In [1]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
```

In [2]:

```
df=pd.read_csv("HRDataset_v14.csv")
```

In [3]:

```
df
```

Out[3]:

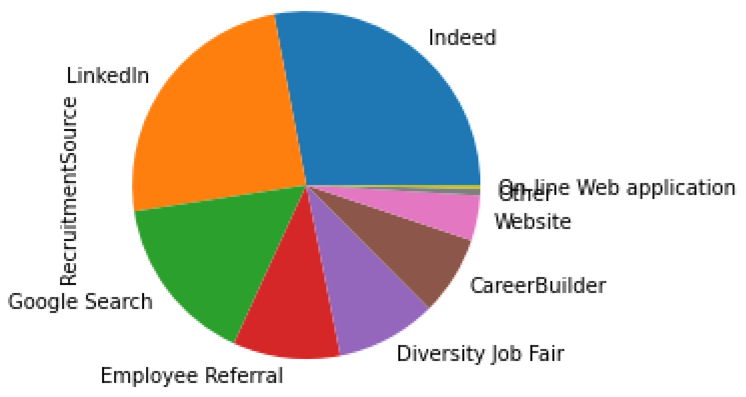
	Employee_Name	EmpID	MarriedID	MaritalStatusID	GenderID	EmpStatusID	DeptID	PerfScoreID	FromDivers
0	Adinolfi, Wilson K	10026	0	0	1	1	5	4	
1	Ait Sidi, Karthikeyan	10084	1	1	1	5	3	3	
2	Akinkuolie, Sarah	10196	1	1	0	5	5	3	
3	Alagbe,Trina	10088	1	1	0	1	5	3	
4	Anderson, Carol	10069	0	2	0	5	5	3	
...	...	...	...	...	...	...	...	...	
306	Woodson, Jason	10135	0	0	1	1	5	3	
307	Ybarra, Catherine	10301	0	0	0	5	5	1	
308	Zamora, Jennifer	10010	0	0	0	1	3	4	
309	Zhou, Julia	10043	0	0	0	1	3	3	
310	Zima, Colleen	10271	0	4	0	1	5	3	

311 rows × 36 columns

In [ ]:

In [4]:

```
df["RecruitmentSource"].value_counts().plot(kind="pie")
plt.show()
```

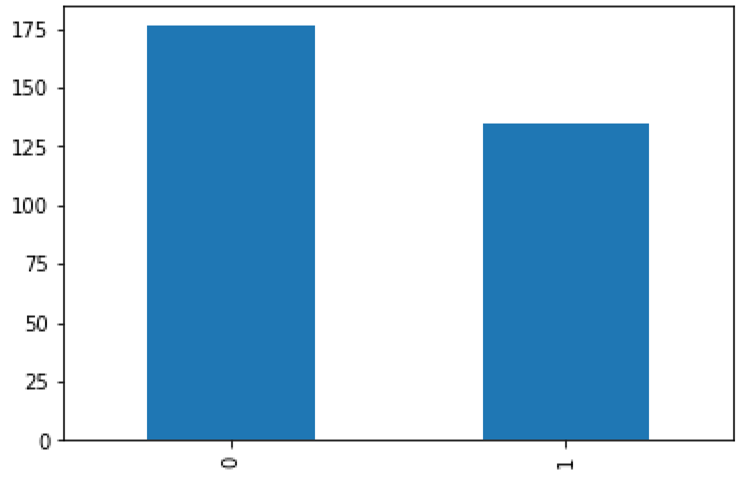


In [5]:

```
df["MarriedID"].value_counts()
#not_married 187
#married 124
```

Out[5]:
```
0    187
1    124
Name: MarriedID, dtype: int64
```

In [6]:
```
df["GenderID"].value_counts().plot(kind="bar")
plt.show()
# 0 represent the male
# 1 represent the female
```



In [7]:
```
df=pd.read_csv("HR_comma_sep.csv")
```

In [8]:
```
df
```

Out[8]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_acciden
0	0.38	0.53	2	157	3	
1	0.80	0.86	5	262	6	
2	0.11	0.88	7	272	4	
3	0.72	0.87	5	223	5	
4	0.37	0.52	2	159	3	
...	...	...	...	...	...	
14994	0.40	0.57	2	151	3	
14995	0.37	0.48	2	160	3	
14996	0.37	0.53	2	143	3	
14997	0.11	0.96	6	280	4	
14998	0.37	0.52	2	158	3	

14999 rows × 10 columns

In [9]:
```
df.pop('satisfaction_level')
```

Out[9]:
```
0        0.38
```

```
1        0.80
2        0.11
3        0.72
4        0.37
         ...
14994    0.40
14995    0.37
14996    0.37
14997    0.11
14998    0.37
Name: satisfaction_level, Length: 14999, dtype: float64
```

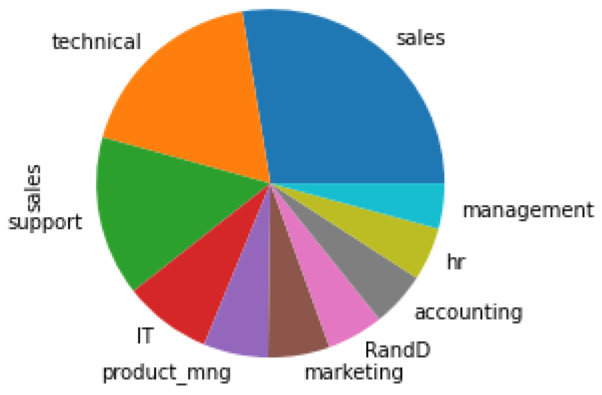
In [10]:
```
df
```

Out[10]:

	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion
0	0.53	2	157	3	0	1	
1	0.86	5	262	6	0	1	
2	0.88	7	272	4	0	1	
3	0.87	5	223	5	0	1	
4	0.52	2	159	3	0	1	
...	...	...	...	...	...	...	
14994	0.57	2	151	3	0	1	
14995	0.48	2	160	3	0	1	
14996	0.53	2	143	3	0	1	
14997	0.96	6	280	4	0	1	
14998	0.52	2	158	3	0	1	

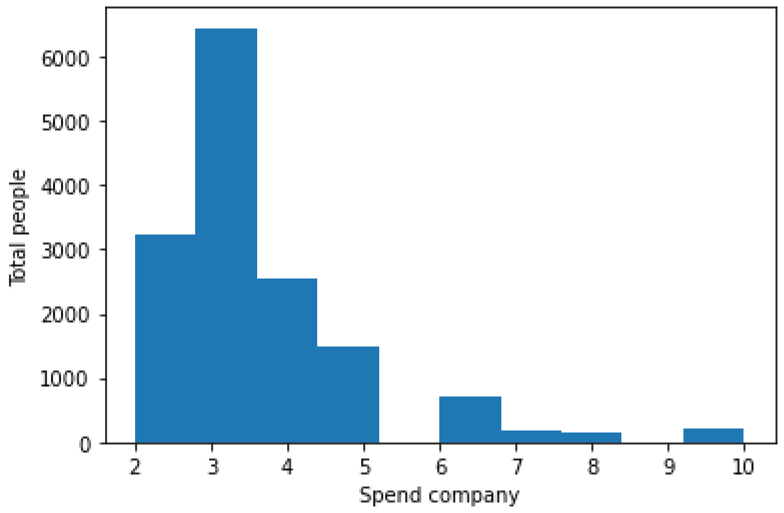
14999 rows × 9 columns

In [11]:
```
df["sales"].value_counts().plot(kind="pie")
plt.show()
```



In [12]:
```
plt.hist(df['time_spend_company'])
plt.xlabel("Spend company")
plt.ylabel("Total people")
```

Out[12]:
```
Text(0, 0.5, 'Total people')
```



In [17]:
```
df.columns
```

Out[17]:
```
Index(['last_evaluation', 'number_project', 'average_montly_hours',
       'time_spend_company', 'Work_accident', 'left', 'promotion_last_5years',
       'sales', 'salary'],
      dtype='object')
```

In [18]:
```
df
```

Out[18]:

	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion
0	0.53	2	157	3	0	1	
1	0.86	5	262	6	0	1	
2	0.88	7	272	4	0	1	
3	0.87	5	223	5	0	1	
4	0.52	2	159	3	0	1	
...	...	...	...	...	...	...	
14994	0.57	2	151	3	0	1	
14995	0.48	2	160	3	0	1	
14996	0.53	2	143	3	0	1	
14997	0.96	6	280	4	0	1	
14998	0.52	2	158	3	0	1	

14999 rows × 9 columns

In [19]:
```
target =df.pop('salary')
```

In [20]:
```
df
```

Out[20]:

	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion
0	0.53	2	157	3	0	1	
1	0.86	5	262	6	0	1	

	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion
2	0.88	7	272	4	0	1	
3	0.87	5	223	5	0	1	
4	0.52	2	159	3	0	1	
...	...	...	...	...	...	...	
14994	0.57	2	151	3	0	1	
14995	0.48	2	160	3	0	1	
14996	0.53	2	143	3	0	1	
14997	0.96	6	280	4	0	1	
14998	0.52	2	158	3	0	1	

14999 rows × 8 columns

In [21]:
```
char_cols =df.dtypes.pipe(lambda x:x[x=='objects']).index
```

In [22]:
```
label_maping={}
for c in char_cols:
    df[c],label_maping[c]=pd.factorize(df[c])
```

In [23]:
```
df.head()
```

Out[23]:

	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last
0	0.53	2	157	3	0	1	
1	0.86	5	262	6	0	1	
2	0.88	7	272	4	0	1	
3	0.87	5	223	5	0	1	
4	0.52	2	159	3	0	1	

In [27]:
```
df.pop('sales')// removing because it contains string datatype
```

Out[27]:
```
0          sales
1          sales
2          sales
3          sales
4          sales
          ...
14994    support
14995    support
14996    support
14997    support
14998    support
Name: sales, Length: 14999, dtype: object
```

In [28]:
```
df
```

Out[28]:

	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion

	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion
0	0.53	2	157	3	0	1	
1	0.86	5	262	6	0	1	
2	0.88	7	272	4	0	1	
3	0.87	5	223	5	0	1	
4	0.52	2	159	3	0	1	
...	...	...	...	...	...	...	
14994	0.57	2	151	3	0	1	
14995	0.48	2	160	3	0	1	
14996	0.53	2	143	3	0	1	
14997	0.96	6	280	4	0	1	
14998	0.52	2	158	3	0	1	

14999 rows × 7 columns

In [29]:
```
scaler =StandardScaler()
df=scaler.fit_transform(df.values)
```

In [30]:
```
from sklearn.model_selection import train_test_split
```

In [31]:
```
X_train, X_test, y_train, y_test = train_test_split(df, target, test_size=0.33, random_sta
```

In [32]:
```
cif=LogisticRegression().fit(X_train,y_train)
pred=cif.predict(X_test)
print("training complete")
```

```
training complete
```

In [34]:
```
y_test
```

Out[34]:
```
6723     medium
6473        low
4679        low
862         low
7286        low
          ...
6889     medium
9187     medium
13352      high
655      medium
14273    medium
Name: salary, Length: 4950, dtype: object
```

In [35]:
```
pred
```

Out[35]:
```
array(['low', 'low', 'low', ..., 'low', 'low', 'low'], dtype=object)
```

In [36]:
```
print(classification_report(y_test,pred))
```

```
              precision    recall  f1-score   support

        high       0.00      0.00      0.00       406
         low       0.51      0.79      0.62      2416
      medium       0.48      0.27      0.35      2128

    accuracy                           0.50      4950
   macro avg       0.33      0.36      0.32      4950
weighted avg       0.46      0.50      0.45      4950

C:\Users\amrit\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Undefi
nedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels wit
h no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\amrit\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Undefi
nedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels wit
h no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\amrit\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1248: Undefi
nedMetricWarning: Precision and F-score are ill-defined and being set to 0.0 in labels wit
h no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [42]:

```
from sklearn import metrics
from sklearn.metrics import confusion_matrix
metrics.f1_score(y_test, pred, average='weighted')
```

Out[42]:

```
0.4527687277614251
```

In [ ]:

In [ ]: