In [1]:

```
import pandas as pd
import numpy as np
from scipy.stats import trim_mean
from statsmodels import robust
import wquantiles

import seaborn as sns
import matplotlib.pylab as plt
```

In [2]:

```
df = pd.read_csv('./IT Salary Survey EU 2018.csv')
```

In [3]:

```
df.columns
```

Out[3]:

```
Index(['Timestamp', 'Age', 'Gender', 'City', 'Position', 'Years of experience',
       'Your level', 'Current Salary', 'Salary one year ago',
       'Salary two years ago', 'Are you getting any Stock Options?',
       'Main language at work', 'Company size', 'Company type'],
      dtype='object')
```

In [4]:

```
df.drop(['Timestamp', 'Salary one year ago', 'Salary two years ago', 'Are you getti
```

Out[4]:

	Age	Gender	City	Position	Years of experience	Your level	Current Salary	Main language at work
0	43.0	M	München	QA Ingenieur	11.0	Senior	77000.0	Deutsch
1	33.0	F	München	Senior PHP Magento developer	8.0	Senior	65000.0	Deutsch
2	32.0	M	München	Software Engineer	10.0	Senior	88000.0	Deutsch
3	25.0	M	München	Senior Frontend Developer	6.0	Senior	78000.0	English
4	39.0	M	München	UX Designer	10.0	Senior	69000.0	English
...	...	...	...	...	...	...	...	...
760	40.0	M	Köln	Java Developer junior	1.0	Junior	44000.0	Deutsch
761	NaN	M	Köln	E.g. C# Developer	1.0	Junior	45000.0	Deutsch
762	NaN	M	Köln	E.g. C# Developer	1.0	Junior	45000.0	Deutsch
763	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
764	31.0	F	München	Pm	10.0	Senior	110000.0	English

765 rows × 8 columns

In [5]:

```
df.rename(columns = {'Current Salary':'salary', 'Years of experience':'experience',
```

In [6]:

```
df.fillna(df.median(), inplace=True)
```

```
C:\Users\dyota\AppData\Local\Temp\ipykernel_7888\3604797450.py:1: FutureWarning: Dr
opping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is de
precated; in a future version this will raise TypeError.  Select only valid columns
before calling the reduction.
  df.fillna(df.median(), inplace=True)
```

In [7]:

```
df = df.dropna()
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 686 entries, 0 to 764
Data columns (total 14 columns):
 #   Column                              Non-Null Count  Dtype  
---  ------                              --------------  -----  
 0   Timestamp                           686 non-null    object 
 1   Age                                 686 non-null    float64
 2   Gender                              686 non-null    object 
 3   City                                686 non-null    object 
 4   Position                            686 non-null    object 
 5   experience                          686 non-null    float64
 6   level                               686 non-null    object 
 7   salary                              686 non-null    float64
 8   Salary one year ago                 686 non-null    float64
 9   Salary two years ago                686 non-null    float64
 10  Are you getting any Stock Options?  686 non-null    object 
 11  lang                                686 non-null    object 
 12  Company size                        686 non-null    object 
 13  Company type                        686 non-null    object 
dtypes: float64(5), object(9)
memory usage: 80.4+ KB
```

Estimate of Location

Location estimate of price and number of reviews

Mean

In [8]:

```
mean_salary = df['salary'].mean()
trimmed_mean_salary = trim_mean(df['salary'], 0.1)
weighted_mean_salary = np.average(df['salary'], weights=df['experience'])

print(f'Mean price = ${mean_salary}')
print(f'Trimmed mean price = ${trimmed_mean_salary}')
print(f'Weighted mean price = ${weighted_mean_salary}')
```

```
Mean price = $67907.03206997084
Trimmed mean price = $66543.78181818181
Weighted mean price = $71915.2170343766
```

Median

In [9]:

```
median_salary = df['salary'].median()
weighted_median_salary = wquantiles.median(df['salary'], weights=df['experience'])

print(f'Median price = ${median_salary}')
print(f'Weighted median price = ${weighted_median_salary}')
```

```
Median price = $65000.0
Weighted median price = $70000.0
```

Estimates of Variablity

In [10]:

```
print(f'Standard Deviation = {df["salary"].std()}')
print(f'Median Absolute Deviation = {robust.scale.mad(df["salary"])}')
```

```
Standard Deviation = 19485.25344692599
Median Absolute Deviation = 14826.02218505602
```

Percentile and Boxplot

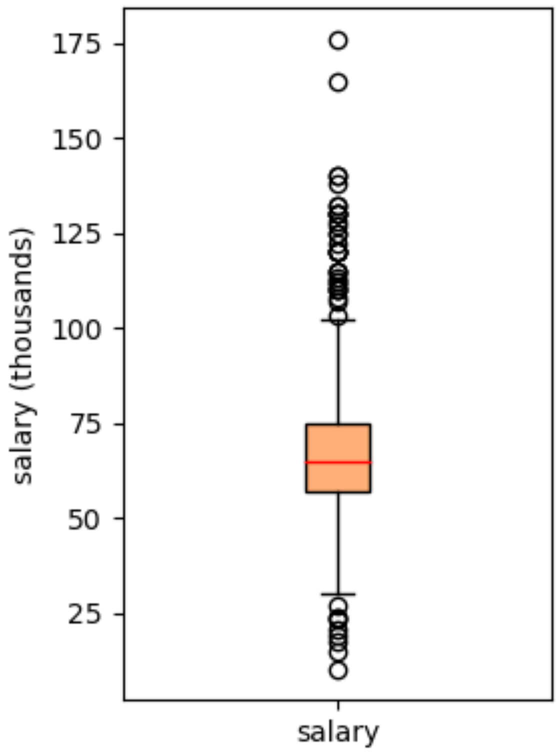
In [11]:

```
print(df['salary'].quantile([0.05, 0.25, 0.5, 0.75, 0.95]))
```

```
0.05     40200.0
0.25     57000.0
0.50     65000.0
0.75     75000.0
0.95    101500.0
Name: salary, dtype: float64
```

In [12]:

```
ax = (df['salary']/1000).plot.box(figsize=(3, 4), color='black',  patch_artist = Tr
ax.set_ylabel('salary (thousands)')

plt.tight_layout()
plt.show()
```

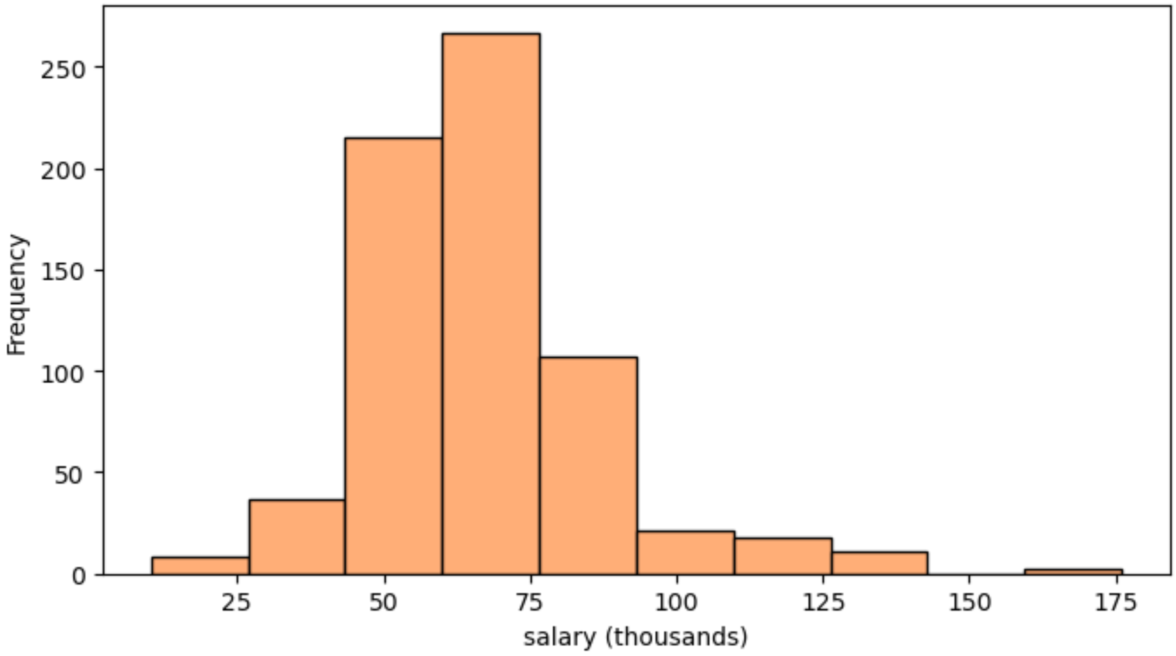


Frequency table and Histogram

In [13]:
```
binnedPopulation = pd.cut(df['salary'], 10)
print(binnedPopulation.value_counts())
```

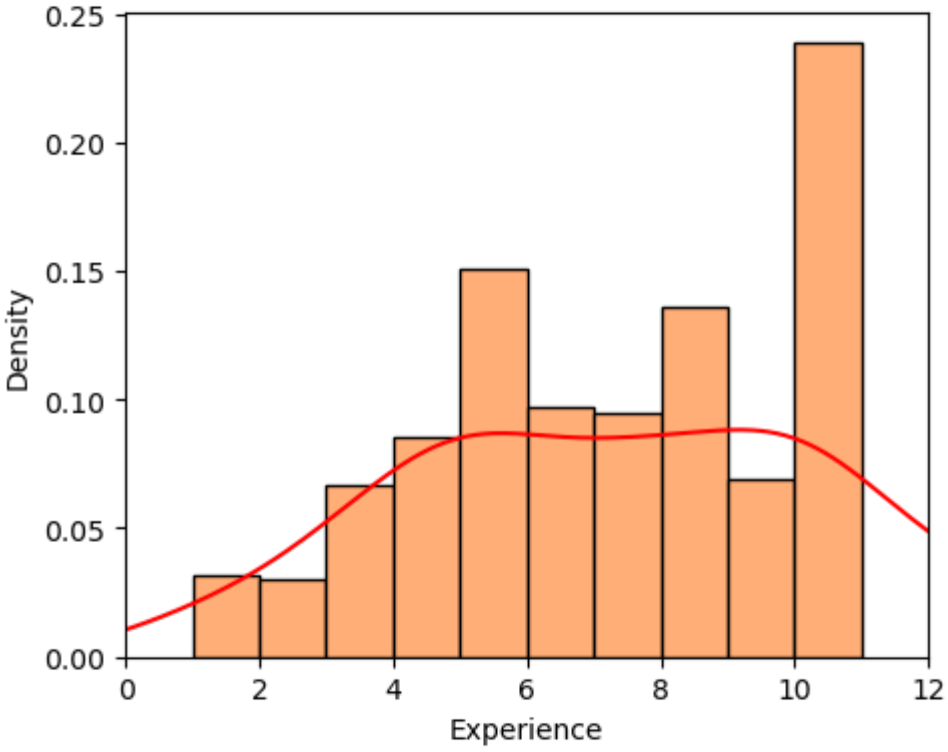
```
(60010.0, 76580.0]      267
(43440.0, 60010.0]      215
(76580.0, 93150.0]      107
(26870.0, 43440.0]       37
(93150.0, 109720.0]      21
(109720.0, 126290.0]     18
(126290.0, 142860.0]     11
(10134.3, 26870.0]        8
(159430.0, 176000.0]      2
(142860.0, 159430.0]      0
Name: salary, dtype: int64
```

In [14]:
```
ax = (df["salary"]/1000).plot.hist(figsize=(7, 4), color='#ffae77', edgecolor='blac
ax.set_xlabel('salary (thousands)')

plt.tight_layout()
plt.show()
```



Density Estimate

In [15]:
```
ax = df['experience'].plot.hist(density=True, xlim=[0, 12], bins=range(1,12), figsi
df['experience'].plot.density(ax=ax, color='red')
ax.set_xlabel('Experience')

plt.tight_layout()
plt.show()
```



Exploring Binary and Categorical Data

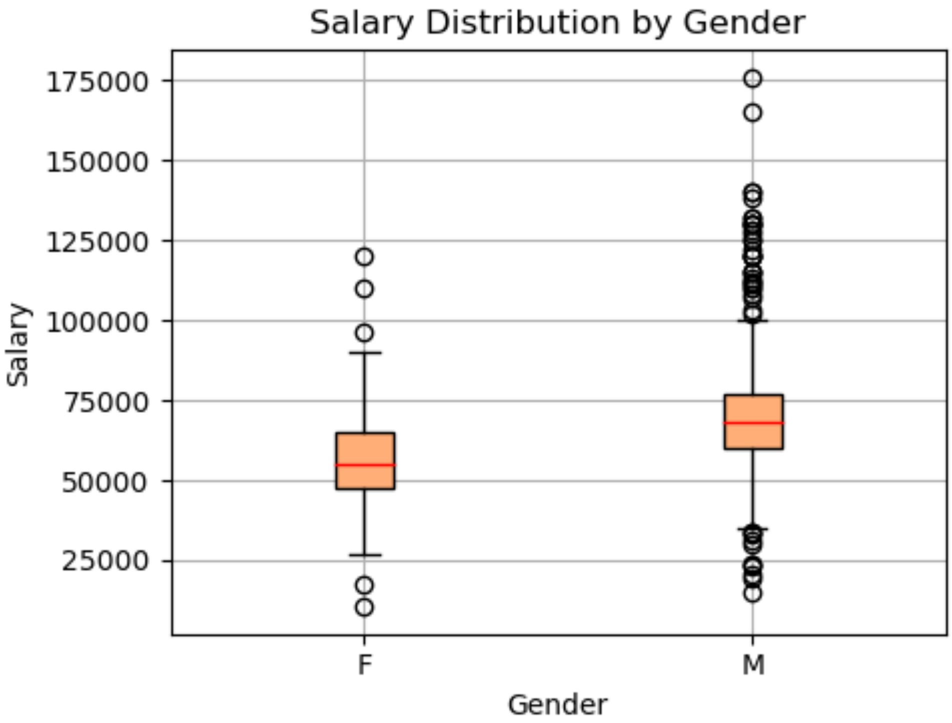
In [16]:

```
print(df["Gender"].unique())
print(df["lang"].unique())
```

```
['M' 'F']
['Deutsch' 'English' 'Russian' 'French'
 'Team - Russian; Cross-team - English;' 'Polish' 'Deutsch/Englisch']
```

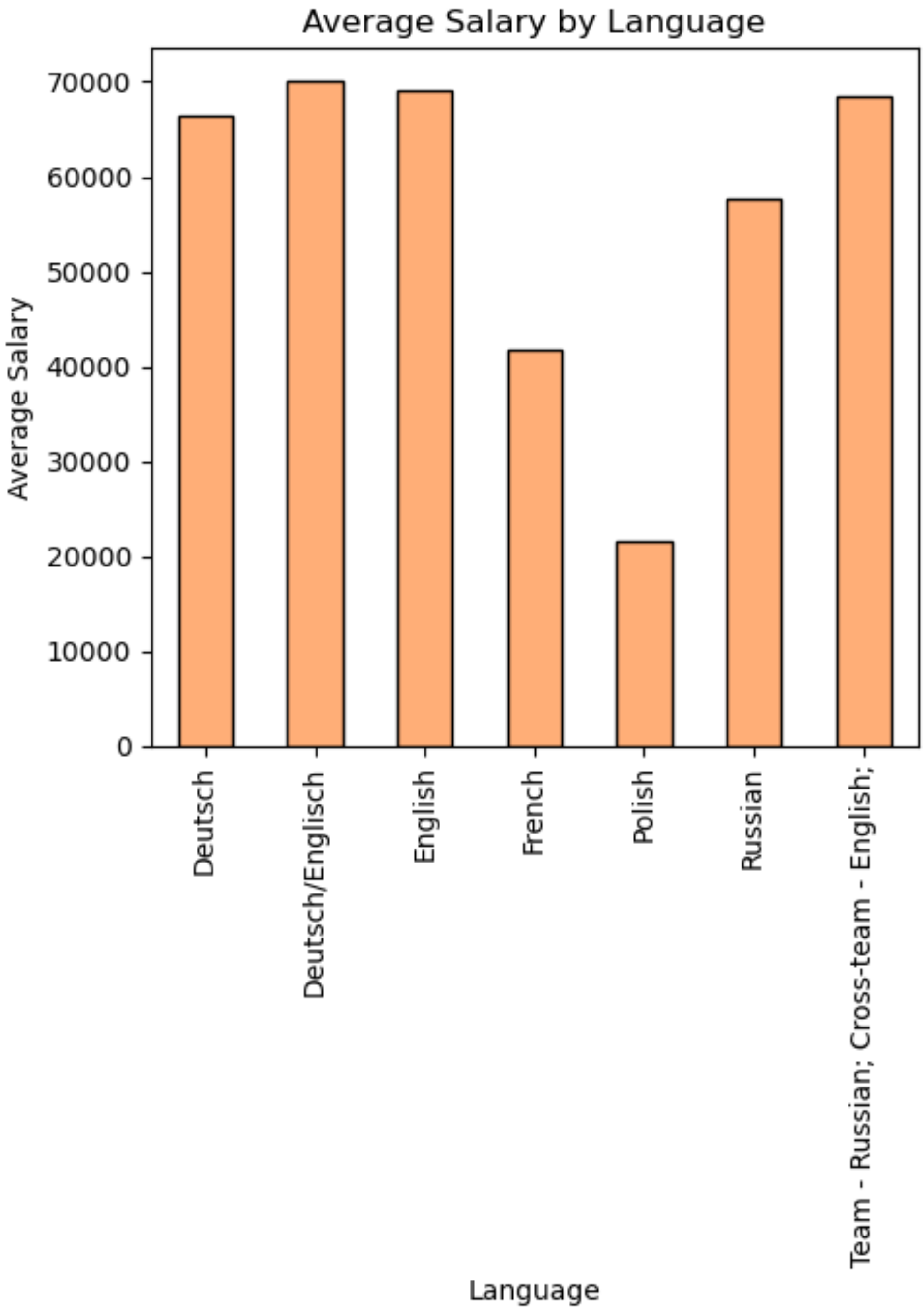
In [17]:

```
ax = df.boxplot(column='salary', by='Gender', figsize=(5, 4), color='black',  patch
ax.set_xlabel('Gender')
ax.set_ylabel('Salary')
plt.title('Salary Distribution by Gender')

plt.suptitle('')
plt.tight_layout()
plt.show()
```



In [18]:

```
language_salary_mean = df.groupby('lang')['salary'].mean()

ax = language_salary_mean.plot(kind='bar', figsize=(5, 7), color='#ffae77', edgecol

ax.set_xlabel('Language')
ax.set_ylabel('Average Salary')
plt.title('Average Salary by Language')

plt.tight_layout()
plt.show()
```

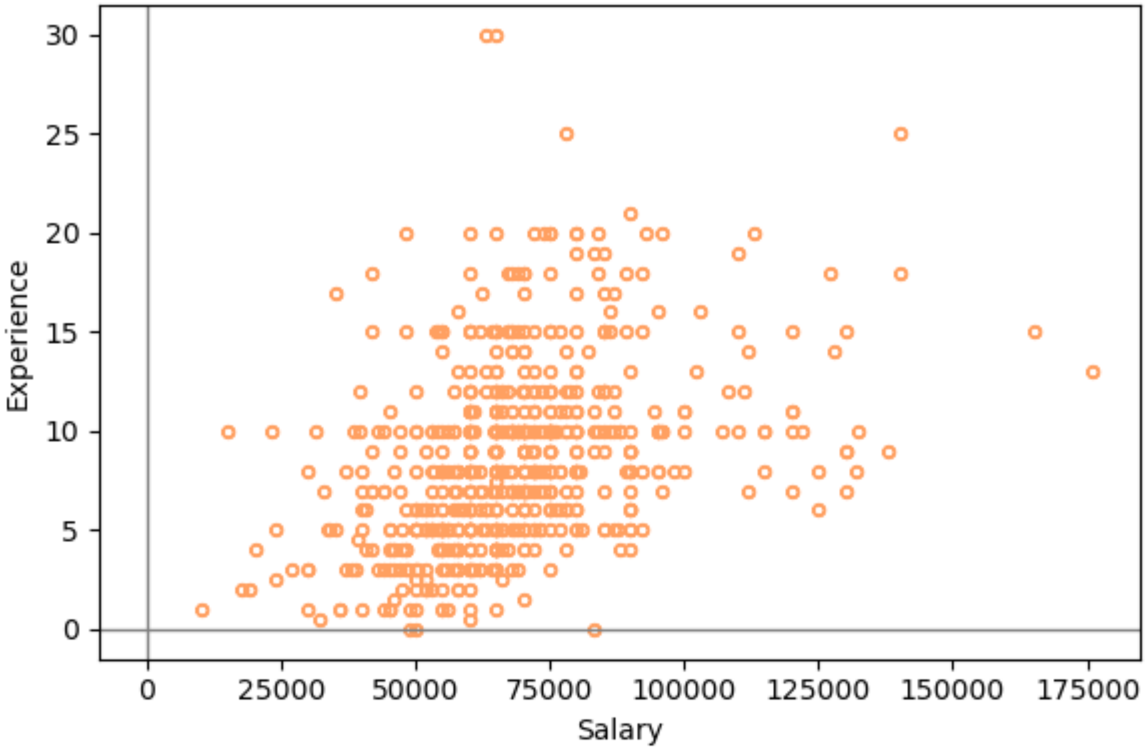


Scatterplots

In [19]:

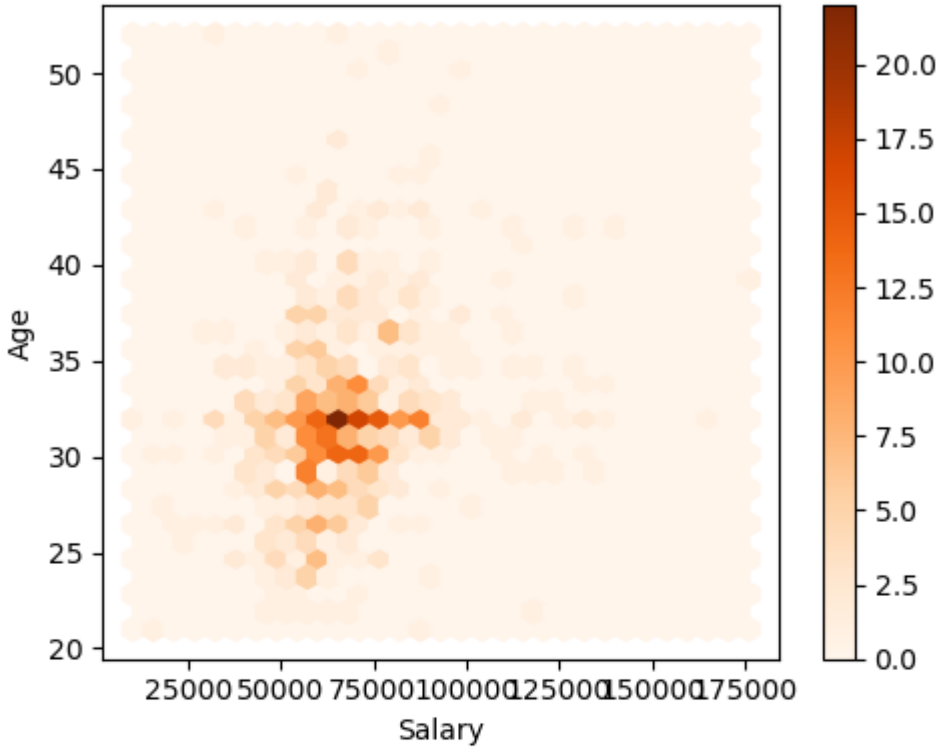
```
ax = df.plot.scatter(x='salary', y='experience', figsize=(6, 4), marker='$\u25EF$',
ax.set_xlabel('Salary')
ax.set_ylabel('Experience')
ax.axhline(0, color='grey', lw=1)
ax.axvline(0, color='grey', lw=1)

plt.tight_layout()
plt.show()
```

```
C:\Users\dyota\anaconda3\lib\site-packages\pandas\plotting\_matplotlib\core.py:111
4: UserWarning: No data for colormapping provided via 'c'. Parameters 'cmap' will b
e ignored
  scatter = ax.scatter(
```

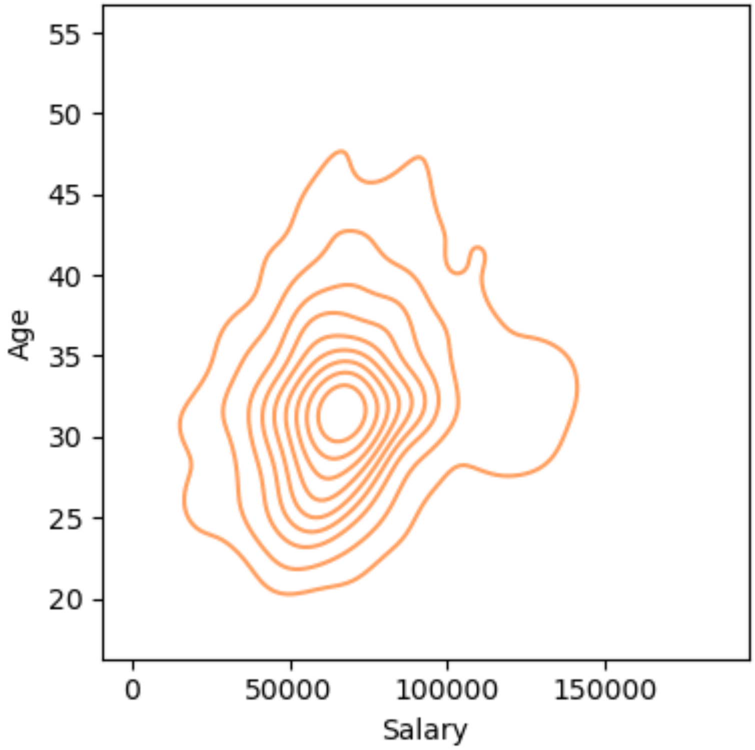


Hexagonal binning and Contour

In [20]:
```
ax = df.plot.hexbin(x='salary', y='Age', gridsize=30, sharex=False, figsize=(5, 4),
ax.set_xlabel('Salary')
ax.set_ylabel('Age')

plt.tight_layout()
plt.show()
```



```
In [21]: fig, ax = plt.subplots(figsize=(4, 4))
         sns.kdeplot(data=df, x='salary', y='Age', ax=ax, color='#ffa061')
         ax.set_xlabel('Salary')
         ax.set_ylabel('Age')

         plt.tight_layout()
         plt.show()
```



Two Categorical Variables

In [22]:

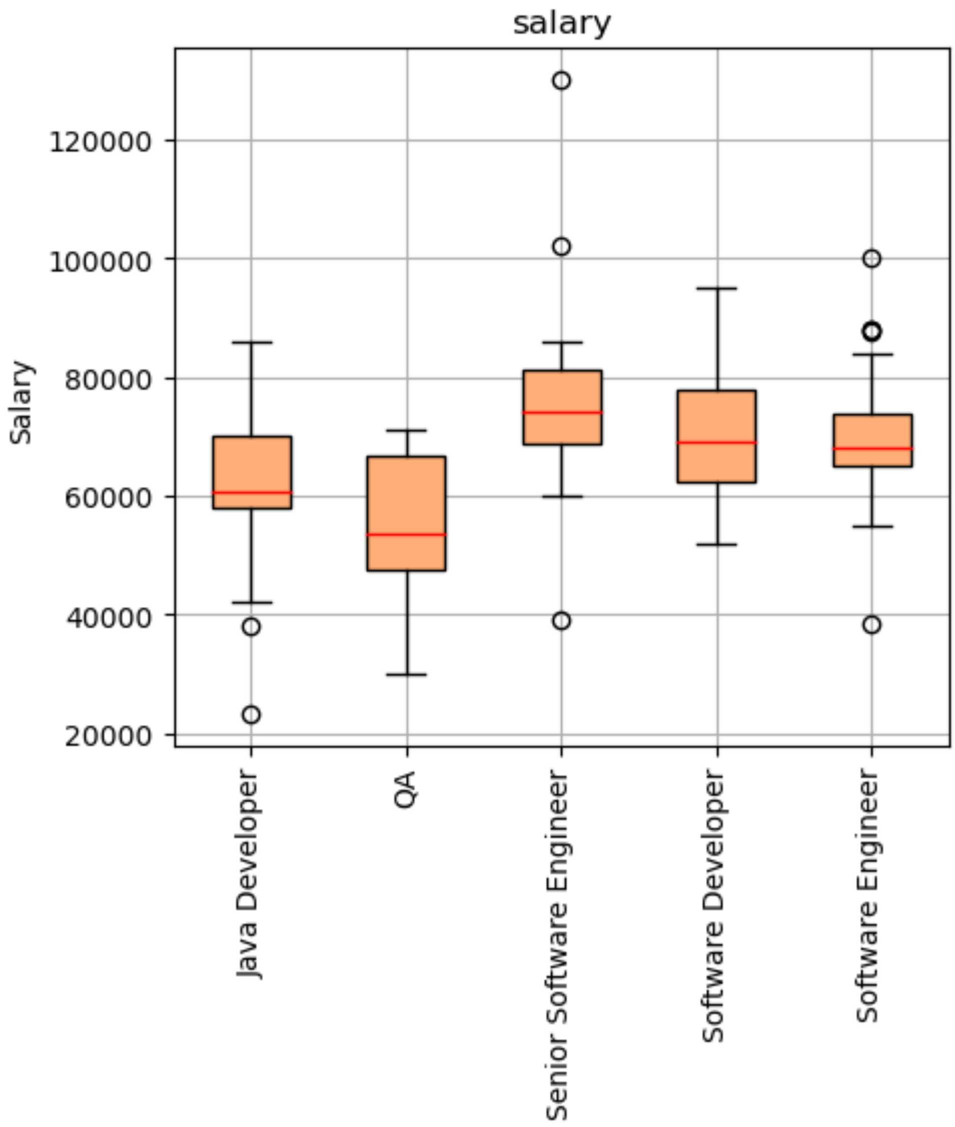
```
df0 = df[['lang', 'level']].copy()
pivot_table_position_lvl = df0.pivot_table(index='lang', columns='level', aggfunc=l

print(pivot_table_position_lvl)
```

```
level                                Junior  Middle  Senior  All
lang
Deutsch                                  13      37      66  116
Deutsch/Englisch                          0       0       1    1
English                                  18     144     377  539
French                                    1       1       0    2
Polish                                    0       2       0    2
Russian                                   1       6      18   25
Team - Russian; Cross-team - English;     0       0       1    1
All                                      33     190     463  686
```

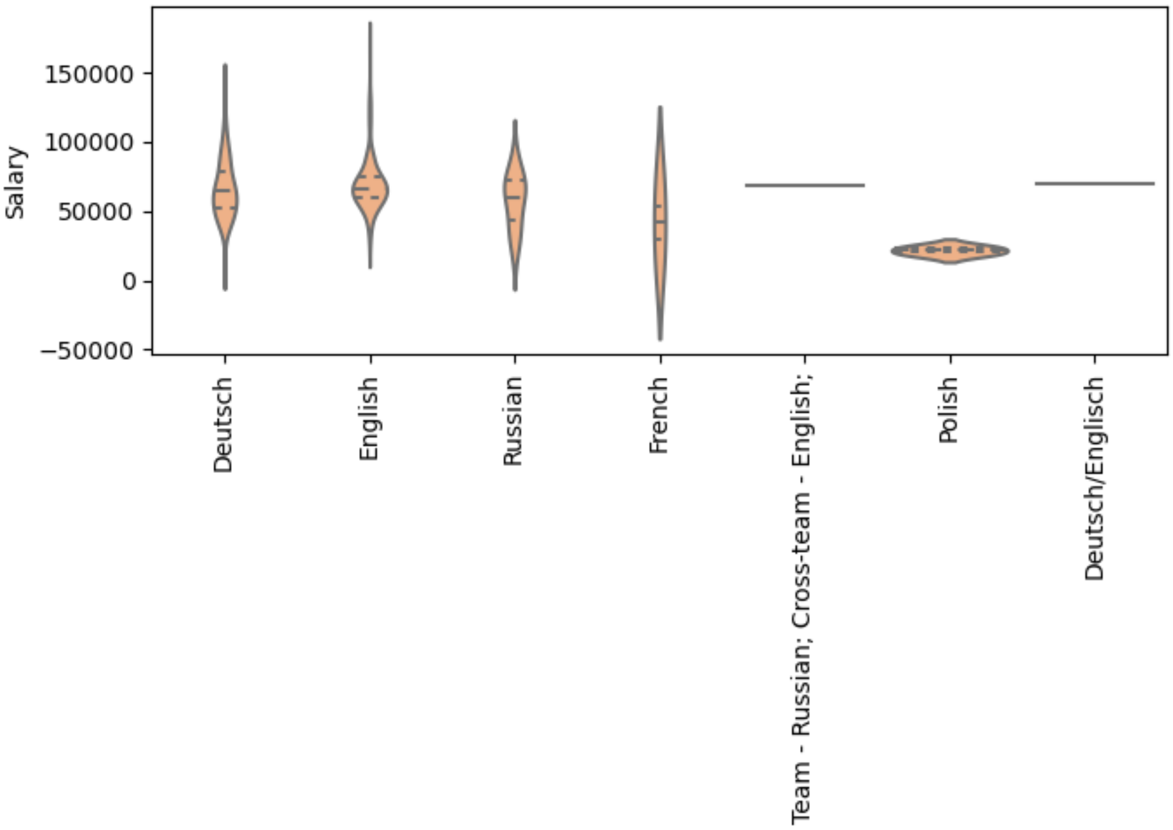
Categorical and Numeric Data

In [23]:

```
position_counts = df['Position'].value_counts()
top_5_positions = position_counts.head(5).index
df_top_5_positions = df[df['Position'].isin(top_5_positions)]

ax = df_top_5_positions.boxplot(by='Position', column='salary', figsize=(5, 6), col
ax.set_xlabel('')
ax.set_ylabel('Salary')
plt.suptitle('')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```



```
In [24]: fig, ax = plt.subplots(figsize=(7, 5))
         sns.violinplot(data=df, x='lang', y='salary',ax=ax, inner='quartile', color='#ffae7
         ax.set_xlabel('')
         ax.set_ylabel('Salary')
         plt.xticks(rotation=90)
         plt.tight_layout()
         plt.show()
```


Salary
150000
100000
50000
0
−50000
Deutsch
English
Russian
French
Team - Russian; Cross-team - English;
Polish
Deutsch/Englisch