Importing Needed Libraries:

In [1]:
```
!pip install country_converter
```

```
Collecting country_converter
  Downloading country_converter-1.0.0-py3-none-any.whl (44 kB)
Requirement already satisfied: pandas>=1.0 in c:\users\titan rafi\anaconda3_python\lib\site-pac
kages (from country_converter) (1.4.2)
Requirement already satisfied: numpy>=1.18.5 in c:\users\titan rafi\anaconda3_python\lib\site-p
ackages (from pandas>=1.0->country_converter) (1.21.5)
Requirement already satisfied: pytz>=2020.1 in c:\users\titan rafi\anaconda3_python\lib\site-pa
ckages (from pandas>=1.0->country_converter) (2021.3)
Requirement already satisfied: python-dateutil>=2.8.1 in c:\users\titan rafi\anaconda3_python\l
ib\site-packages (from pandas>=1.0->country_converter) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\users\titan rafi\anaconda3_python\lib\site-packag
es (from python-dateutil>=2.8.1->pandas>=1.0->country_converter) (1.16.0)
Installing collected packages: country-converter
Successfully installed country-converter-1.0.0
```

In [2]:
```
import pandas as pd
import numpy as np
import country_converter as coco
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import plotly.figure_factory as ff
import plotly.graph_objects as go
import warnings
warnings.filterwarnings('ignore')
pd.options.display.max_columns = None
```

In [3]:
```
df = pd.read_csv("C:/Users/Titan Rafi/Dropbox/PC/Desktop/ML Projects - Self/ds_salaries.csv"
df.head()
```

Out[3]:

	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence
0	2023	SE	FT	Principal Data Scientist	80000	EUR	85847	ES
1	2023	MI	CT	ML Engineer	30000	USD	30000	US
2	2023	MI	CT	ML Engineer	25500	USD	25500	US
3	2023	SE	FT	Data Scientist	175000	USD	175000	CA
4	2023	SE	FT	Data Scientist	120000	USD	120000	CA

In [5]:
```
df.shape
```

Out[5]: (3755, 11)

In [7]:
```
df.isnull().sum()
```

Out[7]:
```
work_year             0
experience_level      0
employment_type       0
job_title             0
salary                0
salary_currency       0
salary_in_usd         0
employee_residence    0
remote_ratio          0
company_location      0
company_size          0
dtype: int64
```

In [8]:
```
df.duplicated().sum()
```

Out[8]:
```
1171
```

In [9]:
```
# drop the duplicated rows
df.drop_duplicates(inplace=True)
```

In [10]:
```
df.shape
```

Out[10]:
```
(2584, 11)
```

In [11]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 2584 entries, 0 to 3754
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   work_year           2584 non-null   int64 
 1   experience_level    2584 non-null   object
 2   employment_type     2584 non-null   object
 3   job_title           2584 non-null   object
 4   salary              2584 non-null   int64 
 5   salary_currency     2584 non-null   object
 6   salary_in_usd       2584 non-null   int64 
 7   employee_residence  2584 non-null   object
 8   remote_ratio        2584 non-null   int64 
 9   company_location    2584 non-null   object
 10  company_size        2584 non-null   object
dtypes: int64(4), object(7)
memory usage: 242.2+ KB
```

In [12]:
```
df.columns
```

Out[12]:
```
Index(['work_year', 'experience_level', 'employment_type', 'job_title',
       'salary', 'salary_currency', 'salary_in_usd', 'employee_residence',
       'remote_ratio', 'company_location', 'company_size'],
      dtype='object')
```

In [13]:
```
# drop 2 columns:  'salary' and 'salary_currency'
df.drop(['salary'], axis=1, inplace=True)

# rename salary_in_usd to salary
df.rename(columns={'salary_in_usd': 'salary'}, inplace=True)
# Inflation rates
```

In [14]:
```
# Inflation rates
us_inflation_rates = {2019: 0.0181, 2020: 0.0123, 2021: 0.0470, 2022: 0.065}
global_inflation_rates = {2019: 0.0219, 2020: 0.0192, 2021: 0.0350, 2022: 0.088}

# Function to adjust salary
def adjust_salary(row):
    year = row['work_year']
    original_salary = row['salary']
    currency = row['salary_currency']

    if year == 2023:
        return original_salary

    adjusted_salary = original_salary
    for y in range(year, 2023):
        if currency == 'USD':
            inflation_rate = us_inflation_rates[y]
        else:
            inflation_rate = global_inflation_rates[y]

        adjusted_salary *= (1 + inflation_rate)

    return adjusted_salary

# Apply the function to the dataset
df['adjusted_salary'] = df.apply(adjust_salary, axis=1)
```

In [15]:
```
df.experience_level.value_counts()
```

Out[15]:
```
SE    1554
MI     664
EN     270
EX      96
Name: experience_level, dtype: int64
```

In [16]:
```
df['experience_level'] = df['experience_level'].replace({
    'SE': 'Senior',
    'EN': 'Junior',
    'EX': 'Executive',
    'MI': 'Mid-level',
})
```

In [17]:
```
converted_country = coco.convert(names=df['employee_residence'], to="name")
df['employee_residence'] = converted_country
```

In [18]:
```
converted_location = coco.convert(names=df['company_location'], to="name")
df['company_location'] = converted_country
```

In [19]:
```
df['remote_ratio'].value_counts()
```

Out[19]:
```
100    1211
0      1186
50      187
Name: remote_ratio, dtype: int64
```

In [21]:
```
df['job_type'] = df['remote_ratio'].replace({0: 'onsite', 50: 'hybrid', 100:'remote'})
```

In [22]:

```
#df.job_title.unique()
def assign_broader_category(job_title):
    data_scientist = ['Principal Data Scientist','Data Scientist','Applied Scientist','Applie
 'Data Science Manager','Director of Data Science', 'Lead Data Scientist',
 'Data Science Lead', 'Data Science Consultant','Head of Data Science',
 'Data Science Engineer', 'Data Science Tech Lead', 'Data Scientist Lead',
 'Product Data Scientist','Staff Data Scientist'
]
    data_engineer = [ 'Data Modeler', 'Data Strategist', 'Data Engineer', 'Data Architect',
  'Head of Data',  'Data Manager','Data Operations Engineer','Azure Data Engineer',
  'Big Data Engineer', 'Cloud Database Engineer','Marketing Data Engineer', 'Data Lead',
  'Data Infrastructure Engineer', 'Software Data Engineer','Data Specialist',
  'BI Data Engineer','BI Developer','Big Data Architect','Cloud Data Engineer',
  'Data Operations Analyst', 'Power BI Developer','Principal Data Architect',
  'Cloud Data Architect', 'Lead Data Engineer','Principal Data Engineer']
    machine_learning= ['ML Engineer','Machine Learning Engineer',
 'Applied Machine Learning Engineer','Machine Learning Researcher','Machine Learning Scientis
 'MLOps Engineer', 'AI Scientist','AI Developer','Applied Machine Learning Scientist',
 'AI Programmer','Deep Learning Researcher',  'Machine Learning Infrastructure Engineer',
 'Deep Learning Engineer', 'Machine Learning Software Engineer','Machine Learning Research Er
 'NLP Engineer','Machine Learning Developer', 'Principal Machine Learning Engineer',
 'Machine Learning Manager', 'Lead Machine Learning Engineer','Head of Machine Learning',
]
    data_analyst = ['Data Analyst','Analytics Engineer','Data Quality Analyst','Compliance Da
 'Data Analytics Manager','Business Data Analyst','Staff Data Analyst',
 'Lead Data Analyst','Financial Data Analyst','BI Analyst','Product Data Analyst',
 'Data Analytics Lead', 'Data Analytics Specialist','BI Data Analyst','Insight Analyst',
 'Data Analytics Engineer','Data Analytics Consultant', 'Marketing Data Analyst',
 'Principal Data Analyst', 'Finance Data Analyst']
    if job_title in data_engineer:
        return "Data Engineer"
    elif job_title in data_scientist:
        return "Data Scientist"
    elif job_title in machine_learning:
        return "ML Engineer"
    elif job_title in data_analyst:
        return "Data Analyst"
    else:
        return "Other"


df['job_category'] = df['job_title'].apply(assign_broader_category)
```

In [23]:

```
df['employment_type'] = df['employment_type'].replace({
    'FL': 'Freelancer',
    'CT': 'Contractor',
    'FT' : 'Full-time',
    'PT' : 'Part-time'
})
df['company_size'] = df['company_size'].replace({
    'S': 'Small',
    'M': 'Medium',
    'L' : 'Large',
})
```

In [24]:
```
1 df.head()
```

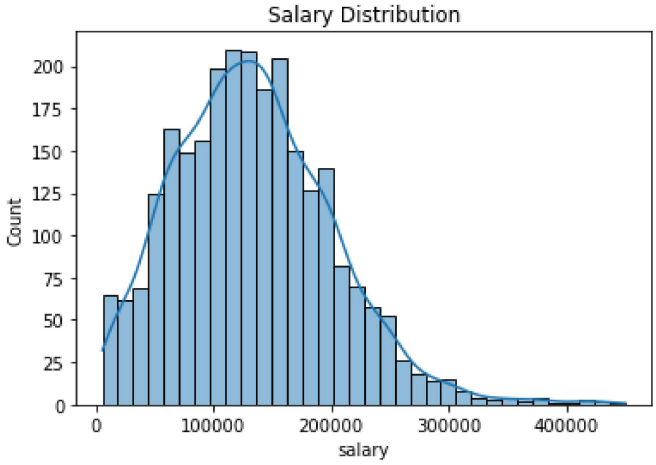
Out[24]:

	work_year	experience_level	employment_type	job_title	salary_currency	salary	employee_residence	remote_ratio
0	2023	Senior	Full-time	Principal Data Scientist	EUR	85847	Spain	100
1	2023	Mid-level	Contractor	ML Engineer	USD	30000	United States	100
2	2023	Mid-level	Contractor	ML Engineer	USD	25500	United States	100
3	2023	Senior	Full-time	Data Scientist	USD	175000	Canada	100
4	2023	Senior	Full-time	Data Scientist	USD	120000	Canada	100

Salary Distribution:

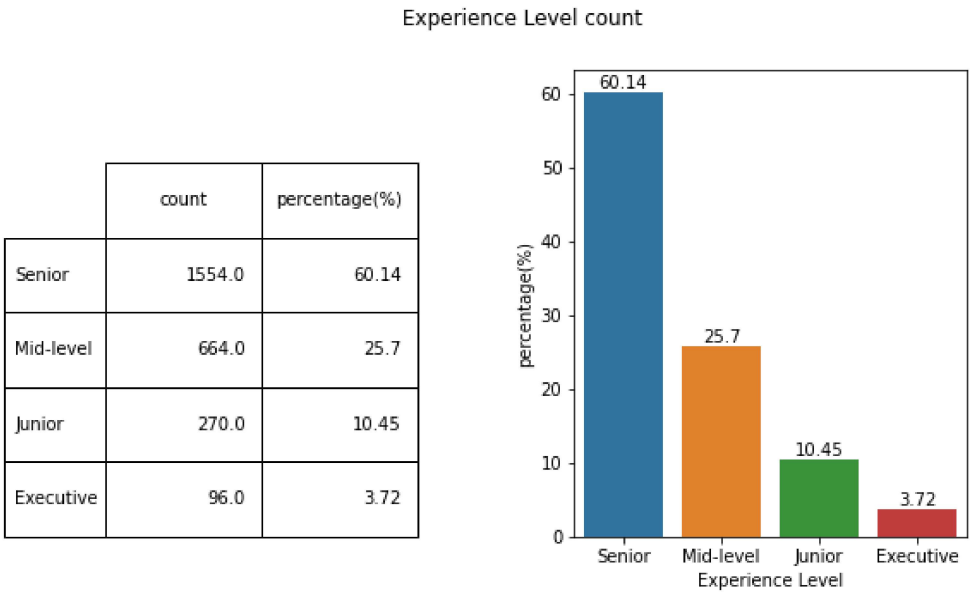
In [25]:
```
1 ax = sns.histplot(df['salary'], kde = True)
2 ax.set_title('Salary Distribution')
```

Out[25]:
```
Text(0.5, 1.0, 'Salary Distribution')
```

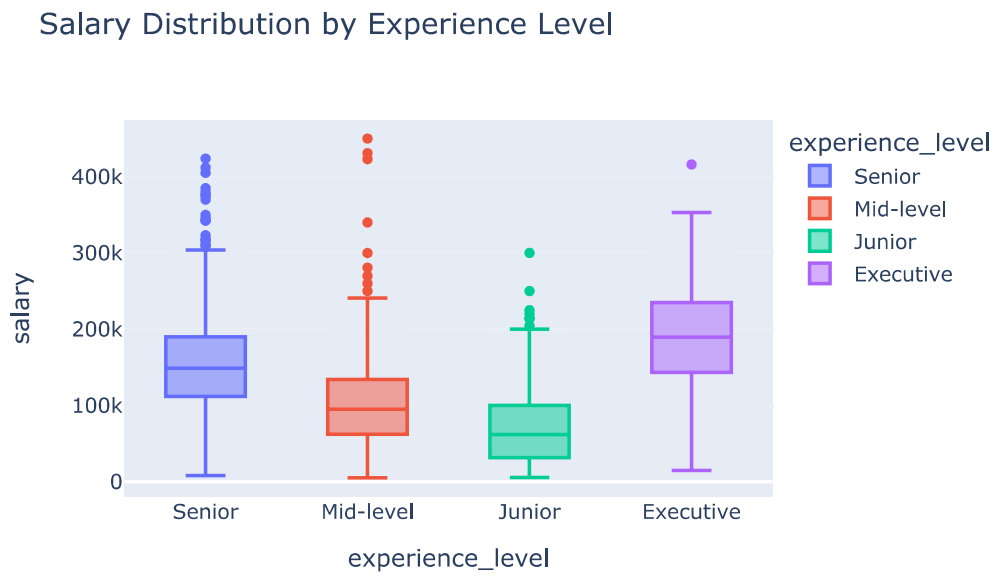


Experience Level:

In [26]:
```
ex_level = df['experience_level'].value_counts().to_frame()
ex_level.columns = ['count']
ex_level['percentage(%)'] =round(100 * df['experience_level'].value_counts(normalize= True),
fig = plt.figure(figsize=(9,5))
ax1 = plt.subplot(121)
ax1.axis('off')
bbox =[0,0,0.8,0.8]
ax1.table(cellText = ex_level.values, rowLabels=ex_level.index, colLabels=ex_level.columns,bl
ax2 = fig.add_subplot(122)
ax2 = sns.barplot(data = ex_level, x=ex_level.index,y=ex_level['percentage(%)'])
ax2.bar_label(ax2.containers[0])
# plt.xticks(rotation= 30)
plt.xlabel('Experience Level')
plt.suptitle('Experience Level count')
plt.show()
```



In [27]:
```
px.box(df, x= 'experience_level', y = 'salary',
       color= 'experience_level',title= 'Salary Distribution by Experience Level',
       width=600, height=400)
```

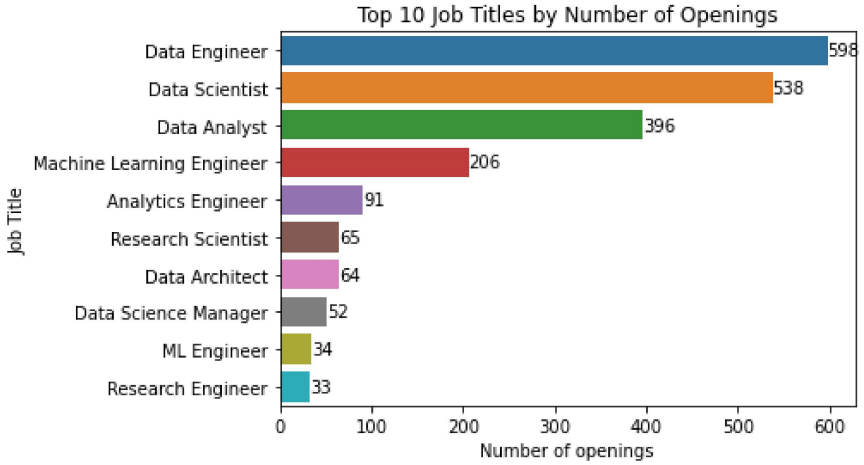


Which Job Roles are Most Popular and Most paid in Data Science:

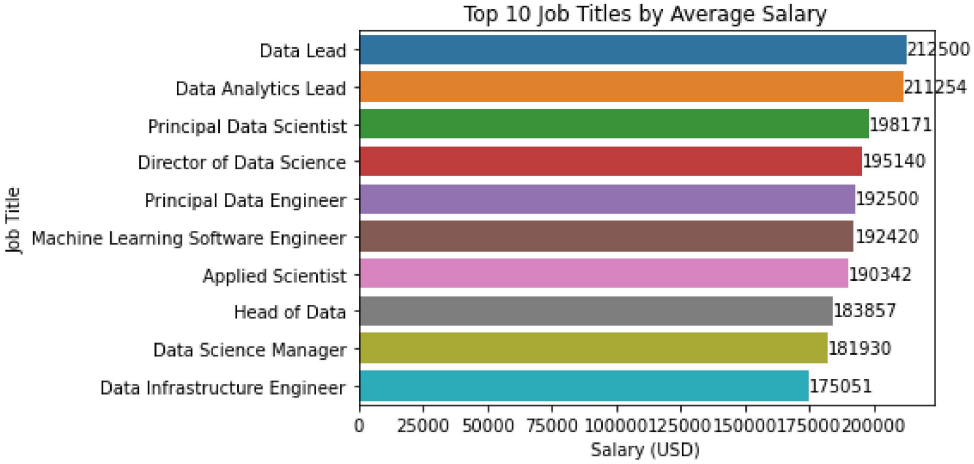
In [28]:
```
print('There are {} job titles in this dataset.'.format(len(df['job_title'].value_counts())))
```

```
There are 93 job titles in this dataset.
```

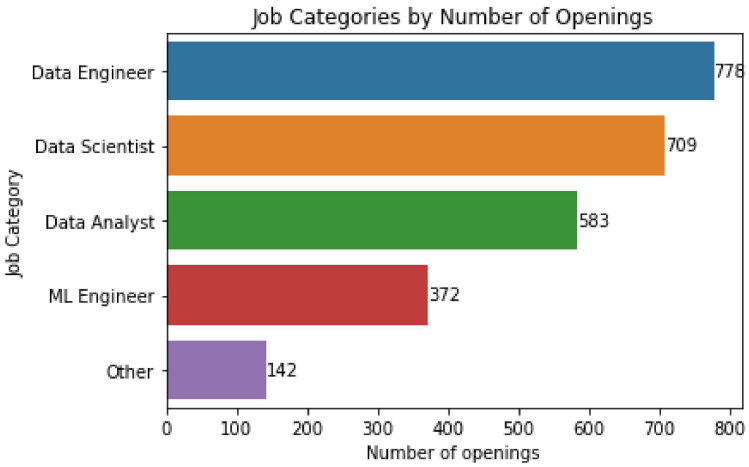
In [29]:
```
top10_count = df['job_title'].value_counts()[:10].to_frame()
top10_count.columns = ['count']

plt.figure(figsize=(6,4))

ax= sns.barplot(data = top10_count, y =  top10_count.index, x= top10_count['count'])
plt.ylabel('Job Title')
plt.xlabel('Number of openings')
plt.title('Top 10 Job Titles by Number of Openings')
ax.bar_label(ax.containers[0], fmt = '%d')
plt.show()
```



In [30]:
```
top_avg_salary = df.groupby('job_title')['salary'].mean().round(2).sort_values(ascending=Fals
role_count = df['job_title'].value_counts()
top10_avg_salary = top_avg_salary[role_count[role_count > 1].index].sort_values(ascending = 

plt.figure(figsize=(6,4))

ax = sns.barplot(data = top10_avg_salary, y=top10_avg_salary.index,x=top10_avg_salary['salary
plt.ylabel('Job Title')
plt.xlabel('Salary (USD)')
plt.title('Top 10 Job Titles by Average Salary')
ax.bar_label(ax.containers[0], fmt = '%d')
plt.show()
```

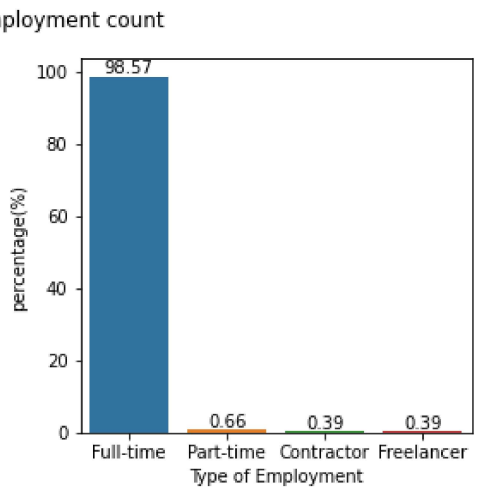


In [31]:
```
temp = df['job_category'].value_counts().to_frame()
temp.columns = ['count']

plt.figure(figsize=(6,4))

ax= sns.barplot(data = temp, y =  temp.index, x= temp['count'])
plt.ylabel('Job Category')
plt.xlabel('Number of openings')
plt.title('Job Categories by Number of Openings')
ax.bar_label(ax.containers[0], fmt = '%d')
plt.show()
```



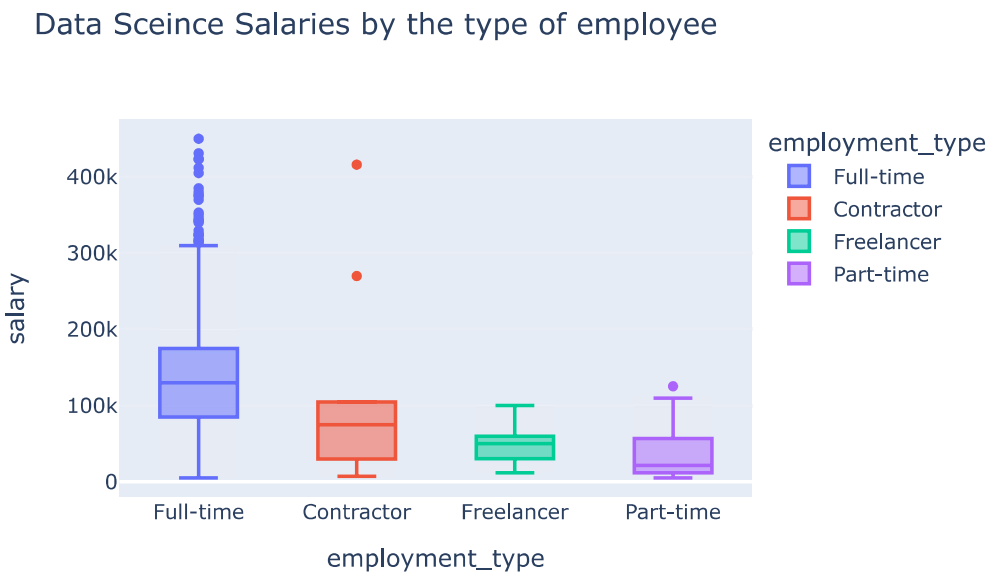
Which type of Employment is Lucrative?

In [32]:
```
employment = df['employment_type'].value_counts().to_frame()
employment.columns = ['count']
employment['percentage(%)'] =round(100 * df['employment_type'].value_counts(normalize= True),

