Project Title	**Data Science Job Salaries**
Tools	ML, Python, SQL, Excel
Domain	Finance Analyst,
Project Difficulties level	intermediate

Dataset : Dataset is available in the given link. You can download it at your convenience.

Click here to download data set

About Dataset

Content

Column	Description
work_year	The year the salary was paid.
experience_level	The experience level in the job during the year with the following possible values: EN Entry-level / Junior MI Mid-level / Intermediate SE Senior-level / Expert EX Executive-level / Director

employment_type	The type of employement for the role: PT Part-time FT Full-time CT Contract FL Freelance
job_title	The role worked in during the year.
salary	The total gross salary amount paid.
salary_currency	The currency of the salary paid as an ISO 4217 currency code.
salary_in_usd	The salary in USD (FX rate divided by avg. USD rate for the respective year via fxdata.foorilla.com).
employee_residence	Employee's primary country of residence in during the work year as an ISO 3166 country code.
remote_ratio	The overall amount of work done remotely, possible values are as follows: 0 No remote work (less than 20%) 50 Partially remote 100 Fully remote (more than 80%)
company_location	The country of the employer's main office or contracting branch as an ISO 3166 country code.
company_size	The average number of people that worked for the company during the year: S less than 50 employees (small) M 50 to 250 employees (medium) L more than 250 employees (large)

Acknowledgements

I'd like to thank ai-jobs.net Salaries for aggregating this data!

Example: You can get the basic idea how you can create a project from here

Step 1: Problem Definition

- **Objective**: Analyze and model data science job salaries to uncover trends, identify salary drivers, and predict salaries based on job-related factors.
- **Data Columns**:
  - `work_year`: Year of the job role.
  - `experience_level`: Entry-level, mid-level, senior, etc.
  - `employment_type`: Full-time, contract, etc.
  - `job_title`: Role title (e.g., Data Scientist, Analyst).
  - `salary`: Reported salary.
  - `salary_currency`: Currency of salary.
  - `salary_in_usd`: Converted salary in USD.
  - `employee_residence`: Country of the employee.
  - `remote_ratio`: 0 (on-site), 50 (hybrid), 100 (remote).
  - `company_location`: Company's country.
  - `company_size`: Small, medium, or large.

Step 2: Load and Understand the Dataset

First, load the data and understand its structure.

Code:

python

code

```python
import pandas as pd
```

```
# Load dataset
file_path = "data_science_job_salaries.csv"  # Replace with
your file path
data = pd.read_csv(file_path)

# Basic information
print(data.info())
print(data.head())
```

Step 3: Data Cleaning

3.1 Handle Missing Values

Identify and handle missing data appropriately.

Code:

python
code

```
# Check for missing values
print(data.isnull().sum())

# Fill missing values (example strategies)
data['salary_in_usd'].fillna(data['salary_in_usd'].median(),
inplace=True)  # Replace with median
data['company_size'].fillna('Unknown', inplace=True)  # Replace
```

```
missing categories with 'Unknown'

# Drop rows with critical missing data
data.dropna(subset=['job_title', 'experience_level'],
inplace=True)

# Verify no missing values remain
print(data.isnull().sum())
```

3.2 Standardize Categorical Columns

Ensure consistent formatting for categorical data.

Code:

python

code

```
# Standardize text case for categorical columns
data['job_title'] = data['job_title'].str.lower()
data['company_size'] = data['company_size'].str.capitalize()

# Verify unique values
print(data['job_title'].unique())
print(data['company_size'].unique())
```

Step 4: Feature Engineering

1. **Encode Categorical Variables**: Convert `experience_level`, `employment_type`, etc., to numeric.
2. **Add Derived Features**:
   - Calculate salary differences between locations (`employee_residence` vs `company_location`).
   - Group salaries by `company_size` and `remote_ratio`.

Code:

python

code

```
# Encode categorical variables
data['experience_level'] = data['experience_level'].map({'EN':
0, 'MI': 1, 'SE': 2, 'EX': 3})
data['employment_type'] = data['employment_type'].map({'PT': 0,
'FT': 1, 'CT': 2, 'FL': 3})

# Add a salary ratio feature
data['salary_ratio'] = data['salary'] / data['salary_in_usd']

# Group salaries by company size
grouped_salary =
data.groupby('company_size')['salary_in_usd'].mean()
print(grouped_salary)
```

Step 5: Exploratory Data Analysis (EDA)

5.1 Summary Statistics

Analyze salary distribution and other numeric columns.

Code:

python

code

```
# Summary statistics
print(data.describe())

# Analyze salary distribution
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 6))
sns.histplot(data['salary_in_usd'], bins=30, kde=True,
color='blue')
plt.title('Salary Distribution (USD)')
plt.xlabel('Salary in USD')
plt.ylabel('Frequency')
plt.show()
```

5.2 Correlation Analysis

Understand relationships between numeric features.

Code:

python

code

```
# Correlation heatmap
plt.figure(figsize=(10, 8))
sns.heatmap(data.corr(), annot=True, cmap='coolwarm')
plt.title('Feature Correlation')
plt.show()
```

5.3 Category-Based Analysis

Examine salaries by `job_title`, `experience_level`, and `remote_ratio`.

Code:

python

code

```
# Boxplot for salaries by experience level
plt.figure(figsize=(12, 6))
sns.boxplot(x='experience_level', y='salary_in_usd', data=data)
plt.title('Salary by Experience Level')
plt.xlabel('Experience Level')
plt.ylabel('Salary in USD')
```

```
plt.show()


# Remote ratio vs salary
plt.figure(figsize=(12, 6))
sns.barplot(x='remote_ratio', y='salary_in_usd', data=data)
plt.title('Salary by Remote Ratio')
plt.xlabel('Remote Ratio')
plt.ylabel('Salary in USD')
plt.show()
```

Step 6: Financial Modeling

Predict Salary Using Linear Regression

1. Train a model to predict `salary_in_usd` based on features like `experience_level`, `job_title`, etc.
2. Split data into training and testing sets.

Code:

python

code

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_absolute_error,
mean_squared_error
```

```
# Select features and target
features = ['experience_level', 'employment_type',
'remote_ratio', 'company_size']
target = 'salary_in_usd'

# Encode categorical columns for model
data = pd.get_dummies(data, columns=features, drop_first=True)

X = data.drop(columns=['salary_in_usd', 'work_year',
'employee_residence'])
y = data['salary_in_usd']

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y,
test_size=0.2, random_state=42)

# Train model
model = LinearRegression()
model.fit(X_train, y_train)

# Predict and evaluate
y_pred = model.predict(X_test)
mse = mean_squared_error(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
```

```
print(f"Mean Squared Error: {mse}")
print(f"Mean Absolute Error: {mae}")
```

Step 7: Data Visualization for Insights

Interactive Dashboards with Streamlit

Build an interactive dashboard to visualize salary trends.

Code:

python

code

```
import streamlit as st

st.title('Data Science Job Salaries')

# Upload summary statistics
st.write(data.describe())

# Visualization
st.line_chart(data['salary_in_usd'])

# Filter by job title
job_filter = st.selectbox('Select Job Title',
data['job_title'].unique())
filtered_data = data[data['job_title'] == job_filter]
```

```
st.bar_chart(filtered_data['salary_in_usd'])
```

Sample Code and output

```
# import requried libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:

```
# get the dataset
df =
pd.read_csv('../input/data-science-job-salaries/ds_salaries.csv
')
df.head()
```

Out[2]:

	Unnamed: 0	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remote_ratio	company_location	company_size

0	0	2020	MI	FT	Data Scientist	70000	EUR	79833	DE	0	DE	L
1	1	2020	SE	FT	Machine Learning Scientist	260000	USD	260000	JP	0	JP	S
2	2	2020	SE	FT	Big Data Engineer	85000	GBP	109024	GB	50	GB	M

					r							
3	3	2020	MI	FT	Product Data Analyst	20000	USD	20000	HN	0	HN	S
4	4	2020	SE	FT	Machine Learning Engineer	150000	USD	150000	US	50	US	L

1- Data Preprocessing

In [3]:

```
# remove the 'Unnamed: 0 ' column
df.drop('Unnamed: 0', axis=1, inplace=True)
```

In [4]:

```
# shape
df.shape
```

Out[4]:

```
(607, 11)
```

The dataset is comprised of 607 instances and 11 variables

In [5]:

```
# columns and data types
df.dtypes
```

Out[5]:

```
work_year                  int64
experience_level          object
employment_type           object
```

```
job_title              object
salary                  int64
salary_currency        object
salary_in_usd           int64
employee_residence     object
remote_ratio            int64
company_location       object
company_size           object

dtype: object
```

Drop duplicates

In [6]:

```
# detect duplications
df.duplicated().sum()
```

Out[6]:

42

In [7]:

```
# drop duplications
df.drop_duplicates(inplace=True)
```

Change abbreviations to complete values

The categorical variables contains some abbreviated values; to better understand, let's change them to their original names.

In [8]:

```
# change country names from ISO2 to original names
# There are two features containing country names,
"company_location" and "employee_residence"
!pip install -q country_converter
import country_converter
cc = country_converter.CountryConverter()
df['company_location'] = cc.convert(df['company_location'],
to='name_short')
df['employee_residence'] = cc.convert(df['employee_residence'],
to='name_short')
```

```
WARNING: Running pip as the 'root' user can result in broken
permissions and conflicting behaviour with the system package
manager. It is recommended to use a virtual environment
instead: https://pip.pypa.io/warnings/venv
```

In [9]:

```
# experience level
df['experience_level'].value_counts()
```

```
Out[9]:
SE    243
MI    208
EN     88
EX     26

Name: experience_level, dtype: int64
```

```
In [10]:
df['experience_level'] = df['experience_level'].map({
    'SE': 'Senior',
    'MI': 'Mid',
    'EN': 'Entry',
    'EX': 'Executive'
})
```

```
In [11]:
# employment type
df['employment_type'].value_counts()
```

```
Out[11]:
FT    546
```

```
PT      10
CT       5
FL       4

Name: employment_type, dtype: int64
```

In [12]:

```python
df['employment_type'] = df['employment_type'].map({
    'FT': 'Full-time',
    'PT': 'Part-time',
    'CT': 'Contract',
    'FL': 'Freelance'
})
```

In [13]:

```python
# company size
df['company_size'].value_counts()
```

Out[13]:

```
M    290
L    193
S     82

Name: company_size, dtype: int64
```

In [14]:

```
df['company_size'] = df['company_size'].map({
    'S': 'Small',
    'M': 'Medium',
    'L': 'Large'
})
```

In [15]:

```
# drop salary and salary_currency features (salary_in_usd is enough to keep on)
df.drop(['salary', 'salary_currency'], axis=1, inplace=True)

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

In [16]:

```
df['work_year'].value_counts()
```

Out[16]:

```
2022    278
```

```
2021    215
2020     72

Name: work_year, dtype: int64
```

Let's look at *remote_ratio* variable

In [17]:

```python
df['remote_ratio'].value_counts()
```

Out[17]:

```
100    346
0      121
50      98

Name: remote_ratio, dtype: int64
```

remote_ratio contains three categorical values: 100 means *fully-remote*, 0 means *fully-onsite* and 50 stands for *hybrid*.

In [18]:

```python
# renmame remote_ratio to job_type
df.rename(columns={'remote_ratio': 'job_type'}, inplace=True)


# change 100 to remote, 0 to onsite, 50 to hybrid
df['job_type'] = df['job_type'].map({
```

```
    100: 'remote',
    0: 'onsite',
    50: 'hybrid',
})
```

In [19]:

```
df['job_type'].value_counts()
```

Out[19]:

```
remote    346
onsite    121
hybrid     98

Name: job_type, dtype: int64
```

In [20]:

```
df.columns
```

Out[20]:

```
Index(['work_year', 'experience_level', 'employment_type',
'job_title',
       'salary', 'employee_residence', 'job_type',
'company_location',
```

```
       'company_size'],
      dtype='object')
```

In [21]:

```
df.head()
```

Out[21]:

	work_year	experience_level	employment_type	job_title	salary	employee_residence	job_type	company_location	company_size
0	2020	Mid	Full-time	Data Scientist	79833	Germany	onsite	Germany	Large
1	2020	Senior	Full-time	Machine Learning Scientist	260000	Japan	onsite	Japan	Small

2	2020	Senior	Full-time	Big Data Engineer	109024	United Kingdom	hybrid	United Kingdom	Medium
3	2020	Mid	Full-time	Product Data Analyst	20000	Honduras	onsite	Honduras	Small
4	2020	Senior	Full-time	Machine Learning Engineer	150000	United States	hybrid	United States	Large

---

2- Analysis

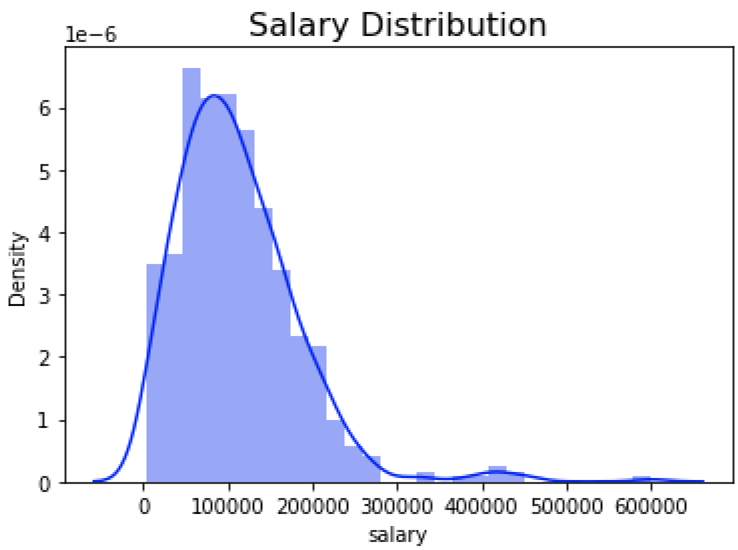
In [22]:

```
# Salary distribution
sns.set_palette('winter')
ax = sns.distplot(df['salary'])
ax.set_title('Salary Distribution', fontdict={'fontsize': 16})
```

```
/opt/conda/lib/python3.7/site-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt your code to use either `displot` (a figure-level function with similar flexibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```

Out[22]:

```
Text(0.5, 1.0, 'Salary Distribution')
```



Few people earn over $300,000

Salary VS experience level

In [23]:

```
# mean salary of employees with different experience levels
mean_s_exp_lv =
df.groupby('experience_level')['salary'].mean().sort_values()
mean_s_exp_lv
```

Out[23]:

```
experience_level
Entry          61643.318182
Mid            87792.995192
Senior        138374.880658
```

```
Executive    199392.038462

Name: salary, dtype: float64
```

In [24]:

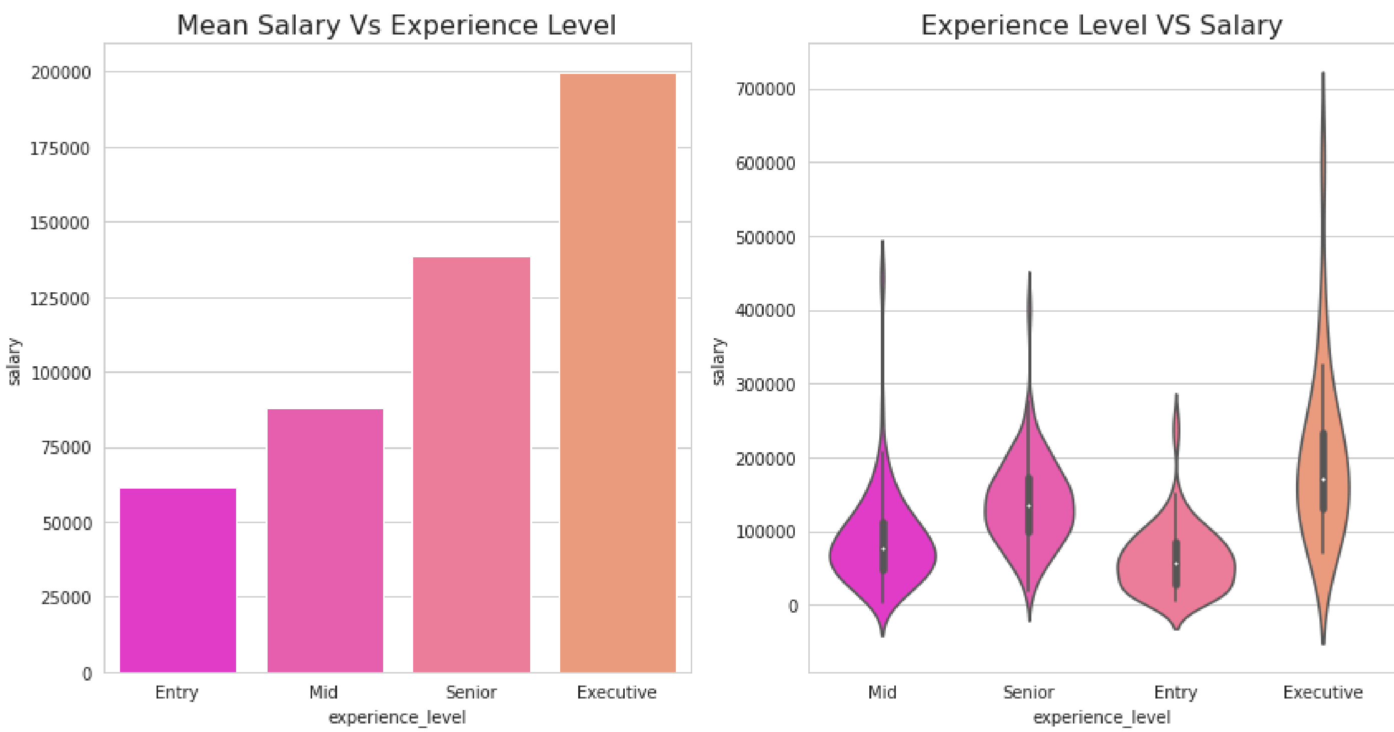
```python
sns.set_style('whitegrid')
```

In [25]:

```python
plt.figure(figsize=(14, 7))
sns.set_palette('spring')

plt.subplot(1, 2, 1)
ax = sns.barplot(x=mean_s_exp_lv.index, y=mean_s_exp_lv)
ax.set_title('Mean Salary Vs Experience Level',
fontdict={'fontsize': 16})

plt.subplot(1, 2, 2)
ax = sns.violinplot(data=df, x='experience_level', y='salary')
ax.set_title('Experience Level VS Salary',
fontdict={'fontsize': 16})
```

Out[25]:

```
Text(0.5, 1.0, 'Experience Level VS Salary')
```



Experience Level VS Salary:

We see that data scientists with the experience level of **Executive** have the highest mean salary, about **$200,000** annualy, and those with **Entry** level have the lowest mean salary, about **$60,000**.

Salary VS Employment Type

In [26]:

```
# mean salary of employees with different employment types
mean_s_emp_type =
df.groupby('employment_type')['salary'].mean().sort_values()
mean_s_emp_type
```

Out[26]:

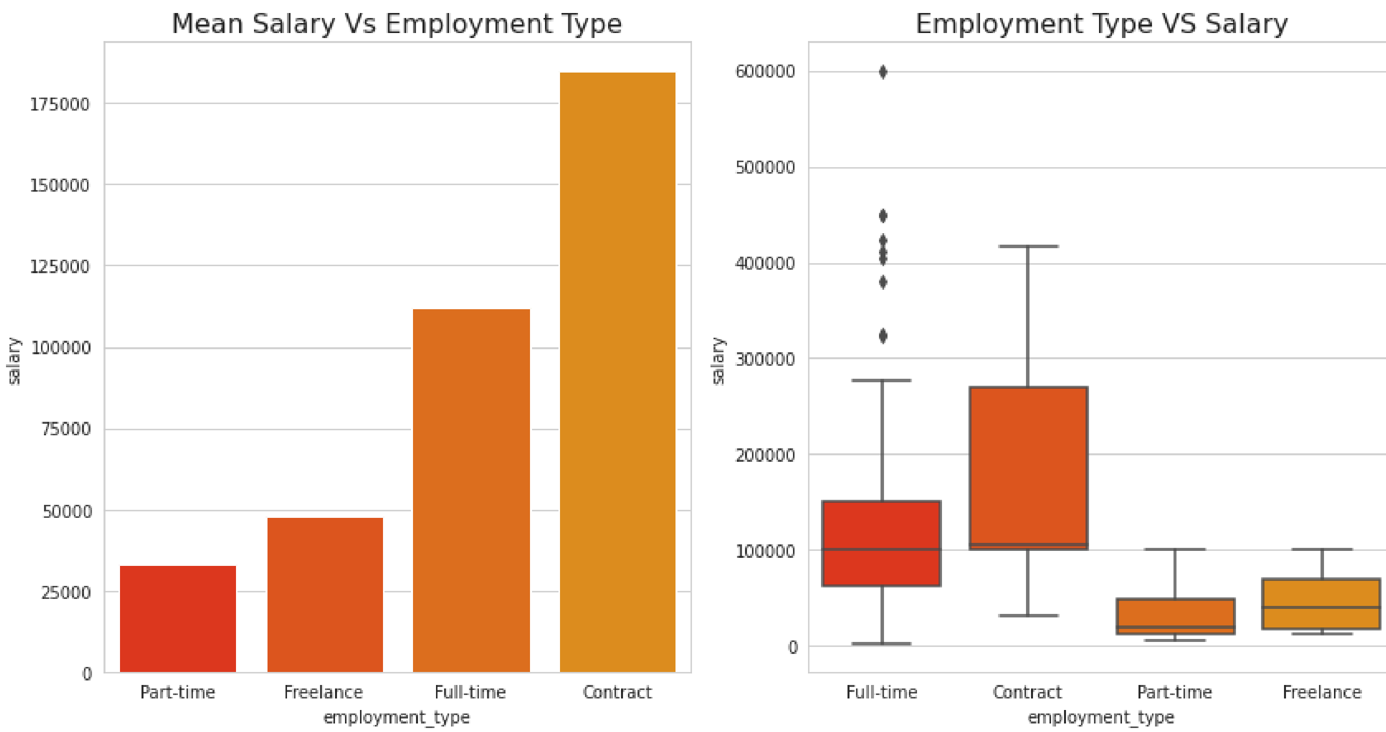
```
employment_type
Part-time     33070.500000
Freelance     48000.000000
Full-time    111811.838828
Contract     184575.000000

Name: salary, dtype: float64
```

In [27]:

```
plt.figure(figsize=(14, 7))
sns.set_palette('autumn')

plt.subplot(1, 2, 1)
ax = sns.barplot(x=mean_s_emp_type.index, y=mean_s_emp_type)
ax.set_title('Mean Salary Vs Employment Type',
fontdict={'fontsize': 16})

plt.subplot(1, 2, 2)
ax = sns.boxplot(data=df, x='employment_type', y='salary')
ax.set_title('Employment Type VS Salary', fontdict={'fontsize':
16})
```

Out[27]:

```
Text(0.5, 1.0, 'Employment Type VS Salary')
```



Employment Type VS Salary:

We see that data scientists with an employment type of ***contract*** have the highest mean salary, about **$180,000**, and those who work **part-time** have the lowest mean salary, about **$30,000** annualy.

Salary VS Company Size

In [28]:

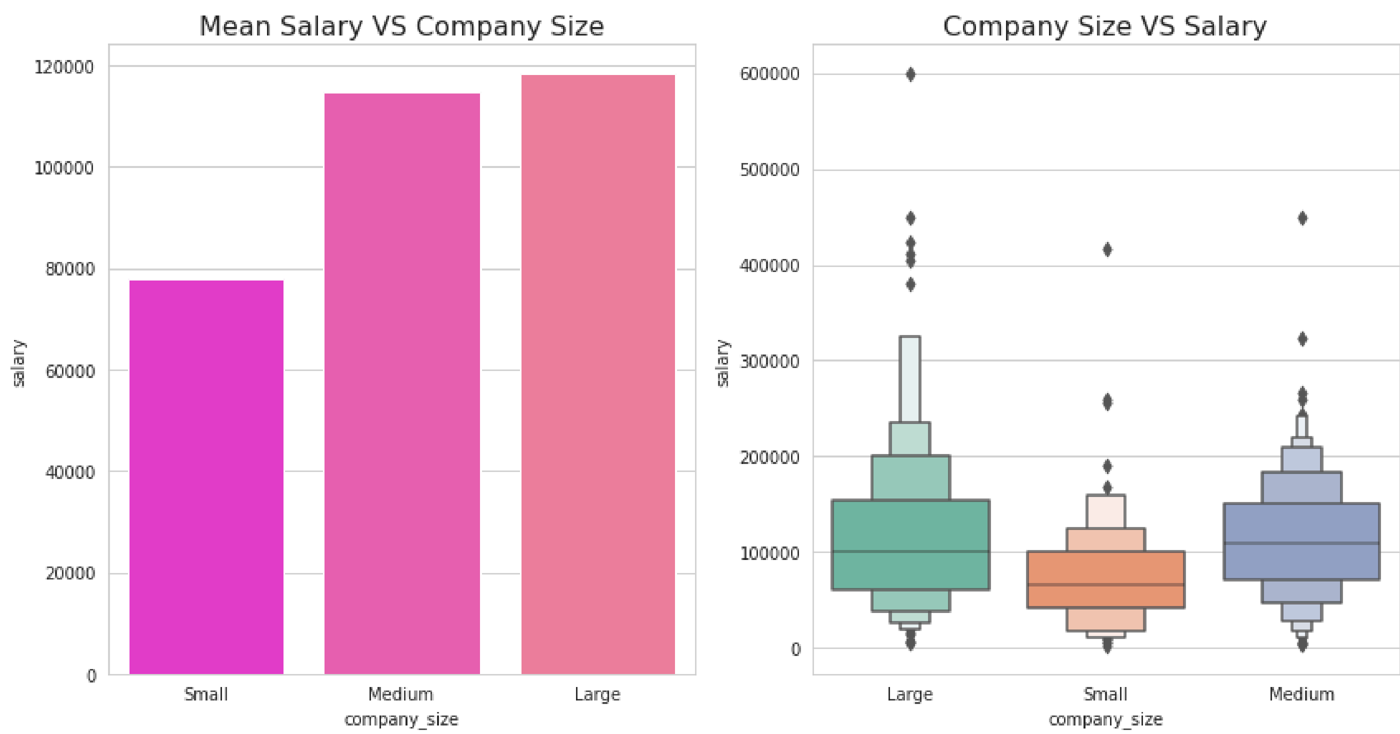
```
# mean salary of employees from different company sizes
mean_s_cmp_size =
df.groupby('company_size')['salary'].mean().sort_values()
mean_s_cmp_size
```

```
Out[28]:
company_size
Small      77872.097561
Medium    114807.079310
Large     118213.880829

Name: salary, dtype: float64
```

```
In [29]:
plt.figure(figsize=(14, 7))
sns.set_palette('spring')

plt.subplot(1, 2, 1)
ax = sns.barplot(x=mean_s_cmp_size.index, y=mean_s_cmp_size)
ax.set_title('Mean Salary VS Company Size',
fontdict={'fontsize': 16})

plt.subplot(1, 2, 2)
sns.set_palette('Set2')
ax = sns.boxenplot(data=df, x='company_size', y='salary')
ax.set_title('Company Size VS Salary', fontdict={'fontsize':
16})
```

Out[29]:

```
Text(0.5, 1.0, 'Company Size VS Salary')
```



Company Size VS Salary:

We see that data scientists working at ***Large*** companies are paid the highest mean salary, about **$120,000**, and those who work at **small** ones have the lowest mean salary, about **$75,000** annualy.

Salary VS Job type (remote, hybrid, onsite)

In [30]:

```
# mean salary of employees with different job types
mean_s_jtype =
df.groupby('job_type')['salary'].mean().sort_values()
```

```
mean_s_jtype
```

Out[30]:

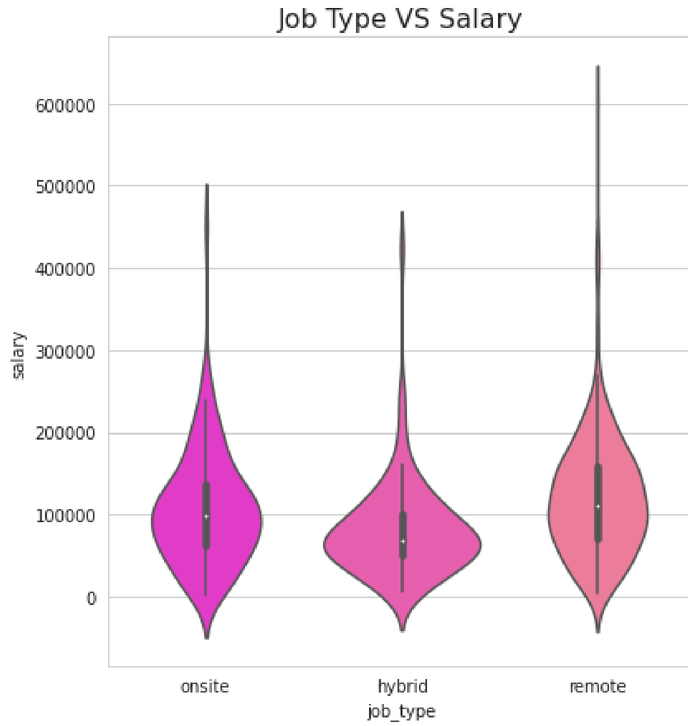
```
job_type
hybrid      80721.897959
onsite     105785.404959
remote     120763.190751

Name: salary, dtype: float64
```

In [31]:

```
plt.figure(figsize=(14, 7))
sns.set_palette('spring')

plt.subplot(1, 2, 1)
ax = sns.barplot(x=mean_s_jtype.index, y=mean_s_jtype)
ax.set_title('Mean Salary VS Job Type', fontdict={'fontsize':
16})

plt.subplot(1, 2, 2)
ax = sns.violinplot(data=df, x='job_type', y='salary')
ax.set_title('Job Type VS Salary', fontdict={'fontsize': 16})
```

Out[31]:

```
Text(0.5, 1.0, 'Job Type VS Salary')
```



Job Type (remote, on-site or hybrid) VS Salary:

We see that data scientists working ***remotely*** (about **$120,000**) have a higher mean salary than those who work **on-site** (about **$105,000**), and **hybrid** workers have a lower mean salary than former two (about **$80,000**) annualy.

In [32]:

```
# job type and company size VS salary
plt.figure(figsize=(14, 7))
sns.set_palette('Set2')
ax = sns.boxenplot(data=df, x='job_type', y='salary',
hue='company_size')
ax.set_title('Job Type & Company Size VS Salary',
```

```
fontdict={'fontsize': 16})
```

Out[32]:

```
Text(0.5, 1.0, 'Job Type & Company Size VS Salary')
```

I would prefer to work **remotely** at a **large** company to get paid higher.

Job Types and Experience Level distributions (Pie)

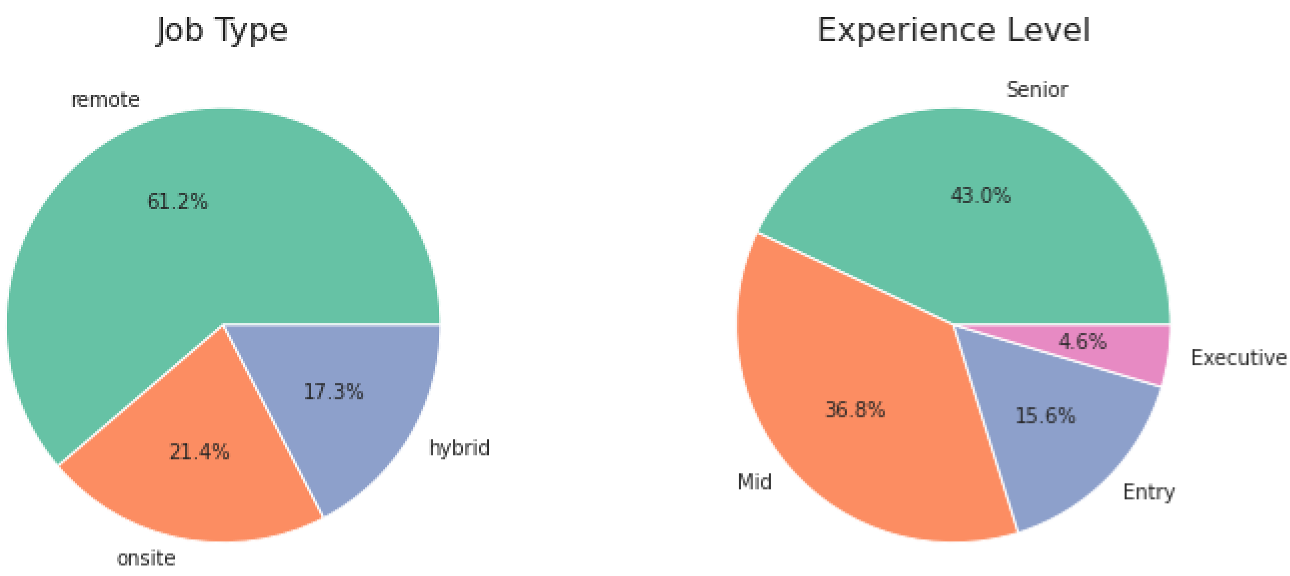
In [33]:

```
plt.figure(figsize=(12, 5))
sns.set_palette('Set2')
```

```python
# job types
plt.subplot(1,2,1)
ax = df['job_type'].value_counts().plot(kind='pie',
autopct='%1.1f%%')
ax.set_title('Job Type', fontdict={'fontsize': 16})
ax.set_ylabel('')

# experience levels
plt.subplot(1,2,2)
ax = df['experience_level'].value_counts().plot(kind='pie',
autopct='%1.1f%%')
ax.set_title('Experience Level', fontdict={'fontsize': 16})
ax.set_ylabel('')
```

Out[33]:

```
Text(0, 0.5, '')
```



- **Remote** jobs have the highest number of openings, **%.**
- Share of Job openings for employees with an experience level of **Senior** is the the highest here, **%.**

Top 10 Data Science Roles

In [34]:

```
# top 10 data science roles according to mean salary
top_ds_roles =
df.groupby('job_title')['salary'].mean().sort_values(ascending=
False)

# ignore those ds roles which happened only once
top_ds_roles_ =
pd.Series(data=list(top_ds_roles.index)).apply(lambda x: x if
list(df['job_title']).count(x) > 1 else 0)
```

```
top_ds_roles_that_happened_gt_1 = top_ds_roles_[top_ds_roles_
!= 0][:9]
top_ds_roles_that_happened_gt_1 =
top_ds_roles[top_ds_roles_that_happened_gt_1]
top_ds_roles_that_happened_gt_1
```

Out[34]:

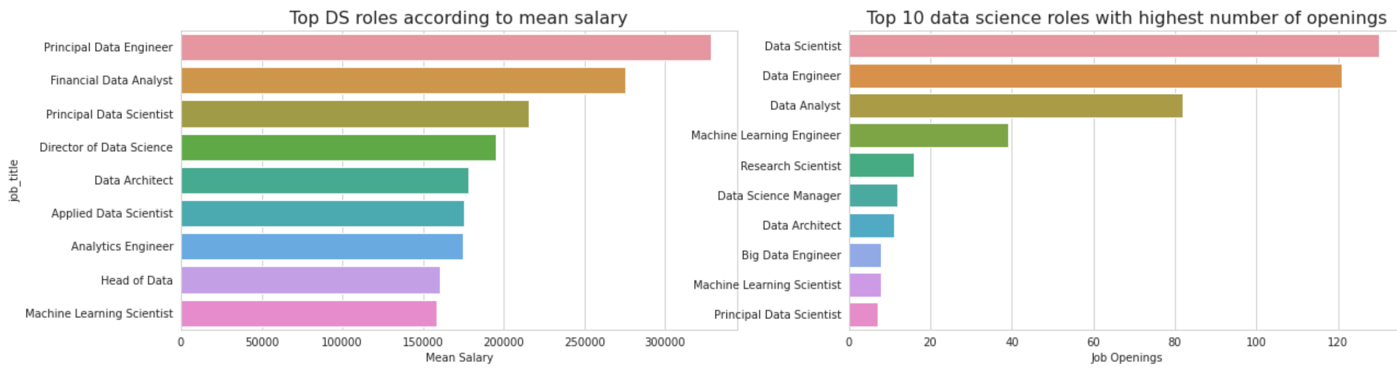
```
job_title
Principal Data Engineer       328333.333333
Financial Data Analyst        275000.000000
Principal Data Scientist      215242.428571
Director of Data Science      195074.000000
Data Architect                177873.909091
Applied Data Scientist        175655.000000
Analytics Engineer            175000.000000
Head of Data                  160162.600000
Machine Learning Scientist    158412.500000

Name: salary, dtype: float64
```

In [35]:

```
plt.figure(figsize=(20, 5))


# top 10 data science roles according to mean salary
plt.subplot(1, 2, 1)
```

```
top_ds_roles = top_ds_roles_that_happened_gt_1
ax = sns.barplot(y=top_ds_roles.index, x=top_ds_roles)
ax.set_xlabel('Mean Salary')
ax.set_title('Top DS roles according to mean salary',
fontdict={'fontsize': 16})

# top 10 data science roles with highest number of openings
plt.subplot(1, 2, 2)
top_dr = df['job_title'].value_counts()[:10]

ax = sns.barplot(x=top_dr, y=top_dr.index)
ax.set_xlabel('Job Openings')
ax.set_title('Top 10 data science roles with highest number of
openings', fontdict={'fontsize': 16})
```

```
Out[35]:

Text(0.5, 1.0, 'Top 10 data science roles with highest number
of openings')
```



- ***Principal Data Engineer***, ***Financial Data Analyst*** and ***Principal Data Scientist*** are the highest paid roles according to this dataset with mean annual salaries of **$405,000**, **$328,333** and **$275,000** respectively.
- **Data Scientist**, **Data Engineer** and **Data Analyst** are the top three Data Science roles with highest number of openings.

Top 10 campany-locations

In [36]:

```
# top 10 company-locations according to mean salary
top_cmp_locations =
df.groupby('company_location')['salary'].mean().sort_values(asc
ending=False)[:10]
top_cmp_locations
```

Out[36]:

```
company_location
Russia                        157500.000000
```

```
United States           144292.993711
New Zealand             125000.000000
Israel                  119059.000000
Japan                   114127.333333
Australia               108042.666667
Canada                  100121.857143
Iraq                    100000.000000
United Arab Emirates    100000.000000
Algeria                 100000.000000

Name: salary, dtype: float64
```

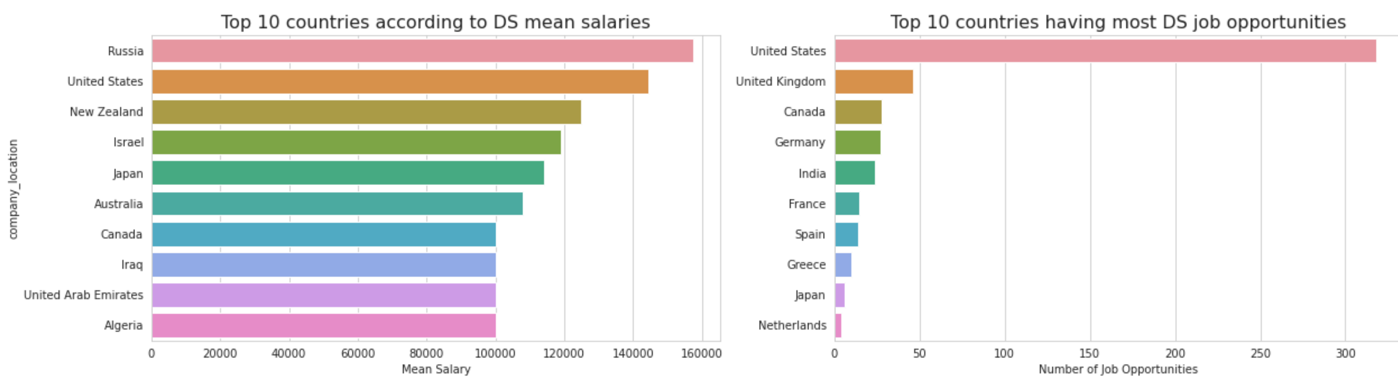
In [37]:

```
plt.figure(figsize=(20, 5))

# top 10 company-locations according to mean salary
plt.subplot(1, 2, 1)
ax = sns.barplot(y=top_cmp_locations.index,
x=top_cmp_locations)
ax.set_xlabel('Mean Salary')
ax.set_title('Top 10 countries according to DS mean salaries',
fontdict={'fontsize': 16})

# top 10 company-locations having most job opportunities
top_cl = df['company_location'].value_counts()[:10]
plt.subplot(1, 2, 2)
```

```
ax = sns.barplot(x=top_cl, y=top_cl.index)
ax.set_xlabel('Number of Job Opportunities')
ax.set_title('Top 10 countries having most DS job
opportunities', fontdict={'fontsize': 16})
```

Out[37]:

```
Text(0.5, 1.0, 'Top 10 countries having most DS job
opportunities')
```



- ***Russia***, ***the United States*** and ***New Zealand*** are the highest paying countries for data science roles according to this dataset, paying mean annual salaries of **$157,500**, **$144,055** and **$125,000** respectively.
- **The US**, **The UK** and **Canada** are the top three countries offering highest number of Data Science job.

Top 10 Employee-residence

In [38]:

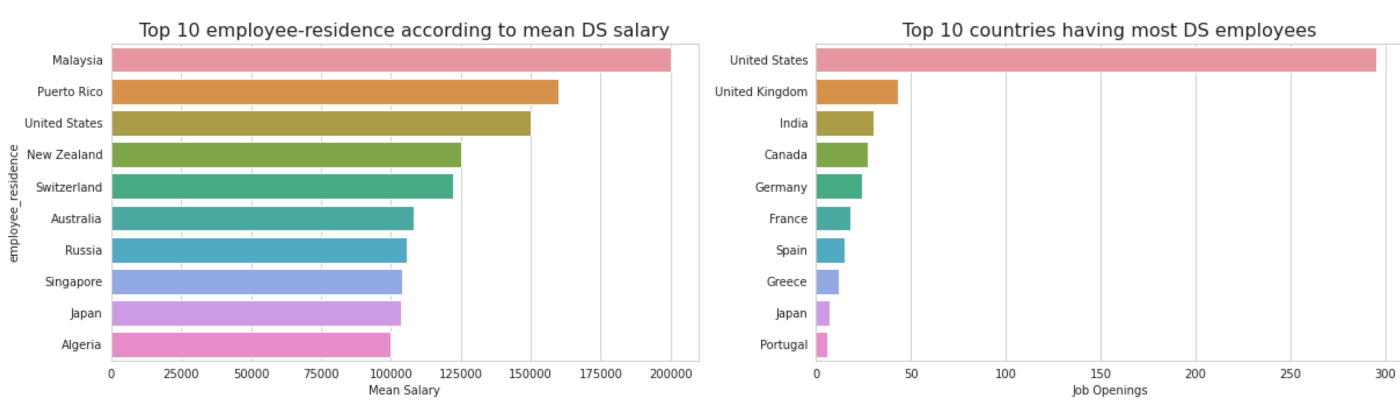
```
# top 10 employee-residence according to mean salary
top_emp_residence =
df.groupby('employee_residence')['salary'].mean().sort_values(a
scending=False)[:10]
top_emp_residence
```

```
Out[38]:
employee_residence
Malaysia          200000.000000
Puerto Rico       160000.000000
United States     150094.918644
New Zealand       125000.000000
Switzerland       122346.000000
Australia         108042.666667
Russia            105750.000000
Singapore         104176.500000
Japan             103537.714286
Algeria           100000.000000

Name: salary, dtype: float64
```

```
In [39]:
plt.figure(figsize=(20, 5))


# top 10 employee-residence according to mean salary
```

```
plt.subplot(1,2,1)
ax = sns.barplot(y=top_emp_residence.index,
x=top_emp_residence)
ax.set_xlabel('Mean Salary')
ax.set_title('Top 10 employee-residence according to mean DS
salary', fontdict={'fontsize': 16})

# top 10 employee-residence according to number of job openings
plt.subplot(1,2,2)
top_er = df['employee_residence'].value_counts()[:10]
ax = sns.barplot(x=top_er, y=top_er.index)
ax.set_title('Top 10 countries having most DS employees',
fontdict={'fontsize': 16})
ax.set_xlabel('Job Openings')
```

```
Out[39]:

Text(0.5, 0, 'Job Openings')
```



- ***Malaysia***, ***Puerto Rico*** and ***the US*** are the highest paid employee-residences (countries where employees live) in data science roles according to this dataset, being paid mean annual salaries of **$200,000**, **$160,000** and **$149,194** respectively.
- **The US**, **the UK** and **India** are the top three countires securing most Data Sceince job.

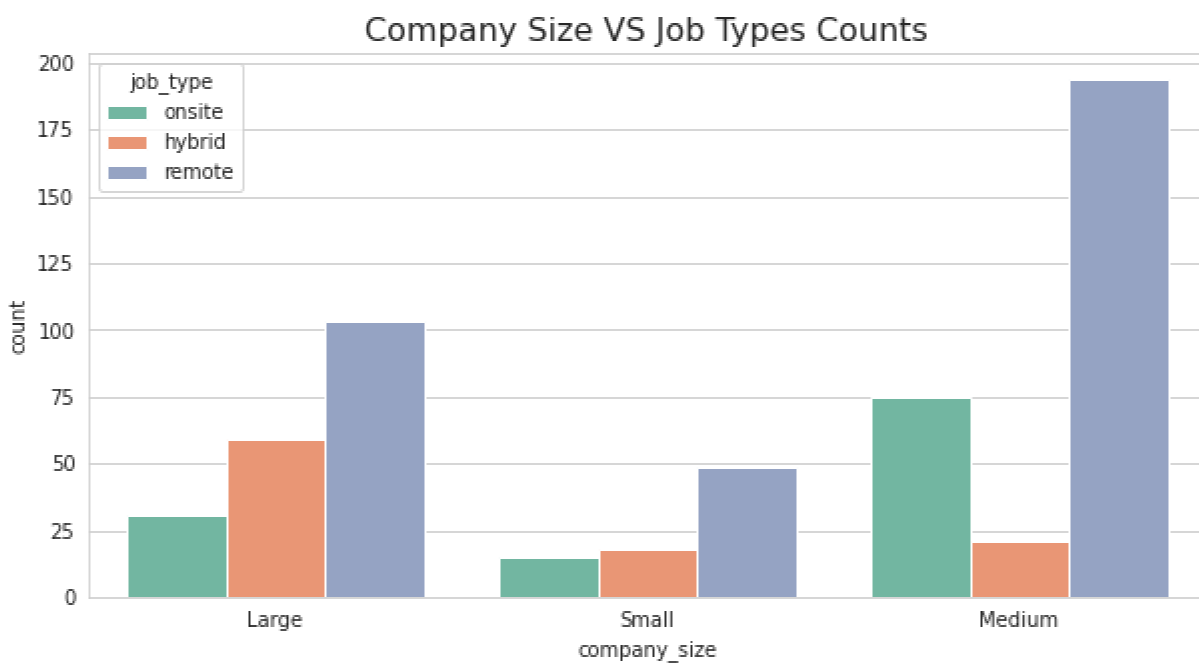
Company Size VS Job Types Counts

In [40]:

```
plt.figure(figsize=(10, 5))
sns.set_palette('Set2')
ax = sns.countplot(data=df, x='company_size', hue='job_type')
ax.set_title('Company Size VS Job Types Counts',
fontdict={'fontsize': 16})
```

Out[40]:

```
Text(0.5, 1.0, 'Company Size VS Job Types Counts')
```



linkcode

In all companies, the number of **remote** workers is **higher** than that of **hybrid** and **on-site**. Furthermore, the number of **hybrid** workers in **small** and **large** companies is **higher** than that of **on-site**, whereas in **medium-sized** companies, **more** people work **on-site** than **hybrid**.

Reference link