Practice

Import Library

In [2]:
```
# Mengimpor library-library yang diperlukan

# Pandas untuk impor, preprocessing data dan manipulasi data
import pandas as pd

# Numpy untuk perhtungan matematika
import numpy as np

# Matplotlib & Seaborn untuk visualisasi data
import matplotlib.pyplot as plt
import seaborn as sns

# Solusi agar visualisasi dapat ditampilkan
# Karena dalam beberapa case visualisasi tidak dapat running
%matplotlib.inline

# Untuk memilih style visualisasi data
plt.style.use('ggplot')
```

Kumpulan style di Matplotlib:

https://matplotlib.org/stable/gallery/style_sheets/style_sheets_reference.html

Import Dataset

In [3]:
```
# Mengimport dataset yang diperlukan, dalam case ini berbentuk csv (comma seperated value)
# Melakukan impor data dengan menggunakan pandas
data = pd.read_csv('HR_comma_sep.csv')
```

Tips:

jika ingin mengetahui fungsi apa saja di dalam pandas (library-library lain), cara tercepatnya adalah setelah tanda titik tekan TAB. Selain itu, jika ingin melihat help atau dokumentasi dapat dilakukan dengan cara SHIFT + TAB setelah tanda titik

Quick Look

In [4]:
```
# Melihat ringkasan 5 observasi data teratas
data.head()
```

Out[4]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_last_5years	s
0	0.38	0.53	2	157	3	0	1	0	s
1	0.80	0.86	5	262	6	0	1	0	s
2	0.11	0.88	7	272	4	0	1	0	s
3	0.72	0.87	5	223	5	0	1	0	s
4	0.37	0.52	2	159	3	0	1	0	s

In [6]:
```
# Melihat info dari masing-masing kolom
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   sales                  14999 non-null  object
 9   salary                 14999 non-null  object
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
```

In [5]:
```
# Melihat info dari masing-masing kolom
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   sales                  14999 non-null  object
 9   salary                 14999 non-null  object
dtypes: float64(2), int64(6), object(2)
memory usage: 1.1+ MB
```

1. Non-Null Count: Untuk mengetahui apakah terdapat data kosong dan berapa banyak jumlahnya. Dalam kasus ini menunjukan hasil non-null yang berarti tidak ada data yang kosong. Selain itu, angka 14999 menunjukan jumlah data.
2. Dtype: float64 menandakan tipe data float (biasanya continuos), int64 menandakan tipe data interger (biasanya discrete), object menandakan tipe data string atau nominal/ordinal.

Standard Describe

In [7]:
```
# Melihat ringkasan statistik
data.describe()
```

Out[7]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left	promotion_
count	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14999.000000	14
mean	0.612834	0.716102	3.803054	201.050337	3.498233	0.144610	0.238083	
std	0.248631	0.171169	1.232592	49.943099	1.460136	0.351719	0.425924	
min	0.090000	0.360000	2.000000	96.000000	2.000000	0.000000	0.000000	
25%	0.440000	0.560000	3.000000	156.000000	3.000000	0.000000	0.000000	
50%	0.640000	0.720000	4.000000	200.000000	3.000000	0.000000	0.000000	
75%	0.820000	0.870000	5.000000	245.000000	4.000000	0.000000	0.000000	
max	1.000000	1.000000	7.000000	310.000000	10.000000	1.000000	1.000000	

Catatan:

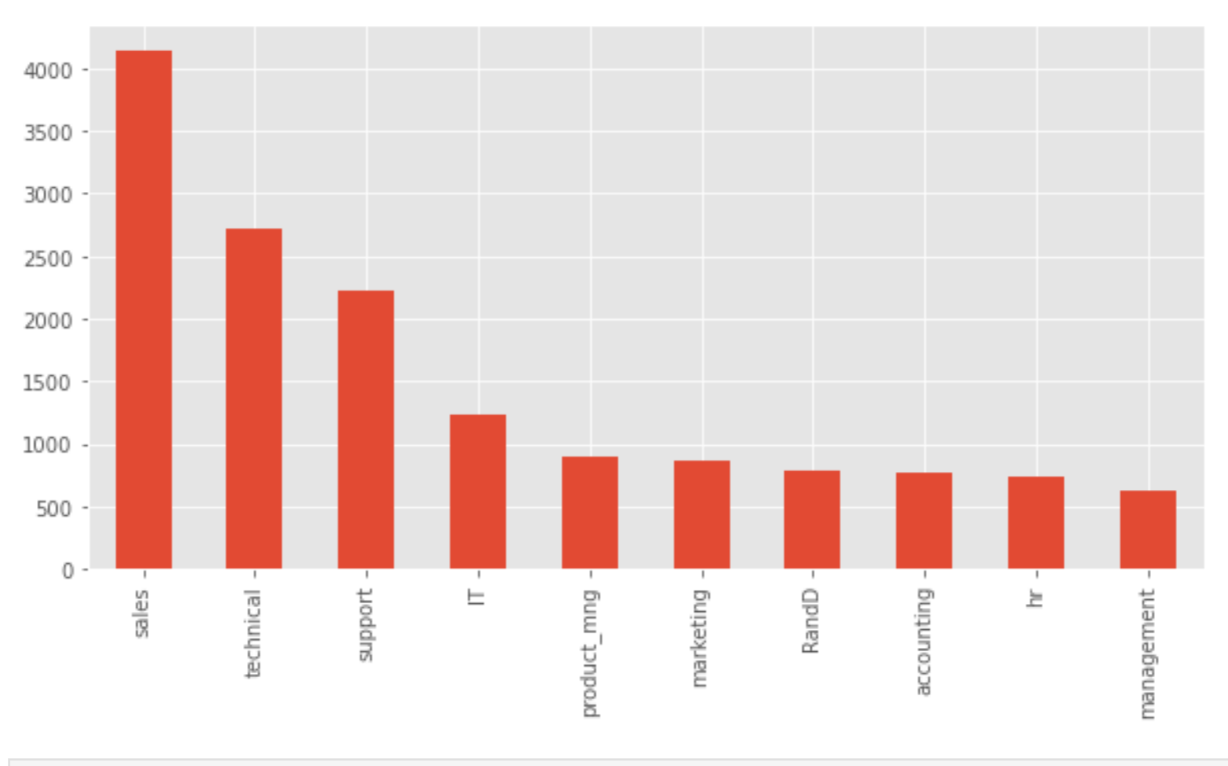
1. count: jumlah data
2. mean: rata-rata
3. std: standard deviasi
4. min: nilai minimum
5. 25%: Q1
6. 50%: Q2 atau median
7. 75%: Q3
8. max: nilai maksimum

Method describe berfungsi untuk mengetahui ukuran pemusatan dan penyebaran data dari masing-masing kolom.

Sales Barchart

In [8]:
```
# Melihat jumlah karyawan dari masing-masing divisi di Perusahaan

# Membuat figura atau bingkai. Dalam kasus ini ukuran figura yaitu panjang 10 inci dan lebar 5 inci
plt.figure(figsize = (10, 5))

# Untuk melihat jumlah karyawan dari masing-masing divisi didapat dari kolom sales
# Untuk mengetahui jumlah karyawan dapat menggunakan method value_counts()
# Method value_counts() yang mana berfungsi untuk mengetahui jumlah data dari masing-masing kategori
# Karena ingin menggunakan plot bar chart untuk visualisasi maka menggunakan plot.bar()
data['sales'].value_counts().plot.bar()
```

Out[8]:
```
<AxesSubplot:>
```



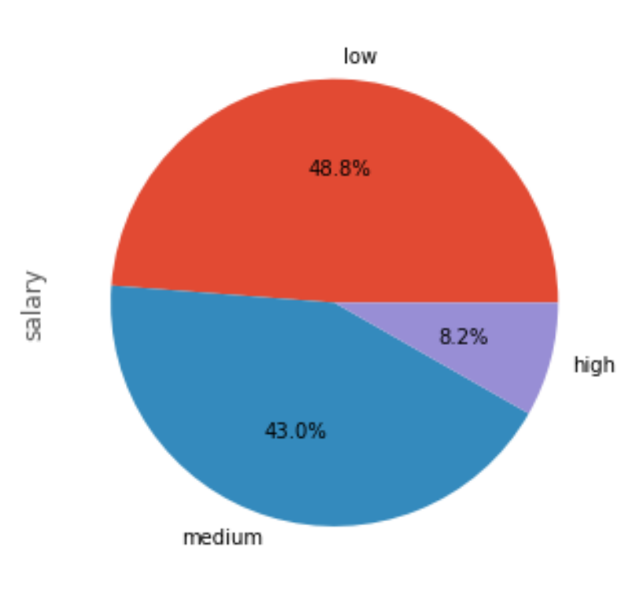
In [10]:
```
# Tanpa visualisasi
data['sales'].value_counts()
```

Out[10]:
```
sales          4140
technical      2720
support        2229
IT             1227
product_mng     902
marketing       858
RandD           787
accounting      767
hr              739
management      630
Name: sales, dtype: int64
```

Percentage of Salary

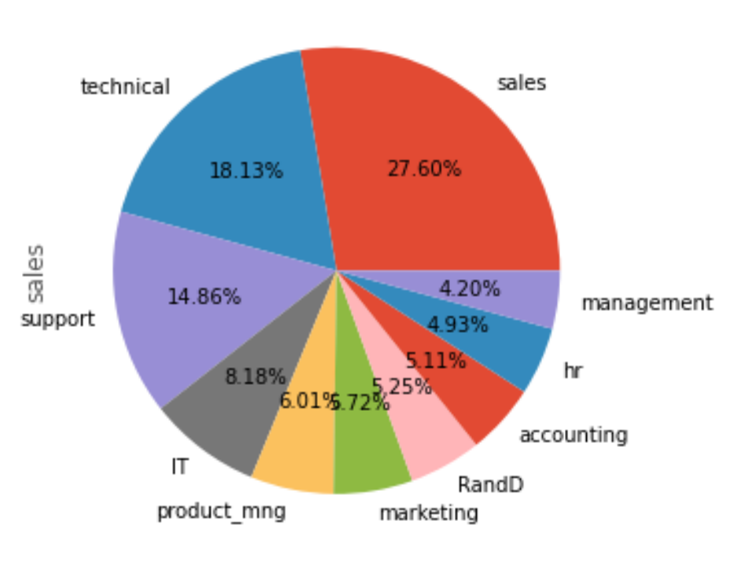
In [11]:
```
# Melihat persentase dari masing-masing tipe gaji (low, medium, high)
# Menggunakan Pie chart
data['salary'].value_counts().plot(kind = 'pie',
                                   figsize = (5, 6),
                                   autopct = '%1.1f%%')

# Kode di atas merupakan cara kedua
# kind menunjukan ingin membuat bentuk visualisasi yang seperti apa, dalam kasus ini pie chart
# figsize untuk bingkai, panjang 5 inci, lebar 6 inc
# autopct digunakan untuk mengatur berapa banyak jumlah angka dibelakang koma. 1.1 artinya ingin membuat hanya
```

Out[11]:
```
<AxesSubplot:ylabel='salary'>
```



In [12]:
```
# Jika ingin dua angka di belakang koma, maka ganti di autopct menjadi 1.2
data['sales'].value_counts().plot(kind = 'pie',
                                  figsize = (5,6),
                                  autopct = '%1.2f%%')
```

Out[12]:
```
<AxesSubplot:ylabel='sales'>
```



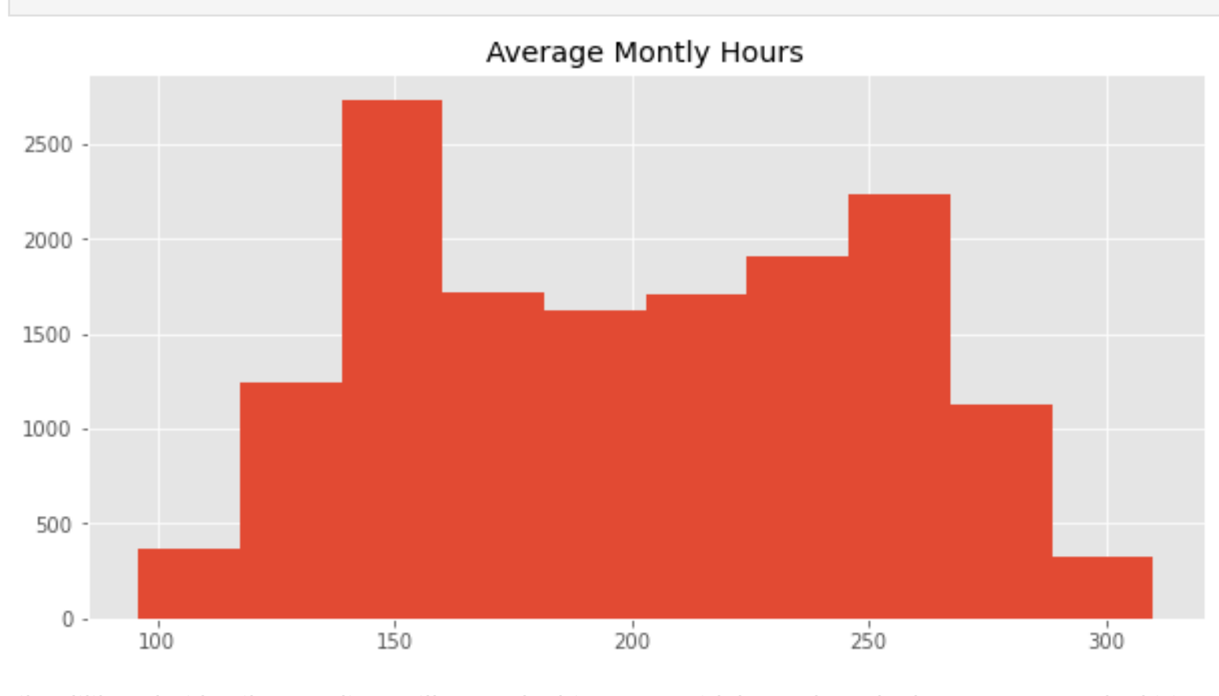
In [16]:
```
data['salary'].value_counts()
```

Out[16]:
```
low       7316
medium    6446
high      1237
Name: salary, dtype: int64
```

Dalam matplotlib dapat menggunakan beberapa cara coding

Average Montly Hours

In [13]:
```
# Cara ketiga coding matplotlib

# Membuat histogram

# 1. Membuat figura atau bingkai panjang 10 inci, lebar 5inci
plt.figure(figsize = (10, 5))

# 2. Membuat histogram, default binnya 10 bisa dilihat di dokumentasi
plt.hist(data['average_montly_hours'])

# 3. Membuat judul
plt.title('Average Montly Hours') # ini bebas ditulis apapun yang kita suka

# 4.  Menampilkan hasil
plt.show()
```



Jika dilihat dari hasil yang ditampilkan maka histogram tidak condong ke kanan maupun ke kiri atau dapat dikatakan terdistribusi normal. Namum memiliki dua puncak, hal tersebut dikenal dengan bimodal. Jika puncaknya hanya satu saja maka disebut unimodal. Jika puncaknya lebih dari dua disebut multimodal.

Memerikasa Pemusatan dan Penyebaran data secara terpisah

Mean

In [14]:
```
# Mengetahui rata-rata atau mean hanya kolom time_spend_company
data['time_spend_company'].mean()
```

Out[14]:
```
3.498233215547703
```

In [15]:
```
# Mengetahui rata-rata atau mean hanya kolom promotion_last_5years
data['promotion_last_5years'].mean()
```

Out[15]:
```
0.021268084538969265
```

Median

In [16]:
```
# Mengetahui nilai Median hanya pada kolom time_spend_company
data['time_spend_company'].median()
```

Out[16]:
```
3.0
```

In [17]:
```
# Mengetahui nilai Median hanya pada kolom
data['promotion_last_5years'].median()
```

Out[17]:
```
0.0
```

Mode

Untuk mencari nilai Modus (Mode) perlu diketik [0] di akhir code.

In [18]:
```
# Mengetahui nilai Modus hanya pada kolom time_spend_company
data['time_spend_company'].mode()[0]
```

Out[18]:
```
3
```

In [19]:
```
# Mengetahui nilai Modus hanya pada kolom promotion_last_5years
data['promotion_last_5years'].mode()[0]
```

Out[19]:
```
0
```

Quartile

In [20]:
```
# Mencari Q1 (Quantile 1) pada kolom time_spend_company
data['time_spend_company'].quantile(0.25)
```

Out[20]:
```
3.0
```

In [29]:
```
# Mencari Q2 (Quantile 2) pada kolom time_spend_company
data['time_spend_company'].quantile(0.50)
```

Out[29]:
```
3.0
```

In [21]:
```
# Mencari Q3 (Quantile 3) pada kolom time_spend_company
data['time_spend_company'].quantile(0.75)
```

Out[21]:
```
4.0
```

Percentile

Catatan: fungsi quantile dapat juga untuk menghitung Percentile maupun decile. Karena konsepnya sama, jika quantile data dibagi 4 bagian, decile 10 bagian dan percentile dibagi 100 bagian.

In [23]:
```
# Mencari Percentile 90 pada kolom time_spend_company
data['time_spend_company'].quantile(0.9)
```

Out[23]:
```
5.0
```

In [24]:
```
# Mencari Percentile 90 pada kolom time_spend_company
data['time_spend_company'].quantile(0.99)
```

Out[24]:
```
10.0
```

Variance

In [25]:
```
# Mencari Variace pada kolom time_spend_company
data['time_spend_company'].var()
```

Out[25]:
```
2.1319978117222864
```

standard deviation

In [34]:
```
# Mencari Standar deviasi pada kolom time_spend_company
data['time_spend_company'].std()
```

Out[34]:
```
1.4601362305354546
```

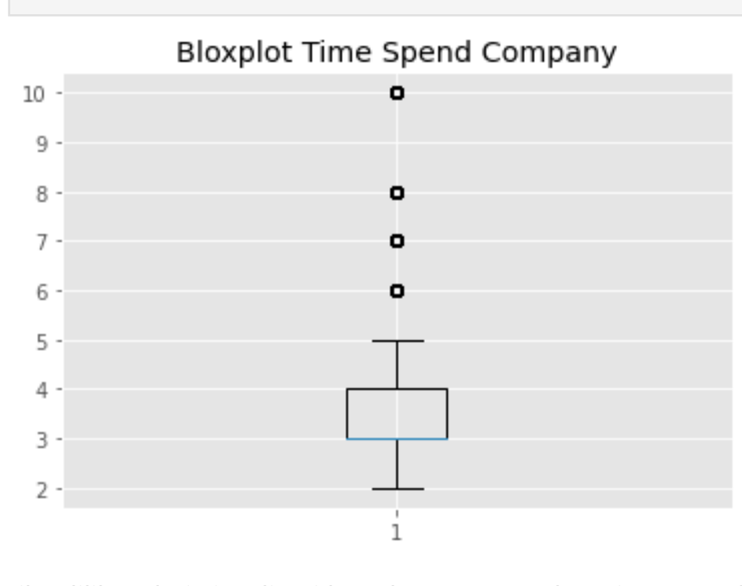
Interquartile rank

In [26]:
```
# Menghitung interquartile rank pada kolom time_spend_company

# 1. mencari Q3
q3 = data['time_spend_company'].quantile(0.75)

# 2. mencari Q1
q1 = data['time_spend_company'].quantile(0.25)

# 3. menghitunga IQR
IQR = q3 - q1
IQR
```

Out[26]:
```
1.0
```

BoxPlot

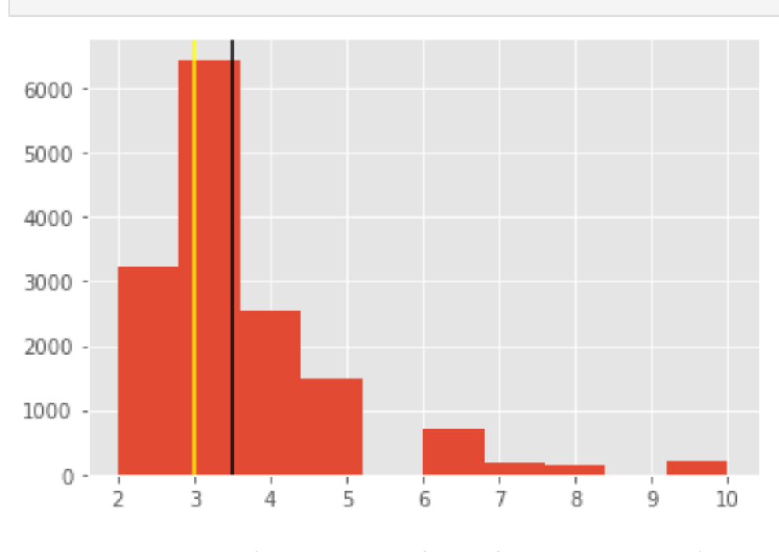
In [27]:
```
# Membuat Boxplot pada kolom time_spend_company
plt.boxplot(data['time_spend_company'])

# Membuat judul
plt.title('Bloxplot Time Spend Company')

# Menampilkan visualisasi
plt.show()
```



Jika dilihat dari visualisasi boxplot, ternyata data time spend company memiliki 4 outlier. Maka jika ingin mengukur pemusatan jangan menggunakan mean tetapi menggunakan median. Selain itu, jika ingin mengukur penyebaran data, jangan menggunakan variance atau standar deviasi tetapi gunakan IQR.

Histogram

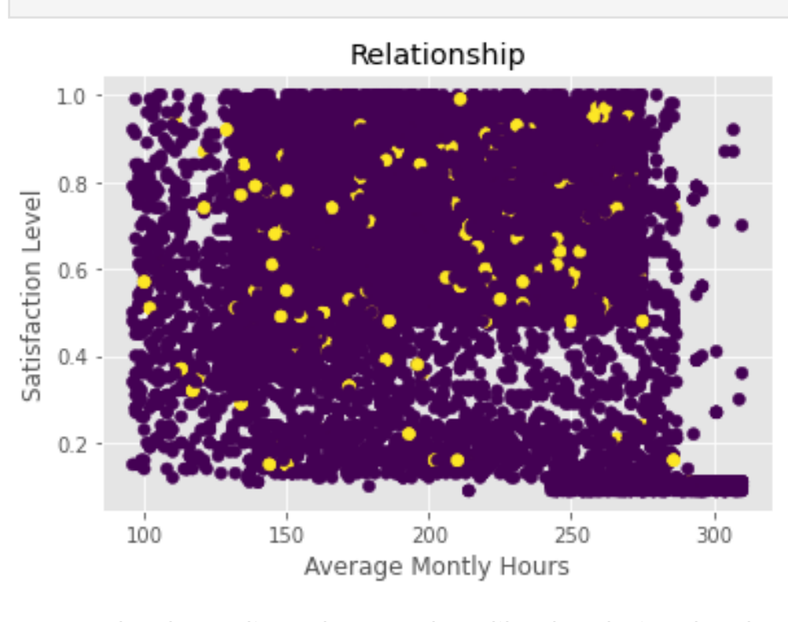
In [28]:
```
# Membuat histogram dari kolom time_spend_company
plt.hist(data['time_spend_company'])

# Menunjukan nilai median dengan garis yang diberi warna kuning
# Caranya dengan menggunakan fungsi axvline
plt.axvline(data['time_spend_company'].median(),color='yellow')

# Menunjukan nilai mean dengan garis yang diberi warna hitam
plt.axvline(data['time_spend_company'].mean(),color='black')

# menampilkan hasil
plt.show()
```



Histogram menunjukan Positive Skewed yang mana terdapat nilai outliers atas. Selain itu dapat diperkuat dengan nilai mean yang lebih besar dari nilai median

Correlation

In [29]:
```
# Membuat scatter plot
# x atau horizontal merupakan nilai pada kolom average monthly hours
# y atau vertical satisfaction level,
# c untuk melihat yang mana yang udah promosi diberi tanda
plt.scatter(x = data['average_montly_hours'], y = data['satisfaction_level'], c = data['promotion_last_5years']

# Membuat x label atau horizontal label
plt.xlabel('Average Montly Hours')

# Membuat y label atau vertical label
plt.ylabel('Satisfaction Level')

# Membuat judul
plt.title('Relationship')

# menampilkan hasil
plt.show()
```



Scatter plot dapat digunakan untuk melihat korelasi. Dalam kasus ini, dapat diketahui bahwa tidak ada korelasi antara average_montly_hour dengan satisfaction_level. Karena visualisasi menunjukan data data yang acak atau tidak membentuk pola tertentu yang dapat diartikan bahwa tidak ada hubungan antar variabel.

In [30]:
```
# Menghitung korelasi antara average_montly_hour dengan satisfaction_level
np.corrcoef(data['average_montly_hours'], data['satisfaction_level'])

# karena nilainya mendekati nol maka tidak berkorelasi
```

Out[30]:
```
array([[ 1.        , -0.02004811],
       [-0.02004811,  1.        ]])
```

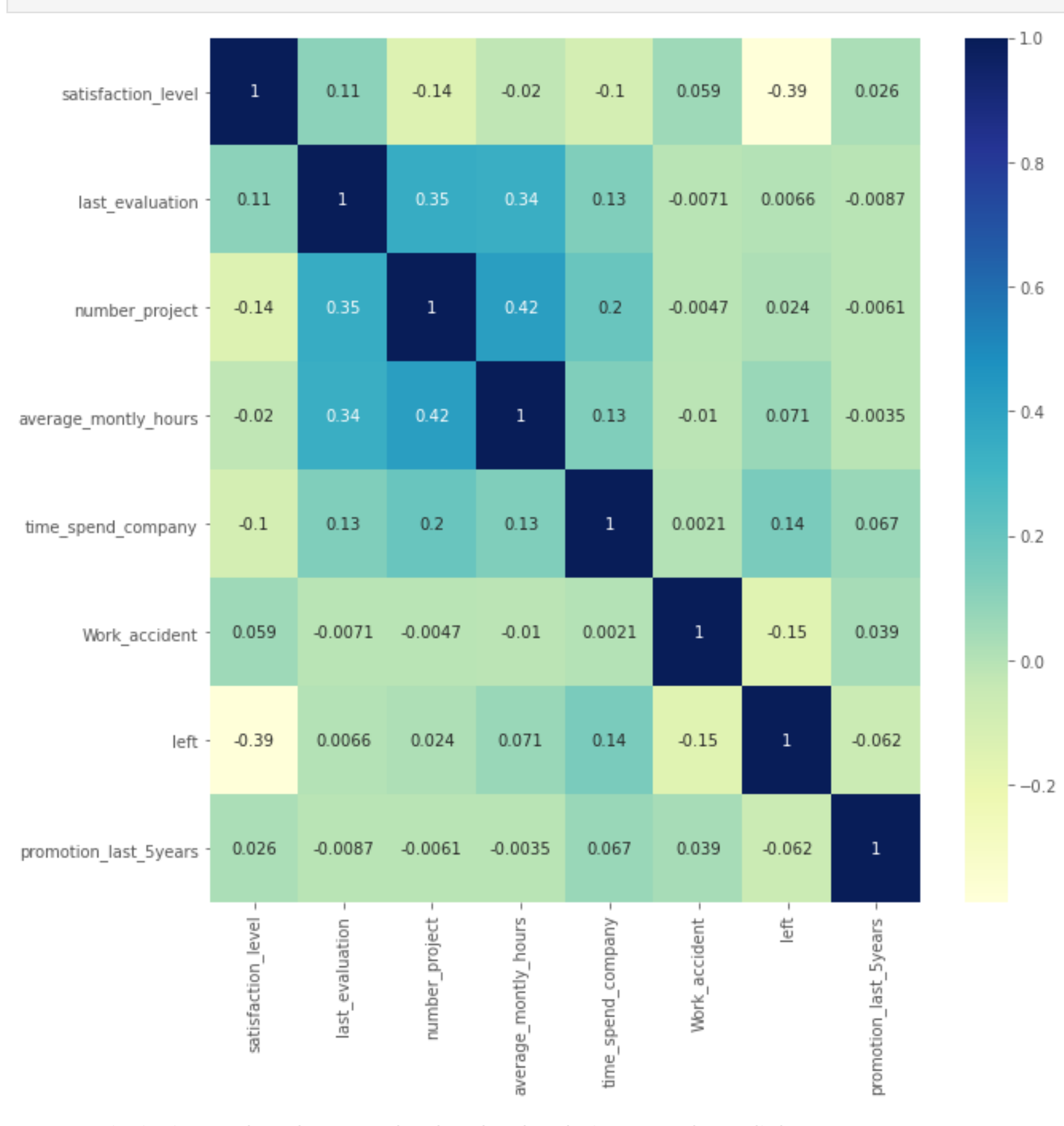
In [31]:
```
# Kita juga dapat menggunakan fungsi korr untuk mengetahui korelasi dari keseluruhan kolom yang ada di dataset
data.corr()
# Jika kolom tersebut object string dianggap tidak sah atau tidak dapat diinterpretasi
```

Out[31]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left
satisfaction_level	1.000000	0.105021	-0.142970	-0.020048	-0.100866	0.058697	-0.388375
last_evaluation	0.105021	1.000000	0.349333	0.339742	0.131591	-0.007104	0.006567
number_project	-0.142970	0.349333	1.000000	0.417211	0.196786	-0.004741	0.023787
average_montly_hours	-0.020048	0.339742	0.417211	1.000000	0.127755	-0.010143	0.071287
time_spend_company	-0.100866	0.131591	0.196786	0.127755	1.000000	0.002120	0.144822
Work_accident	0.058697	-0.007104	-0.004741	-0.010143	0.002120	1.000000	-0.154622
left	-0.388375	0.006567	0.023787	0.071287	0.144822	-0.154622	1.000000
promotion_last_5years	0.025605	-0.008684	-0.006064	-0.003544	0.067433	0.039245	-0.061788

Membuat Heat Map

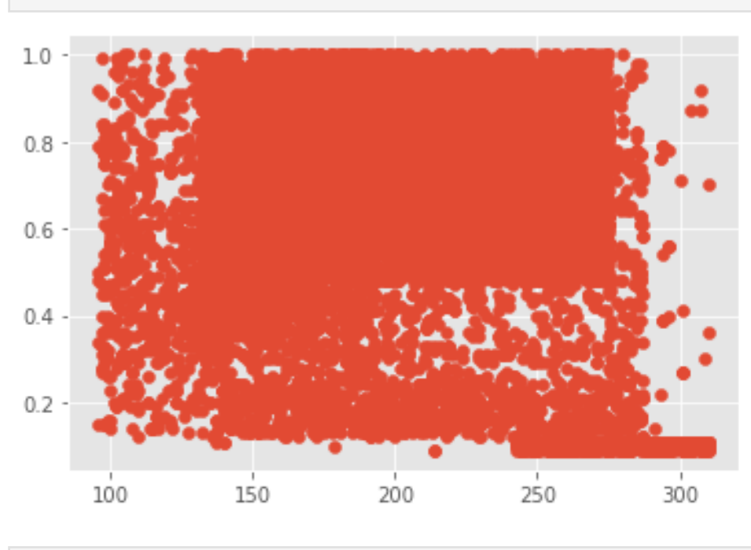
In [32]:
```
# Membuat figure atau bingkai, panjang 10, lebar 10
plt.figure(figsize = (10, 10))

# Membuat heatmap menggunakan seaborn
# Heatmap merepresentasikan korelasi antar kolom secara keseluruhan
sns.heatmap(data.corr(), annot = True, cmap = 'YlGnBu')

plt.show()
```



Catatan: 1. data.corr(): Kita ingin membuat heatmeat berdasarkan korelasi yang terdapat di data 2. Annot = True: annotasi merupakan suatu label nilai seperti 1 , -0.062 dll 3. cmap: merupakan warna. Semakin biru tua semakin besar korelasinya, sedangkan putih atau warna muda semakin kecil korelasinya

Contoh Lain

In [44]:
```
# Mengimpor matplotlib untuk visualisasi data
import matplotlib.pyplot as plt
%matplotlib inline

# menggunakan style tertentu
plt.style.use('ggplot')

# membuat scatter plot
plt.scatter(data['average_montly_hours'], data['satisfaction_level'])

# menampilkan hasil
plt.show()
```



In [45]:
```
# Mencari nilai korelasi antara kolom average_montly_hours dengan satisfaction level
np.corrcoef(data['average_montly_hours'],data['satisfaction_level'])
```

Out[45]:
```
array([[ 1.        , -0.02004811],
       [-0.02004811,  1.        ]])
```

In [46]:
```
data.corr()
```

Out[46]:

	satisfaction_level	last_evaluation	number_project	average_montly_hours	time_spend_company	Work_accident	left
satisfaction_level	1.000000	0.105021	-0.142970	-0.020048	-0.100866	0.058697	-0.388375
last_evaluation	0.105021	1.000000	0.349333	0.339742	0.131591	-0.007104	0.006567
number_project	-0.142970	0.349333	1.000000	0.417211	0.196786	-0.004741	0.023787
average_montly_hours	-0.020048	0.339742	0.417211	1.000000	0.127755	-0.010143	0.071287
time_spend_company	-0.100866	0.131591	0.196786	0.127755	1.000000	0.002120	0.144822
Work_accident	0.058697	-0.007104	-0.004741	-0.010143	0.002120	1.000000	-0.154622
left	-0.388375	0.006567	0.023787	0.071287	0.144822	-0.154622	1.000000
promotion_last_5years	0.025605	-0.008684	-0.006064	-0.003544	0.067433	0.039245	-0.061788

In [47]:
```
plt.figure(figsize=(10,10))
sns.heatmap(data.corr())
plt.show()
```