|  |  |
| --- | --- |
| 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](https://drive.google.com/file/d/1jlayA_UP3pcYdD2zkL_bd2KcPpFHu_ad/view?usp=sharing) [here](https://drive.google.com/file/d/1jlayA_UP3pcYdD2zkL_bd2KcPpFHu_ad/view?usp=sharing) [to](https://drive.google.com/file/d/1jlayA_UP3pcYdD2zkL_bd2KcPpFHu_ad/view?usp=sharing) [download](https://drive.google.com/file/d/1jlayA_UP3pcYdD2zkL_bd2KcPpFHu_ad/view?usp=sharing) [data](https://drive.google.com/file/d/1jlayA_UP3pcYdD2zkL_bd2KcPpFHu_ad/view?usp=sharing) [set](https://drive.google.com/file/d/1jlayA_UP3pcYdD2zkL_bd2KcPpFHu_ad/view?usp=sharing)

**About Dataset**

**Content**

**Column Description**

|  |  |
| --- | --- |
| work\_year | The year the salary was paid. |
| experienc e\_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 |
| employme nt\_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\_cur rency | 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 \_residenc e | Employee's primary country of residence in during the work year as an ISO 3166 country code. |
| remote\_ra  tio | 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. |

The average number of people that worked for the company during

company\_ the year: S less than 50 employees (small) M 50 to 250 employees

size

(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  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]:

Un

na

me

d:

0

wo

rk\_

ye

ar

experi

ence\_

level

emplo

yment

\_type

job

\_tit

le

sa

la

ry

salar

y\_cur

rency

salar

y\_in

\_usd

employ

ee\_resi

dence

rem

ote\_

ratio

compa

ny\_loc

ation

com

pany

\_siz

e

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | | | |
|  | 0 | 0 | 20  20 | MI | FT | Da ta  Sc  ien  tist | 7  0 0 0 0 | EUR | 798  33 | DE | 0 | DE | L |
| 1 | 1 | 20  20 | SE | FT | M  ac hin e Le ar nin g Sc  ien  tist | 2  6  0 0 0 0 | USD | 260  000 | JP | 0 | JP | S |
| 2 | 2 | 20  20 | SE | FT | Bi g Da  ta  En gin ee | 8 5 0 0 0 | GBP | 109  024 | GB | 50 | GB | M |
|  | | | | | | | | | | | | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | | | | | | | | |
|  |  |  |  |  |  | r |  |  |  |  |  |  |  |
| 3 | 3 | 20  20 | MI | FT | Pr od  uct  Da ta  An aly st | 2  0  0 0 0 | USD | 200  00 | HN | 0 | HN | S |
| 4 | 4 | 20  20 | SE | FT | M  ac hin e Le ar nin g En gin ee  r | 1 5 0 0 0 0 | USD | 150  000 | 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]:

df[

'employment\_type'

]

=

df[

'employment\_type'

]

.

map({

'FT'

:

'Full-time'

,

'PT'

:

'Part-time'

,

'CT'

:

'Contract'

,

'FL'

:

'Freelance'

})

In

[13]:

*#*

*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]: 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]:  *# 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 \_yea  r | experien ce\_level | employm ent\_type | job\_  title | sal ary | employee\_ residence | job\_ type | company  \_location | compa  ny\_size | | 0 | 2020 | Mid | Full-time | Dat a  Scie  ntist | 798  33 | Germany | onsi te | Germany | Large | | 1 | 2020 | Senior | Full-time | Mac hine Lear ning Scie  ntist | 260  000 | Japan | onsi te | Japan | Small | |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | 2 | 2020 | Senior | Full-time | Big Dat a Engi neer | 109  024 | United  Kingdom | hyb  rid | United  Kingdom | Mediu  m | | 3 | 2020 | Mid | Full-time | Pro duct Dat a Anal yst | 200  00 | Honduras | onsi te | Honduras | Small | | 4 | 2020 | Senior | Full-time | Mac hine Lear ning Engi neer | 150  000 | United  States | hyb  rid | 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

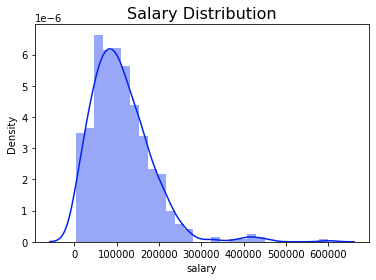
61643.318182

Mid

87792.995192

Senior

138374.880658



|  |
| --- |
| Executive 199392.038462  Name: salary, dtype: float64  In [24]: sns.set\_style('whitegrid')  In [25]:  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'

]

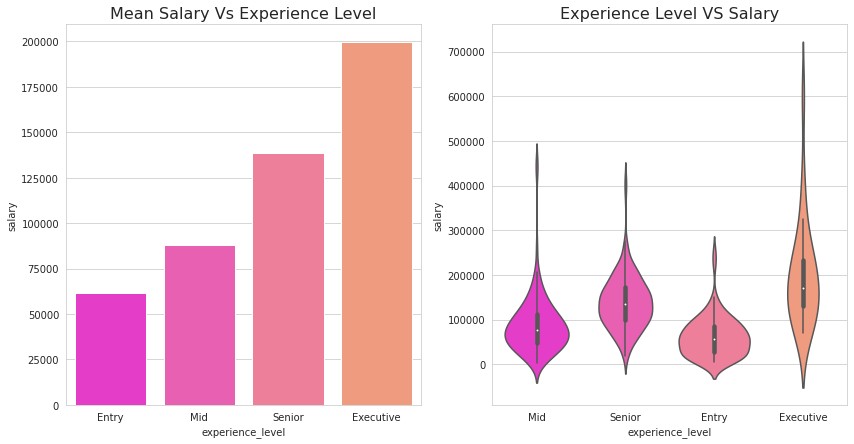
.

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'

]

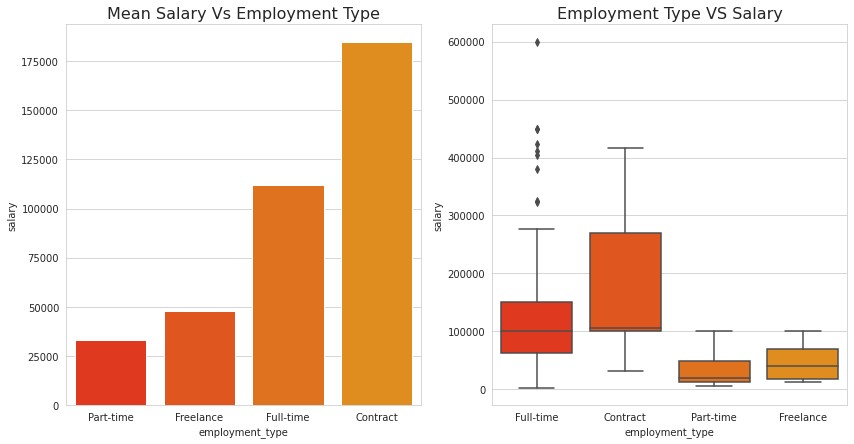
.

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'

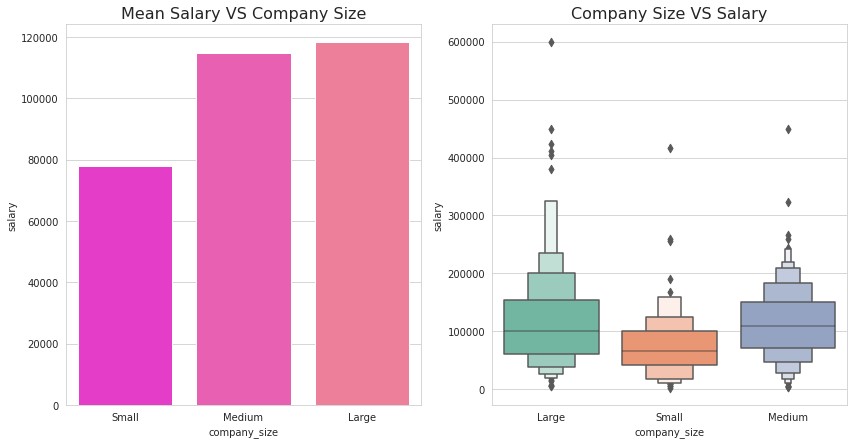
]

.

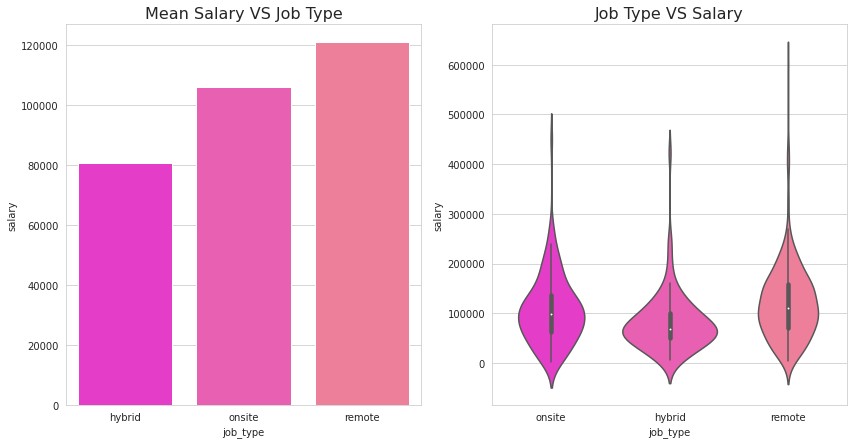
mean()

.

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'

)



|  |
| --- |
| *# 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)

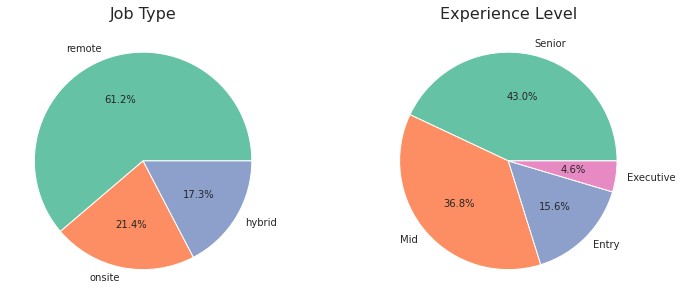
>

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

]

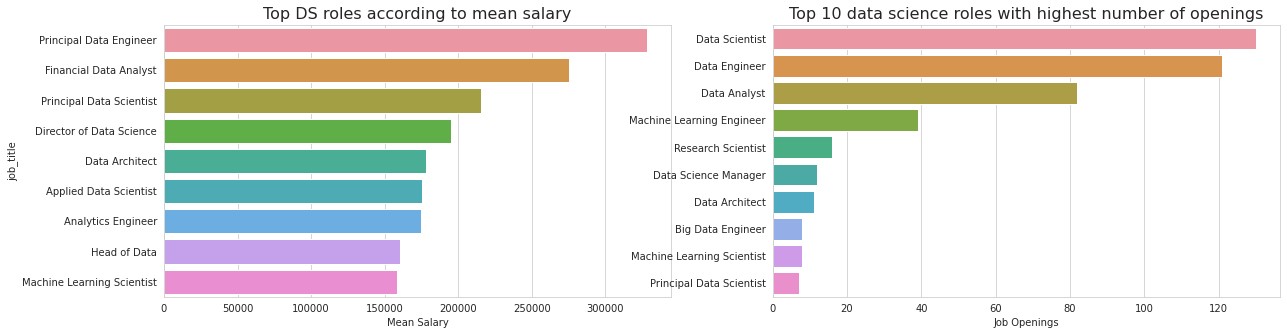
top\_cmp\_locations

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.

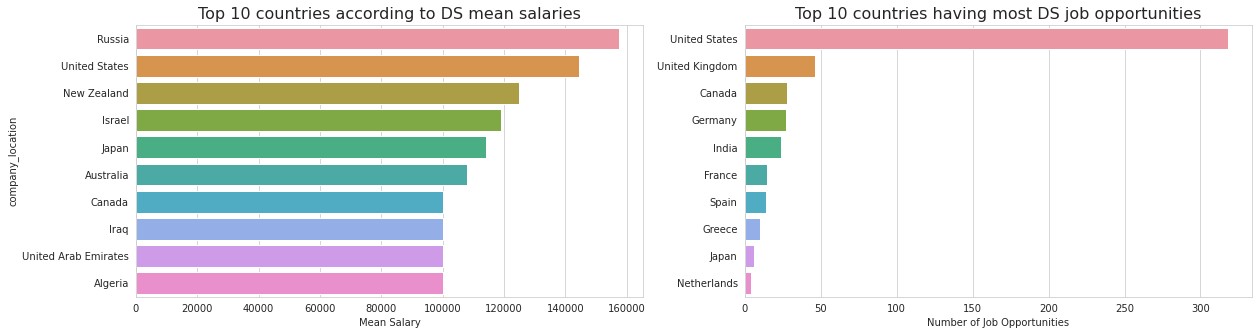
Top

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]:

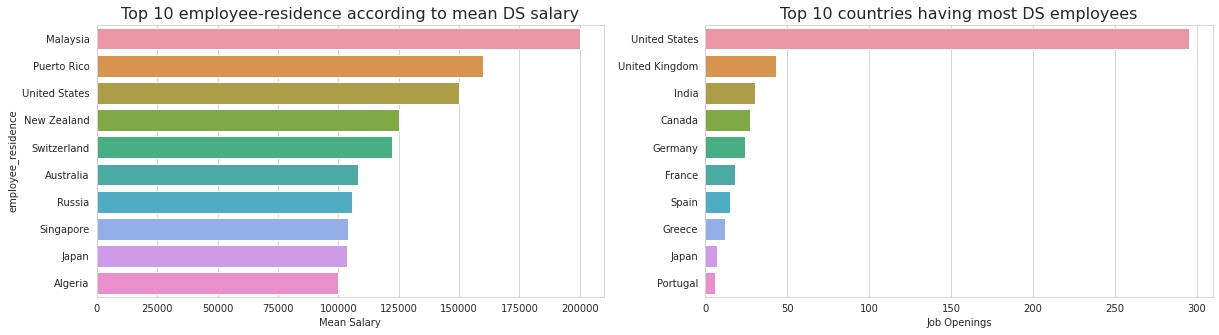
Text(0.5,

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

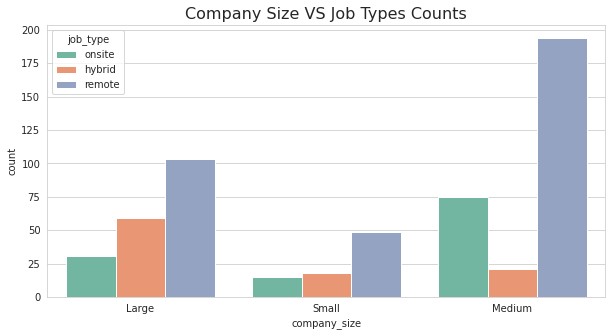
work

**on-site**

than

**hybrid**

.



[Reference](https://github.com/TomaIjatomi/EDA-on-Data-Science-Job-Salaries) [link](https://github.com/TomaIjatomi/EDA-on-Data-Science-Job-Salaries)