DATA SCIENCE INDUSTRY COMPENSATION ANALYSIS

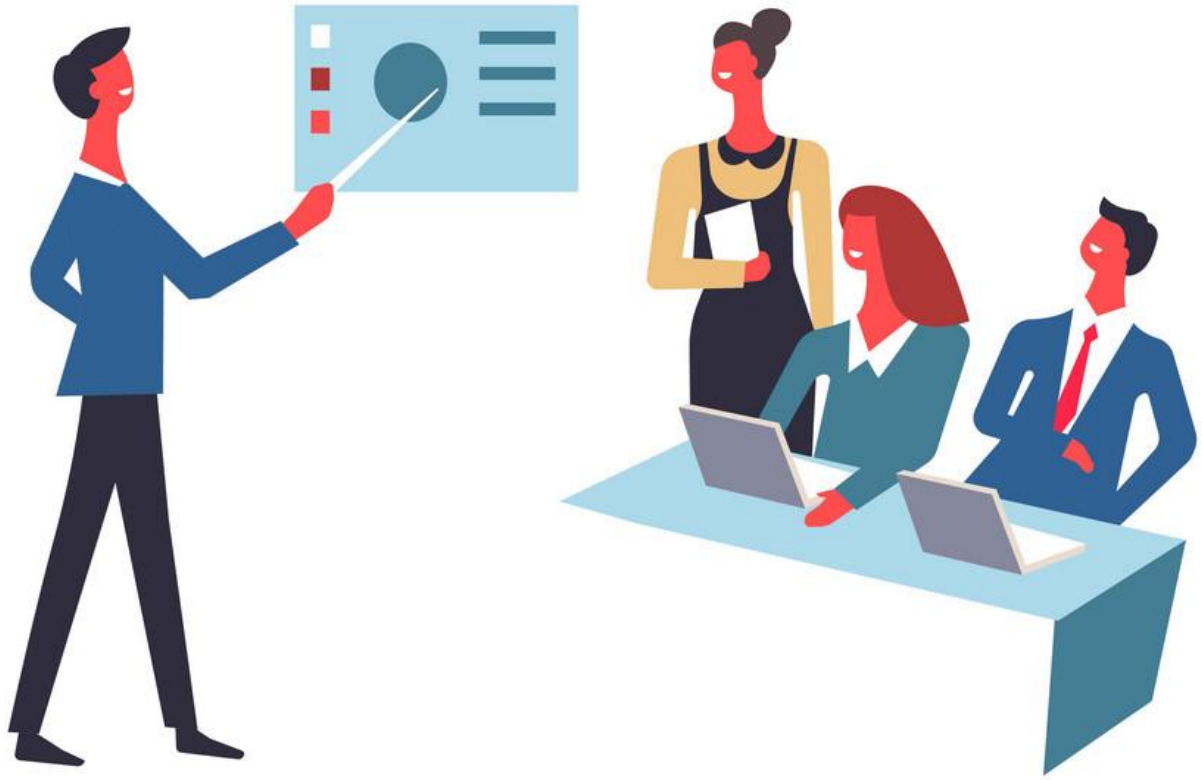
TEAM

Ananyaa Shahi
Anushka Mondal
Vivid Liu
Bhakti Kate

AGENDA

1. Introduction
2. Problem Statement & Objective
3. Data Reading
4. Data Cleaning
5. Exploratory Data Analysis
6. Data Modeling
7. Conclusion

INTRODUCTION



DATASET: DATA SCIENCE JOB SALARIES

- Encompasses 11 key columns capturing multifaceted aspects of job compensation within the realm of data science, covering a wide range of experience levels and job titles.

- The dataset includes information such as:
  - ✓ work year
  - ✓ experience level and type
  - ✓ salary amount
  - ✓ employee residence country
  - ✓ remote work ratio
  - ✓ company location and size

Source: https://www.kaggle.com/datasets/arnabchaki/data-science-salaries-2023

WHY THIS DATASET ?

- Aligns with our data science and analytics career goals, providing real and valuable insights tailored to our field.

- Quality data sourced from aijobs.net, a reputable platform specializing in AI, Machine Learning, and Data Science jobs.

- Enhances our proficiency in Python, statistics, data handling, analysis, and visualization skills.

PROBLEM STATEMENT & OBJECTIVE

- **PROBLEM STATEMENT :**

The Data Science industry is vast and has several factors affecting its entirety.

In this project, we would be determining the key elements that influence the variation in salaries for professionals in the data science and related fields in 2023.

Also, can we predict or analyze salary trends based on these attributes for 2024?

- **OBJECTIVE :**

The project will provide actionable recommendations, empowering us to make informed career decisions.

DATA READING

Summary

```
# summary of the data frame

ds_salaries_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3755 entries, 0 to 3754
Data columns (total 11 columns):
 #   Column              Non-Null Count  Dtype 
---  ------              --------------  ----- 
 0   work_year           3755 non-null   int64 
 1   experience_level    3755 non-null   object
 2   employment_type     3755 non-null   object
 3   job_title           3755 non-null   object
 4   salary              3755 non-null   int64 
 5   salary_currency     3755 non-null   object
 6   salary_in_usd       3755 non-null   int64 
 7   employee_residence  3755 non-null   object
 8   remote_ratio        3755 non-null   int64 
 9   company_location    3755 non-null   object
 10  company_size        3755 non-null   object
dtypes: int64(4), object(7)
memory usage: 322.8+ KB
```

Notable unique values

```
# check unique values of each column

for column in ds_salaries_df.columns:
    print('Column: {} - Unique Values: {}'.format(column, ds_salaries_df[column].unique()))
```

- ✓ **Work years:** 2020, 2021, 2022, 2023
- ✓ **Experience levels:** Entry-level, Senior, Middle, Executive
- ✓ **Employment types:** Full-Time, Contractor, Freelancer, Part-Time
- ✓ **Remote ratios:** 100, 50, 0
- ✓ **Company sizes:** Small, Medium, Large

DATA CLEANING

1. Transformation

```
# transform the column work_year into a string data type

ds_salaries_df['work_year'] = ds_salaries_df['work_year'].astype(str)
```

The work_year column initially had **int64** data type. Since no arithmetic operations is performed on the work_year column, we have transformed the data type to string.

2. Remove duplicates

```
ds_salaries_df.duplicated().sum()
```

```
1171
```

```
# drop duplicate rows

ds_salaries_df = ds_salaries_df.drop_duplicates()
```

```
ds_salaries_df.duplicated().sum()
```

```
0
```

A total of 1171 duplicate rows were found. After dropping, 2584 unique rows were obtained.

3. Remove leading and trailing white spaces

```
# Removing leading and trailing white spaces from all string datatype columns
for col in ds_salaries_df.select_dtypes(include='object'):
    ds_salaries_df[col] = ds_salaries_df[col].str.strip()

# Printing the DataFrame after removing white spaces
ds_salaries_df
```

	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remote_ratio	company_location
0	2023	SE	FT	Principal Data Scientist	80000	EUR	85847	ES	100	ES
1	2023	MI	CT	ML Engineer	30000	USD	30000	US	100	US
2	2023	MI	CT	ML Engineer	25500	USD	25500	US	100	US
3	2023	SE	FT	Data Scientist	175000	USD	175000	CA	100	CA
4	2023	SE	FT	Data Scientist	120000	USD	120000	CA	100	CA
...	...	...	...	...	...	...	...	...	...	...
3750	2020	SE	FT	Data Scientist	412000	USD	412000	US	100	US
3751	2021	MI	FT	Principal Data Scientist	151000	USD	151000	US	100	US
3752	2020	EN	FT	Data Scientist	105000	USD	105000	US	100	US
3753	2020	EN	CT	Business Data Analyst	100000	USD	100000	US	100	US
3754	2021	SE	FT	Data Science Manager	7000000	INR	94665	IN	50	IN

2584 rows × 11 columns

Trimming leading/trailing spaces in datasets ensures data consistency for accurate analysis, sorting, and filtering, improving overall reliability.

4. Replacing abbreviation in job titles

```
# Getting all unique values in the 'job_title' column
unique_job_titles = ds_salaries_df['job_title'].unique()
```

Initially, there were 93 unique job titles. However, there was a subtle repetition that could have affected the result of the analysis.

```
# This dictionary (job_title_mappings) defines the mapping between similar job titles.
# In this case, it indicates that 'ML Engineer' should be replaced with 'Machine Learning Engineer'.

job_title_mappings = {
    'ML Engineer': 'Machine Learning Engineer'
}

ds_salaries_df['job_title'] = ds_salaries_df['job_title'].map(job_title_mappings).fillna(ds_salaries_df['job_title'])

'''
Here, the map function is used to replace values in the 'job_title' column based on the mappings defined in job_title_mappings.
The fillna method is then used to fill any NaN (Not a Number) values with the original values from the 'job_title' column.
This step ensures that if a job title is not found in the mapping, it remains unchanged.
'''

ds_salaries_df
```

After replacing the abbreviation ML Engineer with its full form Machine Learning Engineer, we got 92 unique job titles.

5. Replacing values

```
#  Replace values in experience-level column
ds_salaries_df['experience_level'] = ds_salaries_df['experience_level'].replace('EN', 'Entry-Level')
ds_salaries_df['experience_level'] = ds_salaries_df['experience_level'].replace('EX', 'Executive')
ds_salaries_df['experience_level'] = ds_salaries_df['experience_level'].replace('MI', 'Mid-Level')
ds_salaries_df['experience_level'] = ds_salaries_df['experience_level'].replace('SE', 'Senior')
```

```
#Replace values in employment_type column
ds_salaries_df['employment_type'] = ds_salaries_df['employment_type'].replace('FT', 'Full-Time')
ds_salaries_df['employment_type'] = ds_salaries_df['employment_type'].replace('CT', 'Contractor')
ds_salaries_df['employment_type'] = ds_salaries_df['employment_type'].replace('FL', 'Freelancer')
ds_salaries_df['employment_type'] = ds_salaries_df['employment_type'].replace('PT', 'Part-Time')
```

To enhance comprehension, interpretation, and analysis, we have substituted abbreviations with their corresponding full forms.

EXPLORATORY DATA ANALYSIS

Trendiest Job Title: Which is the trendiest job title by country?

```
df_most_common_job = df_salary_data
# Find the most common job title and highest salary for each country
df_most_common_job['Most Common Job'] = df_most_common_job.apply(lambda row: row.count(), axis=1)

df_most_common_job['Salary (USD)'] = df_most_common_job.max(axis=1)

# Find the index of the row with the highest value in the "Most Common Job" column for each company location
max_common_job_idx = df_most_common_job.groupby('company_location')['Most Common Job'].idxmax()

# Select only the rows with the highest value in the "Most Common Job" column
df_most_common_job = df_most_common_job.loc[max_common_job_idx]
df_most_common_job = df_most_common_job.drop(range(487), axis=1)
# Reset the index to make 'company_location' a column
df_most_common_job = df_most_common_job.reset_index()

# Sort the DataFrame by 'company_location' column
df_most_common_job = df_most_common_job.sort_values(by='company_location')
df_most_common_job = df_most_common_job.rename(columns={'job_title': 'Most Common Job Title', 'company_location': 'Country'})

df_most_common_job = df_most_common_job.drop('Most Common Job', axis=1)
# Print the DataFrame
df_most_common_job
```

This analysis helps us make informed decision about the job market in our "**Dream work location**".

	Country	Most Common Job Title	Salary (USD)
0	AE	Machine Learning Engineer	120000.0
1	AL	3D Computer Vision Researcher	10000.0
2	AM	Machine Learning Engineer	50000.0
3	AR	Data Analyst	50000.0
4	AS	3D Computer Vision Researcher	20000.0
...	...	...	...
67	TH	Data Science Consultant	29453.0
68	TR	Data Scientist	25000.0
69	UA	AI Developer	108000.0
70	US	Data Engineer	324000.0
71	VN	Data Engineer	12000.0

Lucrative Job Title: Which is the most lucrative job title by country?

```
df_highest_paying_job = df_salary_data
# Find the most common job title and highest salary for each country
df_highest_paying_job['Salary (USD)'] = df_highest_paying_job.apply(lambda row: max(row), axis=1)
df_highest_paying_job = df_highest_paying_job.drop(range(487), axis=1)
df_highest_paying_job = df_highest_paying_job.drop('Most Common Job', axis=1)
# Find the index of the row with the highest value in the "Most Common Job" column for each company location
highest_paid_job_idx = df_salary_data.groupby('company_location')['Salary (USD)'].idxmax()

# Select only the rows with the highest value in the "Most Common Job" column
df_highest_paying_job = df_highest_paying_job.loc[highest_paid_job_idx]
df_highest_paying_job = df_highest_paying_job.reset_index()
df_highest_paying_job = df_highest_paying_job.rename(columns={'job_title': 'Highest Paying Job Title', 'company_location': 'Coun
df_highest_paying_job
```

This analysis empowers us with the knowledge necessary to target our "**Dream Salary**".

	Country	Highest Paying Job Title	Salary (USD)
0	AE	Machine Learning Engineer	120000.0
1	AL	3D Computer Vision Researcher	10000.0
2	AM	Machine Learning Engineer	50000.0
3	AR	Data Analyst	50000.0
4	AS	Business Data Analyst	50000.0
...	...	...	...
67	TH	Data Science Consultant	29453.0
68	TR	Data Engineer	28016.0
69	UA	AI Developer	108000.0
70	US	Research Scientist	450000.0
71	VN	Data Engineer	12000.0

By Country, Most Common Job V/S Highest paid Job

```
df1 = df_highest_paying_job.rename(columns={'Salary (USD)': 'Highest Paid Job Salary (USD)'})
df2 = df_most_common_job.rename(columns={'Salary (USD)': 'Most Common Job Salary (USD)'})
df_by_country_most_common_vs_highest_paid = pd.merge(df2, df1, on='Country')
# Print the merged DataFrame
df_by_country_most_common_vs_highest_paid
```

This analysis guides us to make optimal job choice with respect to salary.
Best used during **job switching**.

	Country	Most Common Job Title	Most Common Job Salary (USD)	Highest Paying Job Title	Highest Paid Job Salary (USD)
0	AE	Machine Learning Engineer	120000.0	Machine Learning Engineer	120000.0
1	AL	3D Computer Vision Researcher	10000.0	3D Computer Vision Researcher	10000.0
2	AM	Machine Learning Engineer	50000.0	Machine Learning Engineer	50000.0
3	AR	Data Analyst	50000.0	Data Analyst	50000.0
4	AS	3D Computer Vision Researcher	20000.0	Business Data Analyst	50000.0
...	...	...	...	...	...
67	TH	Data Science Consultant	29453.0	Data Science Consultant	29453.0
68	TR	Data Scientist	25000.0	Data Engineer	28016.0
69	UA	AI Developer	108000.0	AI Developer	108000.0
70	US	Data Engineer	324000.0	Research Scientist	450000.0
71	VN	Data Engineer	12000.0	Data Engineer	12000.0

Demographic Pay Gaps: Are there any pay gaps related to most common job title in the world?

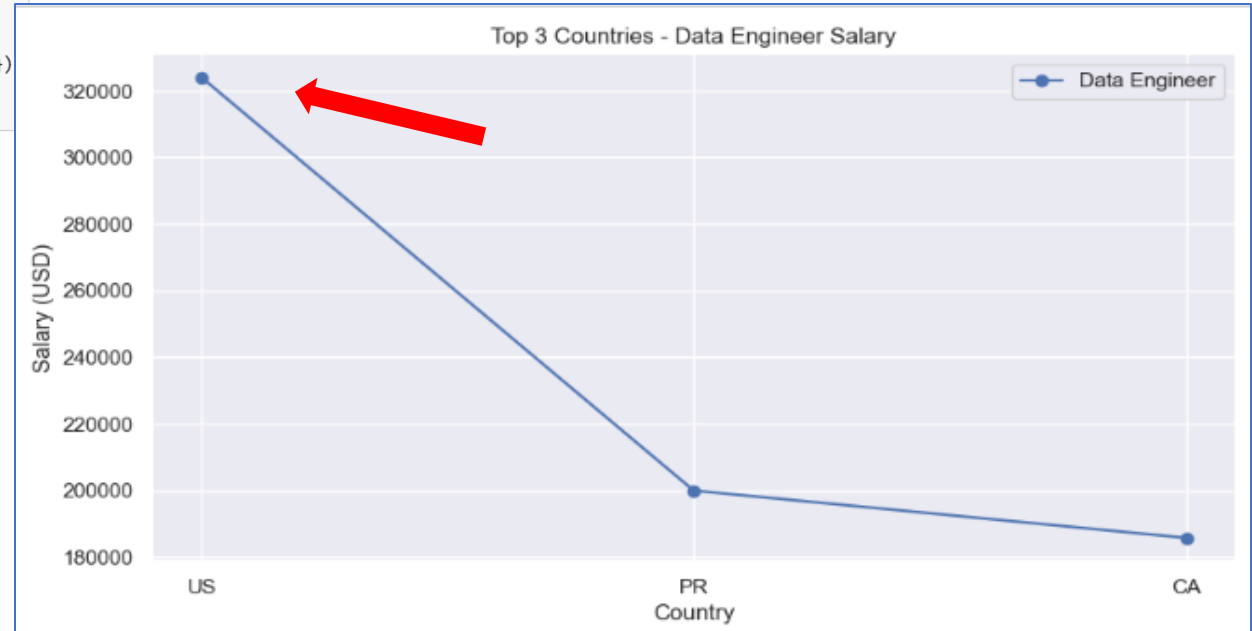
```
df_worlds_most_common_job_idx = df_most_common_job['Most Common Job'].idxmax()
df_worlds_most_common_job = df_most_common_job
df_worlds_most_common_job = df_worlds_most_common_job.rename(columns={'Most Common Job Title': 'Worlds Most Common Job Title'})
df_worlds_most_common_job = df_worlds_most_common_job.loc[df_worlds_most_common_job_idx]
df_worlds_most_common_job
```

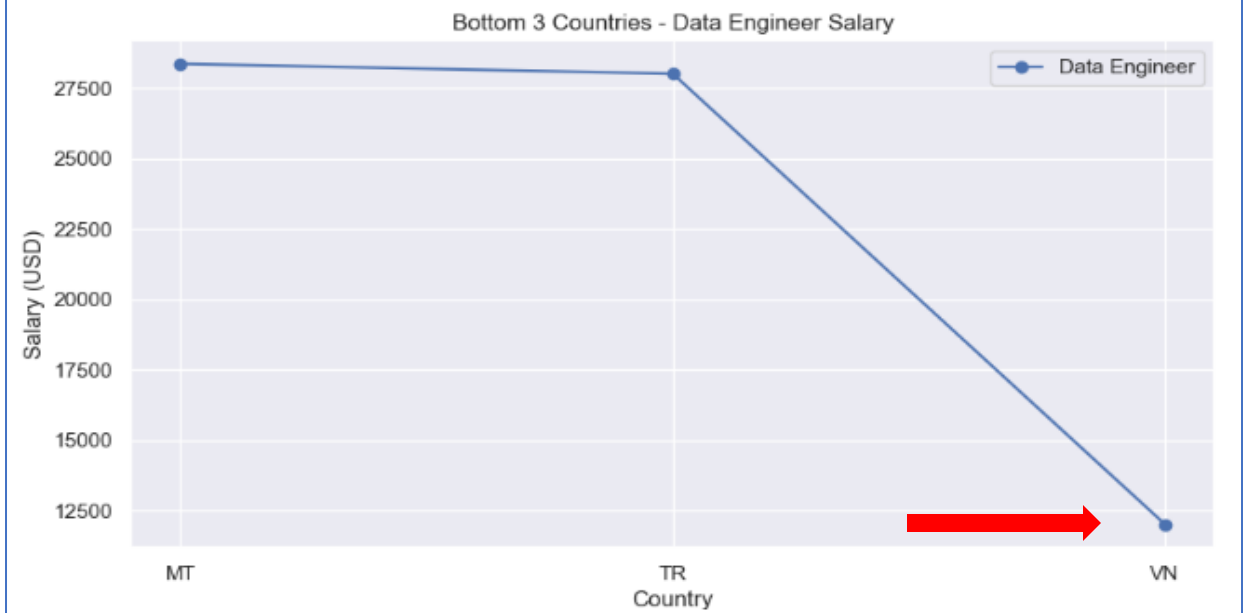
```
Country                              US
Worlds Most Common Job Title    Data Engineer
Most Common Job                           489
Salary (USD)                         324000.0
Name: 70, dtype: object
```

```
# Create a bar plot for the top 3 countries
plt.figure(figsize=(10, 5))
plt.plot(top_df['company_location'], top_df['Salary (USD)'], marker='o')
plt.xlabel('Country')
plt.ylabel('Salary (USD)')
plt.title('Top 3 Countries - Data Engineer Salary')
plt.legend(['Data Engineer'])
plt.show()

# Create a line graph for the bottom 3 countries
plt.figure(figsize=(10, 5))
plt.plot(bottom_df['company_location'], bottom_df['Salary (USD)'], marker='o')
plt.xlabel('Country')
plt.ylabel('Salary (USD)')
plt.title('Bottom 3 Countries - Data Engineer Salary')
plt.legend(['Data Engineer'])
plt.show()
```

We see that there is definite pay-gap for the most common job title in the world.

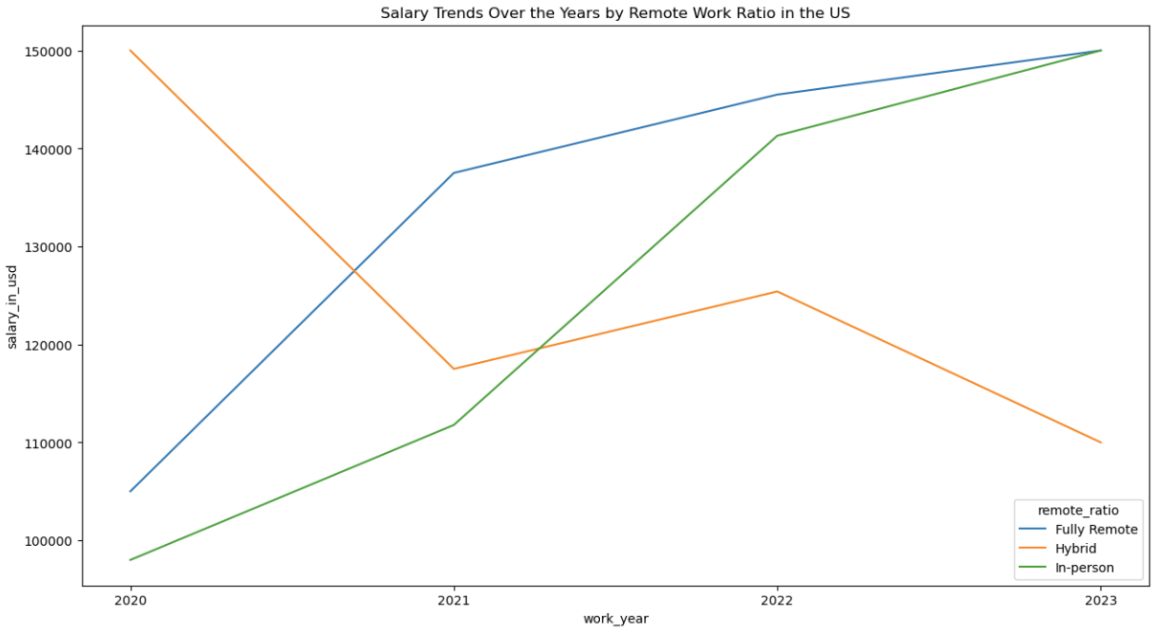
Data Engineer :
US highest ($325k), **Vietnam** lowest ($12k)





Impact of remote_ratio column: Does work modality have any significant impact on salaries?

```python
import seaborn as sns
import matplotlib.pyplot as plt

# Assuming your dataset is stored in a variable df
# Filter data for US locations
us_data = ds_salaries_df[ds_salaries_df['company_location'] == 'US']

# Group by relevant factors
grouped_data = us_data.groupby(['remote_ratio', 'work_year'])

# Calculate median salary for each group
agg_data = grouped_data['salary_in_usd'].median().reset_index()

# Create line plot for salary trends across remote work ratios
plt.figure(figsize=(15, 8))
sns.lineplot(x='work_year', y='salary_in_usd', hue='remote_ratio', data=agg_data)
plt.title('Salary Trends Over the Years by Remote Work Ratio in the US')
plt.show()
```



Work modes: fully remote, hybrid, in-person

- During the peak pandemic years of 2020 and 2021, remote workers' median salaries surged. Surprisingly, in 2022 and 2023, despite the return to in-person work, remote job compensation continued to rise.

- By 2023, median salaries for in-person and remote work equalized, showing no difference. However, hybrid work experienced a sharp decline in median salary, becoming the least lucrative option.

Highest paid remote_ratio entry level positions: Which are the **highest paid** entry level jobs in US in 2023 as per work modality?

NOTE: We have chosen US company location because we believe that majority of our classmates would want to work in the US after graduation.

```
# Select relevant columns from highest_paid_jobs_entry_level DataFrame
selected_columns_entry_level_high = highest_paid_jobs_entry_level[['remote_ratio', 'job_title', 'salary_in_usd']]

# Merge with us_data_entry_level to get additional information
merged_data_entry_level_high = selected_columns_entry_level_high.merge(
    us_data_entry_level[['job_title', 'company_size', 'experience_level', 'employment_type']], on='job_title', how='left')

# Drop duplicates to keep unique rows
unique_rows_entry_level_high = merged_data_entry_level_high.drop_duplicates()

# Print the final result for Entry-Level
print("Highest Paid Entry-Level Job Titles by Remote Work Ratio:")

unique_rows_entry_level_high
```

	remote_ratio	job_title	salary_in_usd	company_size	experience_level	employment_type
0	Fully Remote	Machine Learning Scientist	225000.0	L	Entry-Level	Full-Time
1	Hybrid	Research Scientist	220000.0	L	Entry-Level	Full-Time
2	Hybrid	Research Scientist	220000.0	M	Entry-Level	Full-Time
5	In-person	Computer Vision Engineer	172500.0	M	Entry-Level	Full-Time

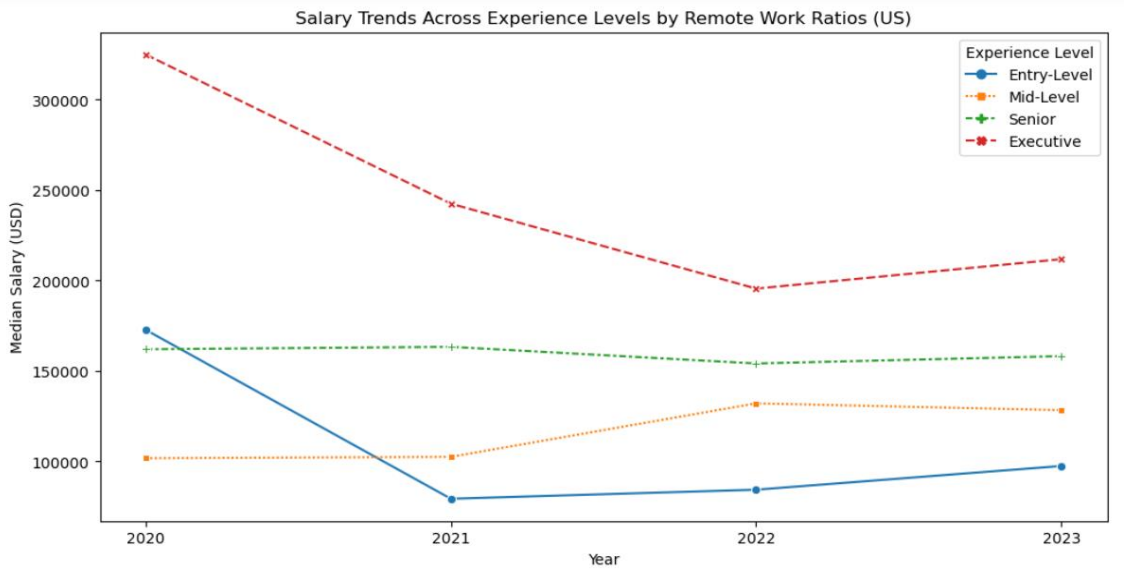
Lowest paid remote_ratio entry level positions: Which are the **lowest paid** entry level jobs in US in 2023 as per work modality?

```
# Select relevant columns from lowest_paid_jobs_entry_level DataFrame
selected_columns_entry_level_low = lowest_paid_jobs_entry_level[['remote_ratio', 'job_title', 'salary_in_usd']]

# Merge with us_data_entry_level to get additional information
merged_data_entry_level_low = selected_columns_entry_level_low.merge(
    us_data_entry_level[['job_title', 'company_size', 'experience_level', 'employment_type']], on='job_title', how='left')

# Drop duplicates to keep unique rows
unique_rows_entry_level_low = merged_data_entry_level_low.drop_duplicates()

# Print the final result for Entry-Level
print("Lowest Paid Entry-Level Job Titles by Remote Work Ratio:")

unique_rows_entry_level_low
```

	remote_ratio	job_title	salary_in_usd	company_size	experience_level	employment_type
0	Fully Remote	AI Scientist	12000.0	M	Entry-Level	Full-Time
1	Fully Remote	AI Scientist	12000.0	M	Entry-Level	Part-Time
2	Fully Remote	AI Scientist	12000.0	S	Entry-Level	Part-Time
3	Fully Remote	BI Analyst	12000.0	L	Entry-Level	Part-Time
4	Fully Remote	BI Analyst	12000.0	L	Entry-Level	Full-Time
5	Hybrid	Business Data Analyst	48000.0	L	Entry-Level	Full-Time
6	Hybrid	Business Data Analyst	48000.0	L	Entry-Level	Contractor
7	In-person	Data Analyst	62500.0	M	Entry-Level	Full-Time
16	In-person	Data Analyst	62500.0	L	Entry-Level	Full-Time
19	In-person	Data Analyst	62500.0	S	Entry-Level	Part-Time
25	In-person	Data Analyst	62500.0	S	Entry-Level	Full-Time
26	In-person	Data Analyst	62500.0	L	Entry-Level	Part-Time

Impact of experience and remote_ratio: Did experience levels play a role in salary trends as per different work modes?

```
# Assuming your dataset is stored in a variable df
# Filter data for US locations
us_data = ds_salaries_df[ds_salaries_df['company_location'] == 'US']

# Specify the order of experience levels
experience_order = ['Entry-Level', 'Mid-Level', 'Senior', 'Executive']

# Group by relevant factors
grouped_data4 = us_data.groupby(['work_year', 'experience_level', 'remote_ratio'])

# Calculate median salary for each group
agg_data4 = grouped_data4['salary_in_usd'].median().reset_index()

# Plotting the salary trends for different remote work ratios across experience levels

# Plotting the salary trends for different remote work ratios with only 'experience_level' in the legend
plt.figure(figsize=(12, 6))
sns.lineplot(x='work_year', y='salary_in_usd', hue='experience_level', style='experience_level', markers=True,
             data=agg_data4, hue_order=experience_order, errorbar = None)
plt.title('Salary Trends Across Experience Levels by Remote Work Ratios (US)')
plt.xlabel('Year')
plt.ylabel('Median Salary (USD)')
plt.legend(title='Experience Level', bbox_to_anchor=(1, 1))
plt.show()
```

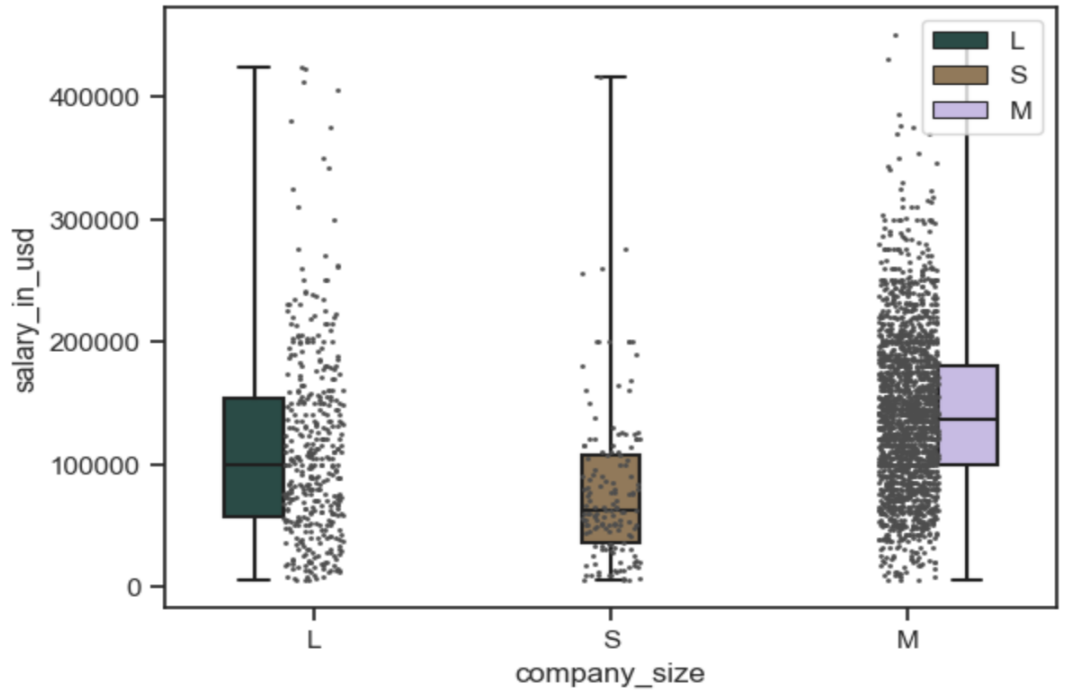


Entry and executive-level median salaries declined significantly during the peak pandemic years from 2020 to 2021 and showed minimal growth from 2022 to 2023. Meanwhile, senior-level salaries consistently increased and mid-level salaries grew from 2021 to 2022 as the pandemic eased.

Salary variation based on company size: Do salaries vary by company size (small, medium, and large)?

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

sns.set_theme(style="ticks")

#plotting box plot
sns.boxplot(ds_salaries_df, x="company_size", y="salary_in_usd", hue="company_size",
    whis=[0, 100], width=0.6, palette="cubehelix",
)
# Add in points to show each observation
sns.stripplot(ds_salaries_df, x="company_size", y="salary_in_usd", size=2, color=".3")

#ax.xaxis.grid(True)
#ax.set(ylabel="")
```

```
<Axes: xlabel='company_size', ylabel='salary_in_usd'>
```



Medium sized companies lead in data science employment rate, with higher median salary ranges compared to both large and small firms, as indicated by the box plot.

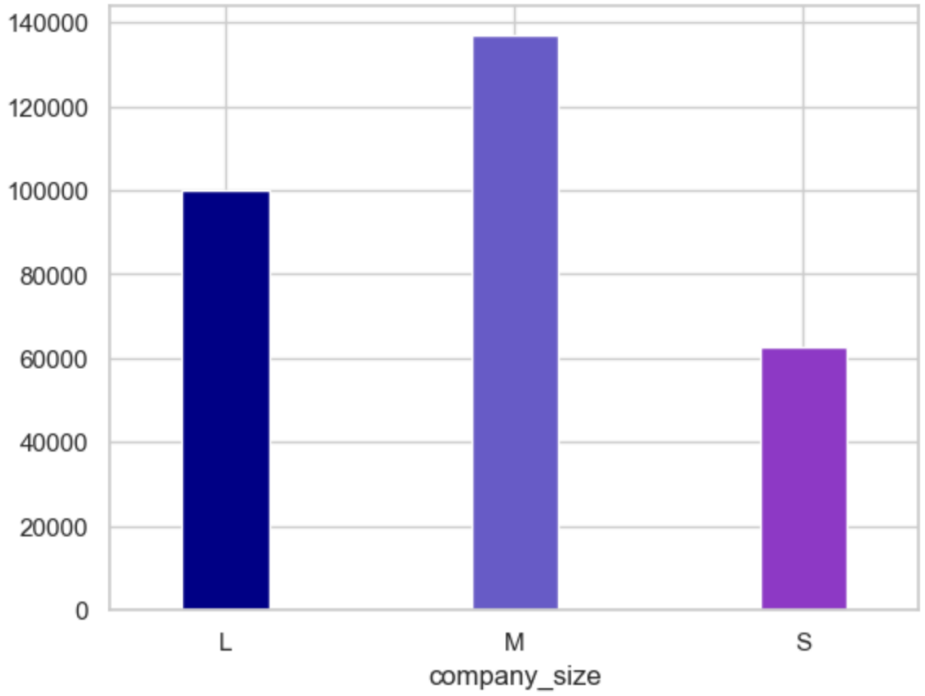
Salary variation based on company size: How is salary distributed across companies based on their sizes?

```
# median of salary vs company size
salary_vs_company_size = ds_salaries_df.groupby('company_size')['salary_in_usd'].median()
salary_vs_company_size
```

```
company_size
L    100000.0
M    136975.0
S     62726.0
Name: salary_in_usd, dtype: float64
```

```
#plotting salary vs company size
import matplotlib.pyplot as plt
import matplotlib.colors as mcolors

fig, ax = plt.subplots()
bar_colors = ['darkblue', 'slateblue', 'darkorchid']
ax = salary_vs_company_size.plot.bar(x='company_size',
   y='salary_in_usd', width= 0.3, rot=0, color=bar_colors)


plt.show()
```

Medium sized companies offer higher median salaries than both large and small companies.



Employment types and experience: What sizes of companies are attracting most of the data science-related jobs? Is there any relationship between company size and experience level?

```
# counting the value of each experience level for employment type
employment_vs_experience = ds_salaries_df.groupby('experience_level')['employment_type'].value_counts()
employment_vs_experience
```

```
employment_vs_experience_dt=pd.DataFrame(employment_vs_experience )

employment_vs_experience_dt.rename(columns = {'employment_type':'Count'}, inplace = True)
employment_vs_experience_dt
```

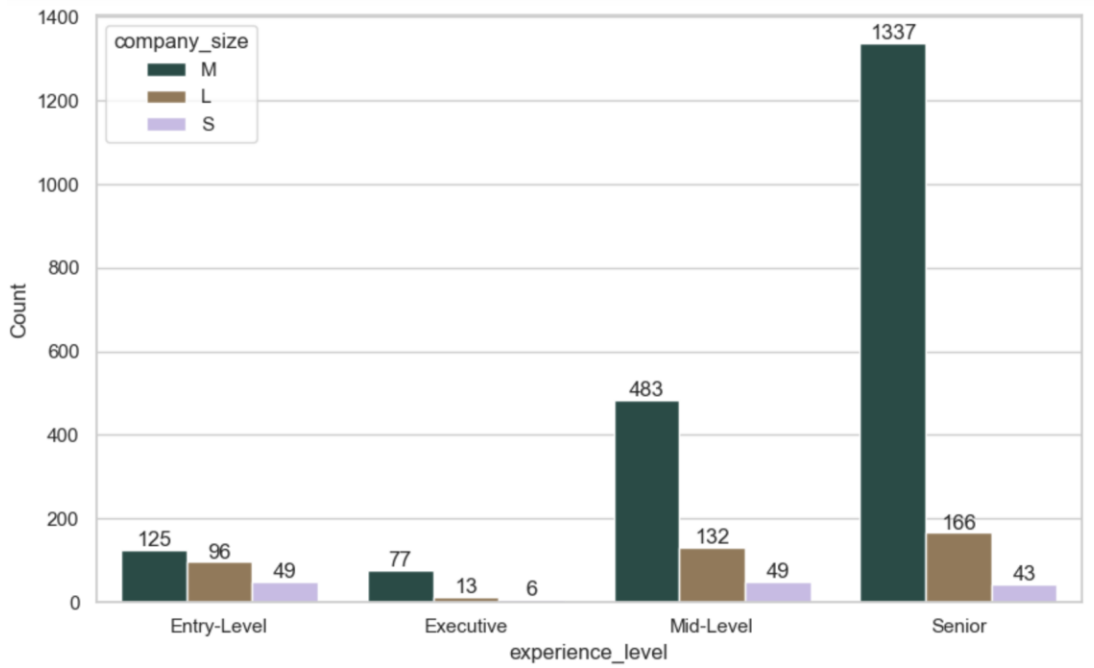
		Count
experience_level	**employment_type**	
Entry-Level	**Full-Time**	252
	Part-Time	14
	Contractor	2
	Freelancer	2
Experienced	**Full-Time**	95
	Contractor	1
Mid-Level	**Full-Time**	651
	Contractor	5
	Freelancer	5
	Part-Time	3
Senior	**Full-Time**	1541
	Freelancer	3
	Contractor	2

Mid-Sized Companies :
Seniors > Mid-level > Entry-Level > Executives

Large Sized Companies:
Seniors > Mid-Level > Entry-Level > Executives

Small-Sized Companies:
Entry-Level = Mid-Level > Senior > Executives

```
sns.set(style="whitegrid")

# Plot the grouped bar chart
plt.figure(figsize=(10, 6))  # Adjust the figure size as needed
ax = sns.barplot(x='experience_level', y='Count', hue='company_size',
                 palette="cubehelix",data=exp_size_df.reset_index())
#plt.show()

# Add values above bars
for i in ax.containers:
    ax.bar_label(i,)
```



Employment types and experience: Do employment types have a dependency on experience?

```
# counting the value of each experience level for employment type
employment_vs_experience = ds_salaries_df.groupby('experience_level')['employment_type'].value_counts()
employment_vs_experience
```

```
employment_vs_experience_dt=pd.DataFrame(employment_vs_experience )

employment_vs_experience_dt.rename(columns = {'employment_type':'Count'}, inplace = True)
employment_vs_experience_dt
```

experience_level	employment_type	Count
Entry-Level	Full-Time	252
	Part-Time	14
	Contractor	2
	Freelancer	2
Experienced	Full-Time	95
	Contractor	1
Mid-Level	Full-Time	651
	Contractor	5
	Freelancer	5
	Part-Time	3
Senior	Full-Time	1541
	Freelancer	3
	Contractor	2

We are using a chi-square test as we are trying to find correlation between two categorical variables.

```
from scipy.stats import chi2_contingency
```

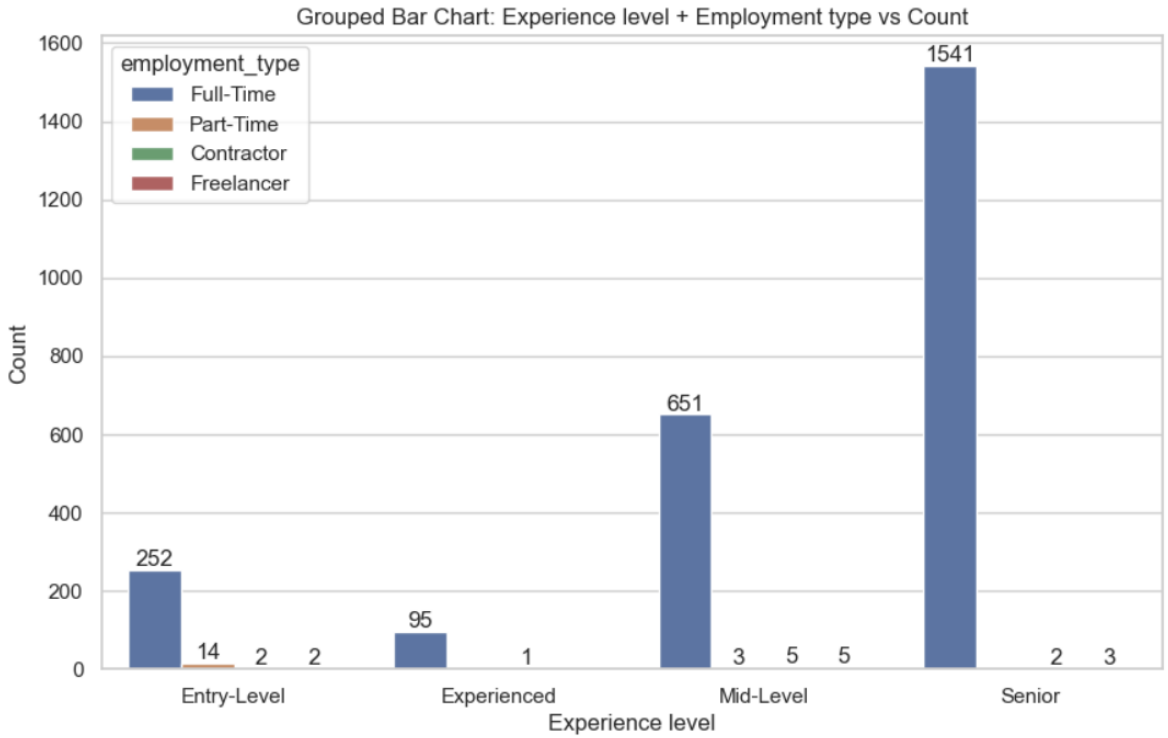
```
cross_tab = pd.crosstab(index=ds_salaries_df['experience_level'],columns=ds_salaries_df['employment_type'])
chi_square_result = chi2_contingency(cross_tab,)
chi_square_result
```

```
Chi2ContingencyResult(statistic=107.89611391840003, pvalue=3.9399062330006367e-19, dof=9, expected_freq=array([[1.0
4813665e+00, 1.04813665e+00, 2.66121894e+02, 1.78183230e+00],
       [3.72670807e-01, 3.72670807e-01, 9.46211180e+01, 6.33540373e-01],
       [2.57763975e+00, 2.57763975e+00, 6.54462733e+02, 4.38198758e+00],
       [6.00155280e+00, 6.00155280e+00, 1.52379425e+03, 1.02026398e+01]]))
```

H0 - The two columns are not correlated ;
H1 - The two columns are correlated
Result - The probability of H0 being true
In this case, since p-value is very less (in comparison of alpha, say 0.05), we will reject the H0 and conclude that the two columns are correlated

Employment types and experience: Do employment types have a dependency on experience?

