DS Job Market Analysis

Bridget, Ian, and Micah

Mission Statement

Our project aims to visualize data on the DS job market from 2020-2022 to analyze trends and help users make employment decisions

Our Data Set

Data_Science_Job.csv - Lifted from Kaggle

- Work Year (2020, 2021, 2022)
- Job Title
- Job Category
- Salary Currency
- Salary
- Salary in USD
- Employee Residence
- Experience Level
- Employment Type
- Work Setting
- Company Location
- Company Size

fx S

A	B	C	D	E	F	G	H	I	J
ork_year	job_title	job_category	salary_currer	salary	salary_in_usc	employee_re	experience_l	employment_	work_setting
2022	Machine Lea	Analysis	EUR	186597	136086	US	MI	CT	Remote
2020	Statistician (	ML/AI	JPY	110630	67982	JP	EX	FL	Remote
2022	Machine Lea	ML/AI	INR	61280	153309	UK	MI	CT	Hybrid
2022	Data Analyst	ML/AI	JPY	154130	135242	DE	SE	FT	Hybrid
2020	Statistician	Data Science	EUR	172312	35156	UK	MI	FT	In-person
2020	Machine Lea	Engineering	JPY	36544	68280	CN	MI	FT	Hybrid
2022	Data Analyst	Data Science	JPY	178404	105324	DE	EX	PT	Remote
2021	Data Scienti	ML/AI	JPY	187908	90706	UK	EX	CT	Remote
2022	Data Analyst			-44388	171043	UK		FL	In-person
2022	Statistician (	Engineering	us dolars	31694	73408	DE	EN	CT	Remote
2022	Data Enginee	Data Science	us dolars	157727	167559	US	SE	FL	Hybrid
2022	Data Scienti	Data Science	INR	118202	50997	MX	EX	FL	In-person
2020	Machine Learning Engineer		(Remote)	143057	76456	MX		FL	In-person
2022	Statistician	Engineering	us dolars	79294	160865	JP	EN	CT	In-person
2021	Data Enginee	Analysis	JPY	68769	107988	JP	EN	FT	Hybrid
2020	Data Analyst	Engineering	JPY	143289	110908	MX	EX	PT	In-person
2021	Data Enginee	Analysis	JPY	132885	107952	US	EX	PT	In-person
2021	Statistician (	ML/AI	JPY	67855	114784	UK	SE	FT	Remote
2021	Statistician	ML/AI	GBP	162532	189266	JP	EX	PT	Remote
2021	Data Enginee	Data Science	INR	77646	194857	US	EX	FL	In-person
2020	Data Analyst	ML/AI	EUR	170907	88676	US	EX	PT	Hybrid
2020	Data Enginee	ML/AI	EUR	143947	38373	JP	SE	CT	Hybrid
2021	Statistician	Analysis	us dolars	40910	191355	MX	MI	FT	Hybrid
2021	Statistician			-112757	124951	DE		FT	Hybrid
2020	Machine Lea	Engineering	EUR	38463	56414	IN	EN	FT	Remote
2020	Machine Lea	Data Science	INR	174408	175450	MX	MI	FT	Hybrid
2020	Data Scientist			56534	147139	DE		CT	Hybrid
2022	Statistician (	Engineering	us dolars	163441	113872	MX	SE	FL	In-person
2022	Machine Lea	Data Science	INR	186453	165772	UK	SE	FL	Hybrid
2022	Data Engineer			-142296	174973	DE		FL	Hybrid
2021	Data Analyst	Data Science	GBP	115094	120374	CN	MI	PT	Hybrid
2022	Data Enginee	Data Science	us dolars	117905	183669	IN	MI	FL	Hybrid
2021	Data Analyst	Analysis	EUR	55407	194500	MX	EN	PT	Remote
2021	Machine Learning Engineer			-92552	196759	IN		CT	In-person
2022	Data Scienti	ML/AI	EUR	197011	59493	US	MI	PT	Remote
2021	Data Enginee	ML/AI	JPY	122303	155074	CN	EX	FL	In-person
2022	Data Enginee	Analysis	GBP	187242	81224	UK	MI	FT	In-person
2022	Data Enginee	Engineering	JPY	101908	61852	US	EX	CT	In-person
2020	Data Analyst	Engineering	EUR	154506	40932	DE	EN	FL	In-person
2022	Machine Lea	ML/AI	INR	52369	118545	DE	EX	FL	Remote

data_science_job +

eady Accessibility: Unavailable

Project Objectives

Analyze Salary Trends and Distributions

- Examine salary variations by experience level, location, job title, and category.
- Identify wage premiums and outliers to uncover key insights.

Visualize Insights Effectively

- Create intuitive, comprehensive graphs and dynamic visualizations for easy interpretation.

Enable Decision-Making

- Provide actionable insights for job seekers, employers, and policymakers to understand and navigate the data science job market.

Methodology

Imported Libraries

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

Used Pandas to read and import data set

```
df = pd.read_csv('data_science_job.csv')
```

Filtered Data using Pandas DataFrame.isin syntax

```
# Filtering by Experience Level
df_EN = df[df['experience_level'].isin(['EN'])]
df_SE = df[df['experience_level'].isin(['SE'])]
df_MI = df[df['experience_level'].isin(['MI'])]
```

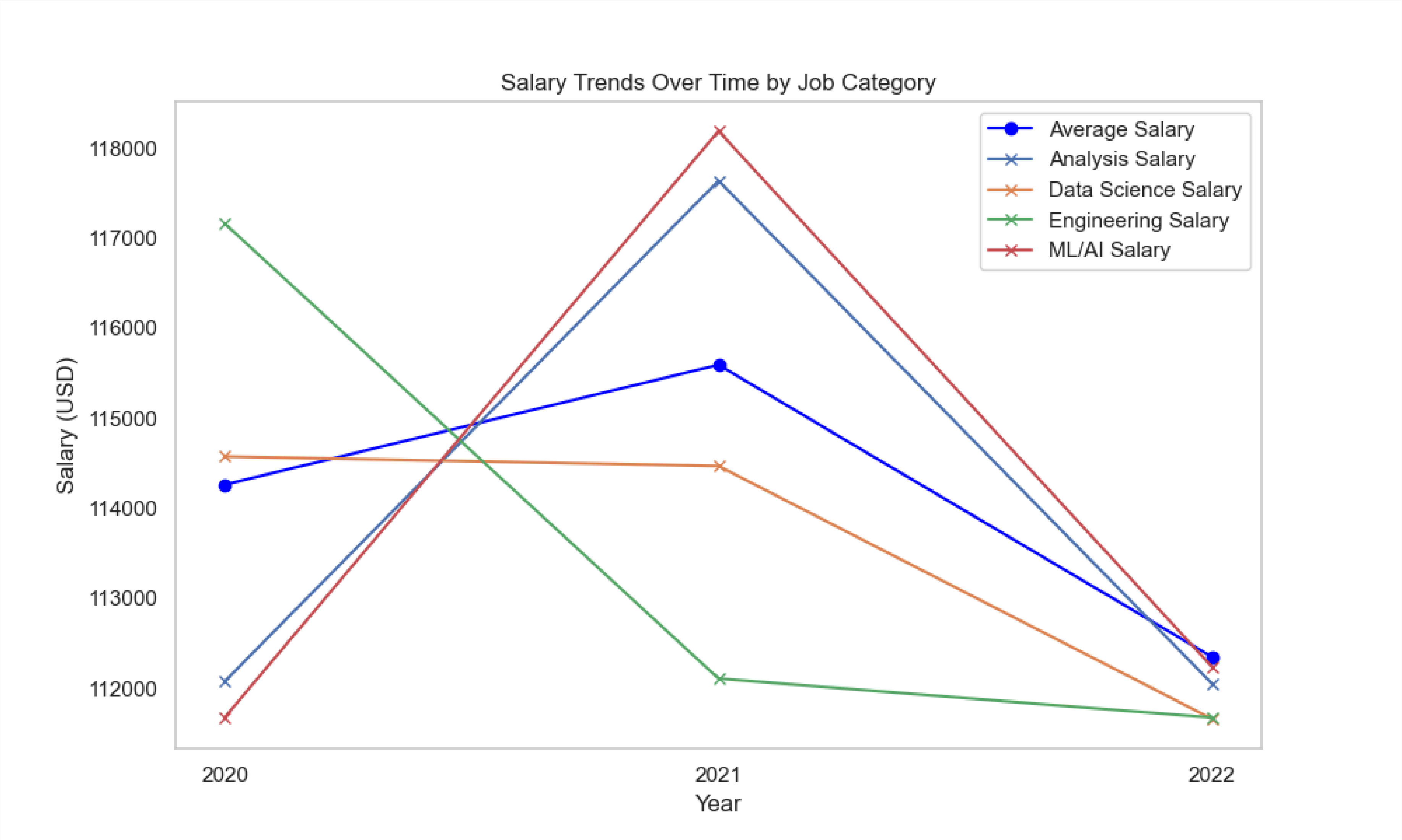
Used Pandas DataFrame.groupby syntax to group data and .mean() to calculate mean

```
# Group by job title and calculate the average salary over the years
salary_trend = df.groupby('work_year')['salary_in_usd'].mean()
```

Used matplotlib and seaborn to visualize data in various forms

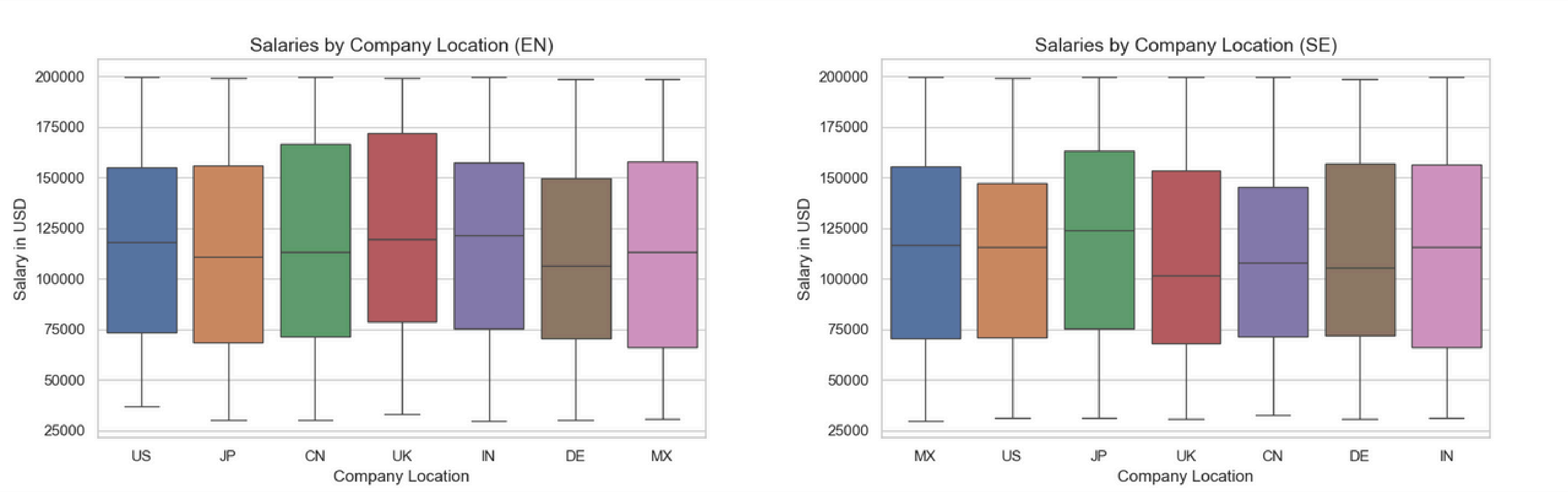
```
# Distrubtion of salaries by year
sns.set(style='whitegrid')
plt.figure(figsize=(8, 5))

sns.boxplot(x='work_year', y='salary_in_usd', data=df)

plt.title('Salaries by Work Year', fontsize=14)
plt.xlabel('Work Year', fontsize=12)
plt.ylabel('Salary in USD', fontsize=12)

plt.savefig('salary_by_work_year_boxplot.png')
plt.show()
```

Salary Over Time




```python
# Section 1: Average Salary Trends
# ----------------------------------
# Average Salary by Year
salary_trend = df.groupby('work_year')['salary_in_usd'].mean()
salary_trend_by_job = df.groupby(['work_year', 'job_category'])['salary_in_usd'].mean().unstack()


plt.figure(figsize=(10, 6))
plt.plot(salary_trend.index, salary_trend, marker='o', label='Average Salary', color='blue')


# Plot the average salary by job category
for job_category in salary_trend_by_job.columns:
    plt.plot(salary_trend_by_job.index, salary_trend_by_job[job_category], marker='x', label=f'{job_category} Salary')


plt.title('Salary Trends Over Time by Job Category')
plt.xlabel('Year')
plt.ylabel('Salary (USD)')
plt.xticks([2020, 2021, 2022])
plt.legend(loc='best')
plt.grid()
plt.savefig('salary_trends_multiple_dimensions.png')
plt.show()
```

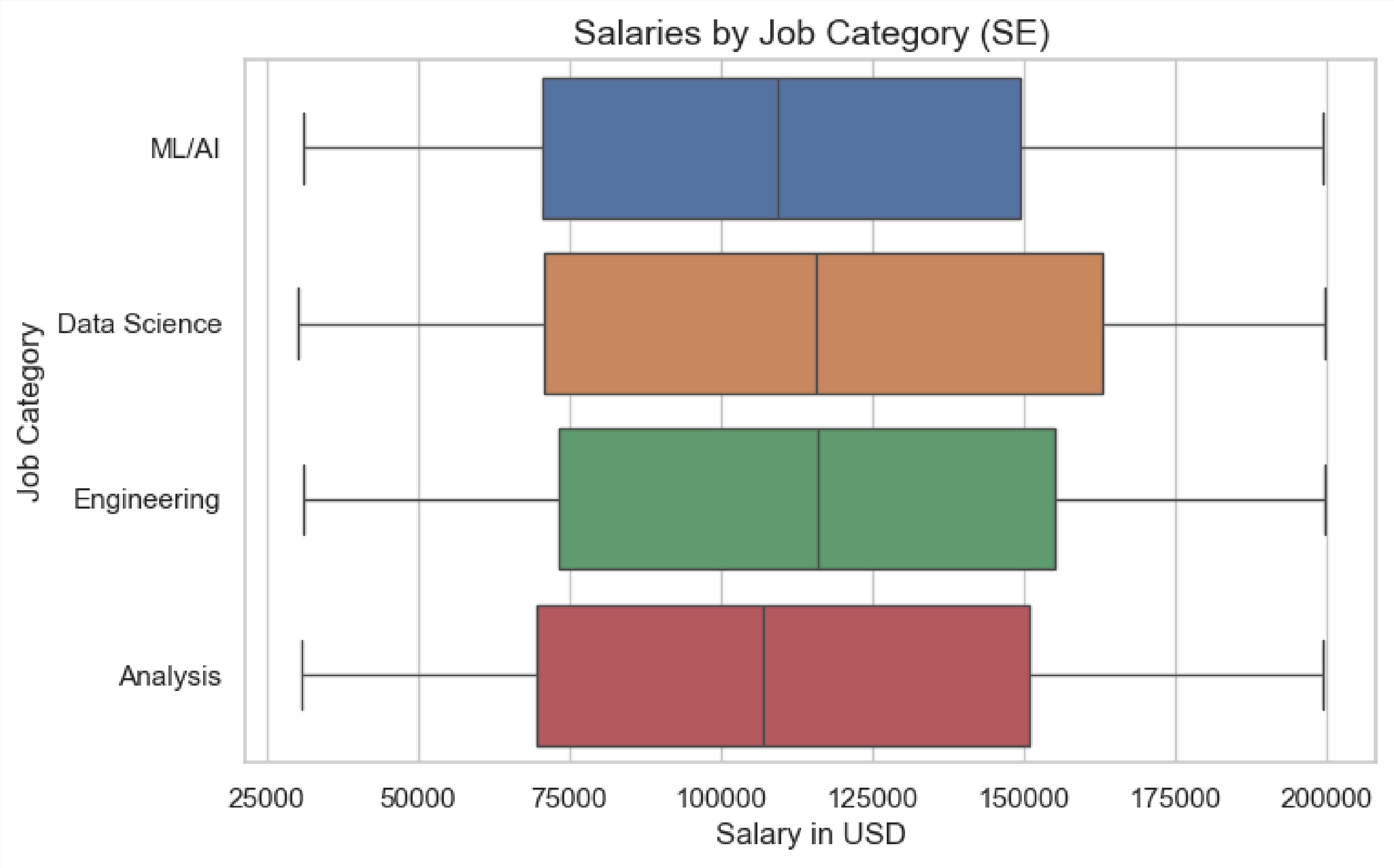
Salary Box Plots

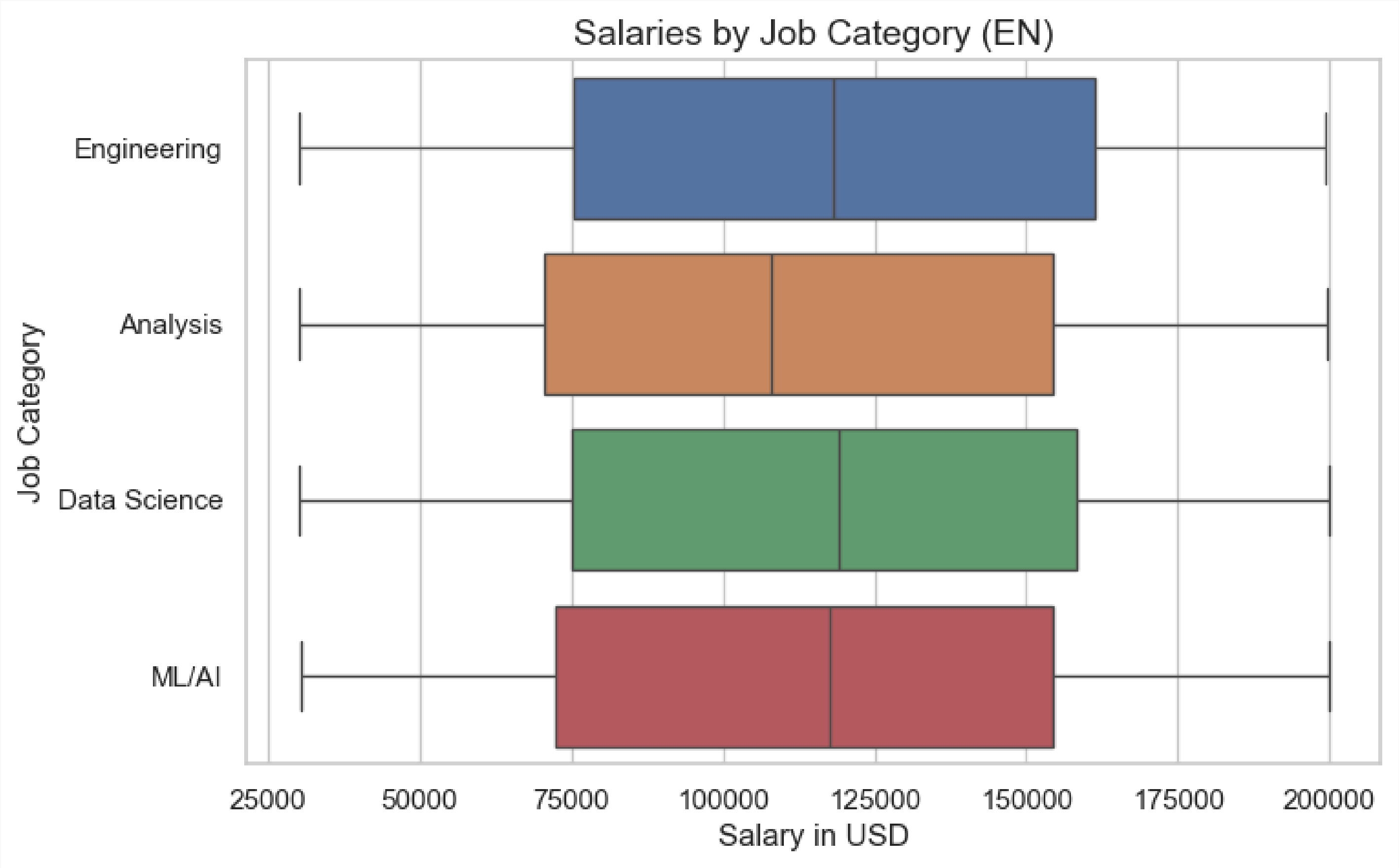



```python
# Section 4: Salary Distributions by Company Location
# ----------------------------------------------------
def plot_salary_distribution_by_location(data,title,filename):
    sns.set(style='whitegrid')
    plt.figure(figsize=(8,5))
    sns.boxplot(x = 'company_location', y = 'salary_in_usd', hue = 'company_location', data=data)
    plt.title(title, fontsize=14)
    plt.xlabel('Company Location', fontsize=12)
    plt.ylabel('Salary in USD', fontsize=12)
    plt.savefig(filename,bbox_inches = 'tight')
    plt.show()

plot_salary_distribution_by_location(df_EN, 'Salaries by Company Location (EN)', 'salaries_by_company_location_EN.png')
plot_salary_distribution_by_location(df_SE, 'Salaries by Company Location (SE)', 'salaries_by_company_location_SE.png')
```

```python
# Section 2: Salary Distributions
# ----------------------------------
# Distribution of salaries by year
sns.set(style='whitegrid')
plt.figure(figsize=(8, 5))
sns.boxplot(x='work_year', y='salary_in_usd', data=df)
plt.title('Salaries by Work Year', fontsize=14)
plt.xlabel('Work Year', fontsize=12)
plt.ylabel('Salary in USD', fontsize=12)
plt.savefig('salary_by_work_year_boxplot.png')
plt.show()
```

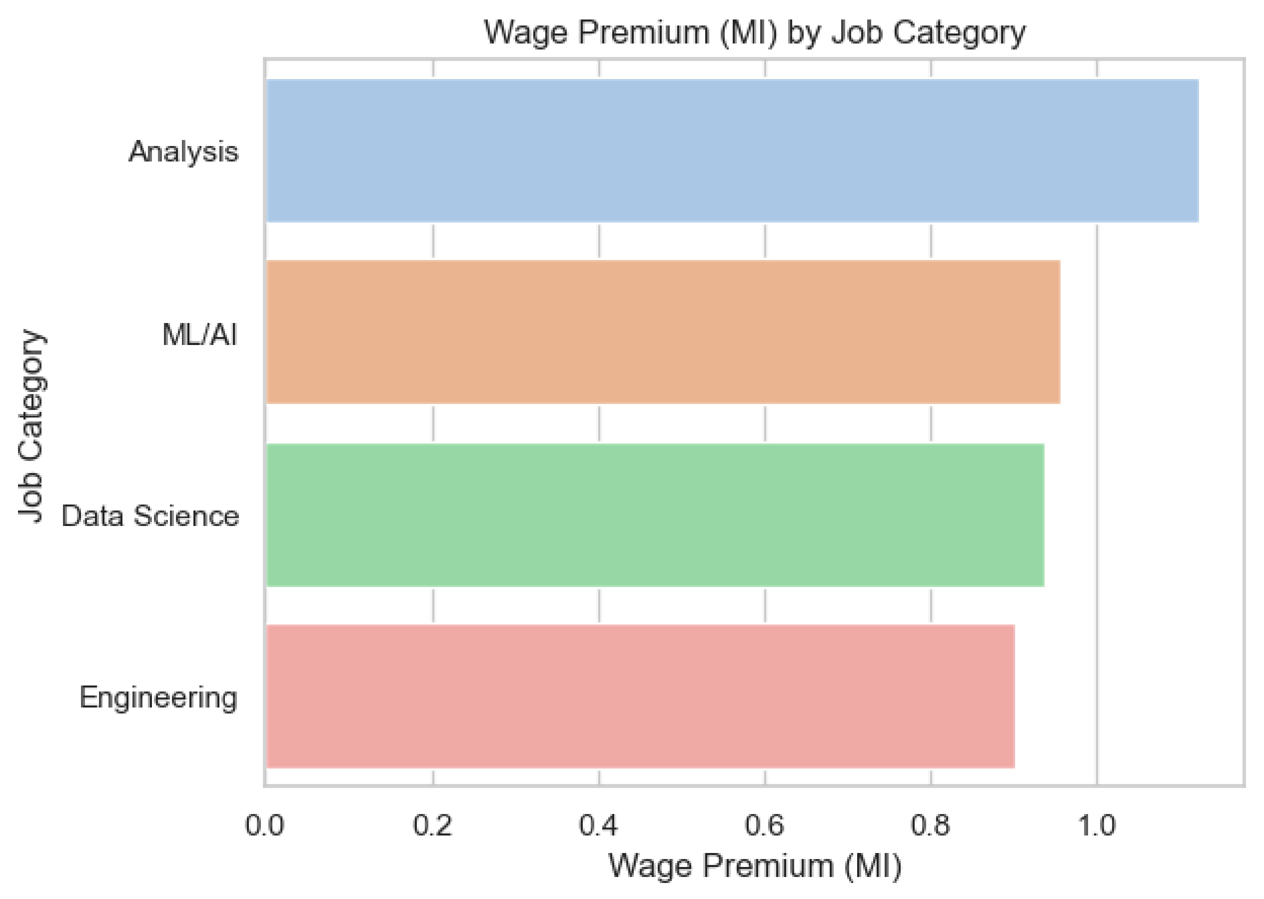
Salary Box Plots



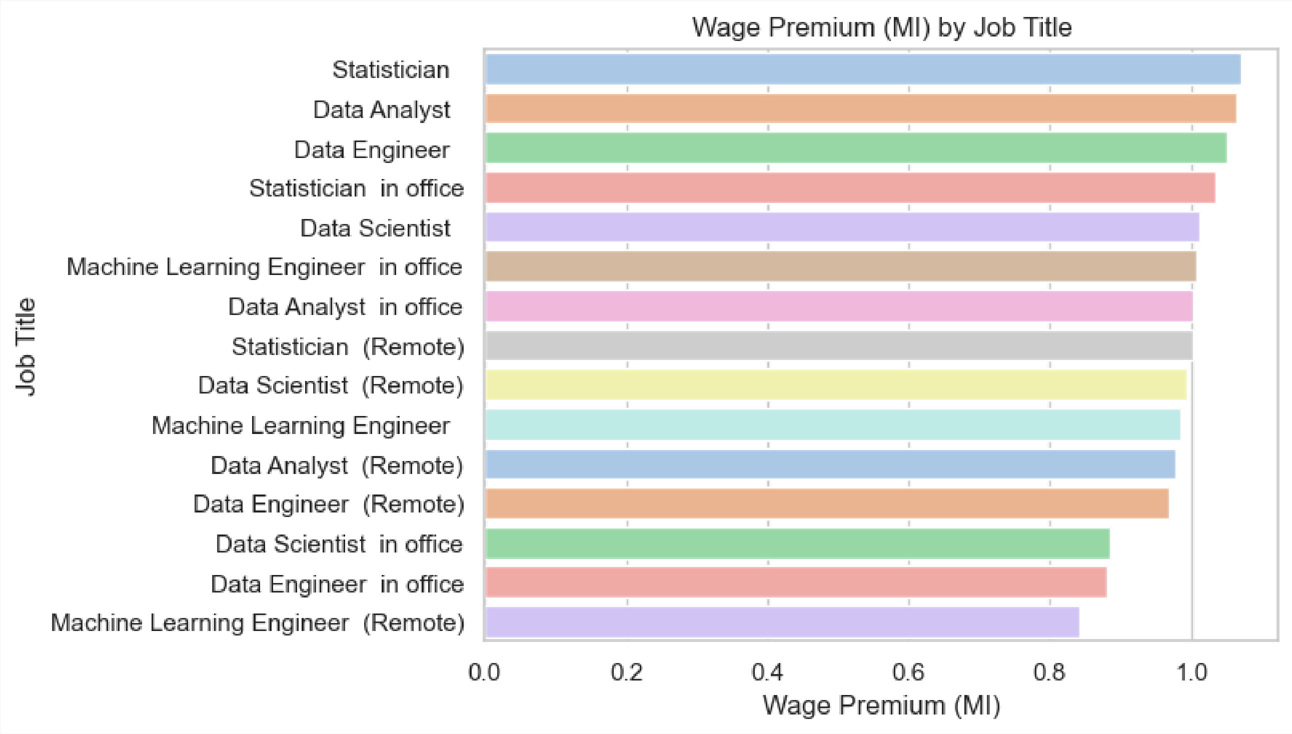


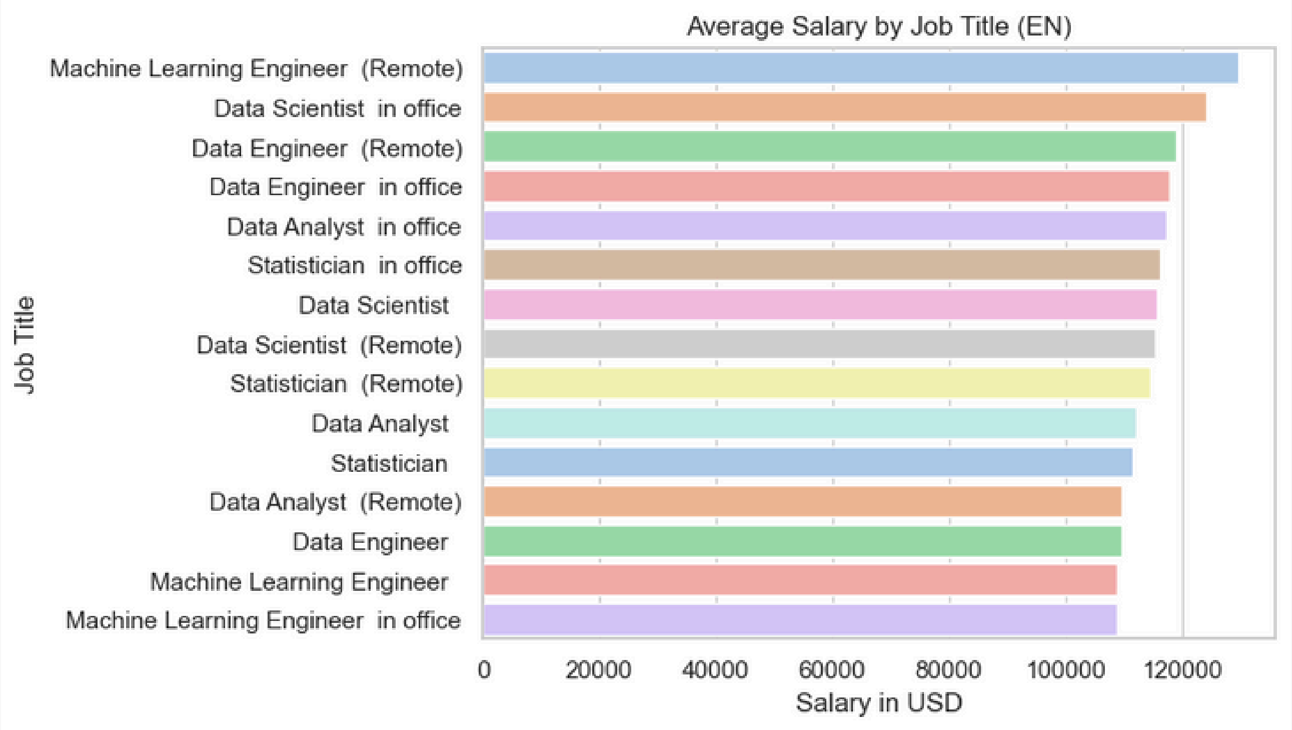

```python
# Section 5: Salary Distrubtion by Job Category
# ----------------------------------------------------

def plot_salary_distribution_by_job_category(data, title, filename):
    sns.set(style='whitegrid')
    plt.figure(figsize=(8, 5))
    sns.boxplot(x='salary_in_usd', y='job_category', hue='job_category', data=data)
    plt.title(title, fontsize=14)
    plt.xlabel('Salary in USD', fontsize=12)
    plt.ylabel('Job Category', fontsize=12)
    plt.savefig(filename, bbox_inches='tight')
    plt.show()

plot_salary_distribution_by_job_category(df_EN, 'Salaries by Job Category (EN)', 'salaries_by_job_category_EN.png')
plot_salary_distribution_by_job_category(df_SE, 'Salaries by Job Category (SE)', 'salaries_by_job_category_SE.png')
```

