0.69      0.71       646

    accuracy                           0.81      1862
```

```
   macro avg       0.79      0.78      0.78      1862
weighted avg       0.80      0.81      0.81      1862
```

```
[ ]: from sklearn.ensemble import RandomForestClassifier
```

```
[ ]: rfc = RandomForestClassifier(n_estimators=1000)
     rfc.fit(X_train,y_train)
     y_random_forest = rfc.predict(X_test)
```

```
[ ]: print("The confusion matrix for Random Forest is : ")
     print("")
     print(confusion_matrix(y_test,y_random_forest))
```

```
The confusion matrix for Random Forest is :

[[1103  113]
 [ 209  437]]
```

```
[ ]: print("The classification report for Decision Tree is : ")
     print("")
     print(classification_report(y_test,y_random_forest))
```

```
The classification report for Decision Tree is :

              precision    recall  f1-score   support

           0       0.84      0.91      0.87      1216
           1       0.79      0.68      0.73       646

    accuracy                           0.83      1862
   macro avg       0.82      0.79      0.80      1862
weighted avg       0.82      0.83      0.82      1862
```

```
[ ]: from sklearn.metrics import accuracy_score
```

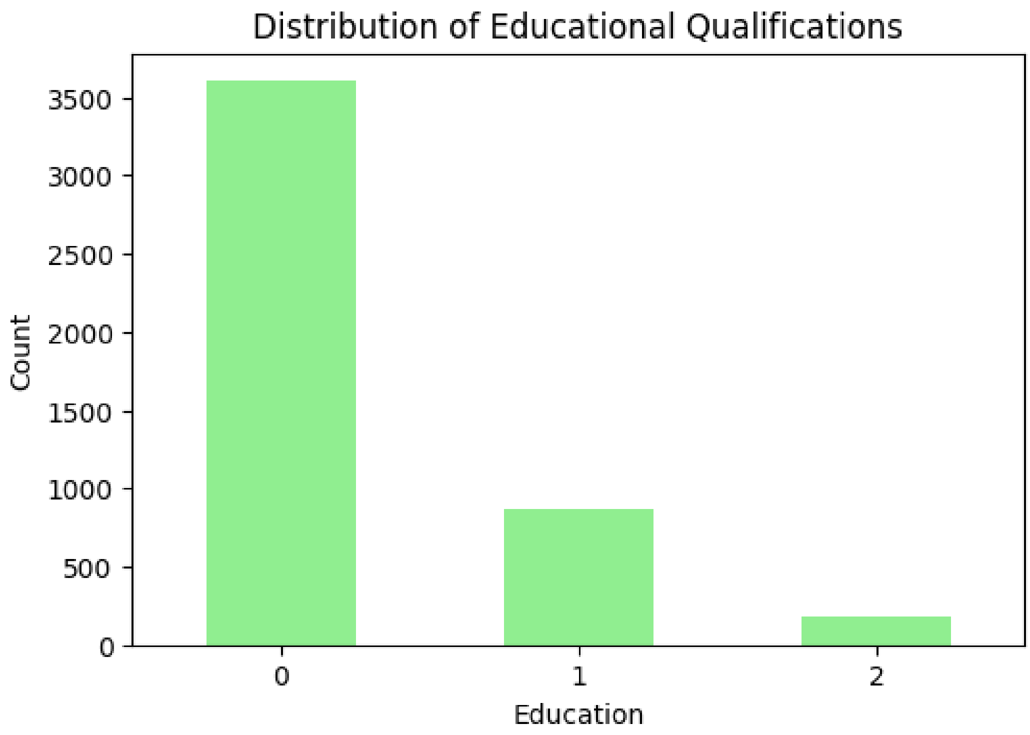
```
[ ]: accuracy_nb = accuracy_score(y_test, y_naive_bayes)
     accuracy_dtc = accuracy_score(y_test, y_decision_tree)
     accuracy_rfc = accuracy_score(y_test, y_random_forest)

     print(f'Accuracy for Naive Bayes: {accuracy_nb:.2f}')
     print(f'Accuracy for Decision Tree 1: {accuracy_dtc:.2f}')
     print(f'Accuracy for Random Forest Classifier 2: {accuracy_rfc:.2f}')
```

```
Accuracy for Naive Bayes: 0.67
Accuracy for Decision Tree 1: 0.81
Accuracy for Random Forest Classifier 2: 0.83
```

Q1. What is the distribution of educational qualifications among employees?

[ ]:

```
education_counts = df['Education'].value_counts()
plt.figure(figsize=(6, 4))
education_counts.plot(kind='bar', color='lightgreen')
plt.title('Distribution of Educational Qualifications')
plt.xlabel('Education')
plt.ylabel('Count')
plt.xticks(rotation=0)
plt.show()
```

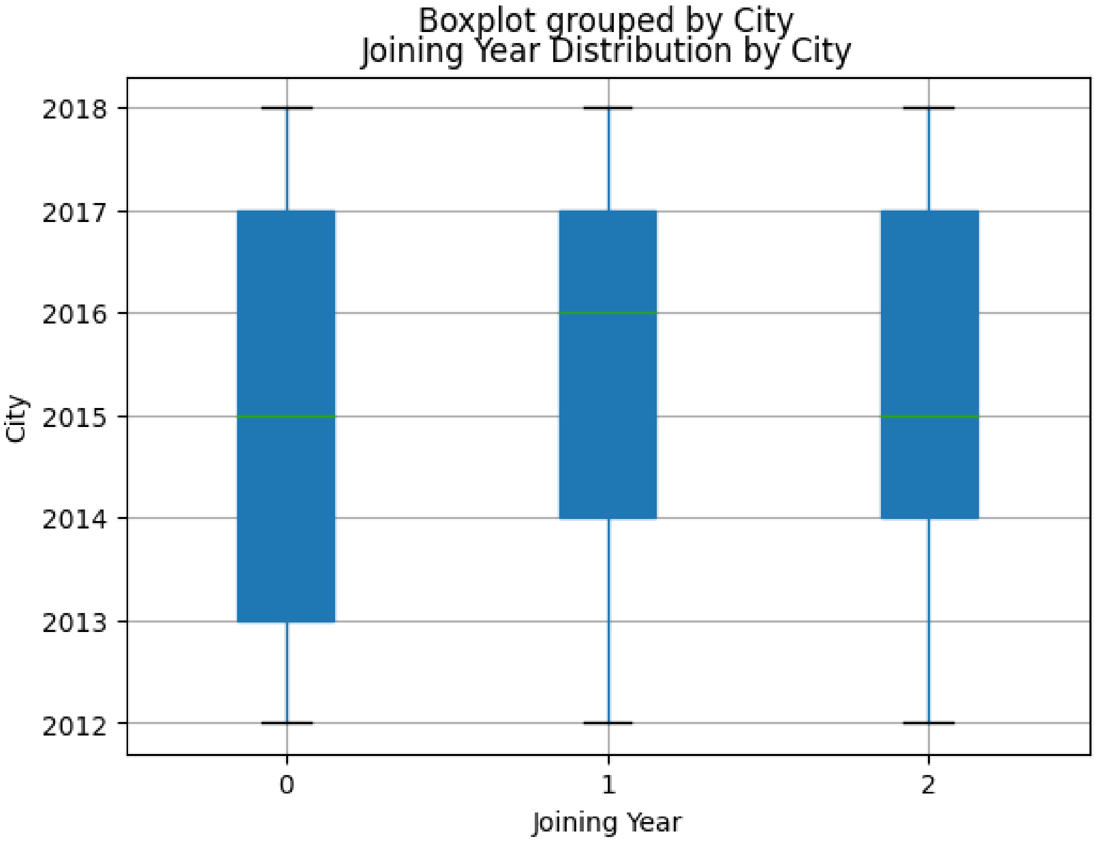


Q2. How does the length of service (Joining Year) vary across different cities?

[ ]:

```
plt.figure(figsize=(10, 6))
df.boxplot(column='JoiningYear', by='City', patch_artist=True)
plt.title('Joining Year Distribution by City')
plt.xlabel('Joining Year')
plt.ylabel('City')

