In [1]:
```
import pandas as pd
from matplotlib import pyplot as plt
%matplotlib inline
```

In [2]:
```
df = pd.read_csv("HR_comma_sep.csv")
```

In [3]:
```
df.head()
```

Out[3]:

satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5ye
0.38	0.53	2	157	3	0	1	
0.80	0.86	5	262	6	0	1	
0.11	0.88	7	272	4	0	1	
0.72	0.87	5	223	5	0	1	
0.37	0.52	2	159	3	0	1	

In [7]:
```
# Renaming certain columns for better readability
```

In [8]:
```
df = df.rename(columns={'satisfaction_level': 'satisfaction',
                        'last_evaluation': 'evaluation',
                        'number_project': 'projectCount',
                        'average_montly_hours': 'averageMonthlyHours',
                        'time_spend_company': 'yearsAtCompany',
                        'Work_accident': 'workAccident',
                        'promotion_last_5years': 'promotion',
                        'sales' : 'department',
                        'left' : 'turnover'
                        })
```

In [9]:
```
turnover = df[df.turnover==1]
turnover.shape
```

Out[9]:
```
(3571, 10)
```

In [10]:
```
retained = df[df.turnover==0]
retained.shape
```

Out[10]:
```
(11428, 10)
```

In [11]:
```
#Average numbers for all columns
```

In [12]:
```
df.groupby("turnover").mean()
```

Out[12]:

turnover	satisfaction	evaluation	projectCount	averageMonthlyHours	yearsAtCompany	workAccident	promotion
0	0.666810	0.715473	3.786664	199.060203	3.380032	0.175009	0.026251
1	0.440098	0.718113	3.855503	207.419210	3.876505	0.047326	0.005321

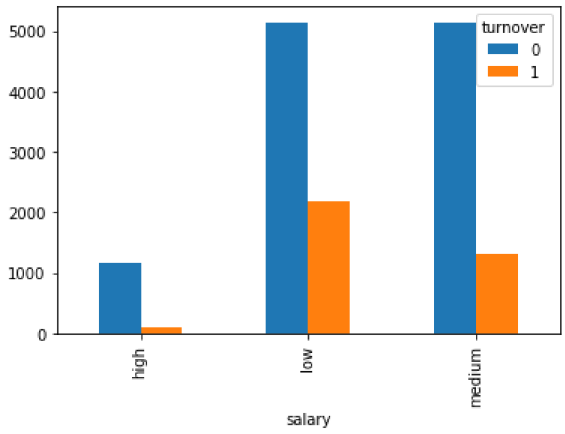
In [14]:
```
#From above table we can draw following conclusions,

#Satisfaction Level**: Satisfaction level seems to be relatively low (0.44) in employees leaving the firm vs
#Average Monthly Hours**: Average monthly hours are higher in employees leaving the firm (199 vs 207)
#Promotion Last 5 Years**: Employees who are given promotion are likely to be retained at firm
```

In [15]:
```
#Impact of salary on employee retention
```

In [16]:
```
pd.crosstab(df.salary,df.turnover).plot(kind="bar")
```

Out[16]:
```
<AxesSubplot:xlabel='salary'>
```

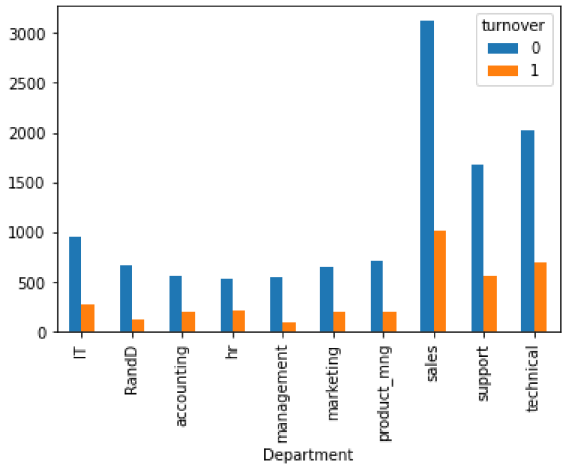


In [ ]:
```
# We can understand from above chart that , High salaried employees are not likley to leave the company.
```

In [17]:
```
#Department wise employee retention rate
```

In [19]:
```
pd.crosstab(df.Department,df.turnover).plot(kind="bar")
```

Out[19]:
```
<AxesSubplot:xlabel='Department'>
```



In [20]:
```
can assume that , there no significant connection between department and turnover.Hence we will ignore departm
```

In [21]:
```
#From the data analysis so far we can conclude that we will use following variables as independant variables
#satisfaction
#averageMonthlyHours
#promotion
#Salary
```

In [22]:
```
subdf = df[['satisfaction','averageMonthlyHours','promotion','salary']]
subdf.head()
```

Out[22]:

	satisfaction	averageMonthlyHours	promotion	salary
0	0.38	157	0	low
1	0.80	262	0	medium
2	0.11	272	0	medium
3	0.72	223	0	low
4	0.37	159	0	low

In [26]:
```
#Salary has all text data. It needs to be converted to numbers and we will use dummy variable for that
```

In [27]:
```
salary_dummies = pd.get_dummies(subdf.salary, prefix="salary")
```

In [28]:
```
df_with_dummies = pd.concat([subdf,salary_dummies],axis='columns')
```

In [29]:
```
df_with_dummies.head()
```

Out[29]:

	satisfaction	averageMonthlyHours	promotion	salary	salary_high	salary_low	salary_medium
0	0.38	157	0	low	0	1	0
1	0.80	262	0	medium	0	0	1
2	0.11	272	0	medium	0	0	1
3	0.72	223	0	low	0	1	0
4	0.37	159	0	low	0	1	0

In [30]:
```
#Now we need to remove salary column which is text data.
```

In [31]:
```
df_with_dummies.drop('salary',axis='columns',inplace=True)
df_with_dummies.head()
```

Out[31]:

	satisfaction	averageMonthlyHours	promotion	salary_high	salary_low	salary_medium
0	0.38	157	0	0	1	0
1	0.80	262	0	0	0	1
2	0.11	272	0	0	0	1
3	0.72	223	0	0	1	0
4	0.37	159	0	0	1	0

In [32]:
```
X = df_with_dummies
X.head()
```

Out[32]:

	satisfaction	averageMonthlyHours	promotion	salary_high	salary_low	salary_medium
0	0.38	157	0	0	1	0
1	0.80	262	0	0	0	1
2	0.11	272	0	0	0	1
3	0.72	223	0	0	1	0
4	0.37	159	0	0	1	0

In [33]:
```
y = df.turnover
```

In [34]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,train_size=0.3)
```

In [36]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
```

In [37]:
```
model.fit(X_train, y_train)
```

Out[37]:
```
LogisticRegression()
```

In [38]:
```
model.predict(X_test)
```

Out[38]:
```
array([0, 0, 1, ..., 0, 0, 0], dtype=int64)
```

In [39]:
```
#Accuracy of the model
```

In [40]:
```
model.score(X_test,y_test)
```

Out[40]:
```
0.7754285714285715
```

In [ ]: