In [1]:

```
#import the useful libraries.
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline

# Read the data set of "Marketing Analysis" in data.
data= pd.read_csv("marketing_analysis.csv")

# Printing the data
data
```

```
/Users/kunalshriwas/opt/anaconda3/lib/python3.8/site-packages/IPython/core/interactiveshell.py:3071: DtypeWarning: Columns (0,1,2,3,11,14,15,16) have mixed types.Specify dtype option on import or set low_memory=False.
  has_raised = await self.run_ast_nodes(code_ast.body, cell_name,
```

Out[1]:

	banking marketing	Unnamed: 1	Unnamed: 2	Unnamed: 3	Unnamed: 4	Unnamed: 5	Unnamed: 6	Unnamed: 7	Unnamed: 8	Unnamed: 9	Unnam
0	customer id and age.	NaN	Customer salary and balance.	NaN	Customer marital status and job with education...	NaN	particular customer before targeted or not	NaN	Loan types: loans or housing loans	NaN	Cont ty
1	customerid	age	salary	balance	marital	jobedu	targeted	default	housing	loan	cont
2	1	58	100000	2143	married	management,tertiary	yes	no	yes	no	unkno
3	2	44	60000	29	single	technician,secondary	yes	no	yes	no	unkno
4	3	33	120000	2	married	entrepreneur,secondary	yes	no	yes	yes	unkno
...	...	...	...	...	...	...	...	...	...	...	
45208	45207	51	60000	825	married	technician,tertiary	yes	no	no	no	cellu
45209	45208	71	55000	1729	divorced	retired,primary	yes	no	no	no	cellu
45210	45209	72	55000	5715	married	retired,secondary	yes	no	no	no	cellu
45211	45210	57	20000	668	married	blue-collar,secondary	yes	no	no	no	telepho
45212	45211	37	120000	2971	married	entrepreneur,secondary	yes	no	no	no	cellu

45213 rows × 19 columns

In [2]:
```
# Read the file in data without first two rows as it is of no use.
data = pd.read_csv("marketing_analysis.csv",skiprows = 2)

#print the head of the data frame.
data.head()
```

Out[2]:

	customerid	age	salary	balance	marital	jobedu	targeted	default	housing	loan	contact	day	month	duration	camp
0	1	58.0	100000	2143	married	management,tertiary	yes	no	yes	no	unknown	5	may, 2017	261 sec	
1	2	44.0	60000	29	single	technician,secondary	yes	no	yes	no	unknown	5	may, 2017	151 sec	
2	3	33.0	120000	2	married	entrepreneur,secondary	yes	no	yes	yes	unknown	5	may, 2017	76 sec	
3	4	47.0	20000	1506	married	blue-collar,unknown	no	no	yes	no	unknown	5	may, 2017	92 sec	
4	5	33.0	0	1	single	unknown,unknown	no	no	no	no	unknown	5	may, 2017	198 sec	

In [3]:
```
# Drop the customer id as it is of no use.
data.drop('customerid', axis = 1, inplace = True)

#Extract job  & Education in newly from "jobedu" column.
data['job']= data["jobedu"].apply(lambda x: x.split(",")[0])
data['education']= data["jobedu"].apply(lambda x: x.split(",")[1])

# Drop the "jobedu" column from the dataframe.
data.drop('jobedu', axis = 1, inplace = True)

# Printing the Dataset
data
```

Out[3]:

	age	salary	balance	marital	targeted	default	housing	loan	contact	day	month	duration	campaign	pdays	previo
0	58.0	100000	2143	married	yes	no	yes	no	unknown	5	may, 2017	261 sec	1	-1	
1	44.0	60000	29	single	yes	no	yes	no	unknown	5	may, 2017	151 sec	1	-1	
2	33.0	120000	2	married	yes	no	yes	yes	unknown	5	may, 2017	76 sec	1	-1	
3	47.0	20000	1506	married	no	no	yes	no	unknown	5	may, 2017	92 sec	1	-1	
4	33.0	0	1	single	no	no	no	no	unknown	5	may, 2017	198 sec	1	-1	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
45206	51.0	60000	825	married	yes	no	no	no	cellular	17	nov, 2017	16.2833333333333 min	3	-1	
45207	71.0	55000	1729	divorced	yes	no	no	no	cellular	17	nov, 2017	7.6 min	2	-1	
45208	72.0	55000	5715	married	yes	no	no	no	cellular	17	nov, 2017	18.7833333333333 min	5	184	
45209	57.0	20000	668	married	yes	no	no	no	telephone	17	nov, 2017	8.46666666666667 min	4	-1	
45210	37.0	120000	2971	married	yes	no	no	no	cellular	17	nov, 2017	6.01666666666667 min	2	188	

45211 rows × 19 columns

In [4]:
```
# Checking the missing values
data.isnull().sum()
```

Out[4]:
```
age          20
salary        0
balance       0
marital       0
targeted      0
default       0
housing       0
loan          0
contact       0
day           0
month        50
duration      0
campaign      0
pdays         0
previous      0
poutcome      0
response     30
job           0
education     0
dtype: int64
```

In [5]:
```
# Dropping the records with age missing in data dataframe.
data = data[~data.age.isnull()].copy()

# Checking the missing values in the dataset.
data.isnull().sum()
```

Out[5]:
```
age           0
salary        0
balance       0
marital       0
targeted      0
default       0
housing       0
loan          0
contact       0
day           0
month        50
duration      0
campaign      0
pdays         0
previous      0
poutcome      0
response     30
job           0
education     0
dtype: int64
```

In [6]:
```
# Find the mode of month in data
month_mode = data.month.mode()[0]

# Fill the missing values with mode value of month in data.
data.month.fillna(month_mode, inplace = True)

# Let's see the null values in the month column.
data.month.isnull().sum()
```

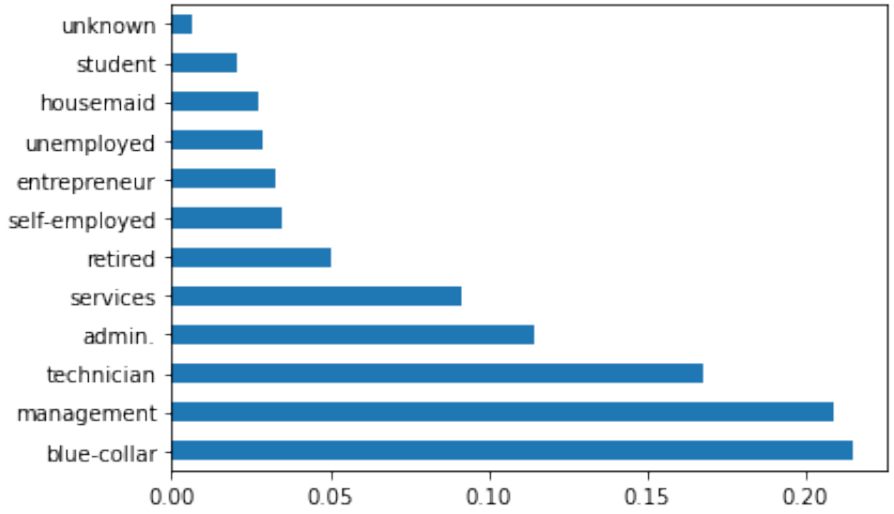
Out[6]:
```
0
```

In [7]:
```
#drop the records with response missing in data.
data = data[~data.response.isnull()].copy()
# Calculate the missing values in each column of data frame
data.isnull().sum()
```

Out[7]:
```
age          0
salary       0
balance      0
marital      0
targeted     0
default      0
housing      0
loan         0
contact      0
day          0
month        0
duration     0
campaign     0
pdays        0
previous     0
poutcome     0
response     0
job          0
education    0
dtype: int64
```

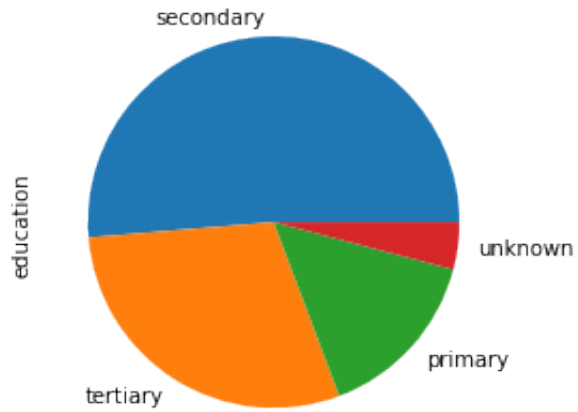
In [8]:
```
# Let's calculate the percentage of each job status category.
data.job.value_counts(normalize=True)

#plot the bar graph of percentage job categories
data.job.value_counts(normalize=True).plot.barh()
plt.show()
```



In [9]:
```
#calculate the percentage of each education category.
data.education.value_counts(normalize=True)

#plot the pie chart of education categories
data.education.value_counts(normalize=True).plot.pie()
plt.show()
```



In [10]:
```
data.salary.describe()
```

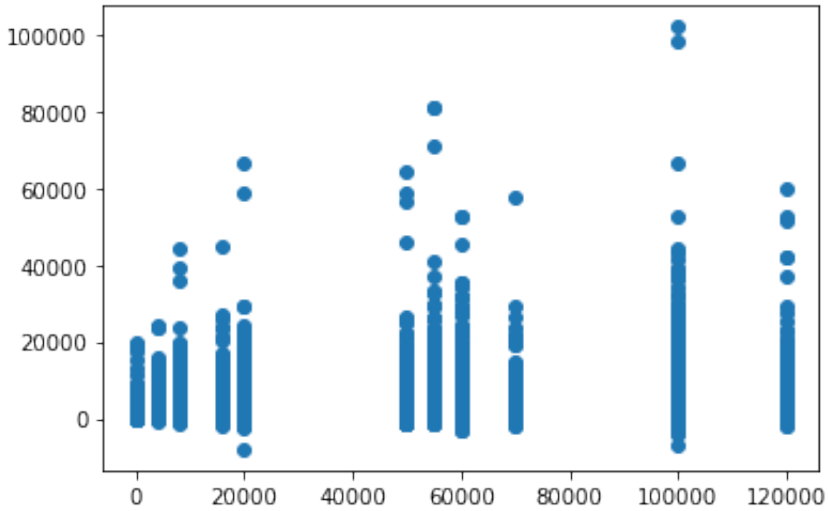
Out[10]:
```
count     45161.000000
mean      57004.849317
std       32087.698810
min           0.000000
25%       20000.000000
50%       60000.000000
75%       70000.000000
max      120000.000000
Name: salary, dtype: float64
```

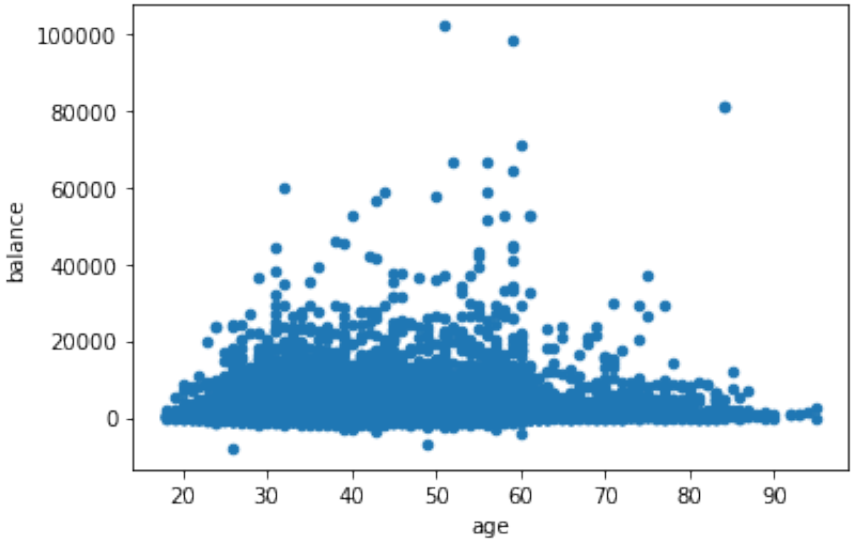
In [ ]:
```

```

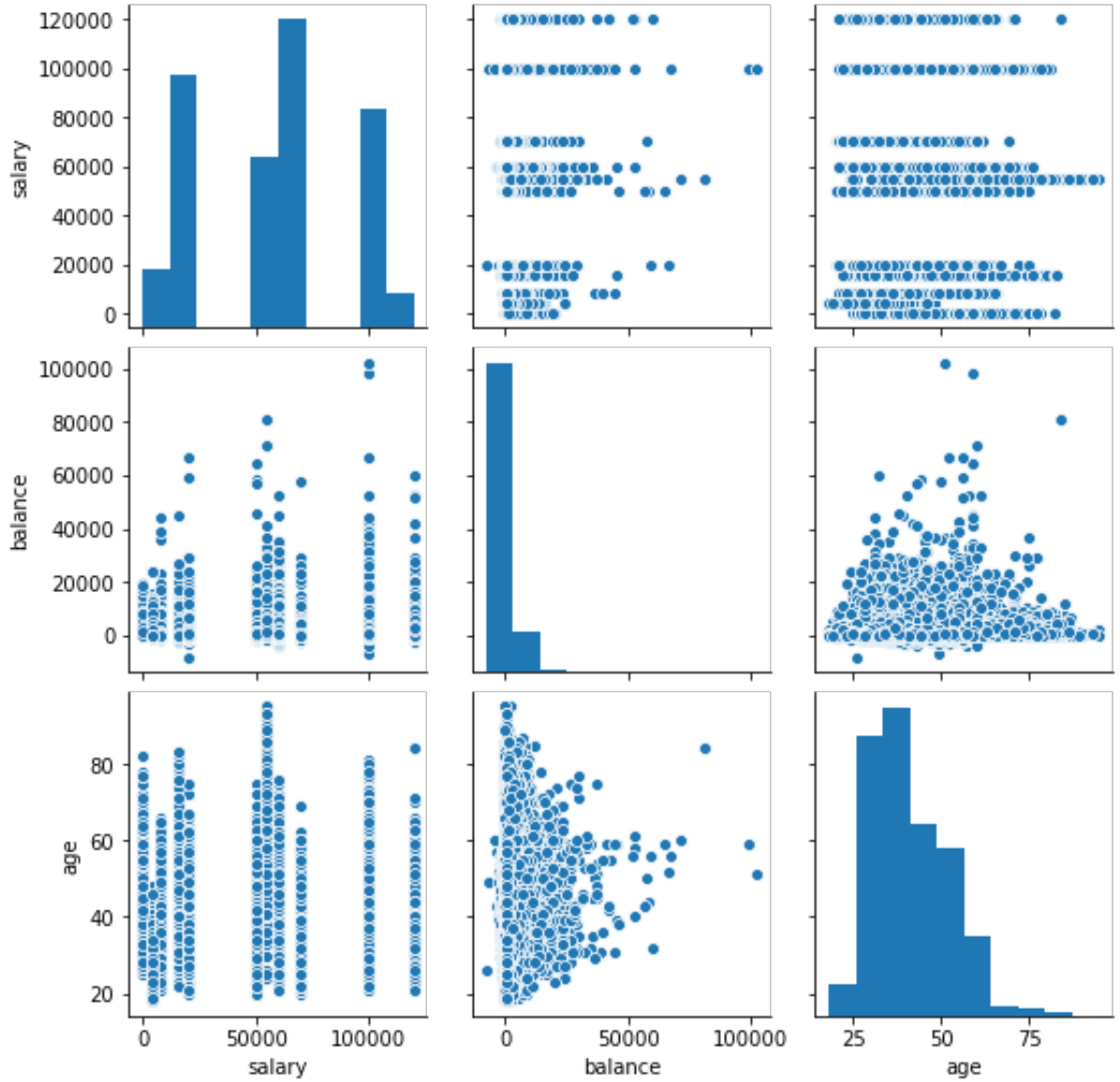
In [11]:
```
#plot the scatter plot of balance and salary variable in data
plt.scatter(data.salary,data.balance)
plt.show()

#plot the scatter plot of balance and age variable in data
data.plot.scatter(x="age",y="balance")
plt.show()
```


100000
80000
60000
40000
20000
0
0
20000
40000
60000
80000
100000
120000

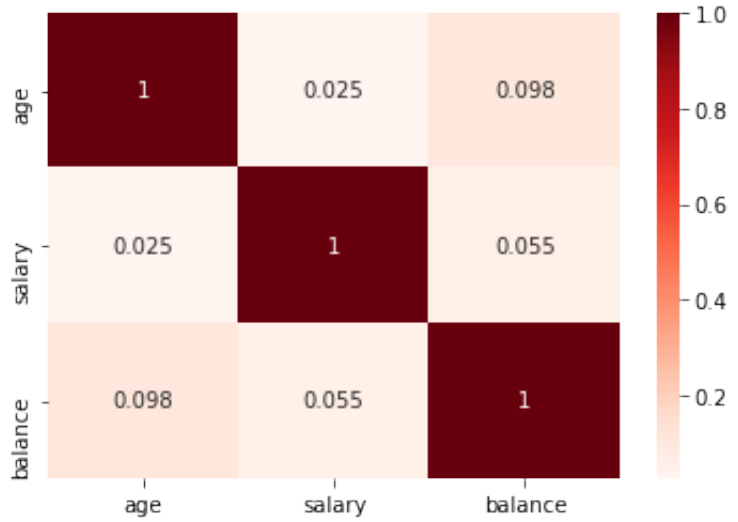

balance
100000
80000
60000
40000
20000
0
20
30
40
50
60
70
80
90
age

In [12]:
```
#plot the pair plot of salary, balance and age in data dataframe.
sns.pairplot(data = data, vars=['salary','balance','age'])
plt.show()
```



In [13]:
```
# Creating a matrix using age, salry, balance as rows and columns
data[['age','salary','balance']].corr()

#plot the correlation matrix of salary, balance and age in data dataframe.
sns.heatmap(data[['age','salary','balance']].corr(), annot=True, cmap = 'Reds')
plt.show()
```



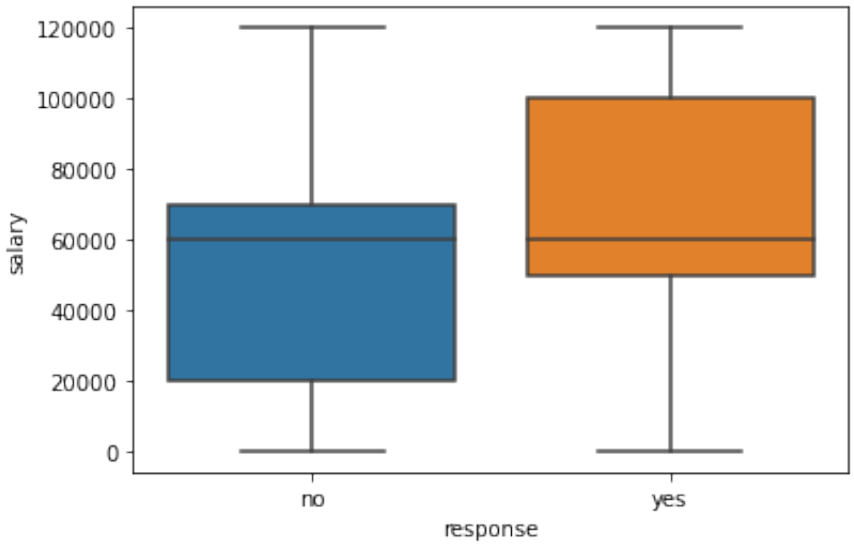
In [14]:
```
#groupby the response to find the mean of the salary with response no & yes separately.
data.groupby('response')['salary'].mean()
```

Out[14]:
```
response
no      56769.510482
yes     58780.510880
Name: salary, dtype: float64
```

In [15]:
```
#groupby the response to find the median of the salary with response no & yes separately.
data.groupby('response')['salary'].median()
```

Out[15]:
```
response
no      60000
yes     60000
Name: salary, dtype: int64
```

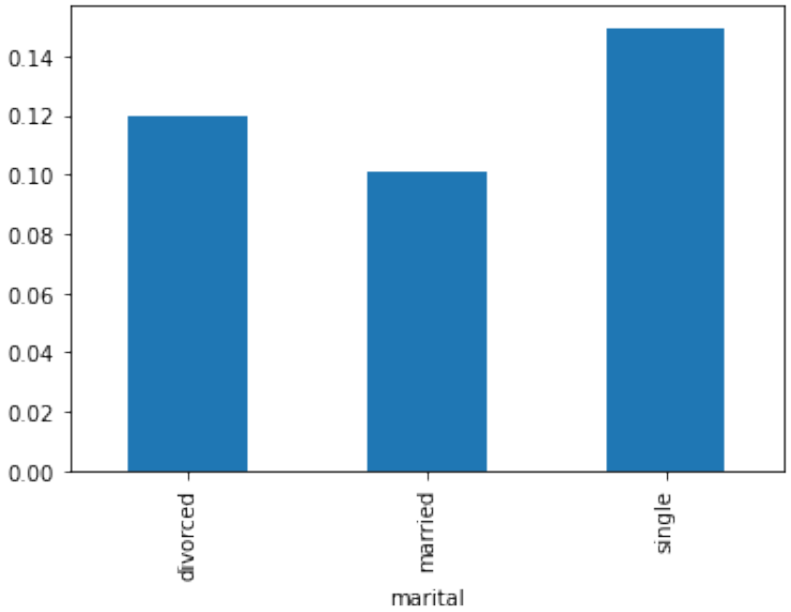
In [16]:
```
#plot the box plot of salary for yes & no responses.
sns.boxplot(data.response, data.salary)
plt.show()
```



In [17]:
```
#create response_rate of numerical data type where response "yes"= 1, "no"= 0
data['response_rate'] = np.where(data.response=='yes',1,0)
data.response_rate.value_counts()
```

Out[17]:
```
0    39876
1     5285
Name: response_rate, dtype: int64
```

In [18]:
```
#plot the bar graph of marital status with average value of response_rate
data.groupby('marital')['response_rate'].mean().plot.bar()
plt.show()
```



In [19]:
```
result = pd.pivot_table(data=data, index='education', columns='marital',values='response_rate')
print(result)

#create heat map of education vs marital vs response_rate
sns.heatmap(result, annot=True, cmap = 'RdYlGn', center=0.117)
plt.show()
```

```
marital     divorced   married    single
education
primary     0.138852  0.075601  0.106808
secondary   0.103559  0.094650  0.129271
tertiary    0.137415  0.129835  0.183737
unknown     0.142012  0.122519  0.162879
```

In [ ]: