EXERCISE 4

Download employee retention data set from Kaggle

1. Now do some exploratory data analysis to figure out which variables have direct and clear impact on employee retention (i.e. whether they leave the company or continue to work)
2. Plot bar charts showing impact of employee salaries on retention
3. Plot bar charts showing corelation between department and employee retention
4. Now build logistic regression model using variables that were narrowed down in step 1
5. Measure the accuracy of the model

In [58]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression
```

In [59]:
```
data = pd.read_csv('HR_comma_sep.csv')
data
```

Out[59]:

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

14999 rows × 10 columns

In [60]:
```
data.isnull()
```

Out[60]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	pron
0	False	False	False	False	False	False	False	
1	False	False	False	False	False	False	False	
2	False	False	False	False	False	False	False	
3	False	False	False	False	False	False	False	
4	False	False	False	False	False	False	False	
...	...	...	...	...	...	...	...	
14994	False	False	False	False	False	False	False	
14995	False	False	False	False	False	False	False	
14996	False	False	False	False	False	False	False	
14997	False	False	False	False	False	False	False	
14998	False	False	False	False	False	False	False	

14999 rows × 10 columns

In [61]:
```
data.tail().duplicated()
```

Out[61]:
```
14994    False
14995    False
14996    False
14997    False
14998    False
dtype: bool
```

In [62]:
```
data.shape
```

Out[62]: (14999, 10)

In [63]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 14999 entries, 0 to 14998
Data columns (total 10 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   satisfaction_level     14999 non-null  float64
 1   last_evaluation        14999 non-null  float64
 2   number_project         14999 non-null  int64
 3   average_montly_hours   14999 non-null  int64
 4   time_spend_company     14999 non-null  int64
 5   Work_accident          14999 non-null  int64
 6   left                   14999 non-null  int64
 7   promotion_last_5years  14999 non-null  int64
 8   Department             14999 non-null  object
 9   salary                 14999 non-null  object
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
```

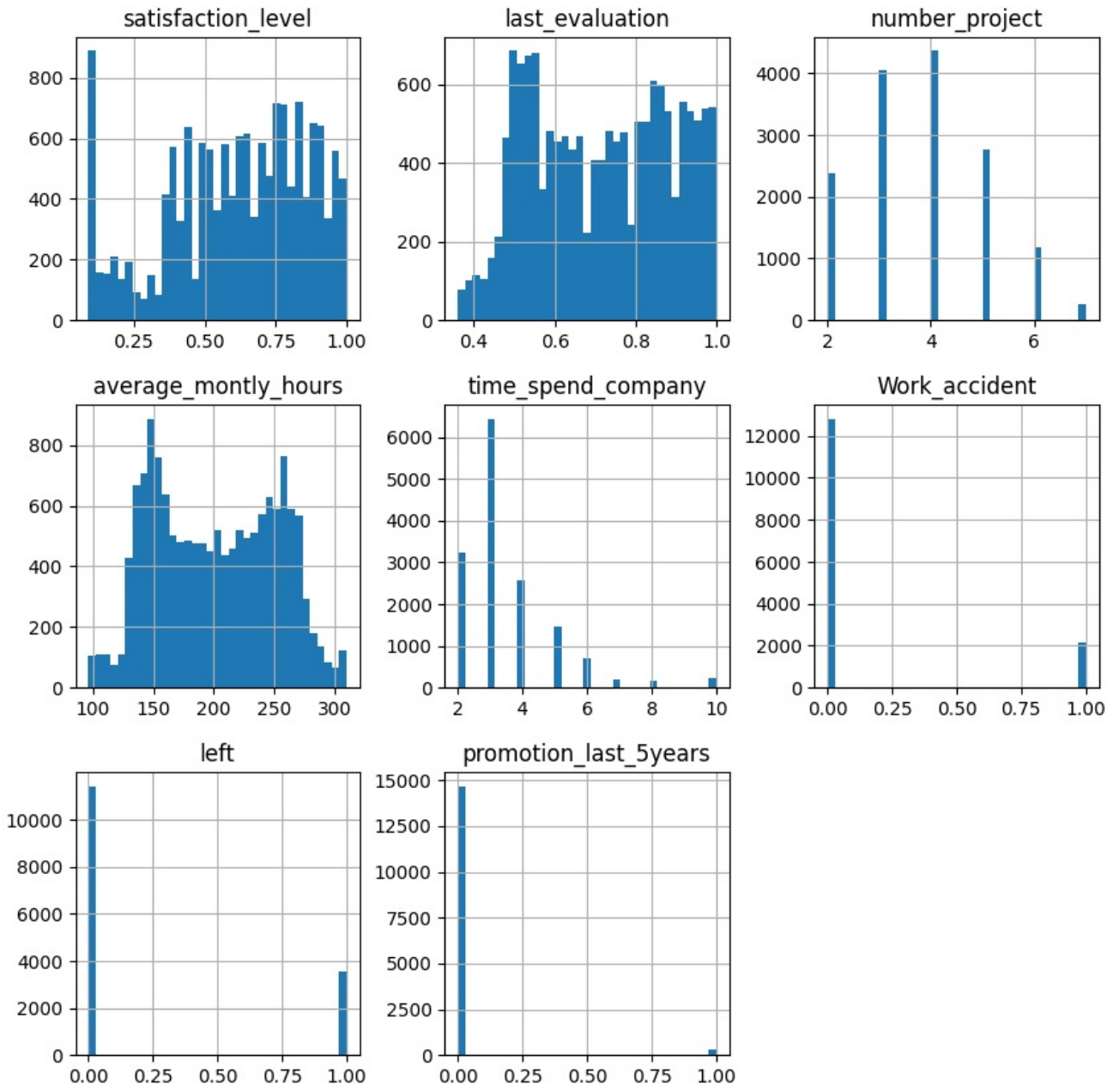
In [64]:
```
data.describe()
```

Out[64]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	le
count	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.0000(
mean	0.612834	0.716102	3.803054	201.050337	3.498233	0.144610	0.2380{
std	0.248631	0.171169	1.232592	49.943099	1.460136	0.351719	0.4259{
min	0.090000	0.360000	2.000000	96.000000	2.000000	0.000000	0.0000(
25%	0.440000	0.560000	3.000000	156.000000	3.000000	0.000000	0.0000(
50%	0.640000	0.720000	4.000000	200.000000	3.000000	0.000000	0.0000(
75%	0.820000	0.870000	5.000000	245.000000	4.000000	0.000000	0.0000(
max	1.000000	1.000000	7.000000	310.000000	10.000000	1.000000	1.0000(

In [65]:
```
data.hist(bins=35, figsize=(10, 10))
```

Out[65]:
```
array([[<Axes: title={'center': 'satisfaction_level'}>,
        <Axes: title={'center': 'last_evaluation'}>,
        <Axes: title={'center': 'number_project'}>],
       [<Axes: title={'center': 'average_montly_hours'}>,
        <Axes: title={'center': 'time_spend_company'}>,
        <Axes: title={'center': 'Work_accident'}>],
       [<Axes: title={'center': 'left'}>,
        <Axes: title={'center': 'promotion_last_5years'}>, <Axes: >]],
      dtype=object)
```



In [66]:
```
left = data[data['left'] == 1]
left.shape
```

Out[66]: (3571, 10)

In [67]:
```
retained = data[data.left == 0]
retained.shape
```

Out[67]: (11428, 10)

Average numbers for all columns

In [71]:
```
test = data.drop(['salary', 'Department'], axis=1)
test.groupby('left').mean()
```

Out[71]:

left	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	promotion_las
0	0.666810	0.715473	3.786664	199.060203	3.380032	0.175009	
1	0.440098	0.718113	3.855503	207.419210	3.876505	0.047326	

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 the retained ones (0.66)

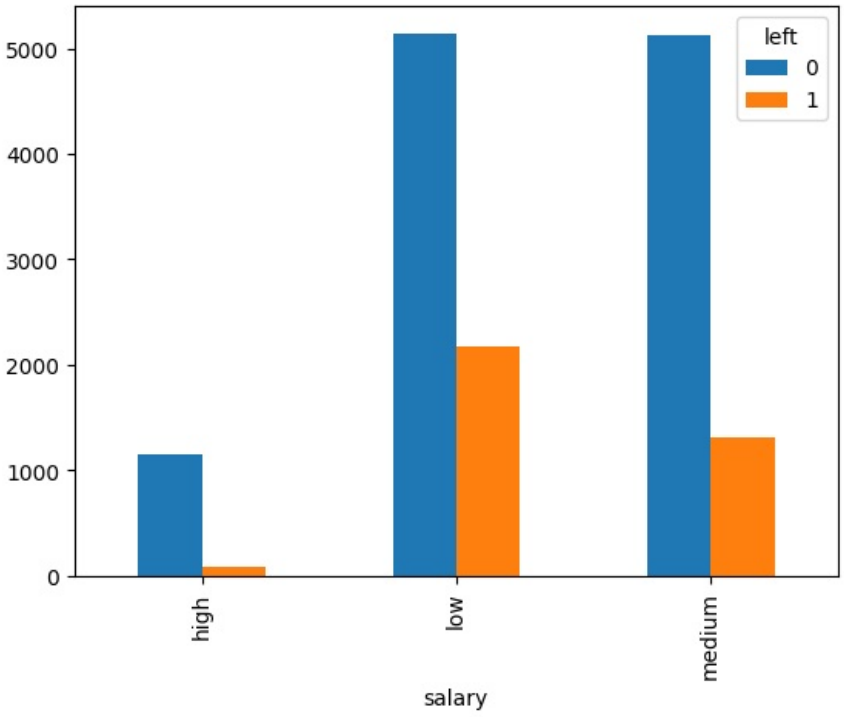
Average Monthly Hours: Average monthly hours are higher in employees leaving the firm (199 vs 207)

Promotion Last 5 Years: Employees who are given promotion are likely to be retained at firm

Impact of salary on employee retention

In [72]:
```
pd.crosstab(data.salary,data.left).plot(kind='bar')
```

Out[72]: <Axes: xlabel='salary'>

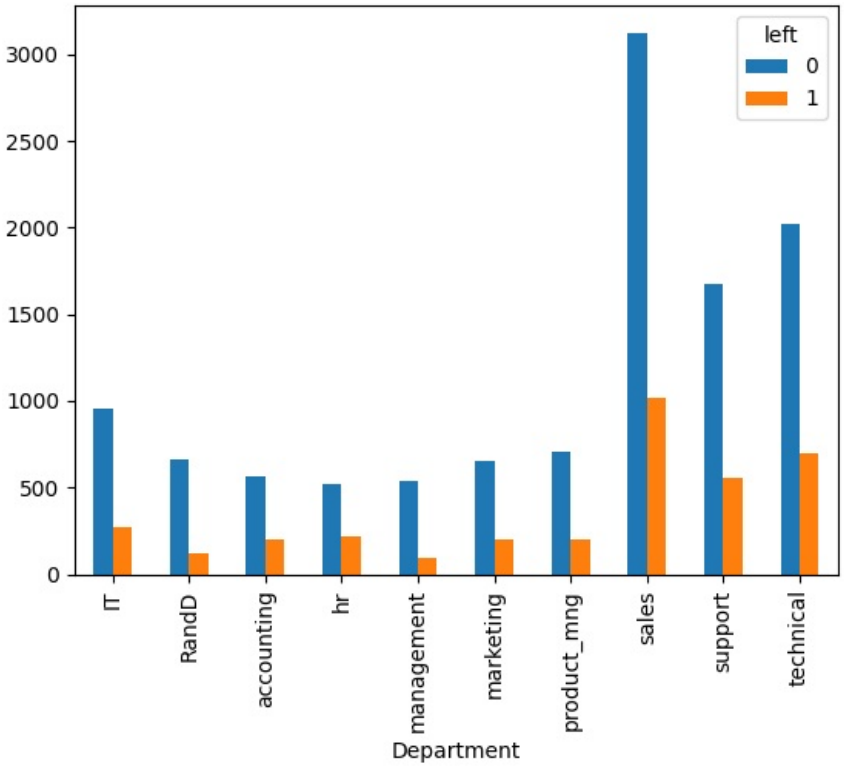


Above bar chart shows employees with high salaries are likely to not leave the company

Department wise employee retention rate

In [74]:
```
pd.crosstab(data.Department,data.left).plot(kind='bar')
```

Out[74]: <Axes: xlabel='Department'>



From above chart there seem to be some impact of department on employee retention but it is not major hence we will ignore department in our analysis

From the data analysis so far we can conclude that we will use following variables as independant variables in our model

1. **Satisfaction Level**
2. **Average Monthly Hours**
3. **Promotion Last 5 Years**
4. **Salary**

In [75]:
```
newdata = data[['satisfaction_level','average_montly_hours','promotion_last_5years','salary']]
newdata.head()
```

Out[75]:

	satisfaction_level	average_montly_hours	promotion_last_5years	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

Tackle salary dummy variable

Salary has all text data. It needs to be converted to numbers and we will use dummy variable for that. Check my one hot encoding tutorial to understand purpose behind dummy variables.

In [87]:
```
dummies = pd.get_dummies(newdata.salary, prefix="salary").astype(int)
dummies.head()
```

Out[87]:

	salary_high	salary_low	salary_medium
0	0	1	0
1	0	0	1
2	0	0	1
3	0	1	0
4	0	1	0

In [88]:
```
merged_data = pd.concat([newdata, dummies], axis=1)
merged_data.head()
```

Out[88]:

	satisfaction_level	average_montly_hours	promotion_last_5years	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 [90]:
```
finaldata = merged_data.drop(['salary', 'salary_medium'], axis=1)
finaldata.head()
```

Out[90]:

	satisfaction_level	average_montly_hours	promotion_last_5years	salary_high	salary_low
0	0.38	157	0	0	1
1	0.80	262	0	0	0
2	0.11	272	0	0	0
3	0.72	223	0	0	1
4	0.37	159	0	0	1

In [91]:
```
X = finaldata
X.head()
```

Out[91]:

	satisfaction_level	average_montly_hours	promotion_last_5years	salary_high	salary_low
0	0.38	157	0	0	1
1	0.80	262	0	0	0
2	0.11	272	0	0	0
3	0.72	223	0	0	1
4	0.37	159	0	0	1

In [92]:
```
y = data.left
y.head()
```

Out[92]:
```
0    1
1    1
2    1
3    1
4    1
Name: left, dtype: int64
```

In [93]:
```
from sklearn.model_selection import train_test_split
```

In [94]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

In [95]:
```
model = LogisticRegression()
```

In [96]:
```
model.fit(X_train, y_train)
```

Out[96]:
```
▾ LogisticRegression
LogisticRegression()
```

In [99]:
```
predict = model.predict(X_test)
```

In [102…
```
model.score(X_test, y_test)
```

Out[102]: 0.7823333333333333

In [105…

```
obj = {
    "Actual" : y_test,
    "Predicted" : predict
}

testdf = pd.DataFrame(obj)
testdf.head()
```

Out[105]:

	Actual	Predicted
10283	0	0
12461	1	0
12404	1	0
2658	0	0
5898	0	1