In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
df=pd.read_csv('Product Purchase data.csv')
df
```

Out[2]:

	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
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

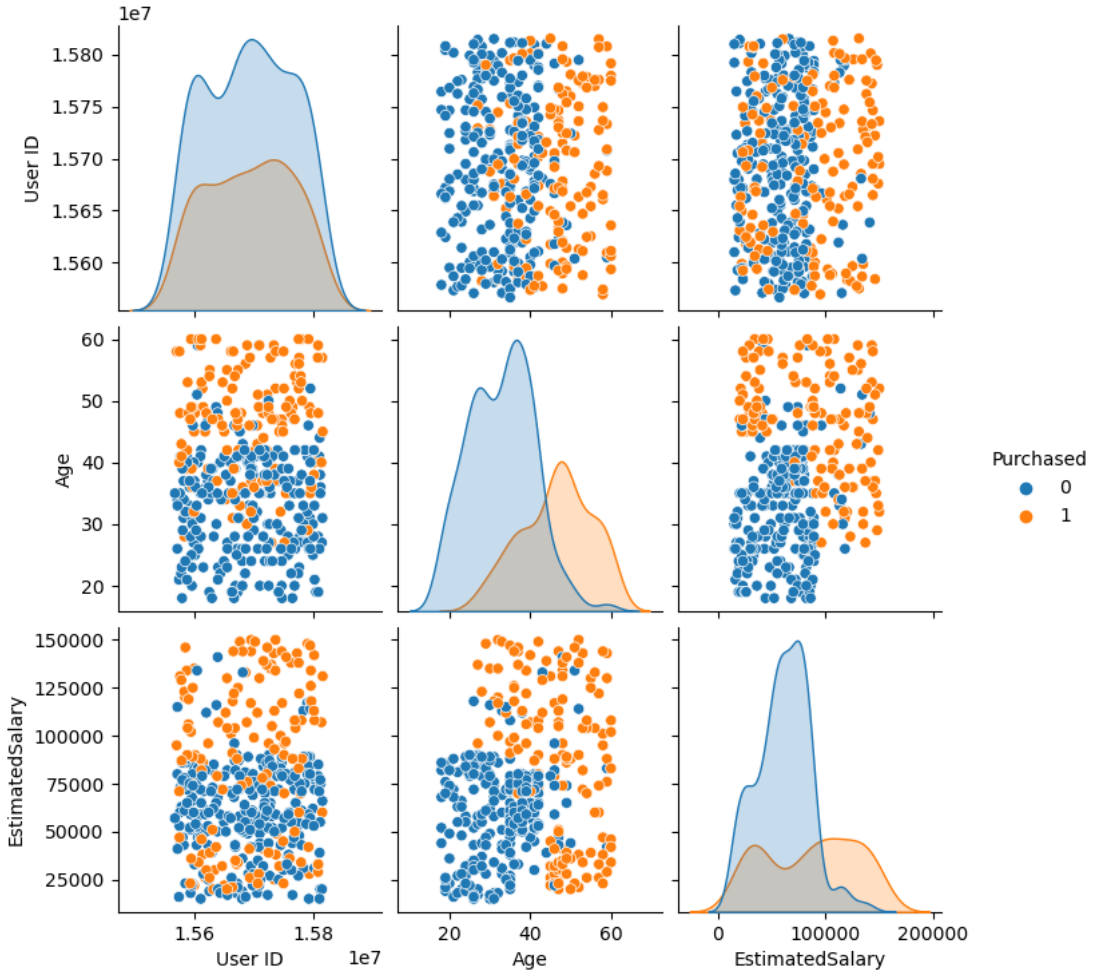
In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   User ID          400 non-null    int64 
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64 
 3   EstimatedSalary  400 non-null    int64 
 4   Purchased        400 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

In [4]:

```
sns.pairplot(df,hue='Purchased')
plt.show()
```



In [5]:

```
x=df.iloc[:,2:4]
y=df.iloc[:,-1:]
```

In [6]:

```
y
```

Out[6]:

	Purchased
0	0
1	0
2	0
3	0
4	0
...	...
395	1
396	1
397	1
398	0
399	1

400 rows × 1 columns

In [7]:

```
x
```

Out[7]:

	Age	EstimatedSalary
0	19	19000
1	35	20000
2	26	43000
3	27	57000
4	19	76000
...	...	...
395	46	41000
396	51	23000
397	50	20000
398	36	33000
399	49	36000

400 rows × 2 columns

In [8]:

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [9]:

```
xtrain
```

Out[9]:

	Age	EstimatedSalary
336	58	144000
64	59	83000
55	24	55000
106	26	35000
300	58	38000
...	...	...
323	48	30000
192	29	43000
117	36	52000
47	27	54000
172	26	118000

320 rows × 2 columns

In [10]:

```
xtest
```

Out[10]:

	Age	EstimatedSalary
132	30	87000
309	38	50000
341	35	75000
196	30	79000
246	35	50000
...	...	...
14	18	82000
363	42	79000
304	40	60000
361	53	34000
329	47	107000

80 rows × 2 columns

In [11]:

```
from sklearn.neighbors import KNeighborsClassifier
```

In [12]:

```
knn=KNeighborsClassifier(n_neighbors=7)
knn.fit(xtrain,ytrain)
ypred=knn.predict(xtest)
```

In [13]:

```
ypred
```

Out[13]:

```
array([0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0,
       1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1], dtype=int64)
```

In [14]:

```
from sklearn.metrics import accuracy_score,confusion_matrix,classification_report
ac=accuracy_score(ytest,ypred)
cm=confusion_matrix(ytest,ypred)
cr=classification_report(ytest,ypred)
```

In [15]:

```
print(ac)
print(cm)
print(cr)
```

```
0.875
[[54  4]
 [ 6 16]]
              precision    recall  f1-score   support

           0       0.90      0.93      0.92        58
           1       0.80      0.73      0.76        22

    accuracy                           0.88        80
   macro avg       0.85      0.83      0.84        80
weighted avg       0.87      0.88      0.87        80
```

In [16]:

```
train=knn.score(xtrain,ytrain)
test=knn.score(xtest,ytest)
```

In [17]:

```
train
```

Out[17]:

```
0.846875
```

In [18]:

```
test
```

Out[18]:

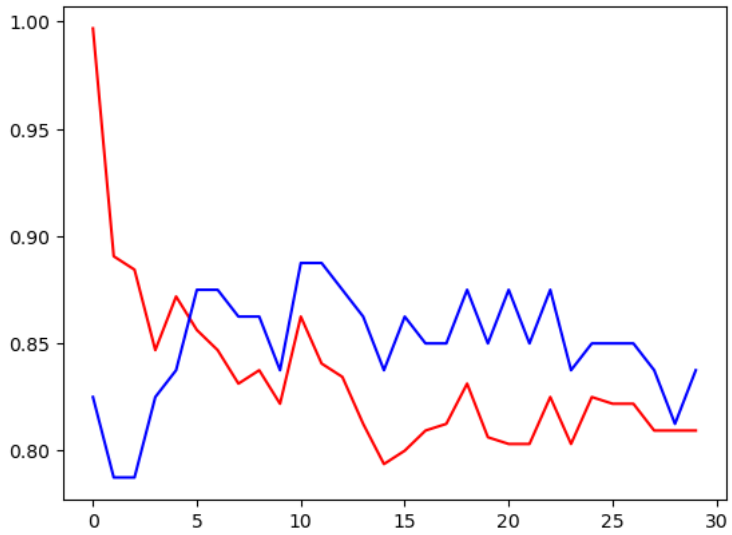
```
0.875
```

In [19]:

```
train1=[]
test1=[]
for i in range(1,31):
    knn1=KNeighborsClassifier(n_neighbors=i)
    knn1.fit(xtrain,ytrain)
    ypred1=knn1.predict(xtest)
    trainac=knn1.score(xtrain,ytrain)
    testac=knn1.score(xtest,ytest)
    train1.append(trainac)
    test1.append(testac)
```

In [20]:

```
plt.plot(train1,color='red')
plt.plot(test1,color='blue')
plt.show()
```



In [21]:

```
knn2=KNeighborsClassifier(n_neighbors=4)
knn2.fit(xtrain,ytrain)
ypred2=knn2.predict(xtest)
ac2=accuracy_score(ytest,ypred2)
cm2=confusion_matrix(ytest,ypred2)
cr2=classification_report(ytest,ypred2)
print(f'Accuracy is {ac2}')
print(f'Confusion matrix is\n {cm2}')
print(f'Classification report\n{cr2}')
train2=knn2.score(xtrain,ytrain)
test2=knn2.score(xtest,ytest)
print(f'Training acc {train2}')
print(f'Testing acc {test2}')
```

```
Accuracy is 0.825
Confusion matrix is
 [[54  4]
 [10 12]]
Classification report
              precision    recall  f1-score   support

           0       0.84      0.93      0.89        58
           1       0.75      0.55      0.63        22

    accuracy                           0.82        80
   macro avg       0.80      0.74      0.76        80
weighted avg       0.82      0.82      0.82        80

Training acc 0.846875
Testing acc 0.825
```

In [22]:

```
def purchase():
    age=int(input('Enter age'))
    salary=int(input('Enter salary'))
    newob=[[age,salary]]
    yp=knn.predict(newob)[0]

    if yp==1:
        print(f'The person will purchase')
    else:
        print(f'The person will not purchase')
```

In [23]:

```
purchase()
```

```
Enter age45
Enter salary100000
The person will purchase
```

In [ ]: