In [1]:

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

In [5]:

```
df= pd.read_csv(r"C:\Users\Roshan Ramdas Kate\Downloads\Social_Network_Ads.csv")
```

In [6]:

```
df.shape
```

Out[6]:

```
(400, 5)
```

In [7]:

```
 df.head()
```

Out[7]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

In [8]:

```
df.dtypes
```

Out[8]:

```
User ID              int64
Gender              object
Age                  int64
EstimatedSalary      int64
Purchased            int64
dtype: object
```

In [9]:

```
df.isnull().sum()
```

Out[9]:

```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

In [10]:

```
df.describe()
```

Out[10]:

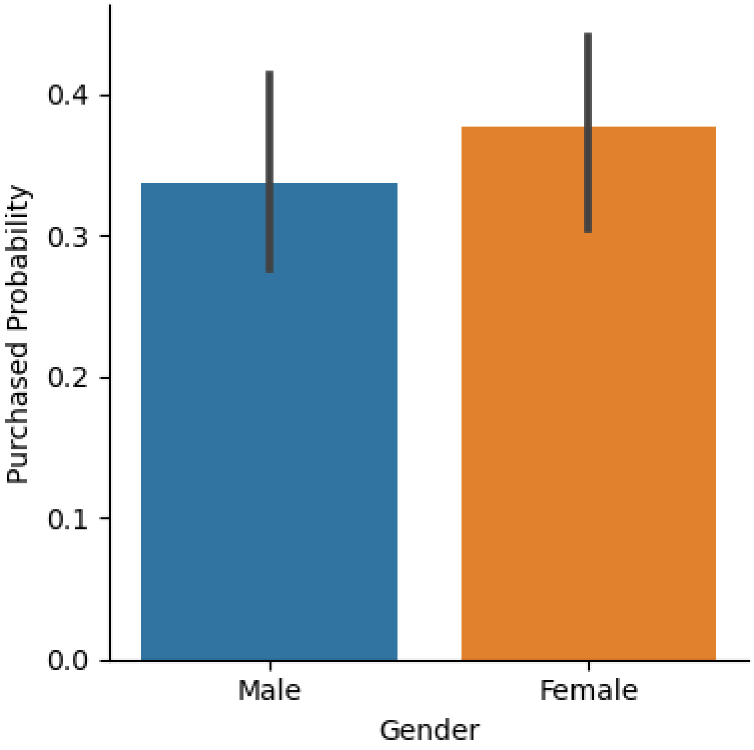
	User ID	Age	EstimatedSalary	Purchased
count	4.000000e+02	400.000000	400.000000	400.000000
mean	1.569154e+07	37.655000	69742.500000	0.357500
std	7.165832e+04	10.482877	34096.960282	0.479864
min	1.556669e+07	18.000000	15000.000000	0.000000
25%	1.562676e+07	29.750000	43000.000000	0.000000
50%	1.569434e+07	37.000000	70000.000000	0.000000
75%	1.575036e+07	46.000000	88000.000000	1.000000
max	1.581524e+07	60.000000	150000.000000	1.000000

In [11]:

```
g = sns.catplot(x = "Gender",y = "Purchased",data = df,kind = "bar",height = 4)
g.set_ylabels("Purchased Probability")
plt.show
```

Out[11]:

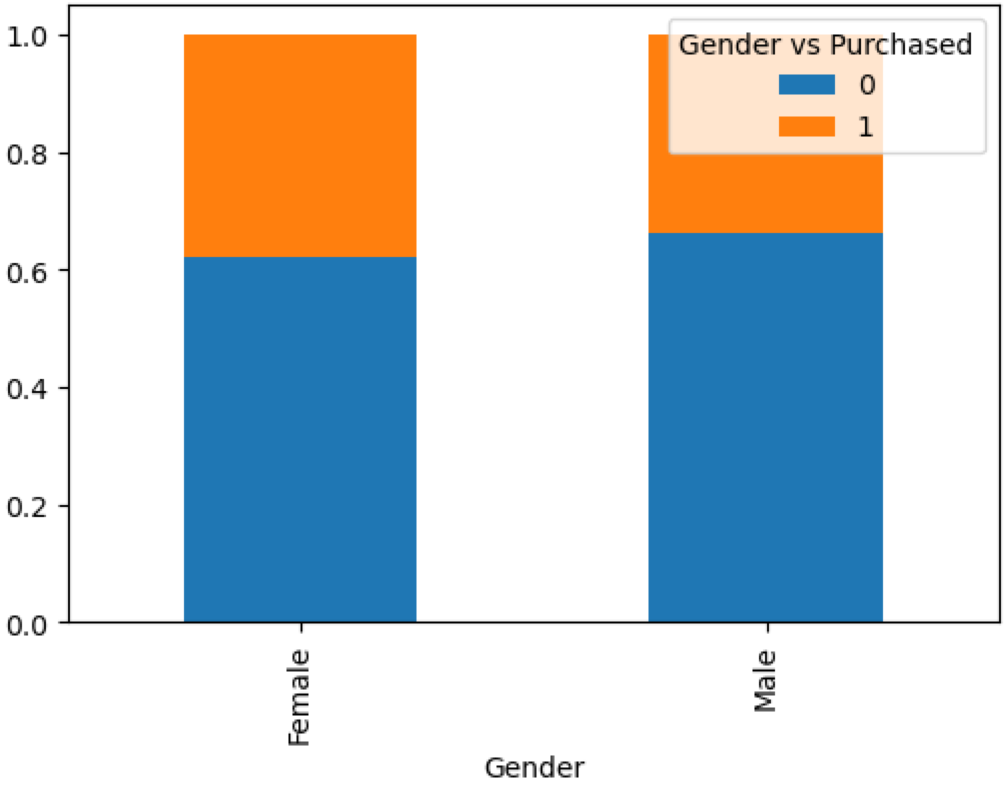
```
<function matplotlib.pyplot.show(close=None, block=None)>
```



In [12]:

```
M2 = pd.crosstab(df.Gender, df.Purchased, normalize='index')
print(M2)
M2.plot.bar(figsize=(6,4),stacked=True)
plt.legend(title='Gender vs Purchased', loc='upper right')
plt.show()
```

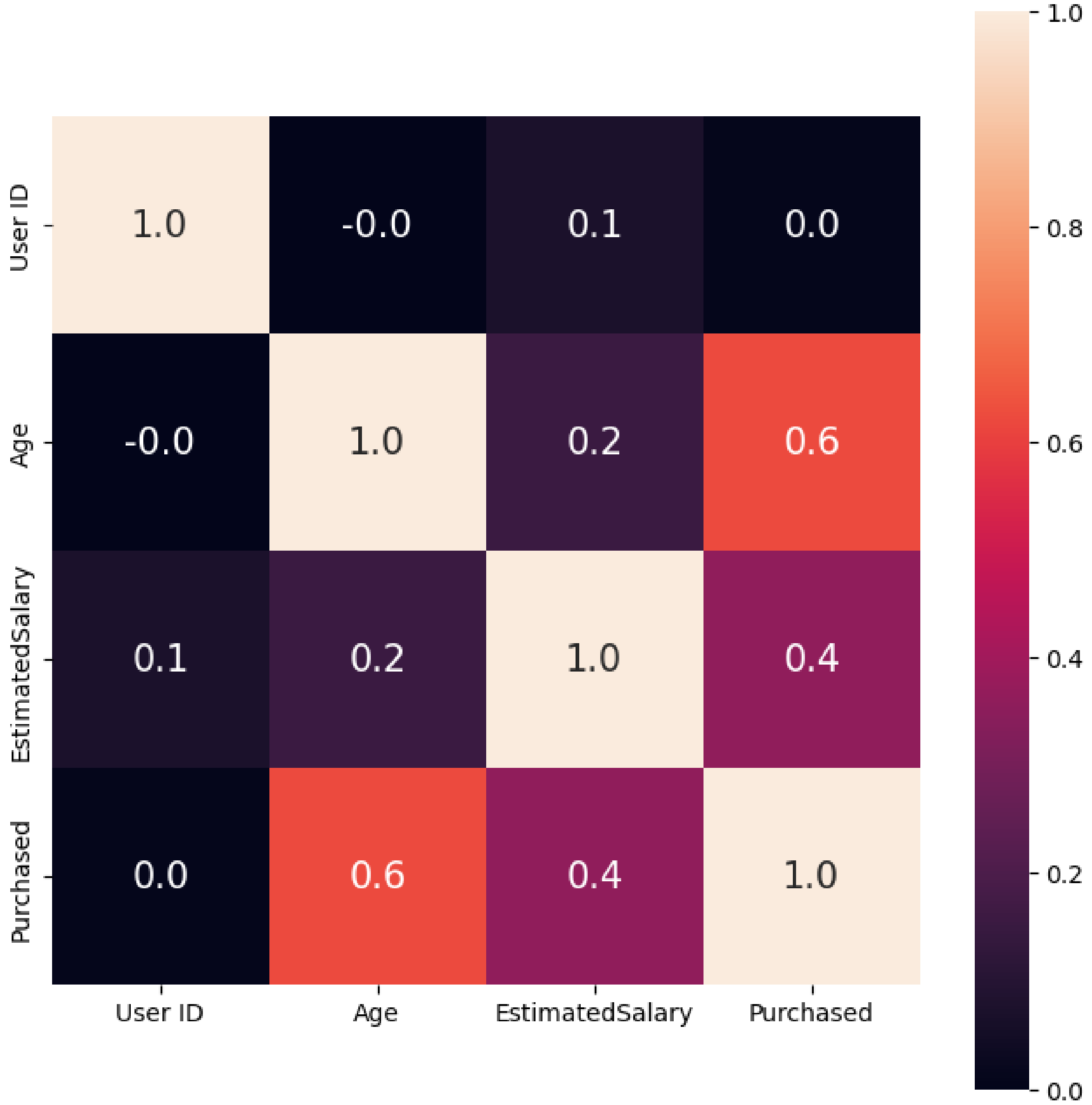
```
Purchased         0         1
Gender
Female     0.622549  0.377451
Male       0.663265  0.336735
```



In [14]:
```
corr = df.corr()
print(corr.shape)
plt.figure(figsize=(8,8))
sns.heatmap(corr, cbar=True, square= True, fmt='.1f', annot=True, annot_kws={'size
```

```
(4, 4)
```

Out[14]:
```
<AxesSubplot:>
```



In [15]:

```
X=df.drop(['Gender','Purchased'],axis=1)
Y= df['Purchased']
X.head()
```

Out[15]:

	User ID	Age	EstimatedSalary
0	15624510	19	19000
1	15810944	35	20000
2	15668575	26	43000
3	15603246	27	57000
4	15804002	19	76000

In [17]:

```
# Split the dataset into training and testing datasets
from sklearn.model_selection import train_test_split
# Shuffle and split the data into training and testing subsets
X_train, X_test, y_train, y_test = train_test_split(X, Y, test_size=0.2, random_st
```

In [19]:

```
from sklearn.linear_model import LogisticRegression
# instantiate the model (using the default parameters)
basemodel= LogisticRegression()
# fit the model with data
```

```
basemodel.fit(X_train,y_train)
print("Training accuracy:", basemodel.score(X_train,y_train)*100)
```

```
Training accuracy: 78.75
```

In [20]:

```
y_predict= basemodel.predict(X_test)
print("Testing accuracy:", basemodel.score(X_test,y_test)*100)
```

```
Testing accuracy: 73.75
```

In [21]:

```
from sklearn.metrics import accuracy_score
Acc=accuracy_score(y_test,y_predict)
print(Acc)
```

```
0.7375
```

In [22]:

```
from sklearn.metrics import confusion_matrix
cm= confusion_matrix(y_test,y_predict)
print(cm)
```

```
[[49  3]
 [18 10]]
```

In [23]:

```
from sklearn.metrics import precision_recall_fscore_support
prf= precision_recall_fscore_support(y_test,y_predict)
print('precision:',prf[0])
print('Recall:',prf[1])
print('fscore:',prf[2])
print('support:',prf[3])
```

```
precision: [0.73134328 0.76923077]
Recall: [0.94230769 0.35714286]
fscore: [0.82352941 0.48780488]
support: [52 28]
```

In [24]:

```
from sklearn.metrics import classification_report
cr= classification_report(y_test,y_predict)
print(cr)
```

```
              precision    recall  f1-score   support

           0       0.73      0.94      0.82        52
           1       0.77      0.36      0.49        28

    accuracy                           0.74        80
   macro avg       0.75      0.65      0.66        80
weighted avg       0.74      0.74      0.71        80
```

In [ ]: