*# This Python 3 environment comes with many helpful analytics libraries installed*

*# It is defined by the kaggle/python Docker image: https://github.com/kaggle/docker-python*

*# For example, here's several helpful packages to load*

import numpy as np *# linear algebra*

import pandas as pd *# data processing, CSV file I/O (e.g. pd.read\_csv)*

import matplotlib.pyplot as plt

import seaborn as sns

*# Input data files are available in the read-only "../input/" directory*

*# For example, running this (by clicking run or pressing Shift+Enter) will list all files under the input directory*

import os

for dirname, \_, filenames **in** os.walk('/kaggle/input'):

for filename **in** filenames:

print(os.path.join(dirname, filename))

*# You can write up to 20GB to the current directory (/kaggle/working/) that gets preserved as output when you create a version using "Save & Run All"*

*# You can also write temporary files to /kaggle/temp/, but they won't be saved outside of the current session*

/kaggle/input/engineering-placements-prediction/collegePlace.csv

In [2]:

df = pd.read\_csv('/kaggle/input/engineering-placements-prediction/collegePlace.csv')

In [3]:

df.head()

Out[3]:

|  | Age | Gender | Stream | Internships | CGPA | Hostel | HistoryOfBacklogs | PlacedOrNot |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 22 | Male | Electronics And Communication | 1 | 8 | 1 | 1 | 1 |
| 1 | 21 | Female | Computer Science | 0 | 7 | 1 | 1 | 1 |
| 2 | 22 | Female | Information Technology | 1 | 6 | 0 | 0 | 1 |
| 3 | 21 | Male | Information Technology | 0 | 8 | 0 | 1 | 1 |
| 4 | 22 | Male | Mechanical | 0 | 8 | 1 | 0 | 1 |

In [4]:

df.shape

Out[4]:

(2966, 8)

In [5]:

df.info()

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 2966 entries, 0 to 2965

Data columns (total 8 columns):

# Column Non-Null Count Dtype

--- ------ -------------- -----

0 Age 2966 non-null int64

1 Gender 2966 non-null object

2 Stream 2966 non-null object

3 Internships 2966 non-null int64

4 CGPA 2966 non-null int64

5 Hostel 2966 non-null int64

6 HistoryOfBacklogs 2966 non-null int64

7 PlacedOrNot 2966 non-null int64

dtypes: int64(6), object(2)

memory usage: 185.5+ KB

In [6]:

df.isna().sum()

Out[6]:

Age 0

Gender 0

Stream 0

Internships 0

CGPA 0

Hostel 0

HistoryOfBacklogs 0

PlacedOrNot 0

dtype: int64

In [7]:

df.Stream.unique()

Out[7]:

array(['Electronics And Communication', 'Computer Science',

'Information Technology', 'Mechanical', 'Electrical', 'Civil'],

dtype=object)

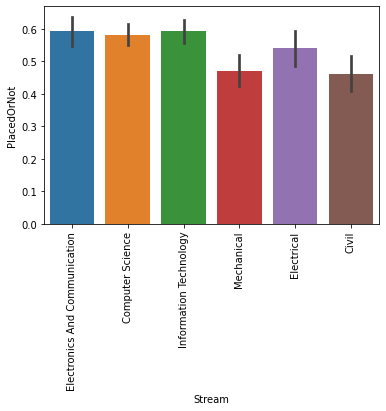
In [8]:

plt.xticks(rotation = 90)

sns.barplot(x = df.Stream, y = df.PlacedOrNot)

Out[8]:

<AxesSubplot:xlabel='Stream', ylabel='PlacedOrNot'>



In [9]:

df.Age.unique()

Out[9]:

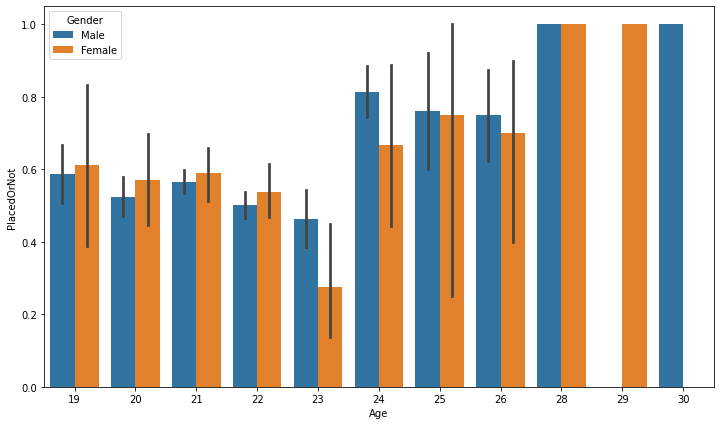
array([22, 21, 23, 24, 28, 30, 25, 26, 20, 19, 29])

In [10]:

plt.figure(figsize = (12,7))

sns.barplot(x = df.Age, y = df.PlacedOrNot, hue = df.Gender)

Out[10]:

<AxesSubplot:xlabel='Age', ylabel='PlacedOrNot'>

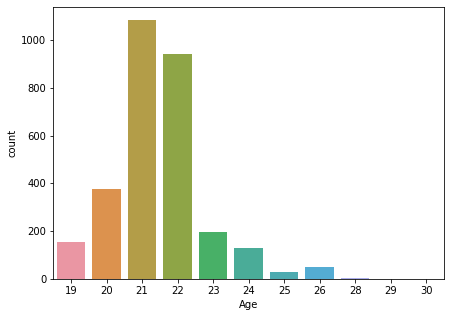
In [11]:

plt.figure(figsize = (7,5))

sns.countplot(x = df.Age)

Out[11]:

<AxesSubplot:xlabel='Age', ylabel='count'>



In [12]:

df.Age.value\_counts()

Out[12]:

21 1084

22 941

20 375

23 195

19 156

24 131

26 50

25 29

28 3

30 1

29 1

Name: Age, dtype: int64

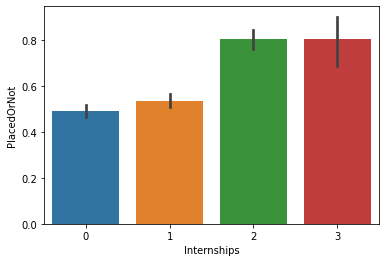
there are only 5 students whose ages are above 26, thats why their placement rate is high

In [13]:

sns.barplot(x = df.Internships, y = df.PlacedOrNot)

Out[13]:

<AxesSubplot:xlabel='Internships', ylabel='PlacedOrNot'>



In [14]:

df.Internships.value\_counts()

Out[14]:

0 1331

1 1234

2 350

3 51

Name: Internships, dtype: int64

In [15]:

df.CGPA.value\_counts()

Out[15]:

7 956

8 915

6 834

9 165

5 96

Name: CGPA, dtype: int64

In [16]:

sns.barplot(x = df.CGPA, y = df.PlacedOrNot)

Out[16]:

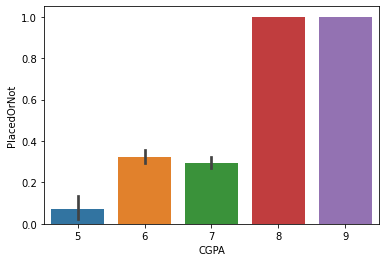
<AxesSubplot:xlabel='CGPA', ylabel='PlacedOrNot'>

In [17]:

sns.barplot(x = df.Hostel, y = df.PlacedOrNot)

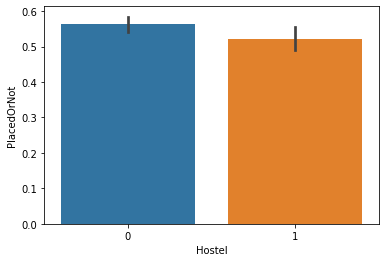
Out[17]:

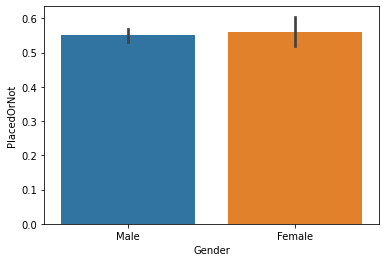
<AxesSubplot:xlabel='Hostel', ylabel='PlacedOrNot'>



In [18]:

sns.barplot(x = df.Gender, y = df.PlacedOrNot)



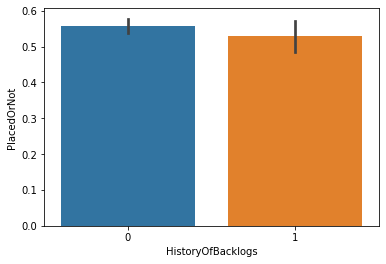


Out[18]:

<AxesSubplot:xlabel='Gender', ylabel='PlacedOrNot'>

In [19]:

sns.barplot(x = df.HistoryOfBacklogs, y = df.PlacedOrNot)



Out[19]:

<AxesSubplot:xlabel='HistoryOfBacklogs', ylabel='PlacedOrNot'>

In [20]:

from sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

In [21]:

df.Gender = le.fit\_transform(df.Gender)

df.Stream = le.fit\_transform(df.Stream)

In [22]:

df.head()

Out[22]:

|  | Age | Gender | Stream | Internships | CGPA | Hostel | HistoryOfBacklogs | PlacedOrNot |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 22 | 1 | 3 | 1 | 8 | 1 | 1 | 1 |
| 1 | 21 | 0 | 1 | 0 | 7 | 1 | 1 | 1 |
| 2 | 22 | 0 | 4 | 1 | 6 | 0 | 0 | 1 |
| 3 | 21 | 1 | 4 | 0 | 8 | 0 | 1 | 1 |
| 4 | 22 | 1 | 5 | 0 | 8 | 1 | 0 | 1 |

In [23]:

x = df.drop(['PlacedOrNot'], axis = 1)

In [24]:

y = df.PlacedOrNot

In [25]:

from sklearn.svm import SVC

from sklearn.tree import DecisionTreeClassifier

from sklearn.linear\_model import LogisticRegression

from sklearn.ensemble import RandomForestClassifier

from sklearn.neighbors import KNeighborsClassifier

In [26]:

from sklearn.model\_selection import cross\_val\_score

In [27]:

cross\_val\_score(SVC(),x, y, cv = 3)

Out[27]:

array([0.73609707, 0.76238625, 0.84817814])

In [28]:

cross\_val\_score(DecisionTreeClassifier(), x, y, cv = 3)

Out[28]:

array([0.84428716, 0.83822042, 0.91497976])

In [29]:

cross\_val\_score(LogisticRegression(), x, y, cv = 3)

Out[29]:

array([0.71991911, 0.74823054, 0.83704453])

In [30]:

cross\_val\_score(RandomForestClassifier(n\_estimators=50), x, y, cv = 3)

Out[30]:

array([0.84428716, 0.84833165, 0.89979757])

In [31]:

cross\_val\_score(KNeighborsClassifier(),x, y ,cv = 3)

Out[31]:

array([0.82103134, 0.82103134, 0.86842105])

Looks like RandomForest is the best model from this problem

In [32]:

from sklearn.model\_selection import train\_test\_split

X\_train, X\_test, y\_train, y\_test = train\_test\_split(x, y, test\_size = 0.2)

In [33]:

model = RandomForestClassifier()

model.fit(X\_train, y\_train)

Out[33]:

RandomForestClassifier()

In [34]:

y\_pred = model.predict(X\_test)

In [35]:

from sklearn.metrics import confusion\_matrix

cm = confusion\_matrix(y\_test, y\_pred)

In [36]:

sns.heatmap(cm, annot = True)

Out[36]:

<AxesSubplot:>

