Assignment 1
DSC 478
Chaonan Shi
1901412

1. Explore the general characteristics of the data as a whole: examine the means, standard deviations, and other statistics associated with the numerical attributes; show the distributions of values associated with categorical attributes; etc.

Basic statistical results for continuous features:

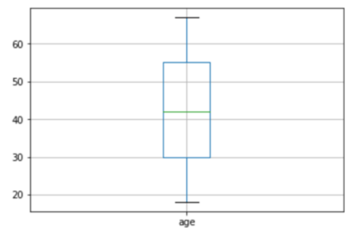
In [33]:
```
pop_df.describe(include="all")
```

Out[33]:

	id	age	income	children	gender	region	married	car	savings_acct	current_acct	mortgage	pep
count	600	600.000000	600.000000	600.000000	600	600	600	600	600	600	600	600
unique	600	NaN	NaN	NaN	2	4	2	2	2	2	2	2
top	ID12637	NaN	NaN	NaN	MALE	INNER_CITY	YES	NO	YES	YES	NO	NO
freq	1	NaN	NaN	NaN	300	269	396	304	414	455	391	326
mean	NaN	42.395000	27524.031217	1.011667	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
std	NaN	14.424947	12899.468246	1.056752	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
min	NaN	18.000000	5014.210000	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
25%	NaN	30.000000	17264.500000	0.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
50%	NaN	42.000000	24925.300000	1.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
75%	NaN	55.250000	36172.675000	2.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
max	NaN	67.000000	63130.100000	3.000000	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

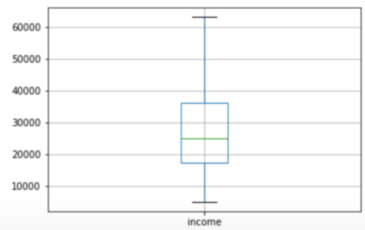
In [27]:
```
pop_df.boxplot(column=["age"], return_type='axes')
```

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1bf34be0>



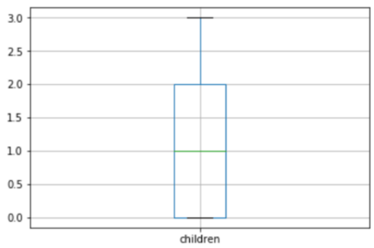
In [30]:
```
pop_df.boxplot(column=["income"], return_type='axes')
```

Out[30]: <matplotlib.axes._subplots.AxesSubplot at 0x1a1be9fe80>



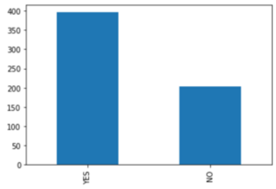
In [31]:
```
pop_df.boxplot(column=["children"], return_type='axes')
```

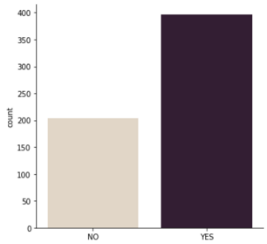
Out[31]: <matplotlib.axes._subplots.AxesSubplot at 0x108999d68>



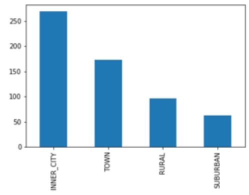
Basic statistical results for categorical features:

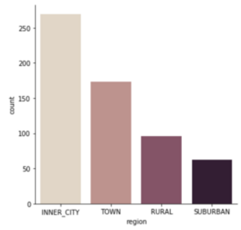
In [68]:
```
bank_df["married"].value_counts().plot(kind='bar')
sns.catplot(x="married", kind="count", palette="ch:.25", data=bank_df);
```



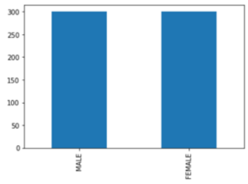


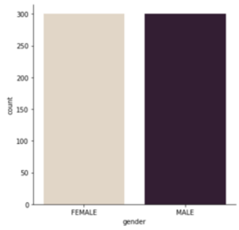
In [67]:
```
bank_df["region"].value_counts().plot(kind='bar')
sns.catplot(x="region", kind="count", palette="ch:.25", data=bank_df);
```



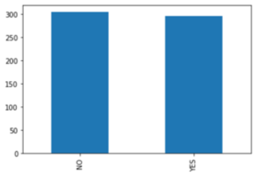


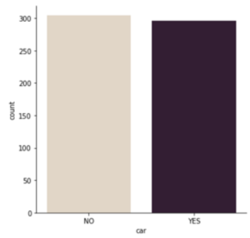
In [65]:
```
bank_df["gender"].value_counts().plot(kind='bar')
sns.catplot(x="gender", kind="count", palette="ch:.25", data=bank_df);
```



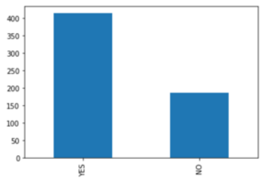


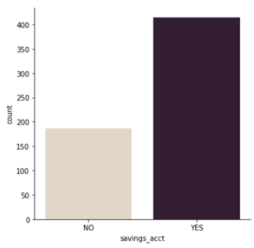
In [69]:
```
bank_df["car"].value_counts().plot(kind='bar')
sns.catplot(x="car", kind="count", palette="ch:.25", data=bank_df);
```



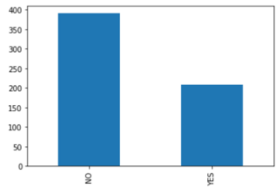


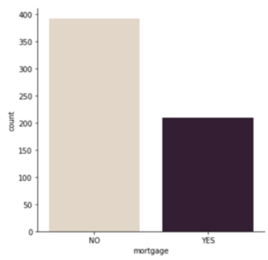
In [70]:
```
bank_df["savings_acct"].value_counts().plot(kind='bar')
sns.catplot(x="savings_acct", kind="count", palette="ch:.25", data=bank_df);
```



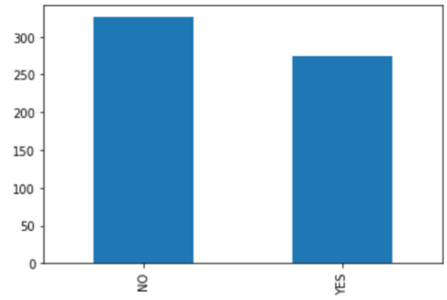


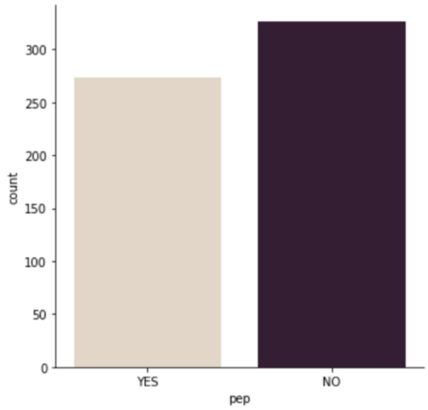
In [72]:
```
bank_df["mortgage"].value_counts().plot(kind='bar')
sns.catplot(x="mortgage", kind="count", palette="ch:.25", data=bank_df);
```





```
In [73]: bank_df["pep"].value_counts().plot(kind='bar')
         sns.catplot(x="pep", kind="count", palette="ch:.25", data=bank_df);
```





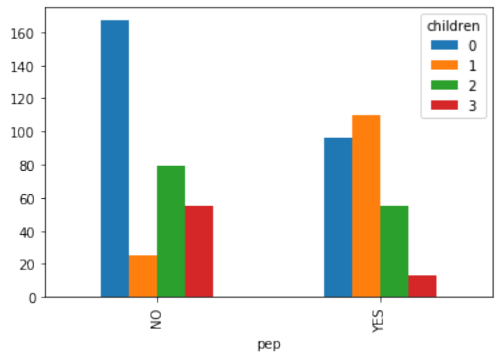
2.Suppose that the hypothetical bank is particularly interested in customers who buy the PEP (Personal Equity Plan) product. Compare and contrast the subsets of customers who buy and don't buy the PEP. Compute summaries (as in part 1) of the selected data with respect to all other attributes. Can you observe any significant differences between these segments of customers? Discuss your observations.

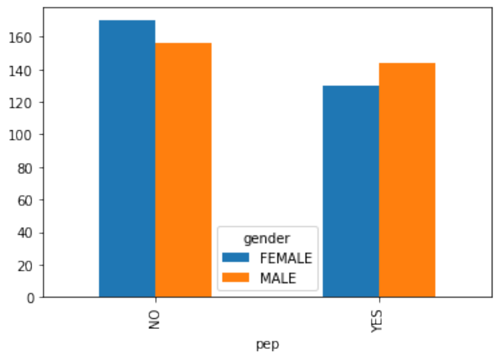
```
In [95]: bank_df.groupby('pep').describe()
```

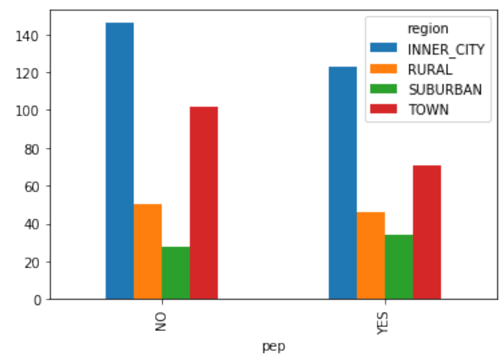
Out[95]:

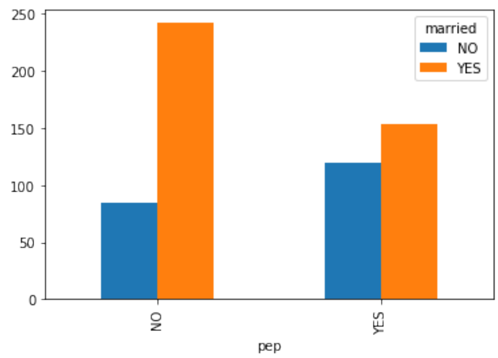
	age								income					children							
	count	mean	std	min	25%	50%	75%	max	count	mean	...	75%	max	count	mean	std	min	25%	50%	75%
pep																				
NO	326.0	40.098160	14.123506	18.0	27.0	40.0	51.0	67.0	326.0	24900.953374	...	31210.90	61554.6	326.0	1.067485	1.195524	0.0	0.0	0.0	2.
YES	274.0	45.127737	14.327991	18.0	34.0	45.0	58.0	67.0	274.0	30644.919453	...	38458.35	63130.1	274.0	0.945255	0.860571	0.0	0.0	1.0	1.

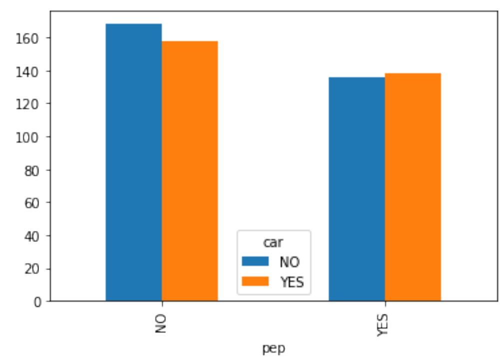
2 rows × 24 columns

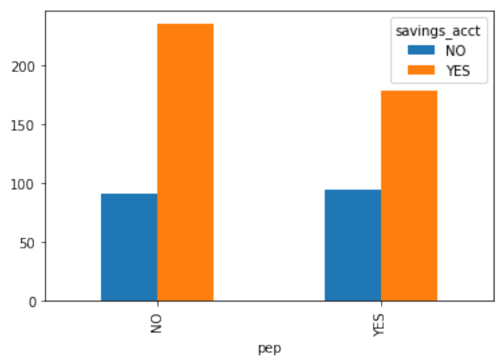


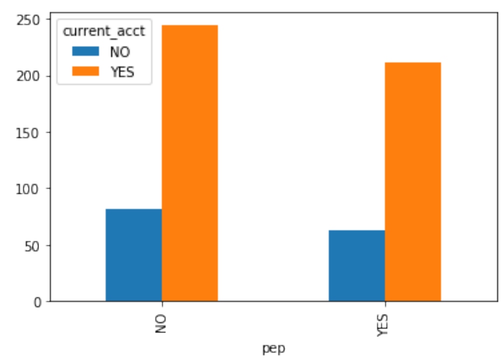


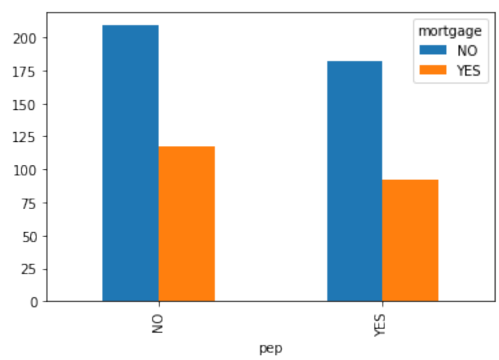












Analysis:

From above analysis, the number of answer of "No" have 16% more than the number of answer of "Yes" (326-274)/326;
Moreover, the standard deviation of answer "No" has less than the answer "Yes", which means answer "No" has less range of answer "Yes";

For the categorical variables 'children', level '0' has more answer for "No"; For answer "yes", most of children with level '1';
For the categorical variables 'gender', more male answer for "No";
For the categorical variables 'car, people who own the car have same level to buy pep, but for the people who do not own the car, they are more likely do not buy pep;
For the categorical variables 'region', level with 'INNER_CITY' has more people for answer both "yes" and "No";
For the categorical variables 'married', married people more likely to do not buy pep;

For the categorical variables ‘savings_acct’, people who do not have savings_acct have more likely to do not buy pep;
For the categorical variables ‘current_acct’, people who do have ‘current_acct’ have more likely to do not buy pep;
For the categorical variables ‘mortgage’, people who do not have ‘mortgage’ have more likely to do not buy pep;

3.Use z-score normalization to standardize the values of the income attribute. [Do not change the original income attribute in the table.]

```
In [103]: income_norm = (bank_df["income"] - bank_df["income"].mean()) / bank_df["income"].std()
          income_norm.head(5)

Out[103]: 0   -0.773523
          1    0.198541
          2   -0.848766
          3   -0.554180
          4    1.787071
          Name: income, dtype: float64
```

4.Discretize the age attribute into 3 categories (corresponding to "young", "mid-age", and "old"). [Do not change the original age attribute in the table.]

```
In [106]: inc_bins = pd.qcut(bank_df.age, [0, 0.33, 0.66, 1], labels = ['young', 'mid-age', 'old'])
          inc_bins

Out[106]: 0       mid-age
          1       mid-age
          2           old
          3         young
          4           old
          5           old
          6         young
          7           old
          8       mid-age
          9           old
          10          old
          11          old
          12      mid-age
          13          old
          14      mid-age
          15      mid-age
          16      mid-age
          17      mid-age
          18          old
          19        young
          20          old
          21          old
          22          old
          23        young
          24        young
          25          old
          26      mid-age
          27      mid-age
          28      mid-age
          29          old
                 ...
```

5.Use Min-Max Normalization to transform the values of all numeric attributes (income, age, children) in the original table (before the transformations in parts 3 and 4 above) onto the range 0.0-1.0.

In [44]:
```python
min_max_scaler = preprocessing.MinMaxScaler()

#bank_df['income', 'age', 'children'].values.reshape(-1, 1)
#bank_df['income'].values.reshape(-1, 1)
inc_mms = min_max_scaler.fit_transform(bank_df['income'].values.reshape(-1, 1))
bank_df['age'] = age_mms = min_max_scaler.fit_transform(bank_df['age'].values.reshape(-1, 1))
bank_df['children'] = chi_mms = min_max_scaler.fit_transform(bank_df['children'].values.reshape(-1, 1))
bank_df['income'] = min_max_scaler.fit_transform(bank_df['income'].values.reshape(-1, 1))
bank_df.describe()
```

Out[44]:

	age	income	children
count	600.000000	600.000000	600.000000
mean	0.497857	0.387326	0.337222
std	0.294387	0.221961	0.352251
min	0.000000	0.000000	0.000000
25%	0.244898	0.210791	0.000000
50%	0.489796	0.342610	0.333333
75%	0.760204	0.536144	0.666667
max	1.000000	1.000000	1.000000

6.Convert the table (after normalization in part 5) into the standard spreadsheet format. Note that this requires converting each categorical attribute into multiple binary ("dummy") attributes (one for each values of the categorical attribute) and assigning binary values corresponding to the presence or not presence of the attribute value in the original record). The numeric attributes should remain unchanged. Save this new table into a file called bank_numeric.csv and submit it along with your assignment. [Hint: you might consider using the get_dummies for Pandas data frames.]

In [27]:
```python
bank_df = pd.get_dummies(bank_df)
bank_df.head()
```

Out[27]:

	age	income	children	id_ID12101	id_ID12102	id_ID12103	id_ID12104	id_ID12105	id_ID12106	id_ID12107	...	car_NO	car_YES	savings_acct_NO
0	0.612245	0.215634	0.333333	1	0	0	0	0	0	0	...	1	0	1
1	0.448980	0.431395	1.000000	0	1	0	0	0	0	0	...	0	1	1
2	0.673469	0.198933	0.000000	0	0	1	0	0	0	0	...	0	1	0
3	0.102041	0.264320	1.000000	0	0	0	1	0	0	0	...	1	0	1
4	0.795918	0.783987	0.000000	0	0	0	0	1	0	0	...	1	0	0

5 rows × 621 columns

In [28]:
```python
bank_df.to_csv("/Users/appobs/Desktop/hw/478/week2/bank_numeric.csv", float_format="%1.2f")
```

7.Using the standardized data set (of the previous part), perform basic correlation analysis among the attributes. Discuss your results by indicating any significant positive or negative correlations among pairs of attributes. You need to construct a complete Correlation Matrix. Be sure to first remove the Customer ID column before creating the correlation matrix. [Hint:you can create the correlation matrix by using the corr() function in Pandas, try at least two corr methods and compare them].

In [30]:
```python
corr = bank_numeric.corr()
corr
```

Out[30]:

	age	income	children	gender_FEMALE	gender_MALE	region_INNER_CITY	region_RURAL	region_SUBURBAN	region_TOWN	ma
age	1.000000	0.752726	0.023572	0.090081	-0.090081	-0.025171	0.018635	0.031345	-0.008510	-
income	0.752726	1.000000	0.036761	0.023845	-0.023845	-0.047564	0.084776	0.029824	-0.036431	
children	0.023572	0.036761	1.000000	0.014206	-0.014206	-0.051222	0.089902	-0.014122	-0.007033	
gender_FEMALE	0.090081	0.023845	0.014206	1.000000	-1.000000	-0.023459	-0.009092	-0.010951	0.040472	
gender_MALE	-0.090081	-0.023845	-0.014206	-1.000000	1.000000	0.023459	0.009092	0.010951	-0.040472	-
region_INNER_CITY	-0.025171	-0.047564	-0.051222	-0.023459	0.023459	1.000000	-0.393444	-0.306032	-0.573814	-
region_RURAL	0.018635	0.084776	0.089902	-0.009092	0.009092	-0.393444	1.000000	-0.148158	-0.277798	
region_SUBURBAN	0.031345	0.029824	-0.014122	-0.010951	0.010951	-0.306032	-0.148158	1.000000	-0.216080	-
region_TOWN	-0.008510	-0.036431	-0.007033	0.040472	-0.040472	-0.573814	-0.277798	-0.216080	1.000000	-
married_NO	-0.010394	0.008386	0.048716	0.021110	-0.021110	-0.003254	0.022649	-0.012483	-0.006369	
married_YES	0.010394	-0.008386	-0.048716	-0.021110	0.021110	0.003254	-0.022649	0.012483	0.006369	-
car_NO	-0.077733	-0.081556	-0.036455	0.006667	-0.006667	0.018143	-0.024006	0.061184	-0.041604	-
car_YES	0.077733	0.081556	0.036455	-0.006667	0.006667	-0.018143	0.024006	-0.061184	0.041604	
savings_acct_NO	-0.184389	-0.266164	-0.041536	0.007207	-0.007207	0.091373	-0.036960	-0.002605	-0.068654	
savings_acct_YES	0.184389	0.266164	0.041536	-0.007207	0.007207	-0.091373	0.036960	0.002605	0.068654	-
current_acct_NO	0.035312	-0.031616	-0.006238	-0.019466	0.019466	-0.007894	0.008496	-0.038157	0.027431	-
current_acct_YES	-0.035312	0.031616	0.006238	0.019466	-0.019466	0.007894	-0.008496	0.038157	-0.027431	
mortgage_NO	0.016154	0.014662	0.074339	0.066465	-0.066465	-0.002098	0.051908	-0.004635	-0.036591	-
mortgage_YES	-0.016154	-0.014662	-0.074339	-0.066465	0.066465	0.002098	-0.051908	0.004635	0.036591	
pep_NO	-0.173825	-0.221991	0.057663	0.046843	-0.046843	-0.001054	-0.019714	-0.062508	0.059115	-
pep_YES	0.173825	0.221991	-0.057663	-0.046843	0.046843	0.001054	0.019714	0.062508	-0.059115	

21 rows × 21 columns

In [34]:
```python
# Filter out all non-high correlated variables which corr < 0.7
# False meaning highly correlated with each other
fil_out = ~(corr.mask(np.eye(len(corr), dtype=bool)).abs() >= 0.7).any()
fil_out
```

Out[34]:
```
age                  False
income               False
children              True
gender_FEMALE        False
gender_MALE          False
region_INNER_CITY     True
region_RURAL          True
region_SUBURBAN       True
region_TOWN           True
married_NO           False
married_YES          False
car_NO               False
car_YES              False
savings_acct_NO      False
savings_acct_YES     False
current_acct_NO      False
current_acct_YES     False
mortgage_NO          False
mortgage_YES         False
pep_NO               False
pep_YES              False
dtype: bool
```

In [41]:
```python
indices = np.where(corr > 0.5)
indices = [(corr.index[x], corr.columns[y]) for x, y in zip(*indices)
                                        if x != y and x < y]
indices
```

Out[41]:
```
[('age', 'income')]
```

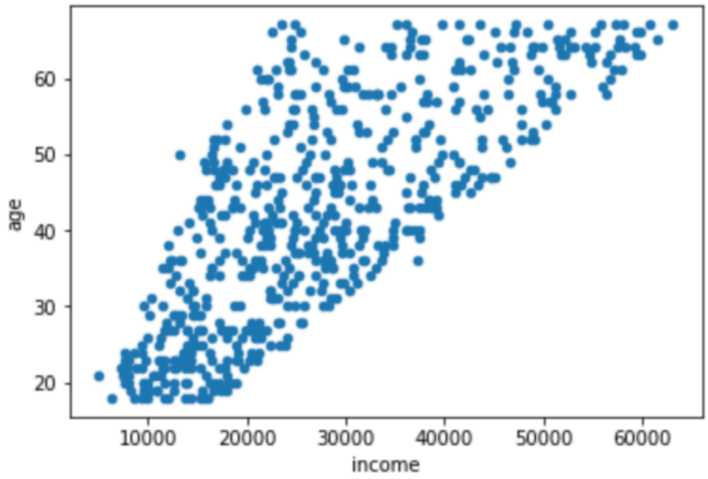
Analysis:

From above analysis, we can summarize that variables: age and income are highly correlated with each other; Which meaning people who have higher age also have income in this case; Moreover, since we created many level of dummies, so they also highly correlated with each other.

8.Using Matplotlib library and/or ploting capabilties of Pandas, create a scatter plot of the (non-normalized) Income attribute relative to Age. Be sure that your plot contains appropriate labels for the axes. Do these variables seem correlated?

```
In [23]: bank_df.plot(x="income", y="age", kind="scatter")

Out[23]: <matplotlib.axes._subplots.AxesSubplot at 0x1a20919e10>
```

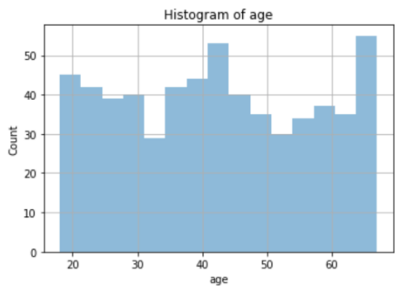


Analysis:

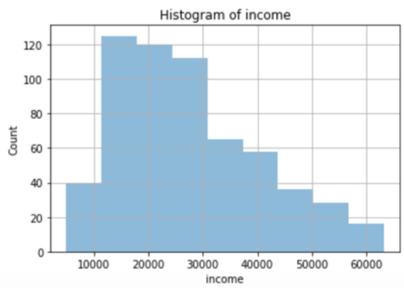
From above graph, we can summarize that variables: age and income are highly correlated with each other. Again, it makes sense since higher age people do have higher income as well.

9.Create histograms for (non-normalized) Income (using 9 bins) and Age (using 15 bins).

```
In [24]: plt.hist(bank_df["age"], bins=15, alpha=0.5)
         plt.xlabel('age')
         plt.ylabel('Count')
         plt.title('Histogram of age')
         plt.grid(True)
```

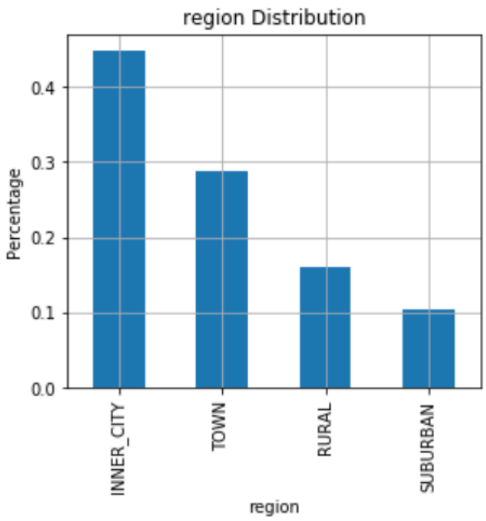



```
In [25]: plt.hist(bank_df["income"], bins=9, alpha=0.5)
         plt.xlabel('income')
         plt.ylabel('Count')
         plt.title('Histogram of income')
         plt.grid(True)
```



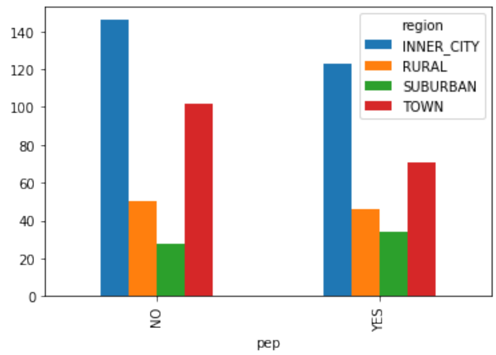
Analysis:
From above graph, we can summarize that variable age is not close to normal distribution, so that we need to transform/normalize it to make it more stationary;
For the variable income, it does close to normal distribution but more close to 'right skew'; in this case, we can summarize that the mood > median > mean for income. Also it makes sense since in the real world, number of people who have lower income more than people who have higher income.

10.Using a bargraph, plot the distribution of the values of the region attribute.



Analysis:
From above graph, we can summarize that for variable 'region', over 40% of records is level with 'INNER_CITY' ; and the following levels are: TOWN, RURAL, and SUBURBAN with roughly 19%, 16%, 11% from high to low, respectively;

11.Perform a cross-tabulation of the region attribute with the pep attribute. This requires the aggregation of the occurrences of each pep value (yes or no) separately for each value of the region attribute. Show the results as a 4 by 2 (region x pep) table with entries representing the counts. [Hint: you can either use Numpy or use aggregations fucntions in Pandas such as groupby() and cross-tab().] Then, either using Matplotlib directly or the plot() function in Pandas create a bar chart graph to visualize of the relationships between these sets of variables. [Hint: This example of creating simple bar charts using Matplotlib may be useful.



Analysis:

From above graph, we can summarize that since majority of people living in the region of 'INNER_CITY', so that whether buy pep (YES) or not buy pep (NO) have more than other regions such as rural, suburban, and town.
Moreover, number of people who living in 'town' also significantly higher that other regions; but both people who living in 'inner_city' and 'town' are more likely do not buy 'pep';
For the level 'suburban', number of people who tend to buy pep are more than people who do not tend to buy pep, and this is only region where more people who willing to buy pep.