fig = plt.figure(figsize=(9,4))
ax1 = plt.subplot(121)
ax1.axis('off')
bbox =[0,0,0.8,0.8]
ax1.table(cellText = employment.values, rowLabels=employment.index, colLabels=employment.colu
ax2 = fig.add_subplot(122)
ax2 = sns.barplot(data = employment, x=employment.index,y=employment['percentage(%)'])
ax2.bar_label(ax2.containers[0])
# plt.xticks(rotation= 30)
plt.xlabel('Type of Employment')
plt.suptitle('Type of Employment count')
plt.show()
```



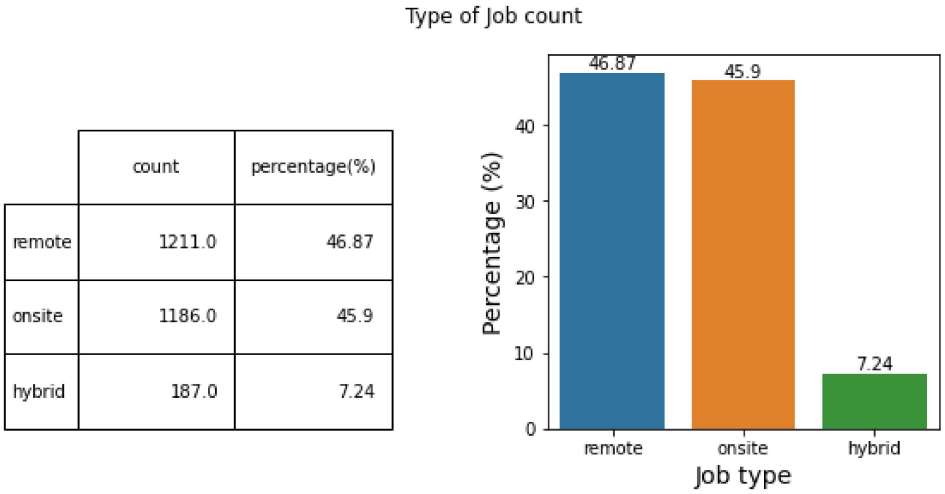
	count	percentage(%)
Full-time	2547.0	98.57
Part-time	17.0	0.66
Contractor	10.0	0.39
Freelancer	10.0	0.39

In [33]:
```
px.box(df,x='employment_type',y='salary',color='employment_type',
      title= 'Data Sceince Salaries by the type of employee',
      width=600, height=400)
```

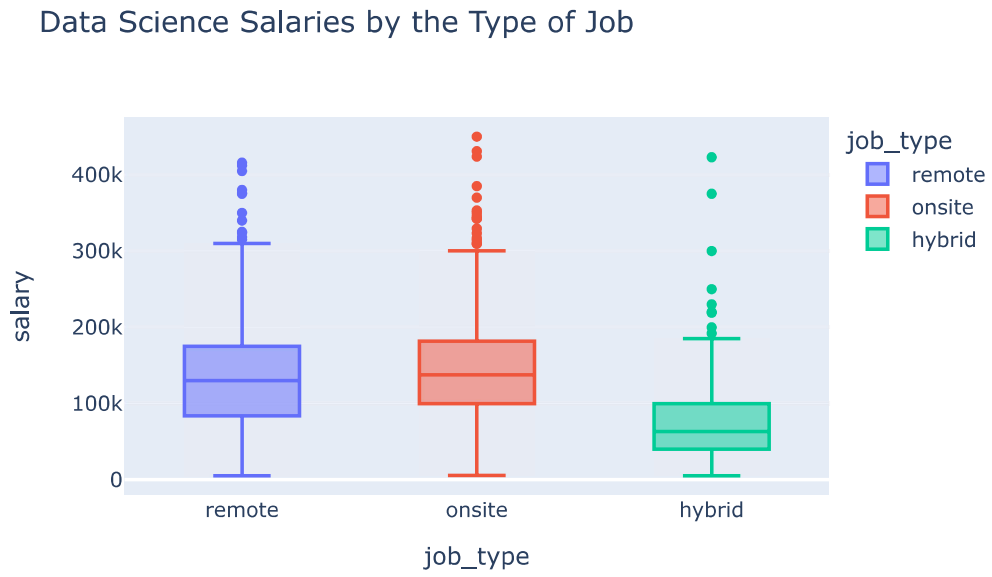


Which type of job has more Average Salary?

In [34]:
```
jobType = df['job_type'].value_counts().to_frame()
jobType.columns = ['count']
jobType['percentage(%)'] =round(100 * df['job_type'].value_counts(normalize= True),2)

fig = plt.figure(figsize=(9,4))
ax1 = plt.subplot(121)
ax1.axis('off')
bbox =[0,0,0.8,0.8]
ax1.table(cellText = jobType.values, rowLabels=jobType.index, colLabels=jobType.columns,bbox
ax2 = fig.add_subplot(122)
ax2 = sns.barplot(data = jobType, x=jobType.index,y=jobType['percentage(%)'])
ax2.bar_label(ax2.containers[0])
# plt.xticks(rotation= 30)
plt.xlabel('Job type', fontdict={'fontsize':14})
plt.ylabel('Percentage (%)', fontdict={'fontsize':14})

plt.suptitle('Type of Job count')
plt.show()
```

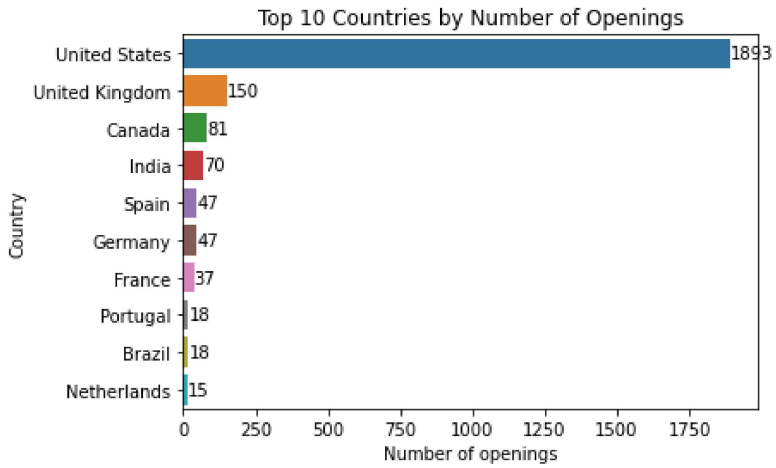


In [35]:
```
px.box(df,x='job_type',y='salary',color='job_type',
           title='Data Science Salaries by the Type of Job',width=600, height=400)
```

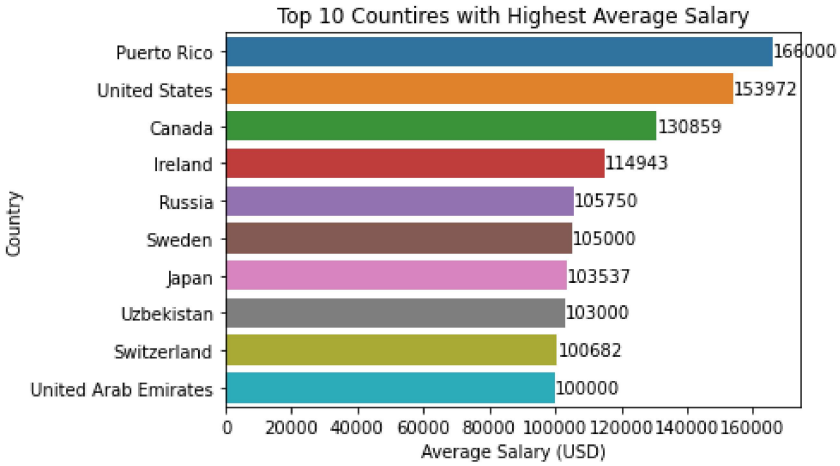


Company Location:

In [36]:
```
top10_count = df['company_location'].value_counts()[:10].to_frame()
top10_count.columns = ['count']

plt.figure(figsize=(6,4))

ax= sns.barplot(data = top10_count, y =  top10_count.index, x= top10_count['count'])
plt.ylabel('Country')
plt.xlabel('Number of openings')
plt.title('Top 10 Countries by Number of Openings')
ax.bar_label(ax.containers[0], fmt = '%d')
plt.show()
```

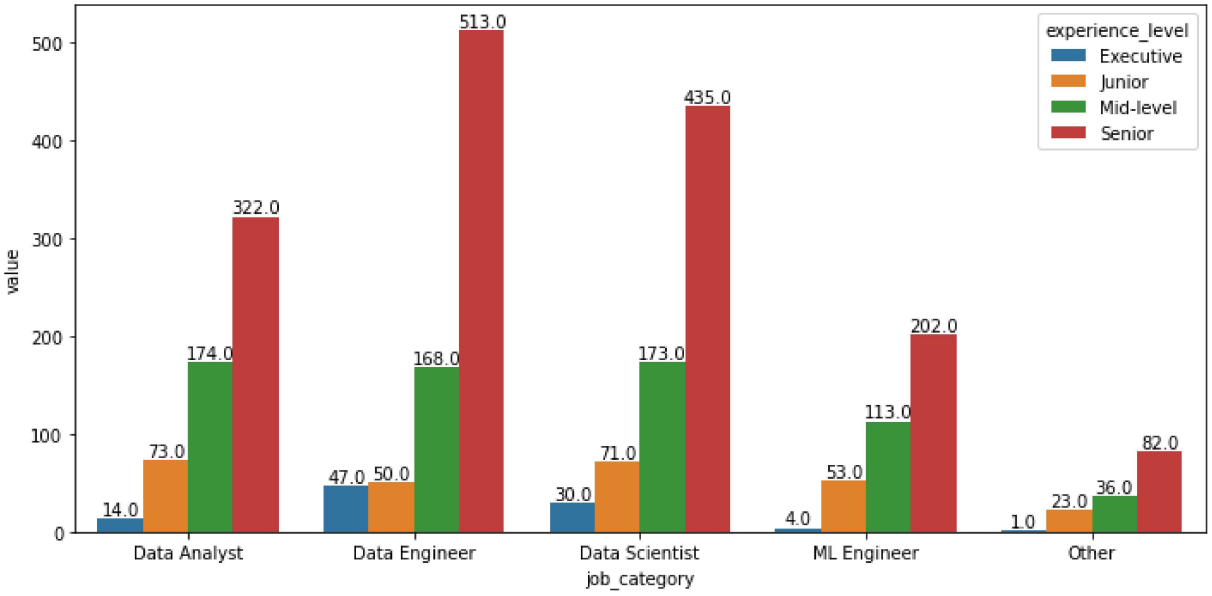


In [37]:
```
top_avg_salary = df.groupby('company_location')['salary'].mean().round(2).sort_values(ascend
role_count = df['company_location'].value_counts()
top10_avg_salary = top_avg_salary[role_count[role_count > 1].index].sort_values(ascending = 

plt.figure(figsize=(6,4))

ax = sns.barplot(data = top10_avg_salary, y=top10_avg_salary.index,x=top10_avg_salary['salary
plt.ylabel('Country')
plt.xlabel('Average Salary (USD)')
plt.title('Top 10 Countires with Highest Average Salary')
ax.bar_label(ax.containers[0], fmt = '%d')
plt.show()
```



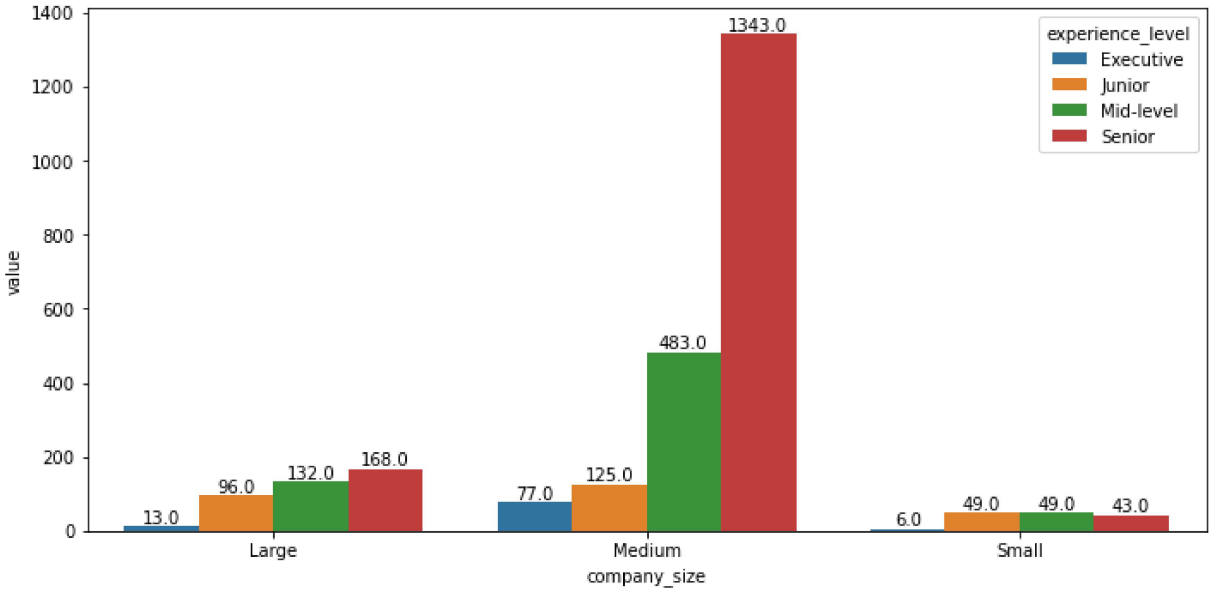
Experience level and Job Category:

In [38]:
```
exlevel_type = df.groupby(['experience_level','job_category']).size().reset_index()
exlevel_type.columns = ['experience_level','job_category', 'value']
plt.figure(figsize=(10,5))
ax = sns.barplot(data = exlevel_type, x=exlevel_type['job_category'], y=exlevel_type['value'
            hue=exlevel_type['experience_level'])
for bars in ax.containers:
        ax.bar_label(bars, fmt='%.1f')
plt.tight_layout()
plt.show()
```



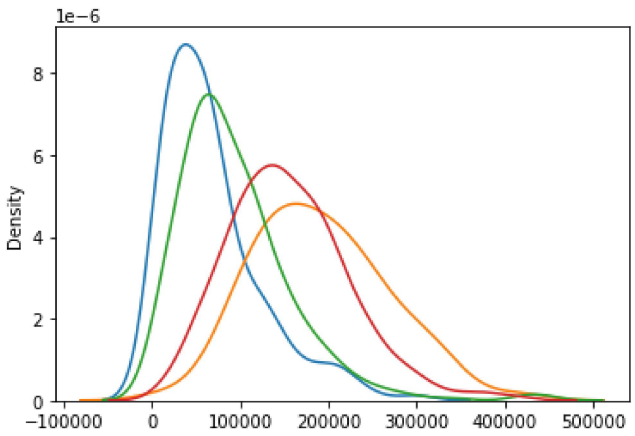
Experience level and Company Size:

In [39]:
```
exlevel_type = df.groupby(['experience_level','company_size']).size().reset_index()
exlevel_type.columns = ['experience_level','company_size', 'value']
plt.figure(figsize=(10,5))
ax = sns.barplot(data = exlevel_type, x=exlevel_type['company_size'], y=exlevel_type['value'
            hue=exlevel_type['experience_level'])
for bars in ax.containers:
        ax.bar_label(bars, fmt='%.1f')
plt.tight_layout()
plt.show()
```



In [40]:
```
exlevel_salary = df.groupby(['experience_level','salary']).size()
entry_salary = exlevel_salary['Junior'].reset_index()
executive_salary = exlevel_salary['Executive'].reset_index()
mid_salary = exlevel_salary['Mid-level'].reset_index()
senior_salary = exlevel_salary['Senior'].reset_index()
sns.distplot(entry_salary[['salary']], hist=False)
sns.distplot(executive_salary[['salary']], hist=False)
sns.distplot(mid_salary[['salary']], hist=False)
sns.distplot(senior_salary[['salary']], hist=False)
```

Out[40]: <AxesSubplot:ylabel='Density'>



In [ ]:
```
1
```