Haberman Survival Status

In [2]:

```
import pandas as pd
import numpy as np
hbrmn = pd.read_csv("haberman.csv", names =["age","op_year","axil_nodes","surv_status"
])
```

In [3]:

```
hbrmn.head()
```

Out[3]:

	age	op_year	axil_nodes	surv_status
0	30	64	1	1
1	30	62	3	1
2	30	65	0	1
3	31	59	2	1
4	31	65	4	1

In [4]:

```
hbrmn.columns
```

Out[4]:

```
Index(['age', 'op_year', 'axil_nodes', 'surv_status'], dtype='object')
```

In [5]:

```
hbrmn.shape
```

Out[5]:

```
(306, 4)
```

In [6]:

```
hbrmn.surv_status.value_counts()
```

Out[6]:

```
1    225
2     81
Name: surv_status, dtype: int64
```

In [40]:

```
# Status of alive patients within 5 years of operation
hbrmn[hbrmn['surv_status']==1].describe()
```

Out[40]:

	age	op_year	axil_nodes	surv_status
count	225.000000	225.000000	225.000000	225.0
mean	52.017778	62.862222	2.791111	1.0
std	11.012154	3.222915	5.870318	0.0
min	30.000000	58.000000	0.000000	1.0
25%	43.000000	60.000000	0.000000	1.0
50%	52.000000	63.000000	0.000000	1.0
75%	60.000000	66.000000	3.000000	1.0
max	77.000000	69.000000	46.000000	1.0

In [41]:

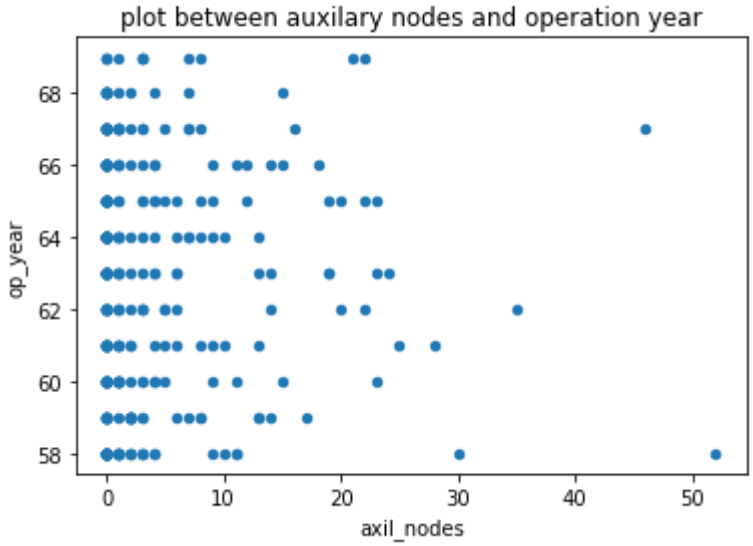
```
# Status of died patients within 5 years of operation
hbrmn[hbrmn['surv_status']==2].describe()
```

Out[41]:

	age	op_year	axil_nodes	surv_status
count	81.000000	81.000000	81.000000	81.0
mean	53.679012	62.827160	7.456790	2.0
std	10.167137	3.342118	9.185654	0.0
min	34.000000	58.000000	0.000000	2.0
25%	46.000000	59.000000	1.000000	2.0
50%	53.000000	63.000000	4.000000	2.0
75%	61.000000	65.000000	11.000000	2.0
max	83.000000	69.000000	52.000000	2.0

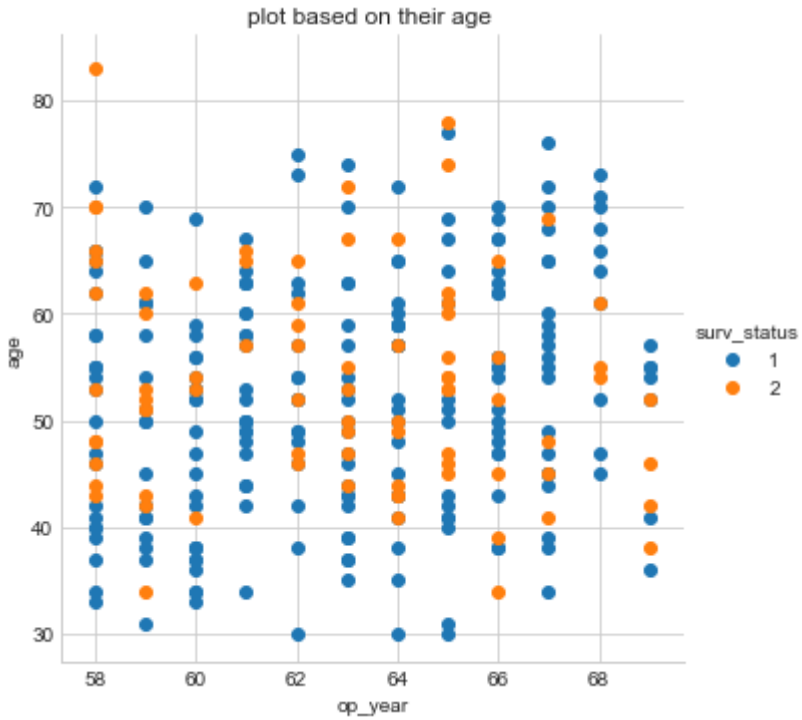
In [10]:

```
import matplotlib.pyplot as plt
import seaborn as sns
hbrmn.plot(kind ='scatter',x = 'axil_nodes' ,y = 'op_year')
plt.title('plot between auxilary nodes and operation year')
plt.show()
```



In [11]:

```
sns.set_style("whitegrid");
sns.FacetGrid(hbrmn, hue="surv_status", size=5) \
   .map(plt.scatter, "op_year","age") \
   .add_legend();
plt.title('plot based on their age ')
plt.show();
```

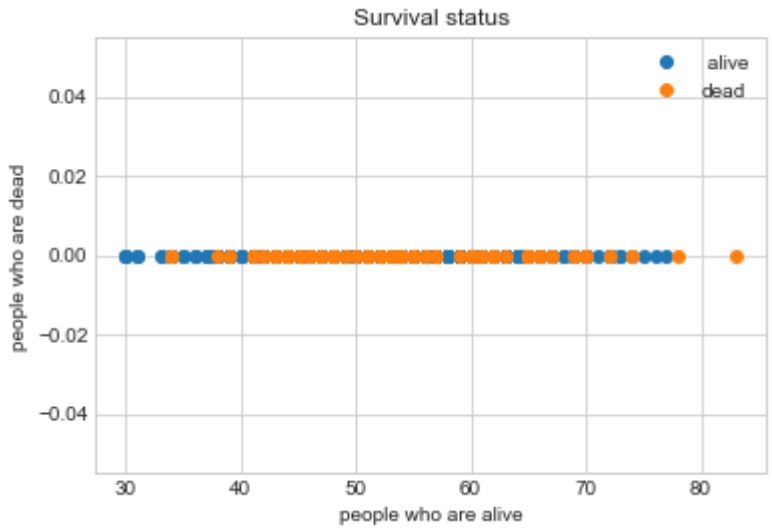


In [20]:

```
hbrmn_alive = hbrmn.loc[hbrmn["surv_status"] == 1];
hbrmn_dead = hbrmn.loc[hbrmn["surv_status"]== 2];
alive, = plt.plot(hbrmn_alive["age"], np.zeros_like(hbrmn_alive['age']), 'o', label = '
alive')
dead, = plt.plot(hbrmn_dead["age"], np.zeros_like(hbrmn_dead['age']),'o', label='dead')
plt.title('Survival status')
plt.xlabel('people who are alive')
plt.ylabel('people who are dead')
plt.legend()
```

Out[20]:

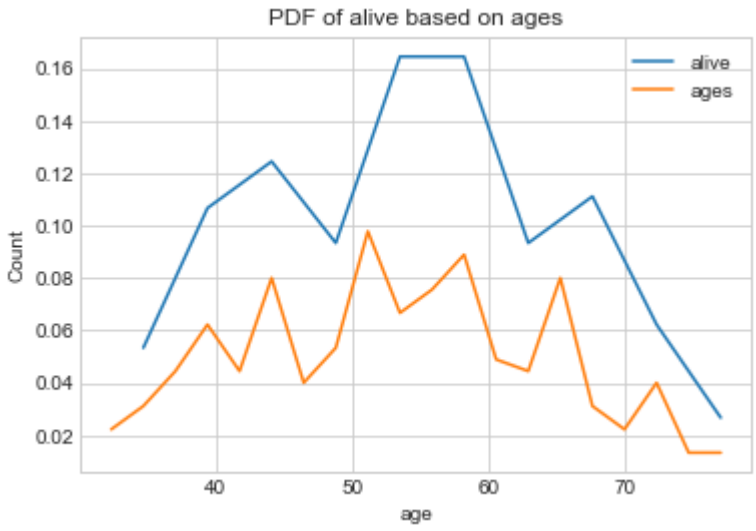
```
<matplotlib.legend.Legend at 0x15ea108b400>
```



In [31]:

```
counts, bin_edges = np.histogram(hbrmn_alive['age'], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges);
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf);


counts, bin_edges = np.histogram(hbrmn_alive['age'], bins=20,
                                 density = True)
pdf = counts/(sum(counts))
plt.plot(bin_edges[1:],pdf);

plt.title('PDF of alive based on ages')
plt.xlabel('age')
plt.ylabel('Count')
plt.legend(['alive', 'ages'])
plt.show();
```

```
[0.05333333 0.10666667 0.12444444 0.09333333 0.16444444 0.16444444
 0.09333333 0.11111111 0.06222222 0.02666667]
[30.  34.7 39.4 44.1 48.8 53.5 58.2 62.9 67.6 72.3 77. ]
```



In [32]:

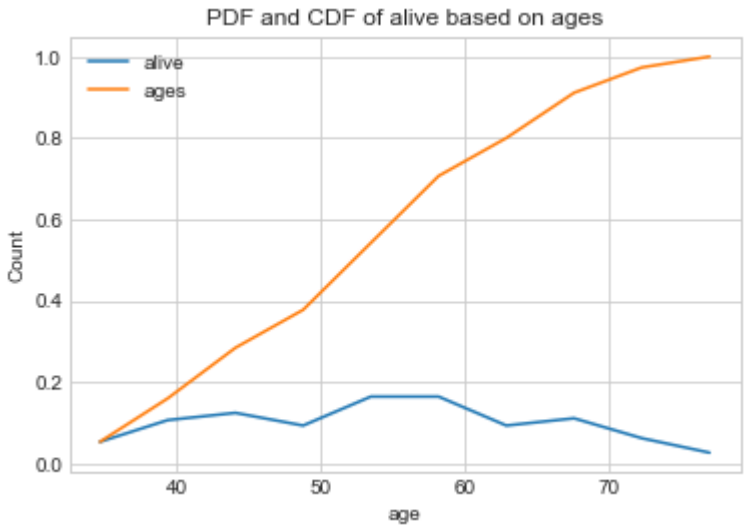
```
counts, bin_edges = np.histogram(hbrmn_dead['age'], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges);
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf);


counts, bin_edges = np.histogram(hbrmn_dead['age'], bins=20,
                                 density = True)
pdf = counts/(sum(counts))
plt.plot(bin_edges[1:],pdf);

plt.title('PDF of dead based on ages')
plt.xlabel('age')
plt.ylabel('Count')
plt.legend(['dead', 'ages'])
plt.show();
```

```
[0.03703704 0.12345679 0.19753086 0.19753086 0.13580247 0.12345679
 0.09876543 0.04938272 0.02469136 0.01234568]
[34.  38.9 43.8 48.7 53.6 58.5 63.4 68.3 73.2 78.1 83. ]
```

In [27]:

```
counts, bin_edges = np.histogram(hbrmn_alive['age'], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)

#compute CDF
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:], cdf)

plt.title('PDF and CDF of alive based on ages')
plt.xlabel('age')
plt.ylabel('Count')
plt.legend(['alive', 'ages'])

plt.show();
```

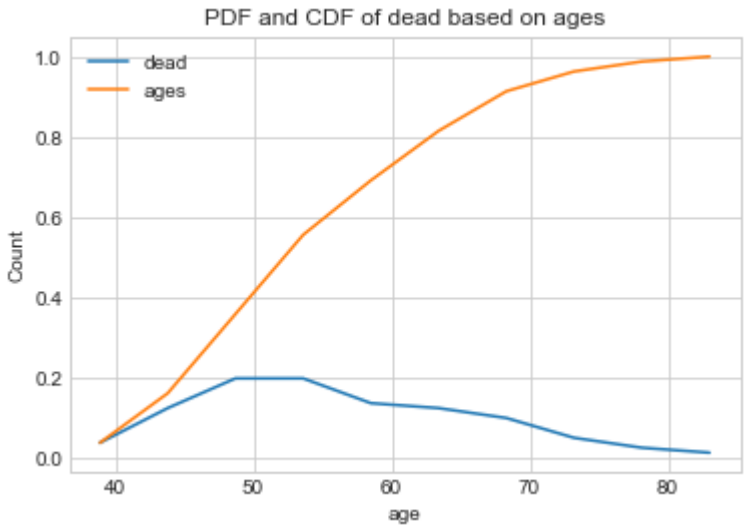
```
[0.05333333 0.10666667 0.12444444 0.09333333 0.16444444 0.16444444
 0.09333333 0.11111111 0.06222222 0.02666667]
[30.  34.7 39.4 44.1 48.8 53.5 58.2 62.9 67.6 72.3 77. ]
```



In [28]:

```
counts, bin_edges = np.histogram(hbrmn_dead['age'], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)

#compute CDF
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:], cdf)

plt.title('PDF and CDF of dead based on ages')
plt.xlabel('age')
plt.ylabel('Count')
plt.legend(['dead', 'ages'])
plt.show();
```

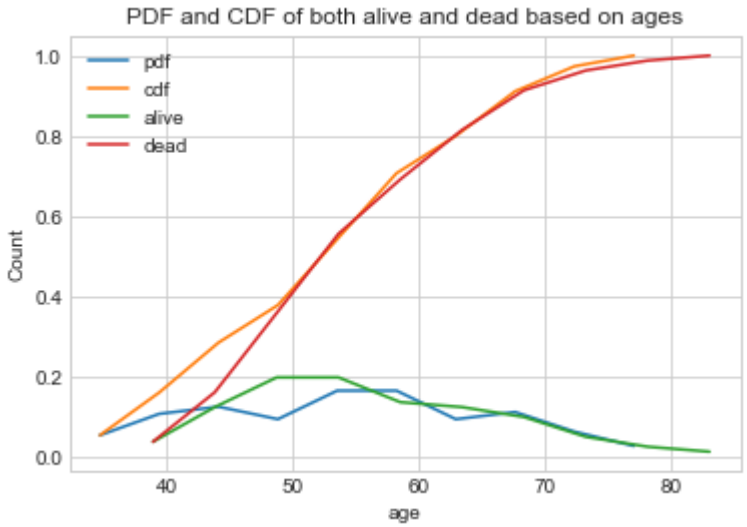
```
[0.03703704 0.12345679 0.19753086 0.19753086 0.13580247 0.12345679
 0.09876543 0.04938272 0.02469136 0.01234568]
[34.  38.9 43.8 48.7 53.6 58.5 63.4 68.3 73.2 78.1 83. ]
```



In [33]:

```
# alive
counts, bin_edges = np.histogram(hbrmn_alive['age'], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:], cdf)


# dead
counts, bin_edges = np.histogram(hbrmn_dead['age'], bins=10,
                                 density = True)
pdf = counts/(sum(counts))
print(pdf);
print(bin_edges)
cdf = np.cumsum(pdf)
plt.plot(bin_edges[1:],pdf)
plt.plot(bin_edges[1:], cdf)

plt.title('PDF and CDF of both alive and dead based on ages')
plt.xlabel('age')
plt.ylabel('Count')
plt.legend(['pdf','cdf', 'alive','dead'])
plt.show()
```

```
[0.05333333 0.10666667 0.12444444 0.09333333 0.16444444 0.16444444
 0.09333333 0.11111111 0.06222222 0.02666667]
[30.  34.7 39.4 44.1 48.8 53.5 58.2 62.9 67.6 72.3 77. ]
[0.03703704 0.12345679 0.19753086 0.19753086 0.13580247 0.12345679
 0.09876543 0.04938272 0.02469136 0.01234568]
[34.  38.9 43.8 48.7 53.6 58.5 63.4 68.3 73.2 78.1 83. ]
```



In [29]:

```
#Mean, Variance, Std-deviation,
print("Means:")
print(np.mean(hbrmn_alive["age"]))
#Mean with an outlier.
print(np.mean(np.append(hbrmn_alive["age"],50)));
print(np.mean(hbrmn_dead["age"]))

print("\nStd-dev:");
print(np.std(hbrmn_alive["age"]))
print(np.std(hbrmn_dead["age"]))
```

```
Means:
52.01777777777778
52.008849557522126
53.67901234567901

Std-dev:
10.98765547510051
10.10418219303131
```

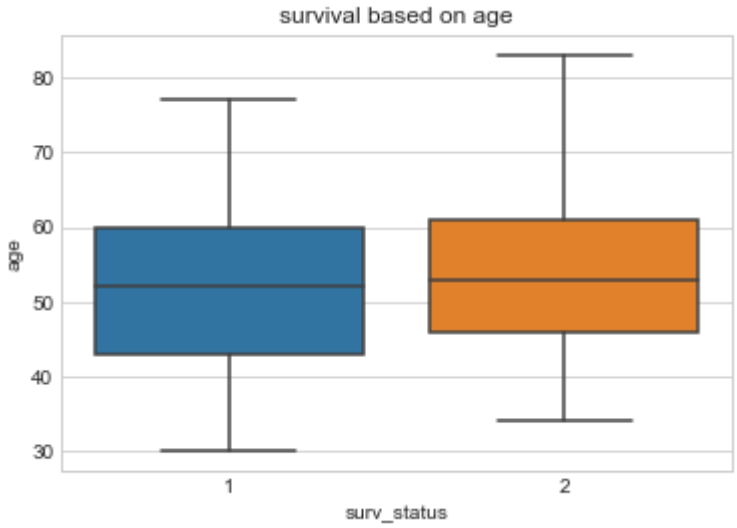
In [30]:

```
#Median, Quantiles, Percentiles, IQR.
print("\nMedians:")
print(np.median(hbrmn_alive["age"]))
#Median with an outlier
print(np.median(np.append(hbrmn_alive["age"],50)));
print(np.median(hbrmn_dead["age"]))

print("\nQuantiles:")
print(np.percentile(hbrmn_alive["age"],np.arange(0, 100, 25)))
print(np.percentile(hbrmn_dead["age"],np.arange(0, 100, 25)))

print("\n90th Percentiles:")
print(np.percentile(hbrmn_alive["age"],90))
print(np.percentile(hbrmn_dead["age"],90))

from statsmodels import robust
print ("\nMedian Absolute Deviation")
print(robust.mad(hbrmn_alive["age"]))
print(robust.mad(hbrmn_dead["age"]))
```

```
Medians:
52.0
52.0
53.0

Quantiles:
[30. 43. 52. 60.]
[34. 46. 53. 61.]

90th Percentiles:
67.0
67.0

Median Absolute Deviation
13.343419966550417
11.860817748044816
```

In [34]:

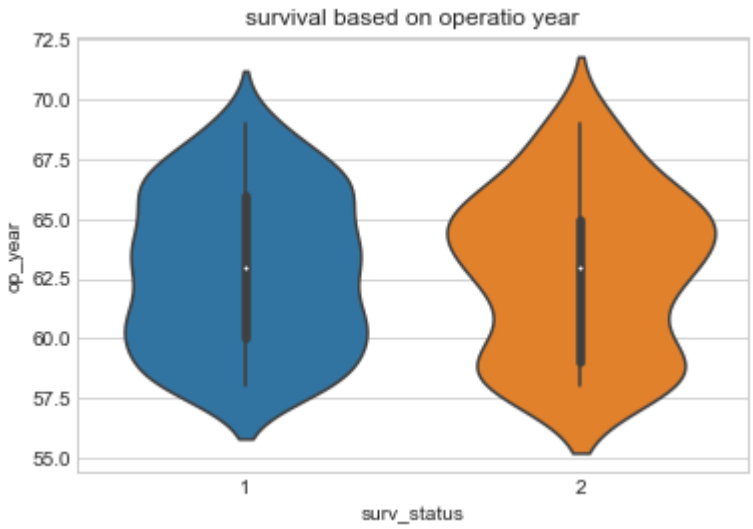
```
sns.boxplot(x='surv_status',y='age', data=hbrmn)
plt.title('survival based on age')
plt.show()
```



In [35]:

```
sns.violinplot(x='surv_status',y='op_year', data=hbrmn, size = 7)
plt.title('survival based on operatio year')
plt.show()
```

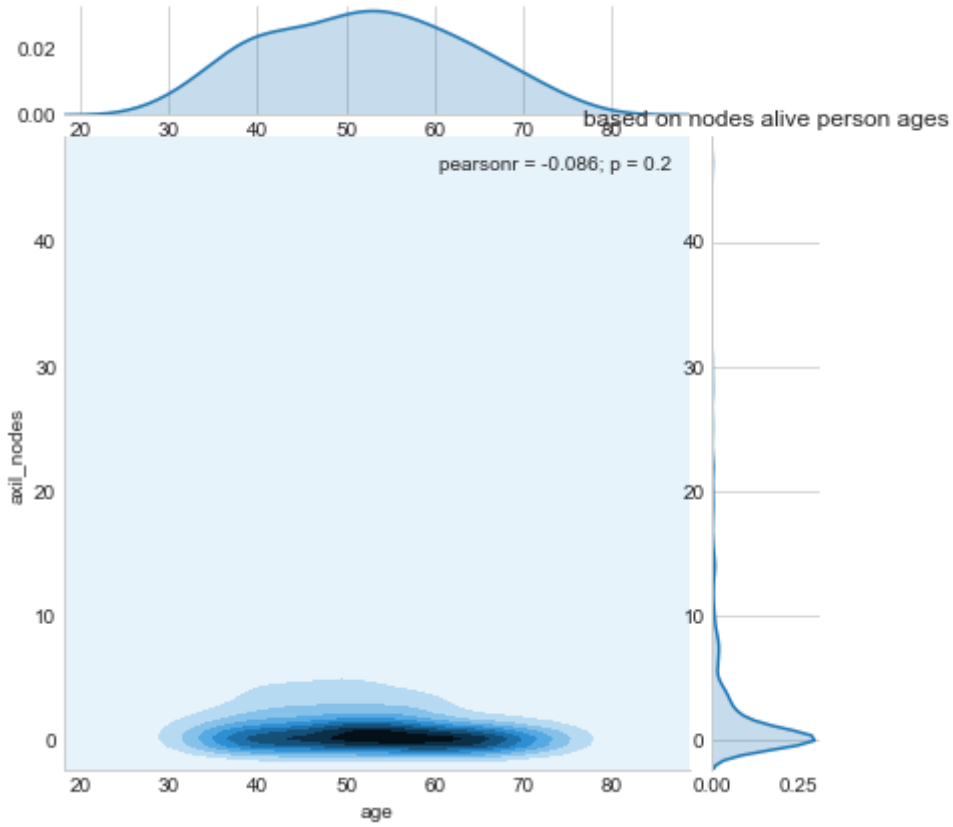
```
C:\Users\SUBHODAYA KUMAR\Anaconda3\lib\site-packages\scipy\stats\stats.py:
1713: FutureWarning: Using a non-tuple sequence for multidimensional index
ing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the fut
ure this will be interpreted as an array index, `arr[np.array(seq)]`, whic
h will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



In [36]:

```
sns.jointplot(x="age", y="axil_nodes", data=hbrmn_alive, kind="kde");
plt.title('based on nodes alive person ages')
plt.show();
```

```
C:\Users\SUBHODAYA KUMAR\Anaconda3\lib\site-packages\scipy\stats\stats.py:
1713: FutureWarning: Using a non-tuple sequence for multidimensional index
ing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the fut
ure this will be interpreted as an array index, `arr[np.array(seq)]`, whic
h will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```



In [37]:

```
sns.jointplot(x="age", y="axil_nodes", data=hbrmn_dead, kind="kde");
plt.title('based on nodes dead person age')
plt.show();
```

```
C:\Users\SUBHODAYA KUMAR\Anaconda3\lib\site-packages\scipy\stats\stats.py:
1713: FutureWarning: Using a non-tuple sequence for multidimensional index
ing is deprecated; use `arr[tuple(seq)]` instead of `arr[seq]`. In the fut
ure this will be interpreted as an array index, `arr[np.array(seq)]`, whic
h will result either in an error or a different result.
  return np.add.reduce(sorted[indexer] * weights, axis=axis) / sumval
```