```
sns.set(style="whitegrid")

# Plot the grouped bar chart
plt.figure(figsize=(10, 6))  # Adjust the figure size as needed
ax = sns.barplot(x='experience_level', y='Count', hue='employment_type',
                 data=employment_vs_experience_dt.reset_index())

# Add labels and title
plt.xlabel('Experience level')
plt.ylabel('Count')
plt.title('Grouped Bar Chart: Experience level + Employment type vs Count')

# Add values above bars
for j in ax.containers:
    ax.bar_label(j,)
# Show the plot
plt.show()
```



Majority of companies offer full time roles for data science jobs, especially for senior level roles.

Finding correlation between numerical values of the dataset

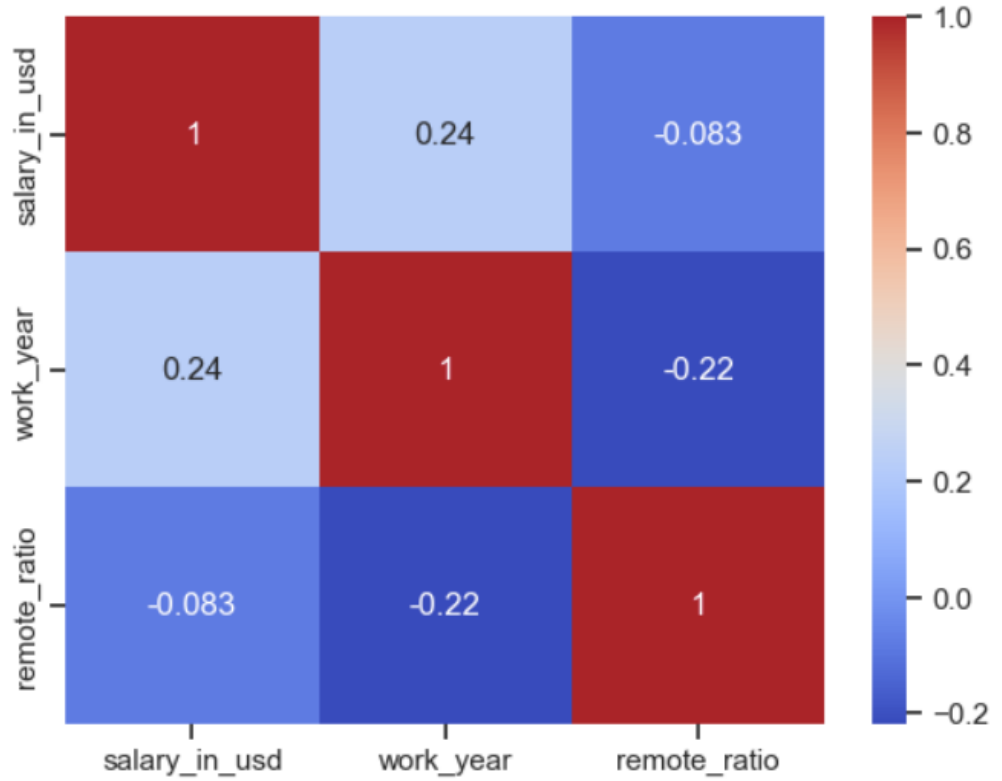
```
ds_salaries_df_cor = ds_salaries_df[['salary_in_usd','work_year','remote_ratio']]

#changing the work year into int
ds_salaries_df_cor['work_year'] = ds_salaries_df_cor['work_year'].astype(int)

#finding the correlation data
corr = ds_salaries_df_cor.corr()

corr
```

	salary_in_usd	work_year	remote_ratio
salary_in_usd	1.000000	0.235672	-0.082760
work_year	0.235672	1.000000	-0.217165
remote_ratio	-0.082760	-0.217165	1.000000

Remote ratio and work year are weak predictors for salary (in USD), given their correlation ratio is quite low: 0.24 and -0.083.

```
sns.heatmap(corr, annot=True, cmap='coolwarm')
ax[0].set_title('Continuous-Continuous Correlation')
```

```
Text(0.5, 1.0, 'Continuous-Continuous Correlation')
```



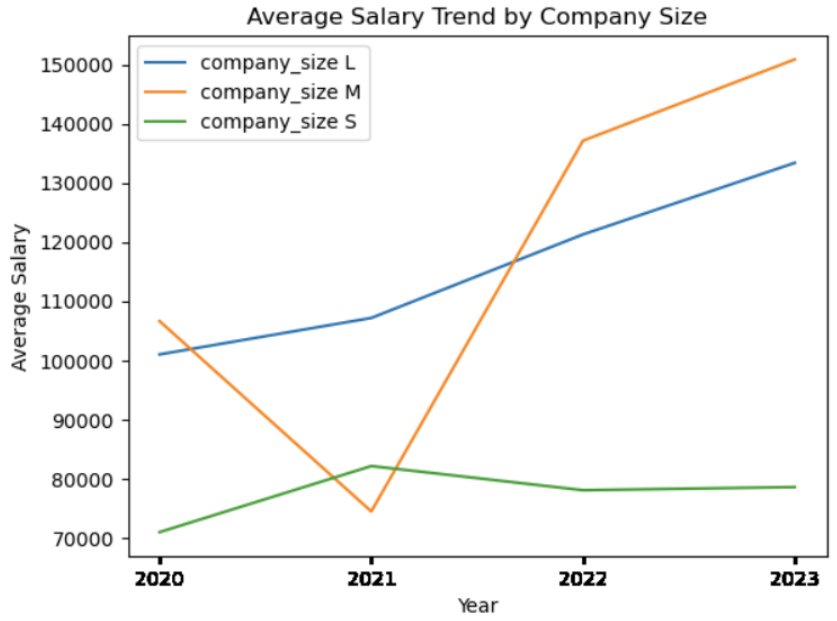
Compensation trends: Are organizations staying up-to-date with the compensation trends in the industry?

```
#Group data by Company Size and Year and calculate the average salaries
avg_salary_by_company_size = ds_salaries.groupby(['company_size', 'work_year'])['salary_in_usd'].mean().reset_index(

print("Average Salary Trend by Company Size")
avg_salary_by_company_size
```

Average Salary Trend by Company Size

	company_size	work_year	salary_in_usd
0	L	2020	101000.371429
1	L	2021	107165.700787
2	L	2022	121281.959391
3	L	2023	133379.400000
4	M	2020	106626.437500
5	M	2021	74463.035714
6	M	2022	137132.464968
7	M	2023	150867.202038
8	S	2020	70958.560000
9	S	2021	82129.446809
10	S	2022	78050.851852
11	S	2023	78579.772727

```
#Plot the Average Salary Trend by Company Size Over the Years
import matplotlib.pyplot as plt

x_labels = ds_salaries['work_year']

for size in ['L', 'M', 'S']:
    subset = avg_salary_by_company_size[avg_salary_by_company_size['company_size'] == size]
    plt.plot(subset['work_year'], subset['salary_in_usd'], label =f'company_size {size}')

plt.xlabel('Year')
plt.ylabel('Average Salary')
plt.title('Average Salary Trend by Company Size')
plt.xticks(x_labels)
plt.legend()

plt.show()
```



The overall average salary trend by company size seems to be **gradually increasing** over the years. However, company size S and M had a spike and a drop in 2021.

Compensation trends: What kind of talent are companies looking for based on the highest paying job title?

```python
#Group data by job title and calculating average salary
avg_salary = ds_salaries.groupby('job_title')['salary_in_usd'].mean()

print("Average Salaries of Each Job Title")
avg_salary
```

```
Average Salaries of Each Job Title

job_title
3D Computer Vision Researcher     21352.250000
AI Developer                     136666.090909
AI Programmer                     55000.000000
AI Scientist                     110120.875000
Analytics Engineer               152368.631068
                                        ...
Research Engineer                163108.378378
Research Scientist               161214.195122
Software Data Engineer            62510.000000
Staff Data Analyst                15000.000000
Staff Data Scientist             105000.000000
Name: salary_in_usd, Length: 93, dtype: float64
```

```python
#Find the top 10 highest paying job titles
top_paying_jobs = avg_salary.sort_values(ascending=False).head(10)

print("Top 10 Highest Paying Job Titles:")
top_paying_jobs
```

```
Top 10 Highest Paying Job Titles:

job_title
Data Science Tech Lead                375000.000000
Cloud Data Architect                  250000.000000
Data Lead                             212500.000000
Data Analytics Lead                   211254.500000
Principal Data Scientist              198171.125000
Director of Data Science              195140.727273
Principal Data Engineer               192500.000000
Machine Learning Software Engineer    192420.000000
Data Science Manager                  191278.775862
Applied Scientist                     190264.482759
Name: salary_in_usd, dtype: float64
```

First, we need to calculate the average salaries of each job title to find the highest paying job title.

Then, retrieve the information about the highest paying job title from the dataset.

```python
# Getting the rows corresponding to the highest paying job title
most_popular_job = ds_salaries[ds_salaries['job_title'] == 'Data Science Tech Lead']

print("The Highest Paying Job Title")
most_popular_job
```

The Highest Paying Job Title

_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remote_ratio	company_location	company_size
2022	SE	FT	Data Science Tech Lead	375000	USD	375000	US	50	US	L

<u>Data Science Tech Lead</u>

Experience level: Senior

Employment tyle: Full time

Employee residence: US

Company size: L

Salary trends over the years: Can we identify trends in salaries over time, such as year-over-year growth or fluctuations?

```
#Calculate the average salary each year
avg_salary_by_year = ds_salaries.groupby('work_year')['salary_in_usd'].mean()
avg_salary_by_year
```

```
work_year
2020     92302.631579
2021     94087.208696
2022    133338.620793
2023    149045.541176
Name: salary_in_usd, dtype: float64
```

```
#Plot the average salary by year using a line graph
import matplotlib.pyplot as plt

plt.figure(figsize=(10, 6))
avg_salary_by_year.plot(marker='o', color='b')
plt.title('Overall Salary Trends Over The Years')
plt.xlabel('Year')
plt.ylabel('Average Salary')
plt.xticks(x_labels)
plt.grid(True)

plt.show()
```

The overall salary trend over the years had the most significant increase in 2022, suggesting that data science is on the rise.

Salary trends over the years: What does the salary trend for **Data Analysts** look like over the years?

```
#Get all data entrys for Data Analysts
data_analyst = ds_salaries[ds_salaries['job_title'] == 'Data Analyst']

data_analyst.head()
```

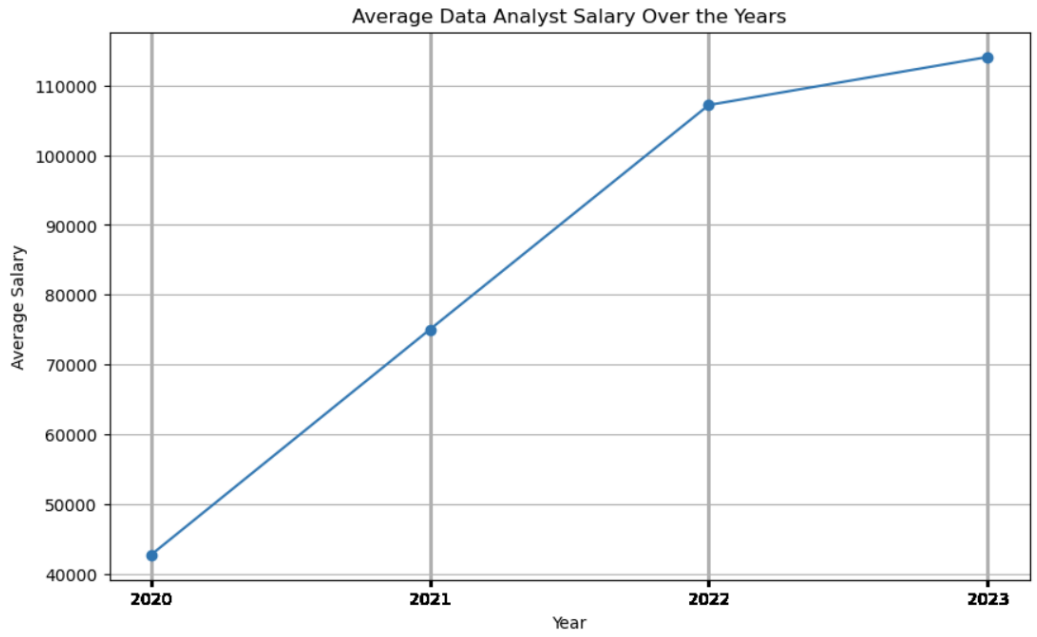
	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remote_ratio	c
11	2023	SE	FT	Data Analyst	130000	USD	130000	US	100	
12	2023	SE	FT	Data Analyst	100000	USD	100000	US	100	
19	2023	MI	FT	Data Analyst	150000	USD	150000	US	100	
20	2023	MI	FT	Data Analyst	110000	USD	110000	US	100	
37	2023	MI	FT	Data Analyst	105380	USD	105380	US	0	

```
#Calculate the average salaries by year
da_avg_salary_over_years = data_analyst.groupby(['work_year'])['salary_in_usd'].mean().reset_index()
da_avg_salary_over_years
```

	work_year	salary_in_usd
0	2020	42705.000000
1	2021	75024.952381
2	2022	107207.398551
3	2023	114097.475570

The salary trend for Data Analysts significantly increased over the years, doubling from 2020 to 2022.

```
#Plot Data Analyst's average salary over the years using a line graph

plt.figure(figsize=(10, 6))
plt.plot(da_avg_salary_over_years['work_year'], da_avg_salary_over_years['salary_in_usd'], marker='o', linestyle='-'
plt.title('Average Data Analyst Salary Over the Years')
plt.xlabel('Year')
plt.ylabel('Average Salary')
plt.xticks(x_labels)
plt.grid(True)
plt.show()
```



Salary trends over the years: What does the salary trend for **BI (Data) Analyst** look like over the years?

```
#Get all data entrys for BI Analysts and BI Data Analysts
BI_analyst = ['BI Analyst', 'BI Data Analyst']
BI_da_analyst = ds_salaries[ds_salaries['job_title'].isin(BI_analyst)]

BI_da_analyst.head()
```

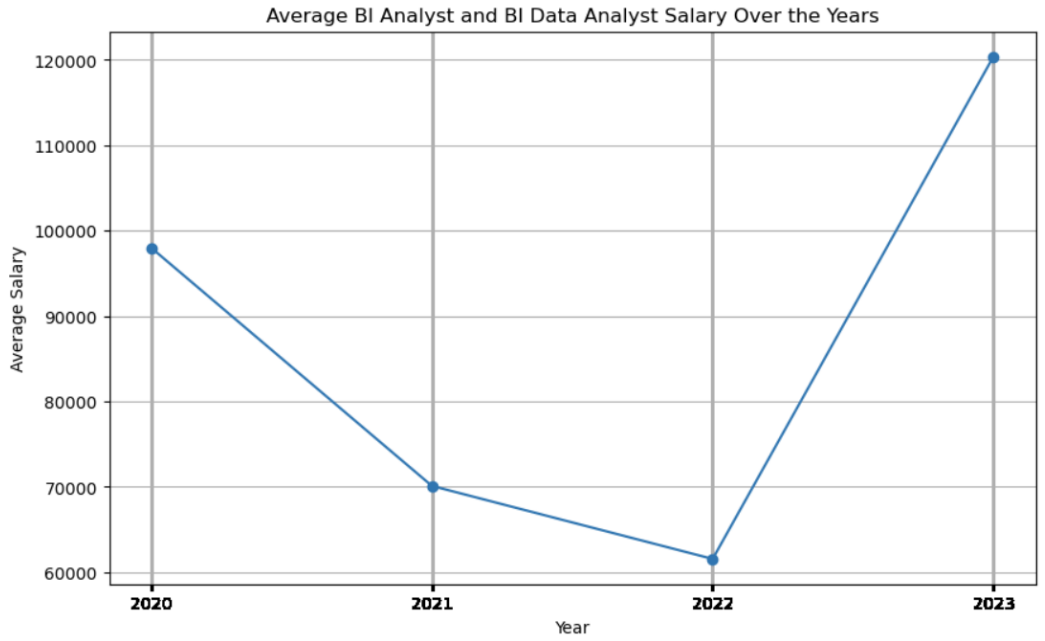
	work_year	experience_level	employment_type	job_title	salary	salary_currency	salary_in_usd	employee_residence	remote_ratio	c
808	2023	SE	FT	BI Analyst	160000	USD	160000	US	0	
809	2023	SE	FT	BI Analyst	135000	USD	135000	US	0	
984	2023	SE	FT	BI Data Analyst	67000	EUR	71897	DE	100	
1521	2023	SE	FT	BI Analyst	125000	USD	125000	US	0	
1522	2023	SE	FT	BI Analyst	110000	USD	110000	US	0	

```
#Calculate the average salaries by year
BI_avg_salary_over_years = BI_da_analyst.groupby(['work_year'])['salary_in_usd'].mean().reset_index()
BI_avg_salary_over_years
```

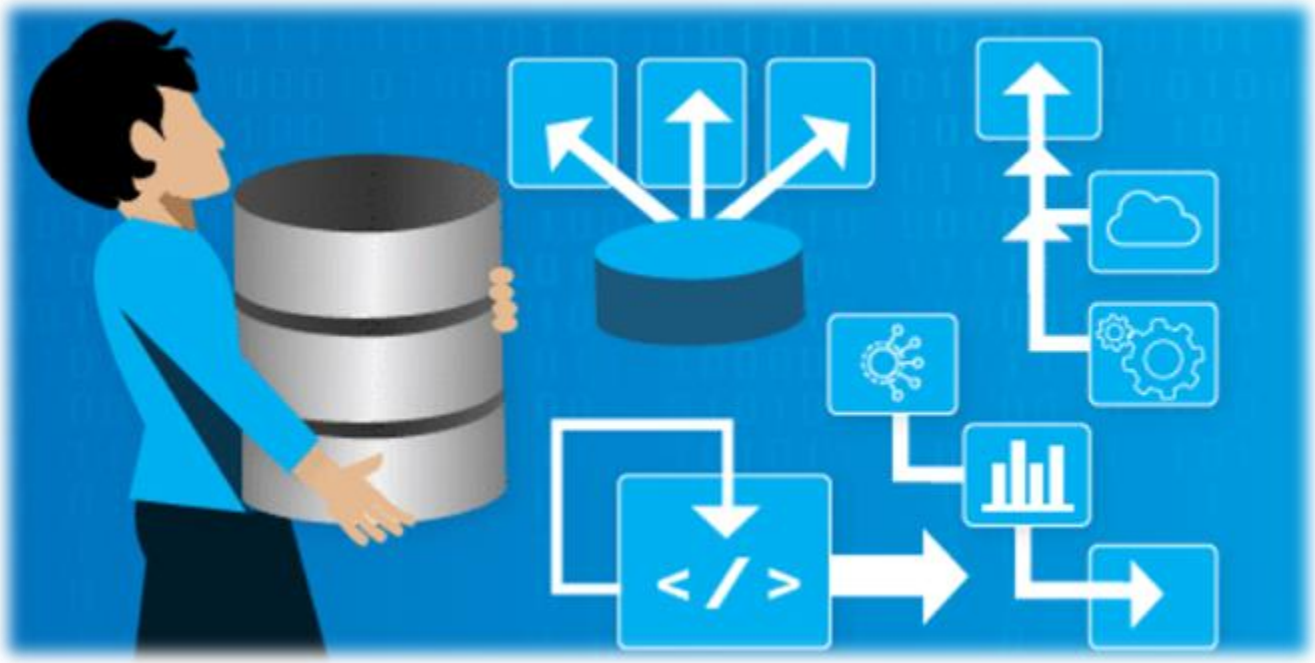
	work_year	salary_in_usd
0	2020	98000.000000
1	2021	70106.200000
2	2022	61551.846154
3	2023	120379.400000

There's an interesting salary trend for **BI Analysts and BI Data Analysts** over the years, declining from 2020 to 2022 and increasing from 2022 to 2023.

```
#Plot BI (Data) Analyst's average salary over the years using a line graph

plt.figure(figsize=(10, 6))
plt.plot(BI_avg_salary_over_years['work_year'], BI_avg_salary_over_years['salary_in_usd'], marker='o', linestyle='-'
plt.title('Average BI Analyst and BI Data Analyst Salary Over the Years')
plt.xlabel('Year')
plt.ylabel('Average Salary')
plt.xticks(x_labels)
plt.grid(True)
plt.show()
```

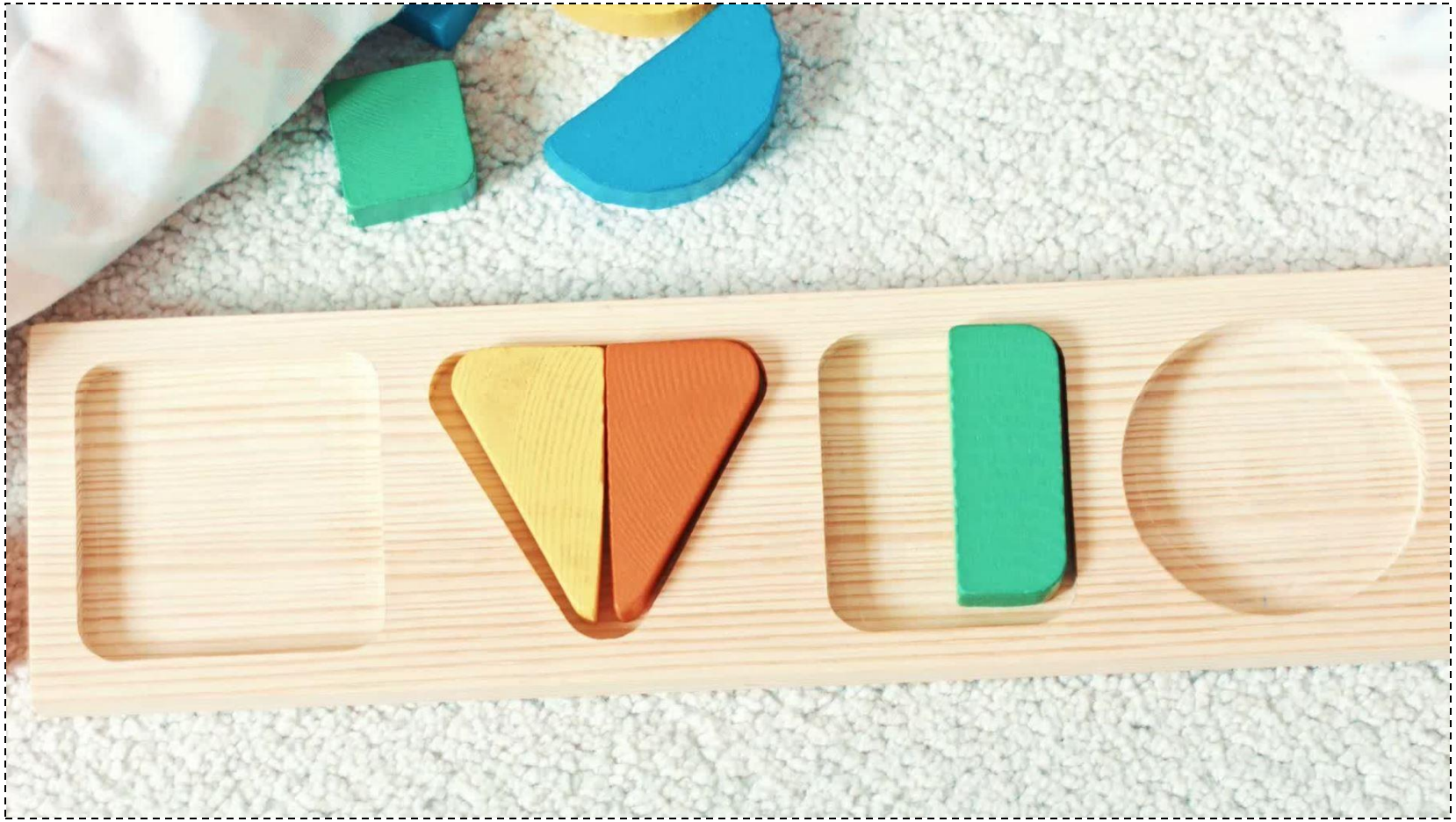


DATA MODELING



- **Chosen model:** Multiple linear regression
- **Purpose:** Predict average salaries for different job titles in the **US** for **2024**
- **Benefit:** Understand how categorical factors influence salaries
- **Insights:** Provide clear understanding of predictors' impact on average salary
- **Aim:** Assist classmates in making informed career decisions by considering all factors during their 2024 job hunt in the US

STEPS FOR CREATING THE MODEL



1. PREPARE DATA

```
# Filter data for US company locations and the required years (2020-2023)
us_predict_data = ds_salaries_df[
    (ds_salaries_df['company_location'] == 'US') &
    ((ds_salaries_df['work_year'] == '2020') |
     (ds_salaries_df['work_year'] == '2021') |
     (ds_salaries_df['work_year'] == '2022') |
     (ds_salaries_df['work_year'] == '2023'))
]

# Group by job title, work_year, and other specified columns to calculate average salary
grouped_data = us_predict_data.groupby(['job_title', 'work_year', 'experience_level', 'employment_type',
                                        'employee_residence', 'remote_ratio', 'company_location'])

# Calculate the mean salary for each group
avg_predict_salary = grouped_data['salary_in_usd'].mean().reset_index()

avg_predict_salary
```

We filtered data exclusively for US company locations and then found out the average salaries of the various job titles identified in US.

	job_title	work_year	experience_level	employment_type	employee_residence	remote_ratio	company_location	salary_in_usd
0	AI Developer	2023	Mid-Level	Full-Time	US	Fully Remote	US	200000.000000
1	AI Scientist	2021	Entry-Level	Part-Time	BR	Fully Remote	US	12000.000000
2	AI Scientist	2021	Entry-Level	Part-Time	PK	Fully Remote	US	12000.000000
3	AI Scientist	2022	Entry-Level	Full-Time	US	Fully Remote	US	50000.000000
4	AI Scientist	2022	Experienced	Full-Time	US	Fully Remote	US	200000.000000
...	...	...	...	...	...	...	...	...
320	Research Scientist	2023	Mid-Level	Full-Time	US	Fully Remote	US	193633.333333
321	Research Scientist	2023	Mid-Level	Full-Time	US	In-person	US	116250.000000

2. MULTIPLE LINEAR REGRESSION USING SKLEARN

```
# Create a feature for the year 2024
avg_predict_salary['work_year'] = pd.to_numeric(avg_predict_salary['work_year'])
avg_salary_2024 = avg_predict_salary.copy()
avg_salary_2024['work_year'] = 2024

# Encoding categorical variables (job_title, work_year) with drop_first=True
avg_salary_encoded = pd.get_dummies(avg_predict_salary[['job_title', 'work_year', 'experience_level',
                                                        'employment_type', 'employee_residence',
                                                       'remote_ratio', 'company_location']],
                                    drop_first=True)
```

```
avg_predict_salary
```

	job_title	work_year	experience_level	employment_type	employee_residence	remote_ratio	company_location	salary_in_usd
0	AI Developer	2023	Mid-Level	Full-Time	US	Fully Remote	US	200000.000000
1	AI Scientist	2021	Entry-Level	Part-Time	BR	Fully Remote	US	12000.000000
2	AI Scientist	2021	Entry-Level	Part-Time	PK	Fully Remote	US	12000.000000
3	AI Scientist	2022	Entry-Level	Full-Time	US	Fully Remote	US	50000.000000
4	AI Scientist	2022	Experienced	Full-Time	US	Fully Remote	US	200000.000000
...	...	...	...	...	...	...	...	...
320	Research Scientist	2023	Mid-Level	Full-Time	US	Fully Remote	US	193633.333333
321	Research Scientist	2023	Mid-Level	Full-Time	US	In-person	US	116250.000000
322	Research Scientist	2023	Senior	Full-Time	US	Fully Remote	US	120000.000000
323	Research Scientist	2023	Senior	Full-Time	US	In-person	US	180548.823529
324	Staff Data Scientist	2021	Senior	Contractor	US	Fully Remote	US	105000.000000

We used one-hot encoding to handle the categorical variables.

3. FIT THE MODEL

```
# Fit the multiple linear regression model
from sklearn import linear_model
lr = linear_model.LinearRegression()
predicted = lr.fit(X=avg_salary_encoded, y=avg_predict_salary['salary_in_usd'])
```

4. PREDICT AVERAGE SALARIES FOR 2024

Reapplying one-hot encoding in the prediction phase ensures the model interprets new data ('avg_salary_2024') accurately, aligning with its learned encoding from training.

```
# Prepare data for predicting salaries in 2024
avg_salary_2024_encoded = pd.get_dummies(avg_salary_2024[['job_title', 'work_year','experience_level',
                                                          'employment_type', 'employee_residence',
                                                         'remote_ratio', 'company_location']], drop_first=True)

# Make predictions for 2024
predictions_2024 = lr.predict(avg_salary_2024_encoded)
```

```
# Create a DataFrame for the predictions
predictions_df = pd.DataFrame({
    'job_title': avg_salary_2024['job_title'],
    'work_year': avg_salary_2024['work_year'],
    'avg_salary': predictions_2024
})
```

```
# Display the predictions in a table format
predictions_df
```

This data frame displays the average salaries for 2024 for data science job titles based in the US.

	job_title	work_year	avg_salary
0	AI Developer	2024	193372.605697
1	AI Scientist	2024	2035.750801
2	AI Scientist	2024	-7882.182908
3	AI Scientist	2024	118344.392849
4	AI Scientist	2024	209293.998827
...	...	...	...
320	Research Scientist	2024	170997.905079
321	Research Scientist	2024	174823.005377

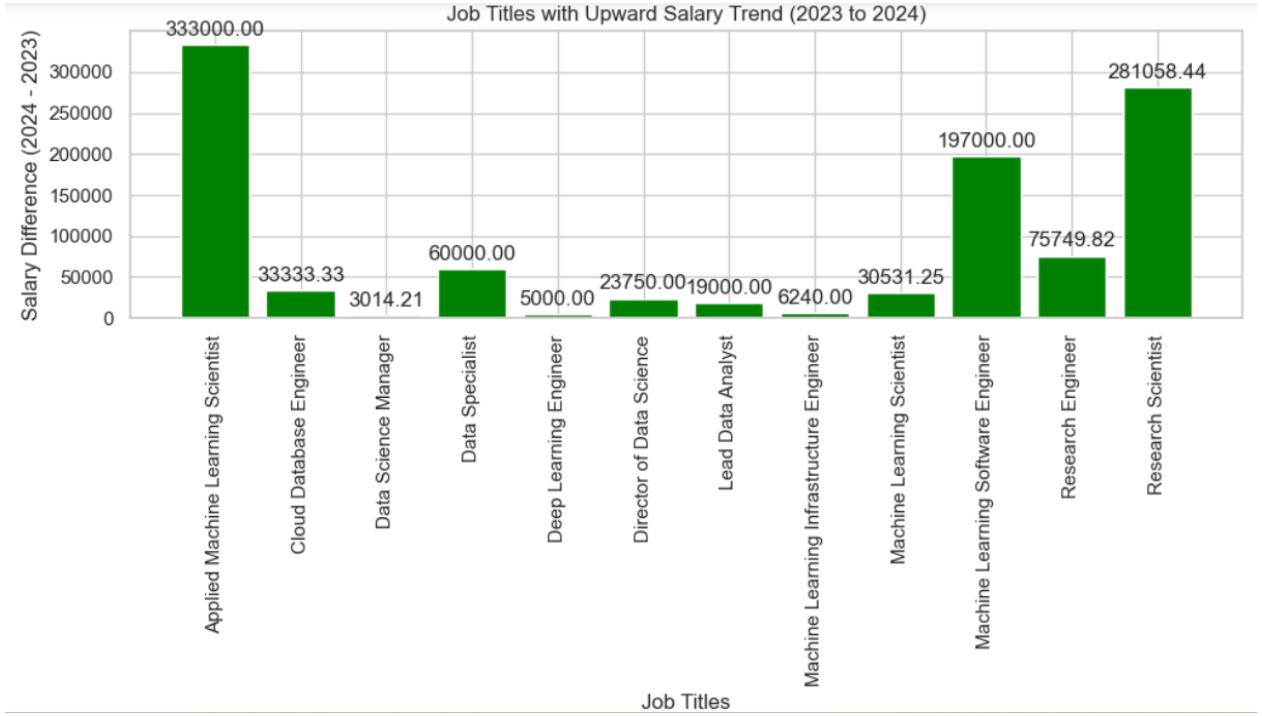
OUTPUT : PART 1

Which job titles have an upward salary trend (2023 to 2024)?

```
# Merge average salaries for 2023 and 2024 based on job titles
comparison_df = pd.merge(avg_salary_2023, cleaned_avg_salary_2024, on='job_title', how='inner')

# Display or analyze the differences between 2023 and 2024 salaries
comparison_df['salary_diff'] = comparison_df['salary_in_usd'] - comparison_df['avg_salary_2023']
comparison_df
```

```
import matplotlib.pyplot as plt

# DataFrames for each trend category
upward_trend_df = comparison_df[comparison_df['salary_diff'] > 0]
downward_trend_df = comparison_df[comparison_df['salary_diff'] < 0]
constant_trend_df = comparison_df[comparison_df['salary_diff'] == 0]

# Function to add value labels to bar plots with adjusted y-coordinate
def add_value_labels(ax):
    for bar in ax.containers:
        for rect in bar:
            height = rect.get_height()
            ax.annotate(f'{height:.2f}',
                        xy=(rect.get_x() + rect.get_width() / 2, height),
                        xytext=(0, 3),  # Adjusted distance above the x-axis
                        textcoords="offset points",
                        ha='center', va='bottom')

# Plotting for upward trend
plt.figure(figsize=(10, 6))
ax1 = plt.bar(upward_trend_df['job_title'], upward_trend_df['salary_diff'], color='green')
plt.xlabel('Job Titles')
plt.ylabel('Salary Difference (2024 - 2023)')
plt.title('Job Titles with Upward Salary Trend (2023 to 2024)')
plt.xticks(rotation=90)
plt.tight_layout()
add_value_labels(plt.gca())
plt.show()
```

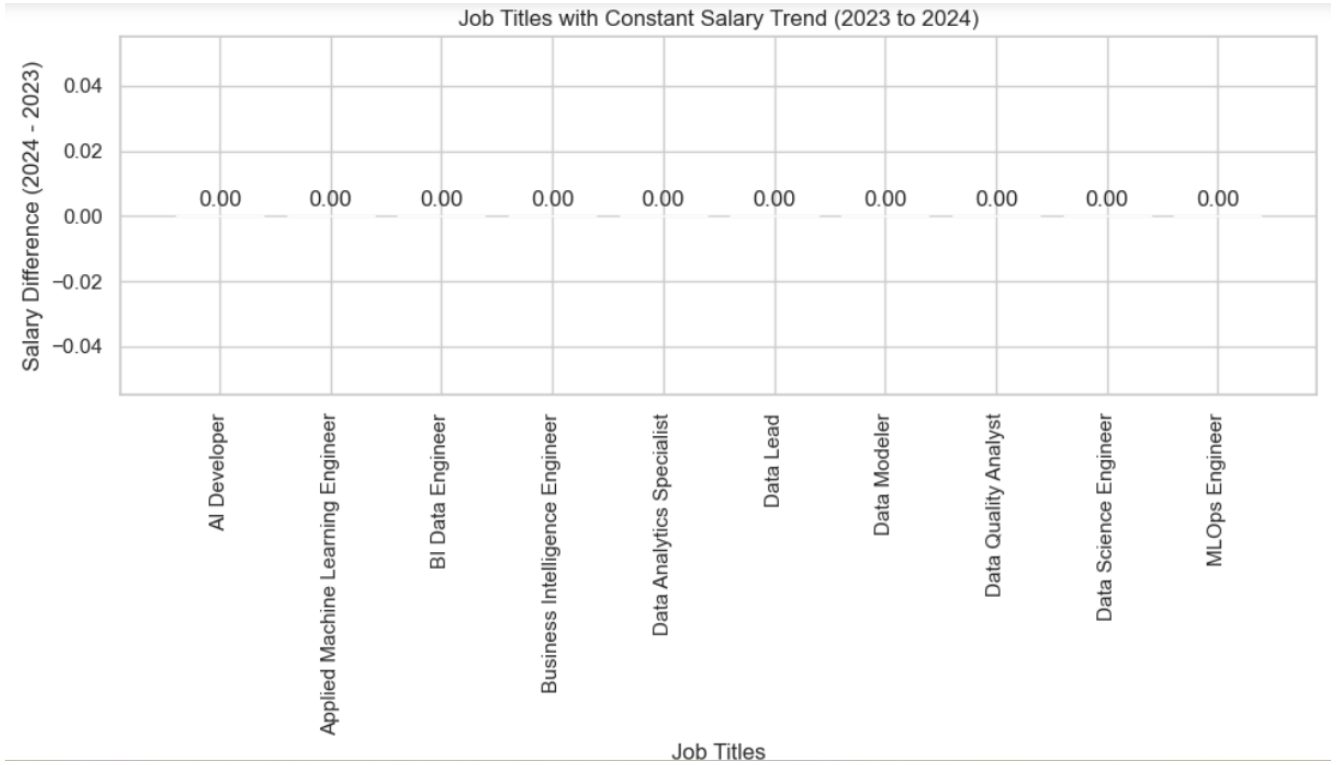
The following job titles all have an **upward salary trend** in 2024, of which Applied Machine Learning Scientist being the highest.



OUTPUT : PART 2

Which job titles have a constant salary trend from 2023 to 2024?

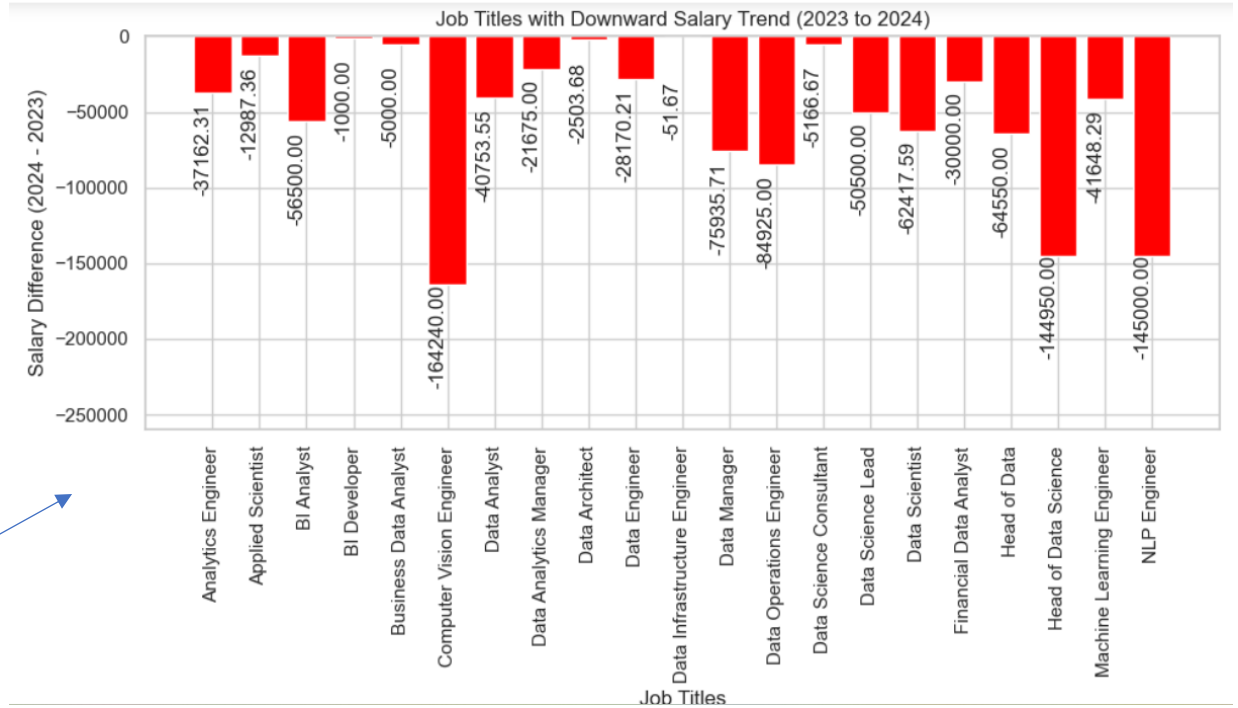
```
# Plotting for constant trend
plt.figure(figsize=(10, 6))
ax3 = plt.bar(constant_trend_df['job_title'], constant_trend_df['salary_diff'], color='blue')
plt.xlabel('Job Titles')
plt.ylabel('Salary Difference (2024 - 2023)')
plt.title('Job Titles with Constant Salary Trend (2023 to 2024)')
plt.xticks(rotation=90)
plt.tight_layout()
add_value_labels(plt.gca())
plt.show()
```



OUTPUT : PART 3

Which job titles have a downward salary trend from 2023 to 2024?

```python
# Function to add value labels with custom rotation and distance below the bars
def add_value_labels(ax, distance):
    for bar in ax.patches:
        height = bar.get_height()
        ax.annotate(f'{height:.2f}',
                    xy=(bar.get_x() + bar.get_width() / 2, height),
                    xytext=(0, -distance),  # Adjusted distance below the bar
                    textcoords="offset points",
                    rotation=90,  # Rotate the text by 45 degrees
                    ha='right',  # Horizontal alignment set to 'right' for better readability
                    va='bottom')

# Plotting for downward trend with adjusted value label position and rotation
plt.figure(figsize=(10, 6))
ax2 = plt.bar(downward_trend_df['job_title'], downward_trend_df['salary_diff'], color='red')
plt.xlabel('Job Titles')
plt.ylabel('Salary Difference (2024 - 2023)')
plt.title('Job Titles with Downward Salary Trend (2023 to 2024)')
plt.xticks(rotation=90)
plt.tight_layout()
add_value_labels(plt.gca(), 60)  # Adjusted distance of 6 cm (60 points) below each bar
plt.ylim(-260000, plt.gca().get_ylim()[1])  # Set y-axis limits from -260000 to current maximum
plt.show()
```



The following job titles all have a **downward salary trend** in 2024, of which Computer Vision Engineer being the lowest.

NOTE: Some bars may not be visible because the salary difference is lower than others.

CONCLUSION

- **Job Title Selection:** Students are recommended to pursue common and high-paying job titles to optimize their career.

- **Influential Factors:** The main factors that have influence on compensation are company location, company size, and experience level.

- **Salary Trends:** For 2024, our predictive model suggests that there could be an overall increase in salary trends in data science job titles such as Research Scientist and Applied Machine Learning Scientist. It is important to note that certain job titles might experience downward trends or no change at all.


THANK
YOU