In [1]:

```
# importing libraries

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder, StandardScaler, OneHotEncoder
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
df = pd.read_csv("bank _M.csv")
df.head()
```

Out[2]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	1042	1
1	56	admin.	married	secondary	no	45	no	no	unknown	5	may	1467	1
2	41	technician	married	secondary	no	1270	yes	no	unknown	5	may	1389	1
3	55	services	married	secondary	no	2476	yes	no	unknown	5	may	579	1
4	54	admin.	married	tertiary	no	184	no	no	unknown	5	may	673	2

In [3]:

```
df.describe()
```

Out[3]:

	age	balance	day	duration	campaign	pdays	previous
count	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000	11162.000000
mean	41.231948	1528.538524	15.658036	371.993818	2.508421	51.330407	0.832557
std	11.913369	3225.413326	8.420740	347.128386	2.722077	108.758282	2.292007
min	18.000000	-6847.000000	1.000000	2.000000	1.000000	-1.000000	0.000000
25%	32.000000	122.000000	8.000000	138.000000	1.000000	-1.000000	0.000000
50%	39.000000	550.000000	15.000000	255.000000	2.000000	-1.000000	0.000000
75%	49.000000	1708.000000	22.000000	496.000000	3.000000	20.750000	1.000000
max	95.000000	81204.000000	31.000000	3881.000000	63.000000	854.000000	58.000000

In [4]:

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

Out[4]:

```
age          0
job          0
marital      0
education    0
default      0
balance      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
deposit      0
dtype: int64
```

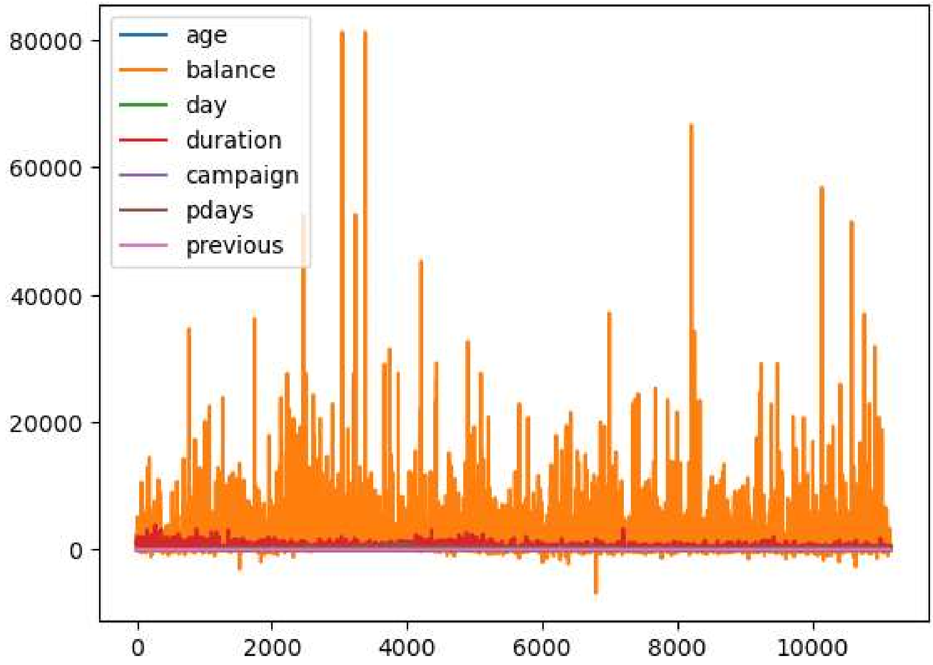
In [5]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11162 entries, 0 to 11161
Data columns (total 17 columns):
 #   Column     Non-Null Count  Dtype 
---  ------     --------------  ----- 
 0   age        11162 non-null  int64 
 1   job        11162 non-null  object
 2   marital    11162 non-null  object
 3   education  11162 non-null  object
 4   default    11162 non-null  object
 5   balance    11162 non-null  int64 
 6   housing    11162 non-null  object
 7   loan       11162 non-null  object
 8   contact    11162 non-null  object
 9   day        11162 non-null  int64 
 10  month      11162 non-null  object
 11  duration   11162 non-null  int64 
 12  campaign   11162 non-null  int64 
 13  pdays      11162 non-null  int64 
 14  previous   11162 non-null  int64 
 15  poutcome   11162 non-null  object
 16  deposit    11162 non-null  object
dtypes: int64(7), object(10)
memory usage: 1.4+ MB
```

In [6]:

```
df.plot()
plt.show()
```



In [7]:

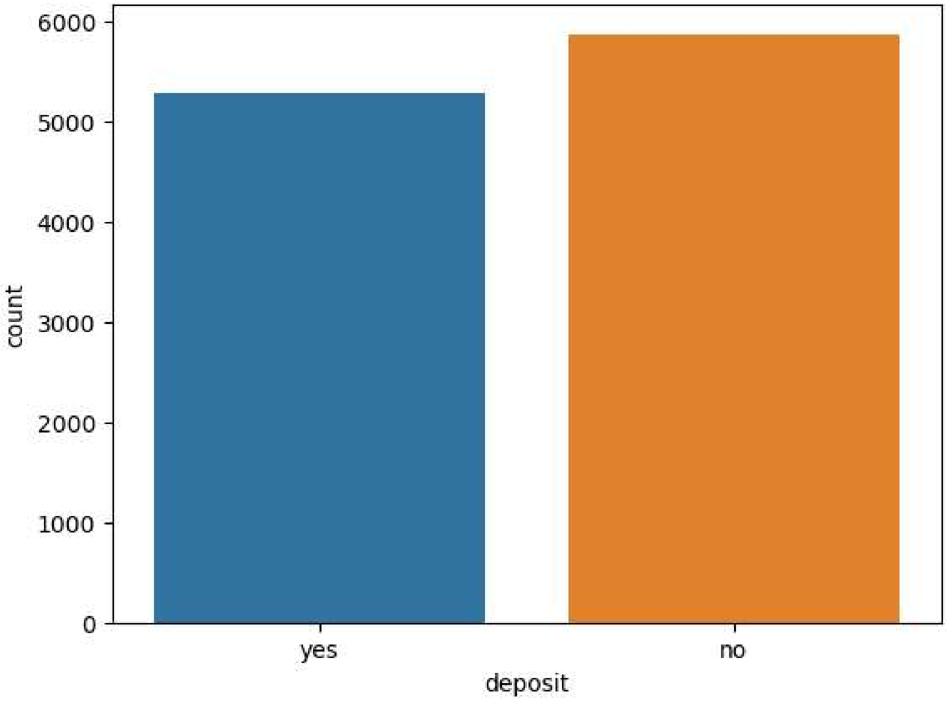
```
df['deposit'].value_counts()
```

Out[7]:

```
deposit
no     5873
yes    5289
Name: count, dtype: int64
```

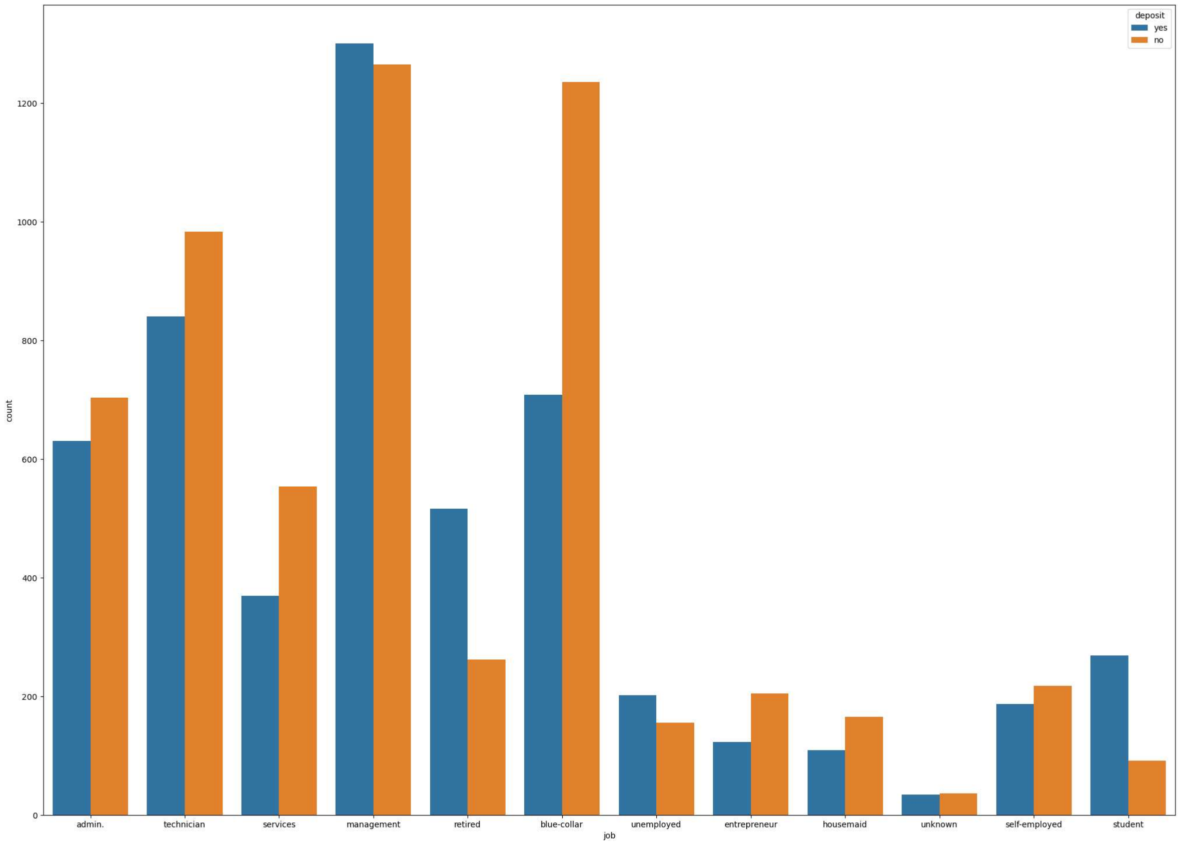
In [8]:

```
sns.countplot(data=df, x ='deposit')
plt.show()
```



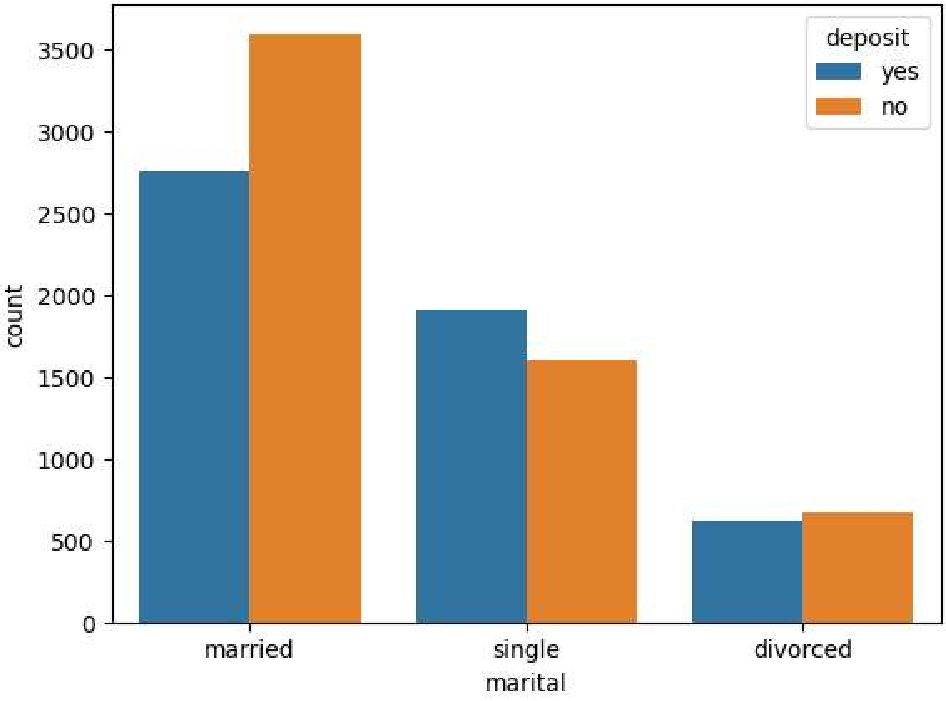
In [9]:

```
plt.figure(figsize=(25,18))
sns.countplot(x=df['job'], y=None, hue=df['deposit'])
plt.show()
```



In [10]:

```
sns.countplot(x=df['marital'], y=None, hue=df['deposit'])
plt.show()
```



In [11]:

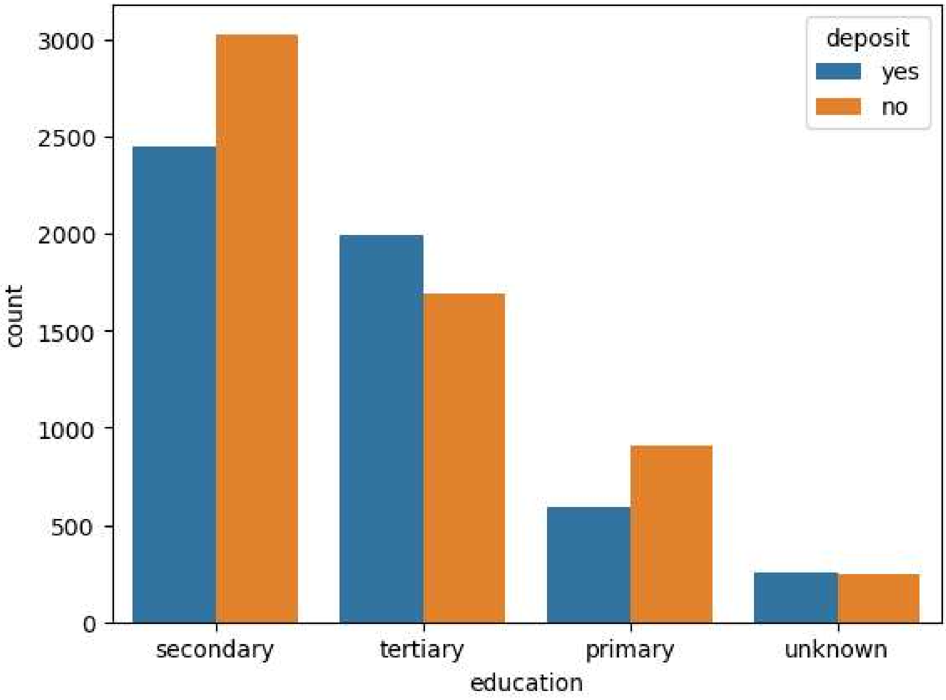
```
df['education'].value_counts()
```

Out[11]:

```
education
secondary    5476
tertiary     3689
primary      1500
unknown       497
Name: count, dtype: int64
```

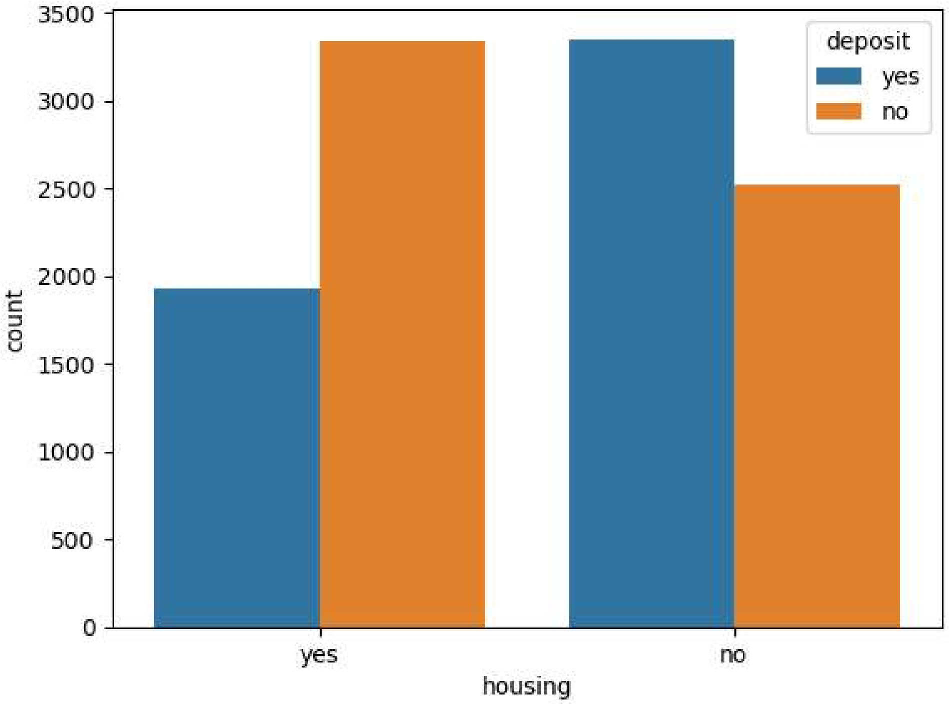
In [12]:

```
sns.countplot(x=df['education'], y=None, hue=df['deposit'])
plt.show()
```



In [13]:

```
sns.countplot(x=df['housing'], y=None, hue=df['deposit'])
plt.show()
```



In [14]:

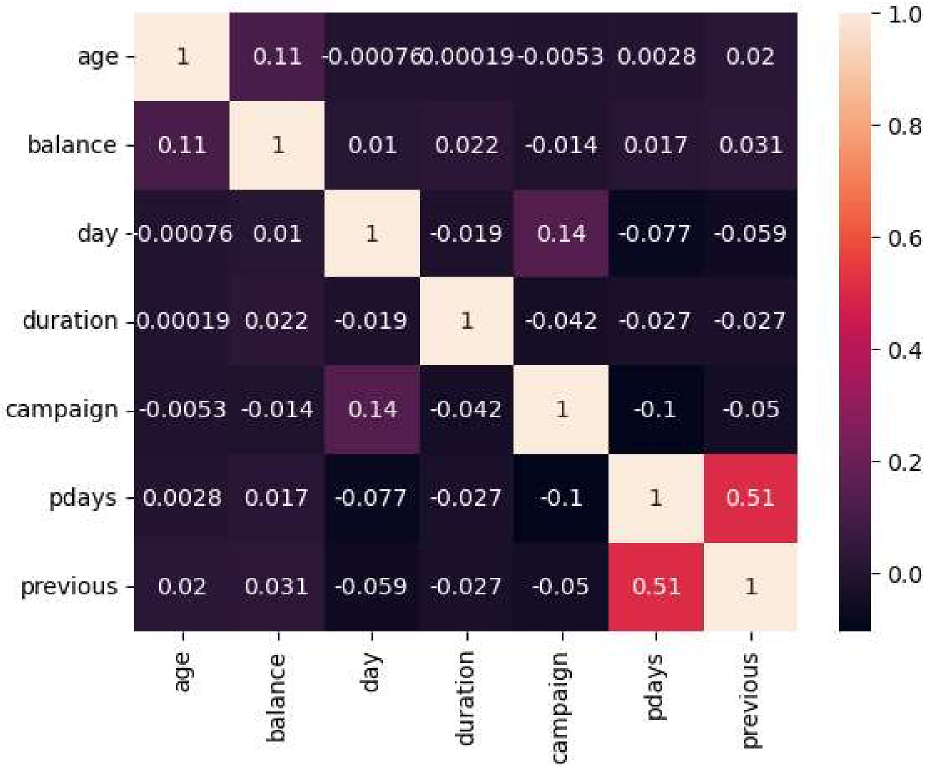
```
df.corr(numeric_only=True)
```

Out[14]:

	age	balance	day	duration	campaign	pdays	previous
age	1.000000	0.112300	-0.000762	0.000189	-0.005278	0.002774	0.020169
balance	0.112300	1.000000	0.010467	0.022436	-0.013894	0.017411	0.030805
day	-0.000762	0.010467	1.000000	-0.018511	0.137007	-0.077232	-0.058981
duration	0.000189	0.022436	-0.018511	1.000000	-0.041557	-0.027392	-0.026716
campaign	-0.005278	-0.013894	0.137007	-0.041557	1.000000	-0.102726	-0.049699
pdays	0.002774	0.017411	-0.077232	-0.027392	-0.102726	1.000000	0.507272
previous	0.020169	0.030805	-0.058981	-0.026716	-0.049699	0.507272	1.000000

In [15]:

```
sns.heatmap(df.corr(numeric_only=True), annot=True)
plt.show()
```



In [16]:

```
le = LabelEncoder()
df['job'] = le.fit_transform(df['job'])
df['marital'] = le.fit_transform(df['marital'])
df['education'] = le.fit_transform(df['education'])
df['default'] = le.fit_transform(df['default'])
df['housing'] = le.fit_transform(df['housing'])
df['loan'] = le.fit_transform(df['loan'])
df['contact'] = le.fit_transform(df['contact'])
df['month'] = le.fit_transform(df['month'])
df['day'] = le.fit_transform(df['day'])
df['deposit'] = le.fit_transform(df['deposit'])
df['poutcome'] = le.fit_transform(df['poutcome'])
```

In [17]:

```
x = df.drop('deposit', axis=1)
y= df['deposit']
```

In [18]:

```
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, random_state=42
```

In [19]:

```
x_train.shape, x_test.shape, y_train.shape, y_test.shape
```

Out[19]:

```
((8929, 16), (2233, 16), (8929,), (2233,))
```

In [20]:

```
# Decision Tree
from sklearn.tree import DecisionTreeClassifier
DT = DecisionTreeClassifier()
DT.fit(x_train, y_train)
DT.score(x_test, y_test)*100
```

Out[20]:

```
75.95163457232422
```

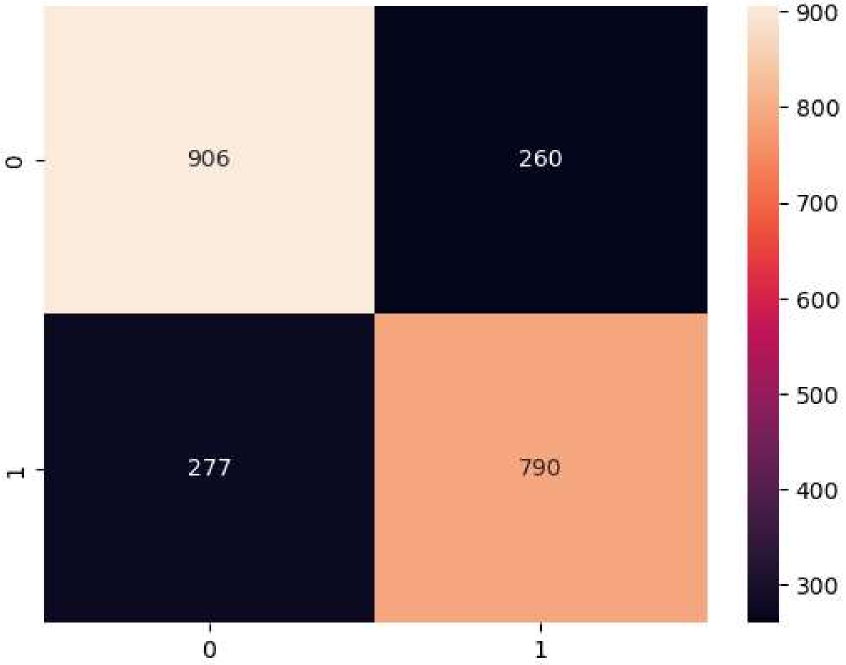
In [21]:

```
y_pred = DT.predict(x_test)
print(y_pred)
```

```
[0 1 1 ... 0 1 0]
```

In [22]:

```
from sklearn.metrics import confusion_matrix
cmf = confusion_matrix(y_test, y_pred)
sns.heatmap(cmf, annot=True, fmt='1')
plt.show()
```



In [23]:

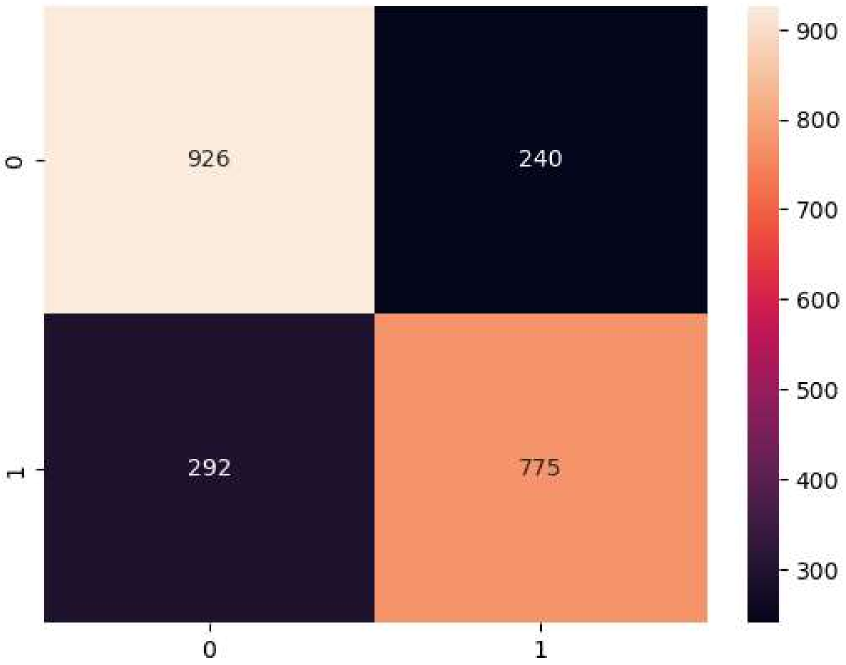
```
# Logistic Regression
from sklearn.linear_model import LogisticRegression
LG = LogisticRegression()
LG.fit(x_train, y_train)
LG.score(x_test, y_test)*100
```

Out[23]:

```
76.17554858934169
```

In [24]:

```
ypred = LG.predict(x_test)
cmf = confusion_matrix(y_test, ypred)
sns.heatmap(cmf, annot=True, fmt='1')
plt.show()
```



In [25]:

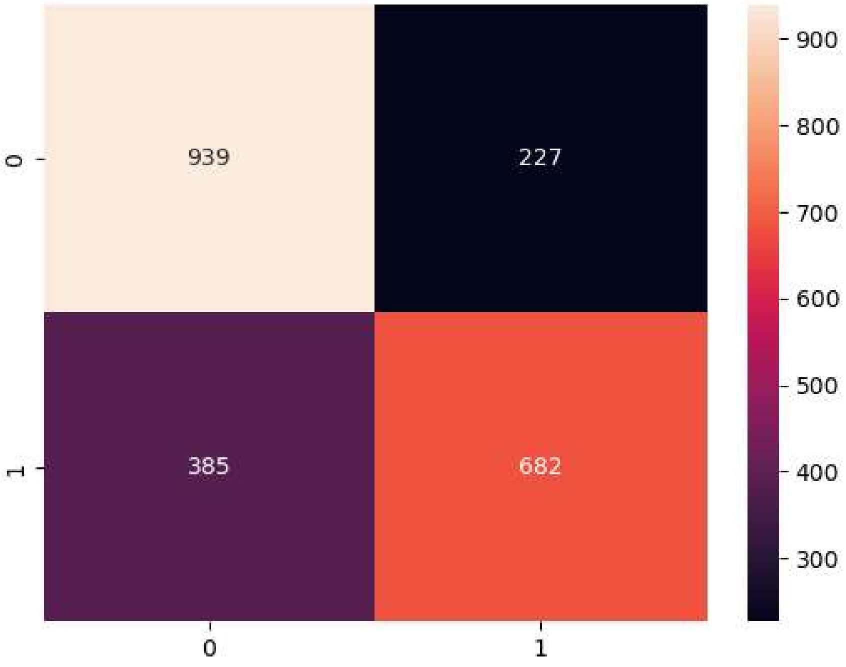
```python
# Support Vetor Machine
from sklearn.svm import SVC
svm = SVC()
svm.fit(x_train, y_train)
svm.score(x_test, y_test)*100
```

Out[25]:

```
72.59292431706224
```

In [26]:

```python
y_predict = svm.predict(x_test)
cmf = confusion_matrix(y_test, y_predict)
sns.heatmap(cmf, annot=True, fmt='1')
plt.show()
```



In [27]:

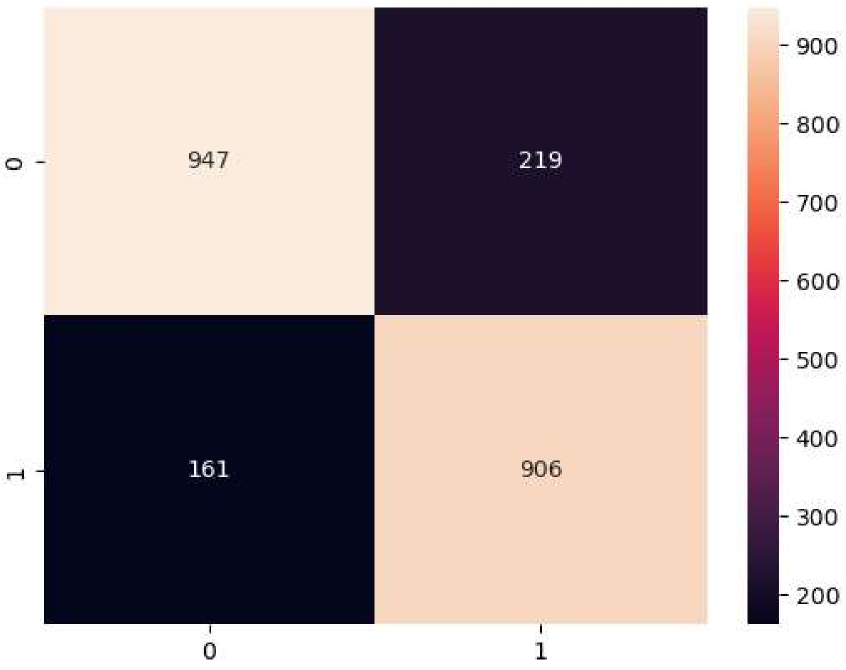
```python
# Random Forest
from sklearn.ensemble import RandomForestClassifier
RF = RandomForestClassifier()
RF.fit(x_train, y_train)
RF.score(x_test, y_test)*100
```

Out[27]:

```
82.98253470667264
```

In [28]:

```
yPredict = RF.predict(x_test)
cmf = confusion_matrix(y_test, yPredict)
sns.heatmap(cmf, annot=True, fmt='1')
plt.show()
```



In [29]:

```
from sklearn.metrics import precision_score
print("Precision Score of Decision Tree :",precision_score(y_test, y_pred))

print("Precision Score of Logistic Regression :",precision_score(y_test, ypred))

print("Precision Score of Support Vetor Machine :",precision_score(y_test, y_predict))

print("Precision Score of Random Forest :",precision_score(y_test, yPredict))
```

```
Precision Score of Decision Tree : 0.7523809523809524
Precision Score of Logistic Regression : 0.7635467980295566
Precision Score of Support Vetor Machine : 0.7502750275027503
Precision Score of Random Forest : 0.8053333333333333
```

In [30]:

```
from sklearn.metrics import recall_score
print("Recall of Decision Tree :",round(recall_score(y_test, y_pred), 2))

print("Recall of Logistic Regression :",round(recall_score(y_test, ypred), 2))

print("Recall of Support Vetor Machine :",round(recall_score(y_test, y_predict), 2))

print("Recall of Random Forest :",round(recall_score(y_test, yPredict), 2))
```

```
Recall of Decision Tree : 0.74
Recall of Logistic Regression : 0.73
Recall of Support Vetor Machine : 0.64
Recall of Random Forest : 0.85
```

In [31]:

```
from sklearn.metrics import accuracy_score
print("Accuracy Score of Decision Tree :",accuracy_score(y_test, y_pred))

print("Accuracy Score of Logistic Regression :",accuracy_score(y_test, ypred))

print("Accuracy Score of Support Vetor Machine :",accuracy_score(y_test, y_predict))

print("Accuracy Score of Random Forest :",accuracy_score(y_test, yPredict))
```

```
Accuracy Score of Decision Tree : 0.7595163457232422
Accuracy Score of Logistic Regression : 0.7617554858934169
Accuracy Score of Support Vetor Machine : 0.7259292431706225
Accuracy Score of Random Forest : 0.8298253470667264
```

In [32]:

```
from sklearn.metrics import classification_report
sc = classification_report(y_test, y_pred)
print(sc)
```

```
              precision    recall  f1-score   support

           0       0.77      0.78      0.77      1166
           1       0.75      0.74      0.75      1067

    accuracy                           0.76      2233
   macro avg       0.76      0.76      0.76      2233
weighted avg       0.76      0.76      0.76      2233
```

In [33]:

```
df.columns
```

Out[33]:

```
Index(['age', 'job', 'marital', 'education', 'default', 'balance', 'housing',
       'loan', 'contact', 'day', 'month', 'duration', 'campaign', 'pdays',
       'previous', 'poutcome', 'deposit'],
      dtype='object')
```

In [34]:

```
df
```

Out[34]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pd
0	59	0	1	1	0	2343	1	0	2	4	8	1042	1	
1	56	0	1	1	0	45	0	0	2	4	8	1467	1	
2	41	9	1	1	0	1270	1	0	2	4	8	1389	1	
3	55	7	1	1	0	2476	1	0	2	4	8	579	1	
4	54	0	1	2	0	184	0	0	2	4	8	673	2	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	
11157	33	1	2	0	0	1	1	0	0	19	0	257	1	
11158	39	7	1	1	0	733	0	0	2	15	6	83	4	
11159	32	9	2	1	0	29	0	0	0	18	1	156	2	
11160	43	9	1	1	0	0	0	1	0	7	8	9	2	
11161	34	9	1	1	0	0	0	0	0	8	5	628	1	

11162 rows × 17 columns

In [35]:

```
def subscripation(age, job, marital, education, default, balance, housing,loan, contact,
    ax = np.array([age, job, marital, education, default, balance, housing,loan, contact
    prediction = DT.predict(ax.reshape(1, -1))
    if prediction == 0:
        return 'Not Subscribed'
    else:
        return "Subsribed"
```

In [36]:

```
subscripation(59, 0, 1, 1, 0, 2343, 1, 0, 2, 4, 8, 1042, 1, -1, 0, 3)
```

Out[36]:

```
'Subsribed'
```

In [37]:

```
subscripation(33, 1, 2, 0, 0, 1, 1, 0, 0, 19, 0, 257, 1, -1, 0, 3,)
```

Out[37]:

```
'Not Subscribed'
```

In [ ]: