Describe

What is the general type of the data (tabular, network, geographical, textual etc.)?

In [17]:

```
import os
import sys
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

m_datafile = "WDB_Accused.csv"
m_dataframe = pd.read_csv(m_datafile)

try:
    # convert those two columns' dtype to 'datetime'
    m_dataframe['Createdate'] = pd.to_datetime(m_dataframe['Createdate'])
    m_dataframe['Lastupdatedon'] = pd.to_datetime(m_dataframe['Lastupdatedon'])
except ValueError as err:
    print("String does not match format! %s" % err)
except:
    print("Unknown error occur! Convert failure!")
```

The general type of the data is fabular.

The types of fields of the table are mainly geographical and textual, but it also contains a small number of numeric attributes, such as 'AccusedID', 'Age', 'Age_estcareer', 'Age_estchild', 'Res_NGR_Easting', 'Res_NGR_Northing'.

Moreover, it also contains 'datetime' type data; for instance, 'Createdate' and 'Lastupdatedon'.

How large and complex is it (rows/columns, size, variation, structure)?

In [18]:

```
row_count = m_dataframe.shape[0]  # get row count
col_count = m_dataframe.shape[1]  # get col count
print("The table has %d rows and %d columns;\nThe size of the table is %d x %d;" %
      (row_count, col_count, row_count, col_count))
print("The table has %d variations." % col_count)
```

```
The table has 3219 rows and 31 columns;
The size of the table is 3219 x 31;
The table has 31 variations.
```

What fields and data types are present (max/min, levels for categorical

values)?

In [19]:

```
def show_columns_info(df):
    """
    this function is used to show the columns' info

    :type df:       pandas.DataFrame
    :params df:     a dataframe
    """
    # get the table columns' name
    table_columns = df.columns.values.tolist()
    HEADER = ["Column's name", "Data type", "Comments"]
    print('-' * 96)
    print("| %s\t| %s| %s|" % (HEADER[0].ljust(18), HEADER[1].ljust(15), HEADER[2].l
    print('-' * 96)
    for col in table_columns:
        data_type = m_dataframe[col].dtype
        describe = None
        if np.issubdtype(data_type, np.number):  # np.int64 or np.float64:
            max_val = np.max(df[col])
            min_val = np.min(df[col])
            describe = "Continuous values: min=%.2f, max=%.2f" % (min_val, max_val)
        else:
            value_set = df[col].unique()
            describe = "Discrete values: levels for categorical values=%d" % len(val
        print("| %s\t| %s| %s|" % (col.ljust(18), str(data_type).ljust(15), describe
    print('-' * 96)


print("Columns' name and corresponding data type are as follow:\n")
show_columns_info(df=m_dataframe)
print("size: %d x %d" % (row_count, col_count))
```

```
Columns' name and corresponding data type are as follow:

------------------------------------------------------------------------
--------------------------
| Column's name          | Data type      | Comments
                          |
------------------------------------------------------------------------
--------------------------
| AccusedRef             | object         | Discrete values: levels for
categorical values=3219 |
| AccusedSystemId        | object         | Discrete values: levels for
categorical values=3    |
| AccusedID              | int64          | Continuous values: min=4.0
0, max=3244.00            |
| FirstName              | object         | Discrete values: levels for
categorical values=457  |
| LastName               | object         | Discrete values: levels for
categorical values=1798 |
| M_Firstname            | object         | Discrete values: levels for
categorical values=160  |
| M_Surname              | object         | Discrete values: levels for
categorical values=1169 |
| Alias                  | object         | Discrete values: levels for
categorical values=101  |
| Patronymic             | object         | Discrete values: levels for
categorical values=15   |
```

```
| DesTitle              | object         | Discrete values: levels for
categorical values=20   |
| Sex                   | object         | Discrete values: levels for
categorical values=3    |
| Age                   | float64        | Continuous values: min=9.0
0, max=100.00             |
| Age_estcareer         | int64          | Continuous values: min=0.0
0, max=1.00               |
| Age_estchild          | int64          | Continuous values: min=0.0
0, max=1.00               |
| Res_settlement        | object         | Discrete values: levels for
categorical values=521  |
| Res_parish            | object         | Discrete values: levels for
categorical values=371  |
| Res_presbytery        | object         | Discrete values: levels for
categorical values=75   |
| Res_county            | object         | Discrete values: levels for
categorical values=35   |
| Res_burgh             | object         | Discrete values: levels for
categorical values=51   |
| Res_NGR_Letters       | object         | Discrete values: levels for
categorical values=7    |
| Res_NGR_Easting       | float64        | Continuous values: min=46.0
0, max=883.00            |
| Res_NGR_Northing      | float64        | Continuous values: min=38.0
0, max=894.00            |
| Ethnic_origin         | object         | Discrete values: levels for
categorical values=6    |
| MaritalStatus         | object         | Discrete values: levels for
categorical values=7    |
| SocioecStatus         | object         | Discrete values: levels for
categorical values=8    |
| Occupation            | object         | Discrete values: levels for
categorical values=32   |
| Notes                 | object         | Discrete values: levels for
categorical values=976  |
| Createdby             | object         | Discrete values: levels for
categorical values=3    |
| Createdate            | datetime64[ns] | Discrete values: levels for
categorical values=916  |
| Lastupdatedby         | object         | Discrete values: levels for
categorical values=3    |
| Lastupdatedon         | datetime64[ns] | Discrete values: levels for
categorical values=3214 |
-------------------------------------------------------------------------
--------------------------
size: 3219 x 31
```