plt.show()
```

```
<Figure size 1000x600 with 0 Axes>
```



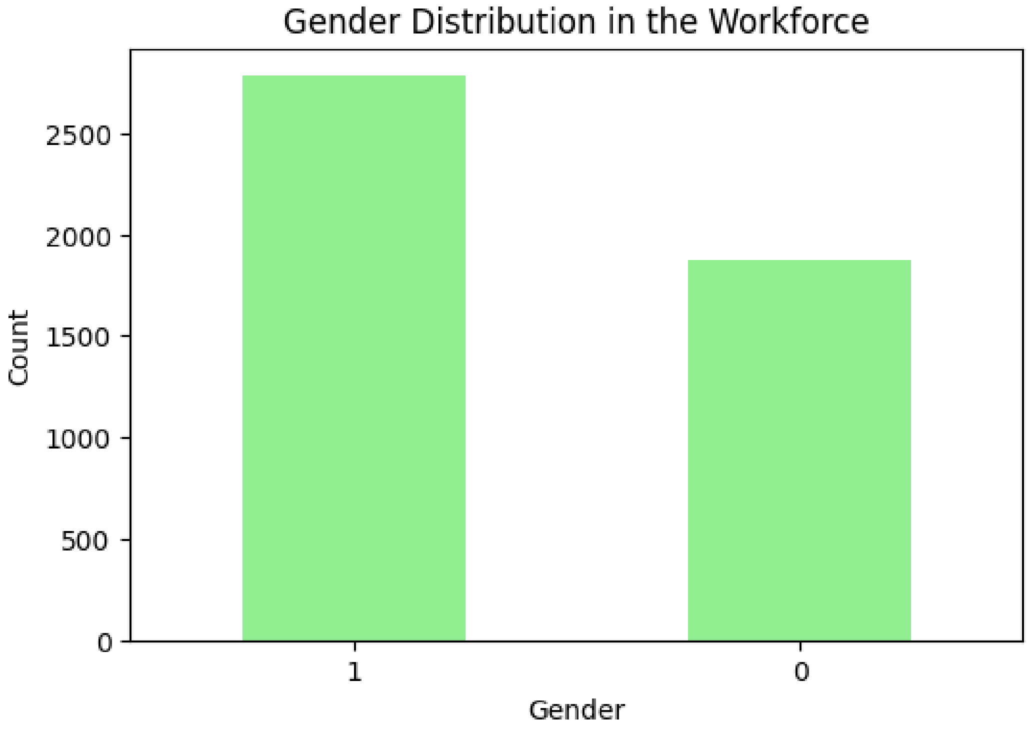
Q3. Is there a correlation between Payment Tier and Experience in Current Domain?

```
[ ]: correlation = df['PaymentTier'].corr(df['ExperienceInCurrentDomain'])

     print(f"Pearson's Correlation Coefficient: {correlation:.2f}")

     if correlation > 0:
         interpretation = "There is a positive correlation."
     elif correlation < 0:
         interpretation = "There is a negative correlation."
     else:
         interpretation = "There is no linear correlation."

     print(interpretation)
```

```
Pearson's Correlation Coefficient: 0.02
There is a positive correlation.
```

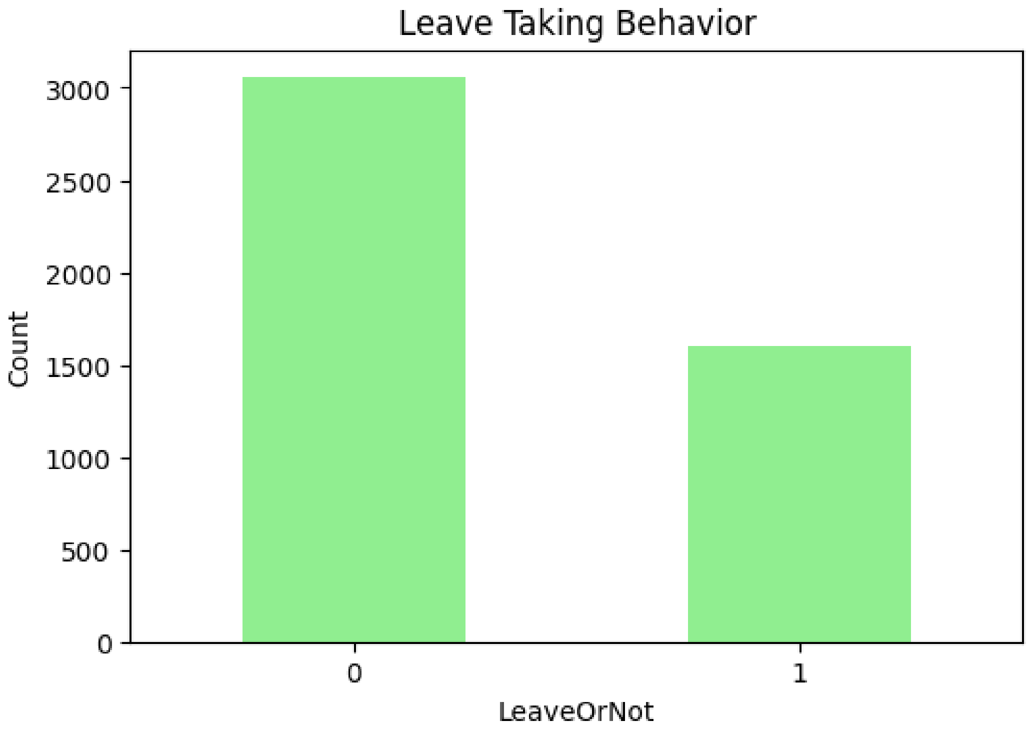
Q4. What is the gender distribution within the workforce?

```
[ ]: gender_counts = df['Gender'].value_counts()
     plt.figure(figsize=(6, 4))
     gender_counts.plot(kind='bar', color='lightgreen')
     plt.title('Gender Distribution in the Workforce')
     plt.xlabel('Gender')
     plt.ylabel('Count')
     plt.xticks(rotation=0)

     plt.show()
```



Q5. Are there any patterns in leave-taking behavior among employees?

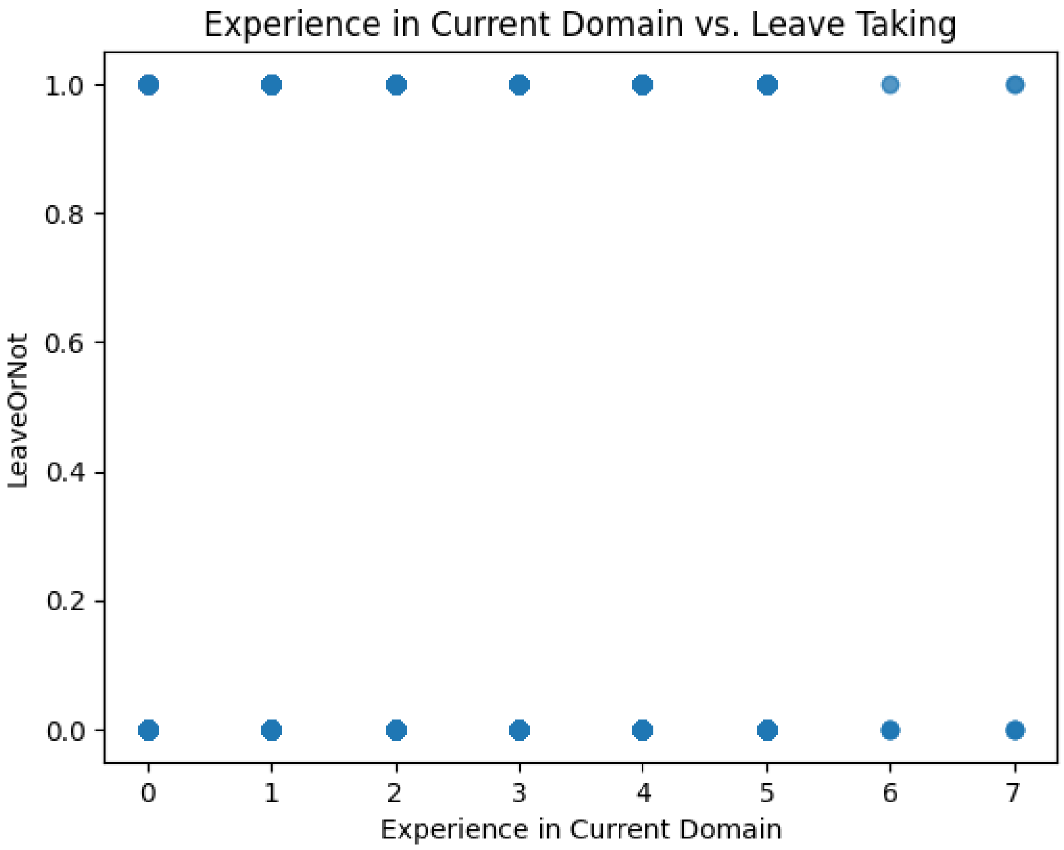
```
[ ]: leave_counts = df['LeaveOrNot'].value_counts()
     plt.figure(figsize=(6, 4))
     leave_counts.plot(kind='bar', color='lightgreen')
     plt.title('Leave Taking Behavior')
     plt.xlabel('LeaveOrNot')
     plt.ylabel('Count')
     plt.xticks(rotation=0)
     plt.show()
```

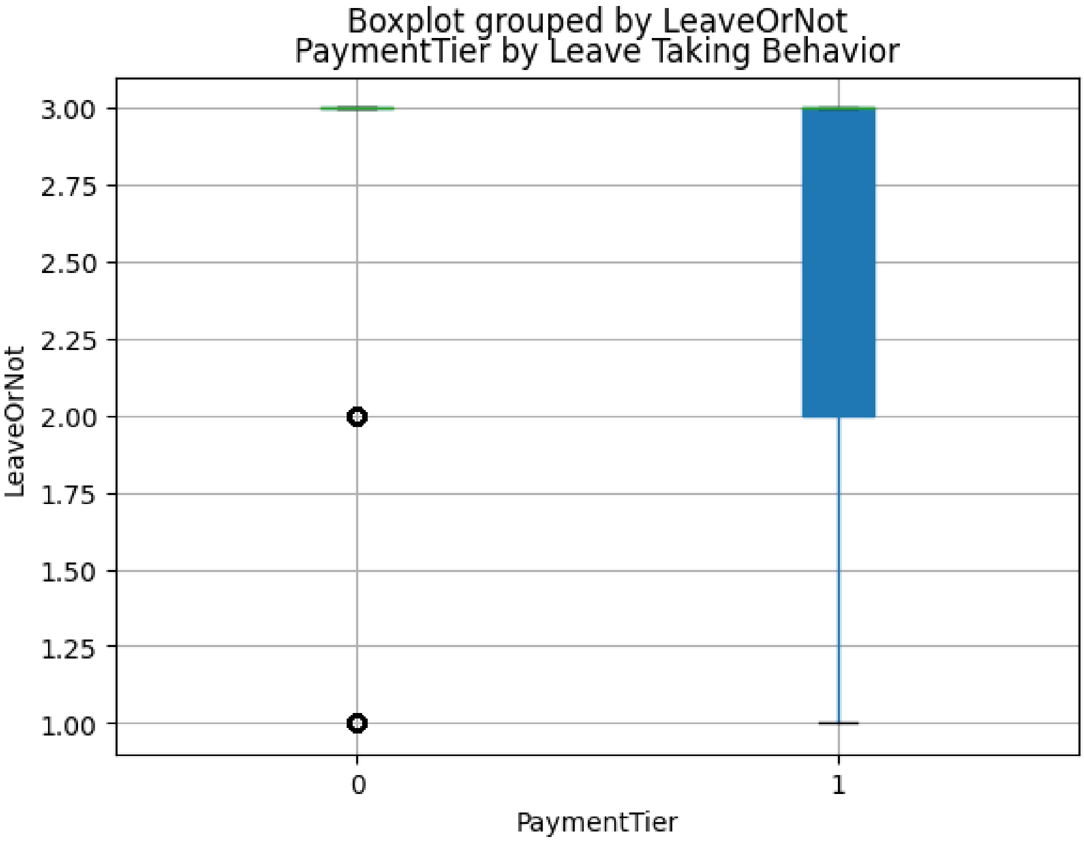


```
[ ]: correlation_leave = df.corr()['LeaveOrNot'].drop('LeaveOrNot')
     print(correlation_leave)
```

```
Education                    0.080497
JoiningYear                  0.181705
City                         0.201058
PaymentTier                 -0.197638
Age                         -0.051126
Gender                      -0.220701
EverBenched                  0.078438
ExperienceInCurrentDomain   -0.030504
Name: LeaveOrNot, dtype: float64
```

```
[ ]: plt.scatter(df['ExperienceInCurrentDomain'], df['LeaveOrNot'], alpha=0.5)
     plt.title('Experience in Current Domain vs. Leave Taking')
     plt.xlabel('Experience in Current Domain')
     plt.ylabel('LeaveOrNot')
     plt.show()

     df.boxplot(column='PaymentTier', by='LeaveOrNot', patch_artist=True)
     plt.title('PaymentTier by Leave Taking Behavior')
     plt.xlabel('PaymentTier')
     plt.ylabel('LeaveOrNot')
plt.show()
```


Experience in Current Domain vs. Leave Taking
LeaveOrNot
1.0
0.8
0.6
0.4
0.2
0.0
0
1
2
3
4
5
6
7
Experience in Current Domain


Boxplot grouped by LeaveOrNot
PaymentTier by Leave Taking Behavior
3.00
2.75
2.50
2.25
2.00
1.75
1.50
1.25
1.00
LeaveOrNot
0
1
PaymentTier