Links between this data and other data (e.g. foreign keys, unique ids)?

In [20]:

```
def get_file_list(dir, format='.csv'):
    """
    Get all the 'csv' file in [dir]

    :param dir:    target dir
    :param format: format type of files
    :return:
    """
    files_list = []
    for root, sub_dirs, files in os.walk(dir):
        for file in files:
            if file.endswith(format):
                files_list.append(os.path.join(root, file))
    return files_list


# get all my columns' name in list
m_header = m_dataframe.columns.values.tolist()[:-5]
# store the links in dict, table's name as key, a list as value which contains the i
links = {}

""" Analyze all database table fields """
for file in get_file_list(dir='./data_witchcraft', format='.csv'):
    # if the file is the same as 'm_datafile', then skip it
    if file.endswith(m_datafile):
        continue
    other_df = pd.read_csv(file)
    header = other_df.columns.values.tolist()
    for col in header:
        # traversing attribute fields
        if col in m_header:
            if file not in links:
                links[file] = [col]
            else:
                links[file].append(col)
    del other_df

# show the links
for key in links.keys():
    print('[%s] contains fields:' % key.split('/')[-1])
    print('%s\n' % links[key])
```

As far as I know, the field 'AccusedRef' should be the unique ids for table ***WDB_Accused***.

At the mean time, 'AccusedRef' also acts as a foreign key to ***WDB_Case*** and ***WDB_Accused_family***.

Although ***WDB_Accused_family*** and ***WDB_Accused*** have some other same fields, I don't think there are any links between them, and it can be inferred from common sense that those are just duplications of names.

In addition, the existence of redundant fields for ***WDB_Person*** and ***WDB_Accused*** maybe just to improve retrieval efficiency and avoid associated queries.

Summary statistics about the data - how many people, what time frame, field averages etc. ?

How many people were be accused? How many people are there in which men and women respectively?

Firstly, I would like to check the data from two aspects:

1. Considering there might be both empty in column ['FirstName', 'LastName', 'M_Firstname', 'M_Surname'], so this is one of the checks.
2. Another thing is that we don't know the label values in column 'Sex', so it is the second check.

In [21]:

```
# select rows with all name related fields('FirstName', 'LastName', 'M_Firstname',
selected_rows = (
        (m_dataframe['FirstName'].isnull()) &
        (m_dataframe['LastName'].isnull()) &
        (m_dataframe['M_Firstname'].isnull()) &
        (m_dataframe['M_Surname'].isnull())
    )
selected_index = m_dataframe[selected_rows].index.tolist()
selected_count = np.sum(selected_rows.astype(np.int))
print('The count of invalid row is %d. Index as follow:\n%s\n' % (selected_count, se

# check the available values in column 'Sex'
sex_label = m_dataframe['Sex'].unique()
print("The label of 'Sex' are as follow:\n%s\n" % sex_label)
```

```
The count of invalid row is 1. Index as follow:
[1489]

The label of 'Sex' are as follow:
['Female' 'Male' nan]
```

Secondly, clean the data according to what I found.

In [22]:

```
# remove the dirty data, creating a copy of the data
cleaned_dataframe = m_dataframe.drop(selected_index, axis=0, inplace=False)

# check the rows of the table
assert cleaned_dataframe.shape[0] == row_count - selected_count, "remove dirty data
print("Data has been cleaned according to name related fields.\nThere are %d rows le

# remove the nan row according to 'Sex'
cleaned_dataframe.dropna(subset=['Sex'], inplace=True)
print("Data has been cleaned according to column 'Sex'. \nThere are %d rows left.\n"
```

```
Data has been cleaned according to name related fields.
There are 3218 rows left.

Data has been cleaned according to column 'Sex'.
There are 3170 rows left.
```

Count the number of people, female and male accused. At the same time, calculate the corresponding proportion.

In [23]:

```
cleaned_dataframe['full_name'] = cleaned_dataframe['FirstName'] + '.' + \
                    cleaned_dataframe['LastName'] + '.' + \
                    cleaned_dataframe['M_Firstname'] + '.' + \
                    cleaned_dataframe['M_Surname']
old_row_count = cleaned_dataframe.shape[0]

print("Given that the list of accused could be duplicated, so I would like to remove
# eliminate duplicate row data
cleaned_dataframe.drop_duplicates(subset=['full_name'], inplace=True)
count_accused_people = cleaned_dataframe.shape[0]

print("There were %d row in total. However, after drop duplicates, %d rows left.\nSo
      % (old_row_count, count_accused_people, old_row_count - count_accused_people)

"""
Count the number of Female and Male accused
"""
count_accused_male = cleaned_dataframe[cleaned_dataframe['Sex'] == 'Male'].shape[0]
count_accused_female = cleaned_dataframe[cleaned_dataframe['Sex'] == 'Female'].shape
template = "\n%d people in total. Among them:\n%4d are  male , accounted for %.2f%%;
print(template % (count_accused_people,
                  count_accused_male, count_accused_male / count_accused_people * 10
                  count_accused_female, count_accused_female / count_accused_people
print("\nThe number of female is %.2f times that of male." % (count_accused_female /
```

```
Given that the list of accused could be duplicated, so I would like to
remove duplicates.
There were 3170 row in total. However, after drop duplicates, 2950 row
s left.
So, there are 220 duplicates in total.

2950 people in total. Among them:
 448 are  male , accounted for 15.19%;
2502 are female, accounted for 84.81%;

The number of female is 5.58 times that of male.
```

What is the average of 'Age' for accused? In addition, what is the average 'Age' of male and female accused? Median is also included.

First, I would like to remove the row that field 'Age' is Nan. After cleaning, we calculate the average value.

In [24]:

```
age_dataframe = m_dataframe.dropna(subset=['Age'], inplace=False)
```

Secondly, we have to make sure that all the values of 'Age' make sense. For instance, decimal and negative are not allowed, etc.

In [25]:

```
selected_rows = (age_dataframe['Age'] < 0) | (age_dataframe['Age'] > 120)
print("Abnormal row count is %d;" % np.sum(selected_rows.astype(np.int)))
selected_rows = np.ceil(age_dataframe['Age']) != age_dataframe['Age']
print("Decimal  row count is %d;" % np.sum(selected_rows.astype(np.int)))
print("Not found any illegal data.")
```

```
Abnormal row count is 0;
Decimal  row count is 0;
Not found any illegal data.
```

In [26]:

```
template = "Average age is %.2f. Among them:\nFemale's average age is %.2f;\nMale's
accused_avg_age = np.mean(age_dataframe['Age'])
female_accused_avg_age = np.mean(age_dataframe[age_dataframe['Sex'] == 'Female']['Ag
male_accused_avg_age = np.mean(age_dataframe[age_dataframe['Sex'] == 'Male']['Age'])

print(template % (accused_avg_age, female_accused_avg_age, male_accused_avg_age))

print("median of accused age is %.2f;" % np.median(age_dataframe['Age']))
print("median of accused female age is %.2f;" % np.median(age_dataframe[age_datafram
print("median of accused  male  age is %.2f;\n" % np.median(age_dataframe[age_datafr

print("Std of accused age is %.2f;" % np.std(age_dataframe['Age']))
print("Std of accused female age is %.2f;" % np.std(age_dataframe[age_dataframe['Sex
print("Std of accused  male  age is %.2f;" % np.std(age_dataframe[age_dataframe['Sex
```

```
Average age is 43.13. Among them:
Female's average age is 43.27;
Male's   average age is 42.45;

median of accused age is 45.00;
median of accused female age is 45.00;
median of accused  male  age is 45.00;

Std of accused age is 14.16;
Std of accused female age is 13.57;
Std of accused  male  age is 16.68;
```

What is the date range of the table data?

In [27]:

```
create_date = m_dataframe.dropna(subset=['Createdate'], inplace=False)['Createdate']
update_date = m_dataframe.dropna(subset=['Lastupdatedon'], inplace=False)['Lastupdat
create_date = np.sort(create_date)
update_date = np.sort(update_date)
date_range = np.array([create_date[0], create_date[-1], update_date[0], update_date[
date_range = np.sort(date_range)

def get_datetime(date):
    return np.datetime_as_string(date, unit='s').replace('T', ' ')

print("Date of the data ranges from '%s' to '%s'" % (get_datetime(date_range[0]),
                                                     get_datetime(date_range[-1])))
```

```
Date of the data ranges from '2001-01-08 11:07:04' to '2002-12-11 17:1
5:26'
```

How does the data relate to the questions that the data owner has discussed with you?

One of the questions the data holder discussed with me was geographic information of most people accused of being witches. The data I deal with contains the geographic information of those accused. In the explore module, I will perform data visualization of the geographic information in the form of barplot to further explore the specific geographic location where more people are accused of witchcraft.

Explore

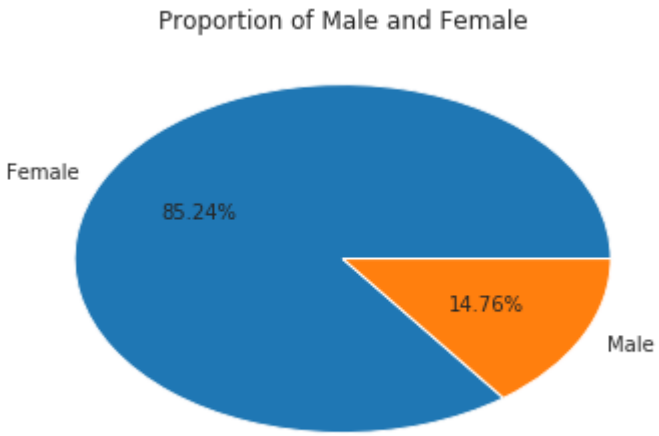
carry out a deeper exploration of the data. This includes looking at individual fields/variables to see the distribution of values they take (e.g. evenly distributed, bell curves, bi-modal) or how they are distributed in time. It also includes relationships between variables in your dataset: are there correlations? In which direction? Complex curves? We would expect to see roughly:

1. 4-5 exploratory visualisations, presented in a readable form, with an explanation about what you have found;
2. 1-2 relationships between variables analysed;
3. Ideas about trends, outliers, clusters;
4. Reference to statistics, i.e. a sense of which relationships are significant, and what claims you can back up;

In [28]:

```
# set the style of chart
sns.set_style("whitegrid")

sex_df = m_dataframe.dropna(subset=['Sex'], inplace=False)
sexuality = sex_df.Sex.value_counts()
sex_df = pd.DataFrame(data={'Sex': sexuality.index,
                            'Frequency': sexuality.values})
fig = plt.figure()
plt.pie(sex_df.Frequency, labels=sex_df.Sex, autopct='%1.2f%%')
plt.title("Proportion of Male and Female")
plt.show()
print(sexuality)
```




```
Female    2702
Male       468
Name: Sex, dtype: int64
```

According to the pie chart, female are far more likely to be accused than male.

So sex is a significant factor in the accused.

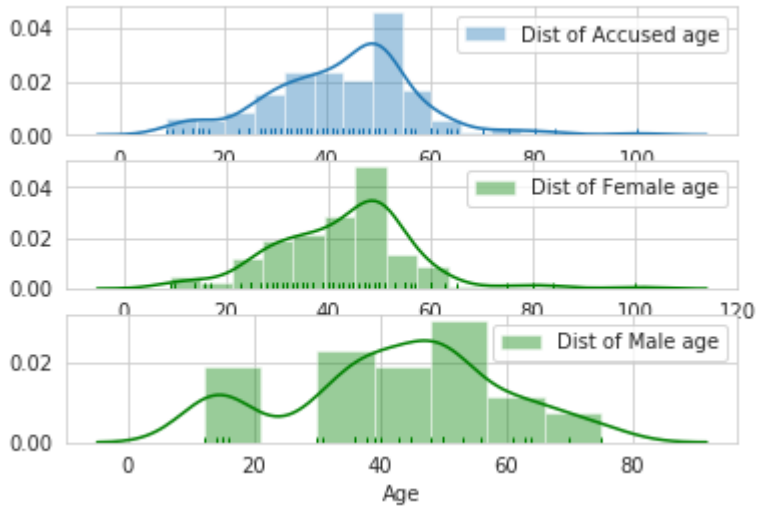
In [29]:

```
tmp_df = m_dataframe.dropna(subset=['Age', 'Sex'], inplace=False)
# creating the 1st subplot
axes_of_all = plt.subplot(3, 1, 1)
sns.distplot(tmp_df['Age'], ax=axes_of_all, kde=True, rug=True, label='Dist of Accus
axes_of_all.legend(loc='upper right')

# creating the 2nd subplot
axes_of_female = plt.subplot(3, 1, 2)
sns.distplot(tmp_df[tmp_df['Sex'] == 'Female']['Age'], ax=axes_of_female, color='g',
axes_of_female.legend(loc='upper right')

# creating the 3rd subplot
axes_of_male = plt.subplot(3, 1, 3)
sns.distplot(tmp_df[tmp_df['Sex'] == 'Male']['Age'], ax=axes_of_male, color='g', kde
axes_of_male.legend(loc='upper right')

plt.show()
```



According to the data I have, the distribution of age is basically accord with normal distribution, regardless the sexuality. Both female and male have the same median of 45 and mean of 43. And the standard deviation of female and male is very close.

Sex	Median	Mean	Standard deviation
Male	45.00	42.45	16.68
Female	45.00	43.27	13.57

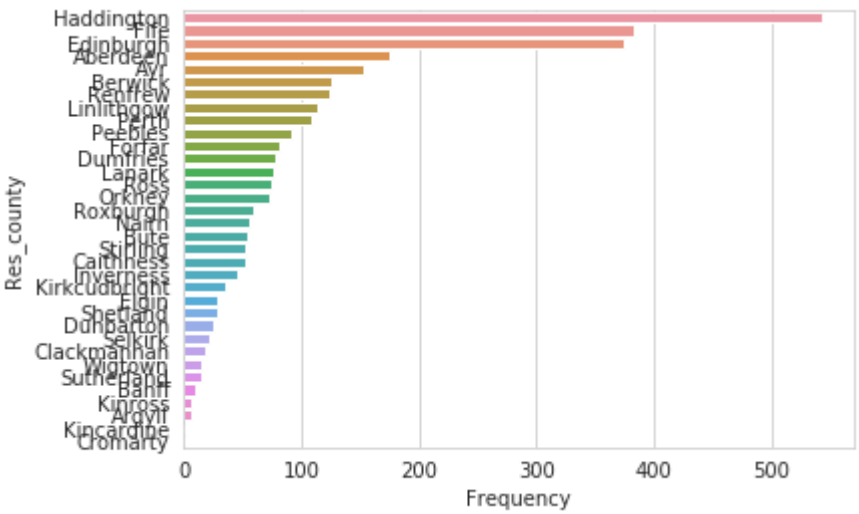
So I came to this conclusion:

Age is not affected by sexuality. And the middle-aged people, such as around 45, are more likely to be accused as a witch.

In [30]:

```
def barplot(column_name, threshold=0.5):
    """ show the barplot figure according to the column_name """
    tmp_df = m_dataframe.dropna(subset=[column_name], inplace=False)
    sum_of_each_col = {}
    for key in tmp_df[column_name].unique():
        selected_rows = tmp_df[column_name] == key
        sum_of_each_col[key] = np.sum(selected_rows.astype(np.int))
    sum_of_each_col = dict(sorted(sum_of_each_col.items(), key=lambda x: x[1], reve
    # create new dataframe to show
    tmp_df = pd.DataFrame(data={column_name: list(sum_of_each_col.keys()),
                                'Frequency': list(sum_of_each_col.values())})
    fig, ax = plt.subplots()
    g = sns.barplot(y=column_name, x='Frequency', data=tmp_df, orient='h')
    plt.ylabel(column_name)
    plt.xlabel('Frequency')
    plt.xticks(rotation='horizontal')
    plt.show()
    tmp_df['proportion'] = tmp_df['Frequency'] / np.sum(tmp_df['Frequency'])
    print(tmp_df)
    sum = 0
    top_n = []
    for index, row in tmp_df.iterrows():
        sum += row.proportion
        top_n.append(row.at[column_name])
        if sum >= threshold:
            break
    print("\nAs shown below, these %d '%s' occupy more than %.2f%% of the total amou
          (len(top_n), column_name, threshold * 100))
    index = 1
    for item in top_n:
        print(" - %d. %s" % (index, item))
        index += 1


barplot('Res_county', threshold=0.7)
```



	Res_county	Frequency	proportion
0	Haddington	543	0.174936
1	Fife	382	0.123067
2	Edinburgh	374	0.120490

```
3         Aberdeen        175    0.056379
4              Ayr        153    0.049291
5          Berwick        126    0.040593
6          Renfrew        124    0.039948
7       Linlithgow        114    0.036727
8            Perth        109    0.035116
9          Peebles         91    0.029317
10          Forfar         82    0.026418
11        Dumfries         78    0.025129
12          Lanark         77    0.024807
13            Ross         74    0.023840
14          Orkney         72    0.023196
15        Roxburgh         60    0.019330
16           Nairn         55    0.017719
17            Bute         54    0.017397
18        Stirling         53    0.017075
19       Caithness         52    0.016753
20       Inverness         45    0.014497
21  Kirkcudbright          35    0.011276
22           Elgin         28    0.009021
23        Shetland         28    0.009021
24       Dunbarton         25    0.008054
25         Selkirk         21    0.006765
26     Clackmannan         18    0.005799
27         Wigtown         15    0.004832
28      Sutherland         15    0.004832
29           Banff          9    0.002899
30         Kinross          7    0.002255
31          Argyll          6    0.001933
32      Kincardine          2    0.000644
33        Cromarty          2    0.000644

As shown below, these 10 'Res_county' occupy more than 70.00% of the t
otal amount:
 - 1. Haddington
 - 2. Fife
 - 3. Edinburgh
 - 4. Aberdeen
 - 5. Ayr
 - 6. Berwick
 - 7. Renfrew
 - 8. Linlithgow
 - 9. Perth
 - 10. Peebles
```

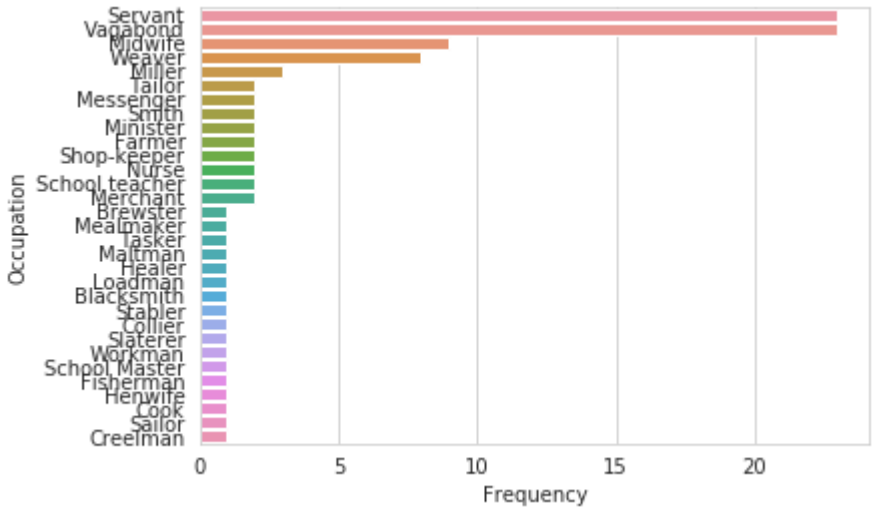
As we can see it clearly, in frequency order, the top 7 counties are **Haddington, Fife, Edinburgh, Aberdeen, Ayr, Berwick and Renfrew.**

I'm not surprised by the high frequency of Edinburgh. **The main reason is that the information is missing badly and the location information is not available in the past. Therefore, Edinburgh was taken as the best option to fill out.**

Haddington and Fife occupy such a high proportion, 17.49% and 12.31%, perhaps related to the historical background at the time.

In [31]:

```
barplot('Occupation', threshold=0.7)
```



	Occupation	Frequency	proportion
0	Servant	23	0.227723
1	Vagabond	23	0.227723
2	Midwife	9	0.089109
3	Weaver	8	0.079208
4	Miller	3	0.029703
5	Tailor	2	0.019802
6	Messenger	2	0.019802
7	Smith	2	0.019802
8	Minister	2	0.019802
9	Farmer	2	0.019802
10	Shop-keeper	2	0.019802
11	Nurse	2	0.019802
12	School teacher	2	0.019802
13	Merchant	2	0.019802
14	Brewster	1	0.009901
15	Mealmaker	1	0.009901
16	Tasker	1	0.009901
17	Maltman	1	0.009901
18	Healer	1	0.009901
19	Loadman	1	0.009901
20	Blacksmith	1	0.009901
21	Stabler	1	0.009901
22	Collier	1	0.009901
23	Slaterer	1	0.009901
24	Workman	1	0.009901
25	School Master	1	0.009901
26	Fisherman	1	0.009901
27	Henwife	1	0.009901
28	Cook	1	0.009901
29	Sailor	1	0.009901
30	Creelman	1	0.009901

As shown below, these 8 'Occupation' occupy more than 70.00% of the to
tal amount:
- 1. Servant
- 2. Vagabond
- 3. Midwife

- 4. Weaver
- 5. Miller
- 6. Tailor
- 7. Messenger
- 8. Smith

As we can see from the top 8 occupations, I can roughly come to this conclusion:

these positions belong to the service industry, and their social status is low. They are unable to resist and are thus persecuted.

Analyse variables

I want to analyze the relationship between 'Res_county' and 'SocioecStatus'. Because we have already got the sorted list of counties according to the frequence of accused, so here we only analyze the top 20 counties.

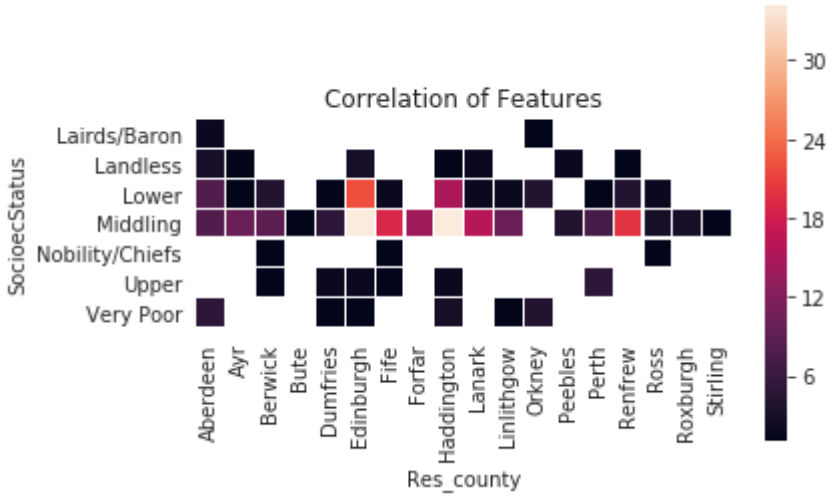
In [32]:

```
subset = ['SocioecStatus', 'Res_county']
county = ['Haddington', 'Fife', 'Edinburgh', 'Aberdeen', 'Ayr', 'Berwick', 'Renfrew
          'Peebles', 'Forfar', 'Dumfries', 'Lanark', 'Ross', 'Orkney', 'Roxburgh',
          'Caithness']

tmp_df = m_dataframe.dropna(subset=subset, inplace=False)
tmp_df = tmp_df[['SocioecStatus', 'Res_county']]
tmp_df = tmp_df[tmp_df['Res_county'].isin(county)]
tmp_df['count'] = np.ones(shape=tmp_df.shape[0])

pt = tmp_df.pivot_table(index='SocioecStatus', columns='Res_county', values='count',

sns.heatmap(pt, linewidths=0.1, robust=True, square=True)
plt.title("Correlation of Features")
plt.show()
```



These is a **negative correlation** between **the social status** and **the number of accusations**.

According to the heat map, we can see that the social status of the people in counties with high frequency of accusations is mostly in the middling and lower classes. And the top 3 counties have higher heat values and lighter colors in 'Middling' and 'Lower' row, which explains that **social class is also an important factor affecting accusations.**

Significant factor & trends

Based on the above analysis of the data, we can confirm that 'sex' is the most significant factor in accusation, and the following factors are social status and occupation.

Female are more likely to be accused than male and people with lower social status are also more likely to be accused. At the same time, we can also see a trend in which the high proportion of social status is a small proportion of the accused, and the larger proportion is the people with lower social status.

Reflect and Hypothesise

Reflection on the data (200 words)

According to the analysis of my data in the module above, several important results are that the number of women being accused of witchcraft is much higher than that of men, the number of people being accused of witchcraft is much higher in some regions than in other regions, and the lower the social class people in the existing data samples, the easier it is to be accused of witchcraft. From these outstanding results, it can be found that the people accused of witchcraft are not randomly selected, the people with certain characteristics are vulnerable to prosecution. Gender, geographical location and social class are all important indicators of political persecution. Due to the influence of the age of geographical information, the geographical names at that time may no longer exist, and other reasons, some geographical information is not accurate enough. Therefore, the problems and assumptions reflected by geographical information are not accurate enough, so I don't consider too much about the conclusions obtained by analyzing geographical information. But it is certain that women and those of lower social rank are more likely to be accused of witchcraft. This reflects the social situation of Scotland at that time. The deep meaning of witch hunting may be the persecution and oppression of women and people under the social hierarchy by the patriarchal bourgeoisie in the name of religious belief, these data are the epitome of the dark social background of Scotland at that time.

What your hypothesis is?

Hypothesis 1 and Explanation

Hypothesis: The middle and lower classes and the proletariat are more likely to be accused of witches because the bourgeoisie dominates.

Explanation: According to the above figure named "Occupation", eight kinds of occupations including Servant, Vagabond, Midwife, Weaver, Miller, Tailor, Messenger and Smith, which are more likely to be accused of witchcraft. In the social environment at that time, these occupations were of middle or lower social class, so one of my hypothesis is The middle and lower classes and the proletariat are more likely to be accused of witchcraft because of the dominance of the bourgeoisie.

Test method: Consulting literature for further studies about the social background at that time, and obtain more relevant data from the data holder to use graph to test my hypothesis.

Hypothesis 2 and Explanation

Hypothesis: Older people are more likely to accused of witchcraft.

Explanation: According to the above figures named "Dist of Accused age","Dist of Female age" and "Dist of Male age", the average age and the median age of people accused of witchcraft in the available data are all above 40. After consulted literature I know the average life expectancy of people living near this area at that time was between 40 and 50, so one of my hypothesis is older people are more likely to accused of witchcraft.

Test method: Consulting literature for further studies about the social background at that time, and obtain more relevant data from the data holder to use graph to test my hypothesis.

Hypothesis 3 and Explanation

Hypothesis: Women were more likely to be accused of witchcraft because of the male-dominated society's persecution of women

Explanation: According to the above figure named "Proportion of Male and Female", of all the genders in the data, women were far more likely to be accused of being witches than men. According to literature review, the male and female birthrates were similar at that time, excluding the possibility that the total number of females was more than that of males. And since the rise of Christianity was mentioned in the literature as a result of a large number of men working in the priesthood, so that men's rights grew. so one of my hypothesis is Women were more likely to be accused of witchcraft because of the male-dominated society's persecution of women.

Test method: Consulting literature for further studies about the social background at that time, and obtain more relevant data from the data holder to use graph to test my hypothesis.