Salary Bar Graphs




```
# Section 7: Wage Premium Calculations
# ---------------------------------------
salary_by_title_EN = df_EN.groupby('job_title')['salary_in_usd'].mean().reset_index()
salary_by_title_MI = df_MI.groupby('job_title')['salary_in_usd'].mean().reset_index()
salary_by_title_SE = df_SE.groupby('job_title')['salary_in_usd'].mean().reset_index()

merged_salary = salary_by_title_EN.merge(salary_by_title_MI, on='job_title', how='outer', suffixes=('_EN', '_MI'))
merged_salary = merged_salary.merge(salary_by_title_SE, on='job_title', how='outer', suffixes=('', '_SE'))

print(merged_salary.head())
print()

merged_salary['wage_premium_MI'] = merged_salary['salary_in_usd_MI']/merged_salary['salary_in_usd_EN']
print(merged_salary.head())
```

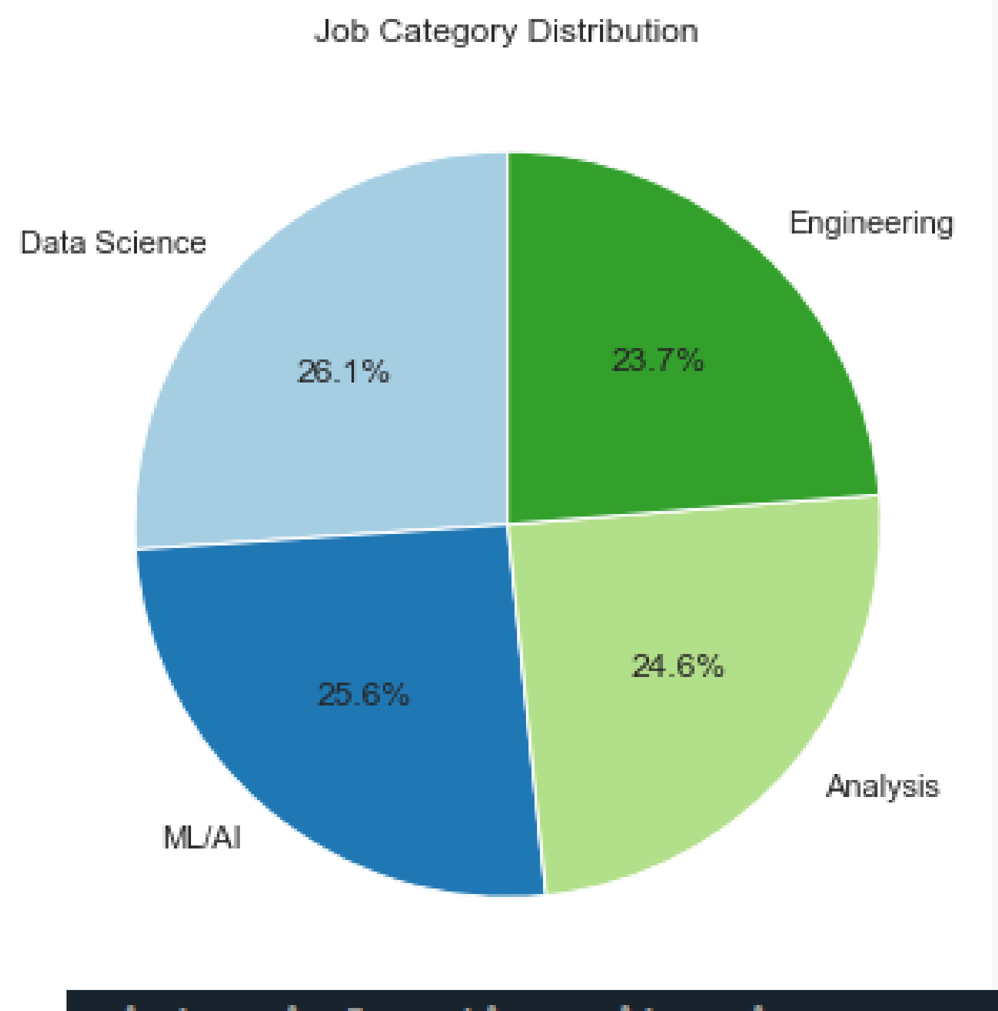


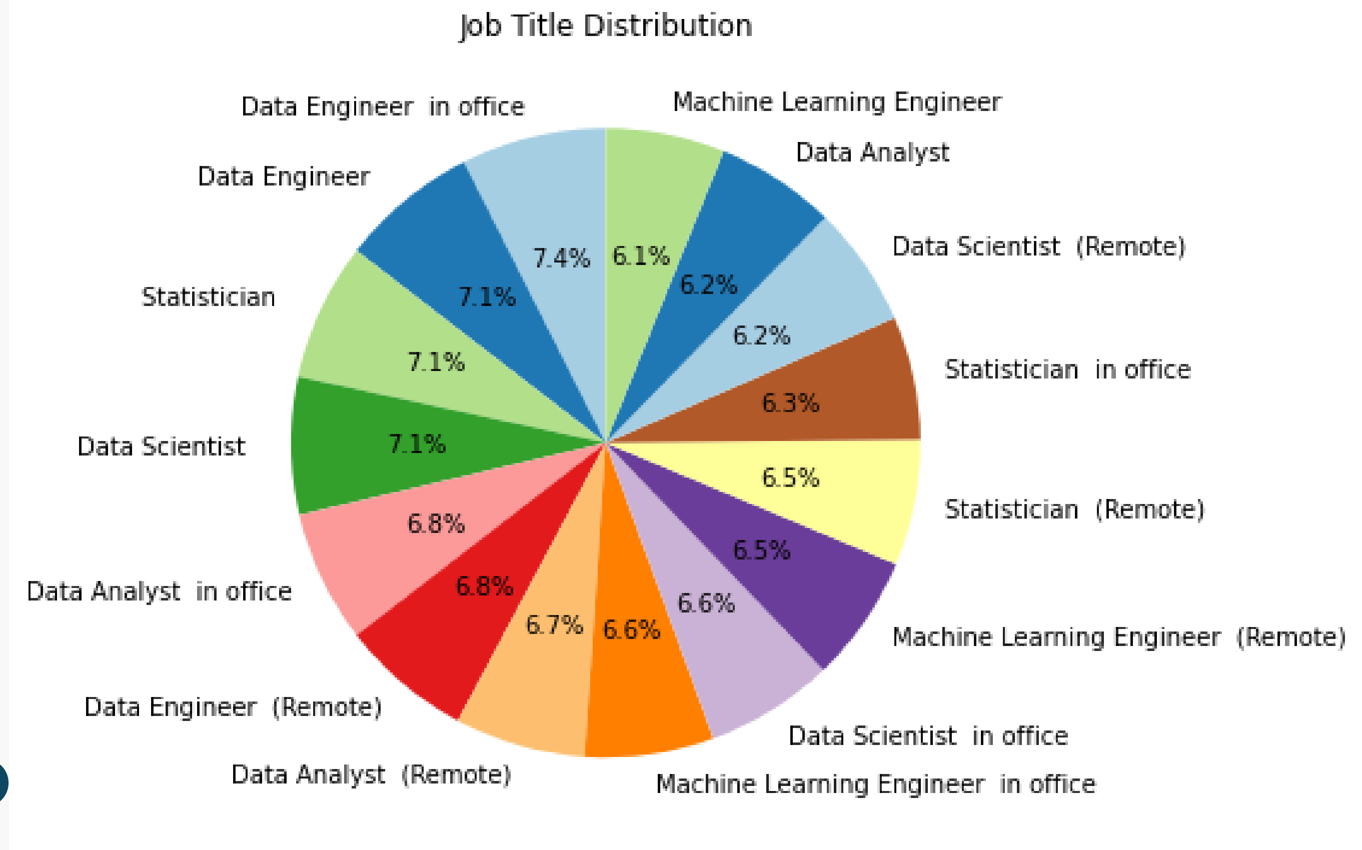



```
# Section 6: Salary by Job Title
# ----------------------------------
salary_by_title = df_EN.groupby('job_title')['salary_in_usd'].mean().reset_index()
salary_by_title_sorted = salary_by_title.sort_values(by='salary_in_usd', ascending=False).reset_index(drop=True)

sns.barplot(x='salary_in_usd', y='job_title', data=salary_by_title_sorted, hue = 'job_title', palette='pastel', legend = False)
plt.title('Average Salary by Job Title (EN)')
plt.xlabel('Salary in USD')
plt.ylabel('Job Title')
plt.savefig('salaries_by_job_title_EN.png',bbox_inches = 'tight')
plt.show()
```

Pie charts

```
#used the matplotlib lecture but asked google to help
#Job Category Distribution
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('data_science_job.csv')
#job category distribution
f = pd.read_csv('data_science_job.csv')

# Calculate the frequencies for job categories
job_category_counts = df['job_category'].value_counts()
# Plot as a pie chart
job_category_counts.plot(kind='pie', autopct='%1.1f%%', figsize=(6, 6), s
plt.title('Job Category Distribution')
plt.ylabel('')  # Removes the default ylabel
plt.savefig("job_category_distribution.png")
plt.show()
```






```
#job title distribution
# Calculate the frequencies for job categories
job_category_counts = df['job_title'].value_counts()

# Plot as a pie chart
job_category_counts.plot(kind='pie', autopct='%1.1f%%
plt.title('Job Title Distribution')
plt.ylabel('')  # Removes the default ylabel
plt.savefig("job_title_distribution.png")
plt.show()
```

User Input Graphs

With this code, I tried to create a system where the user can compare their desired variables

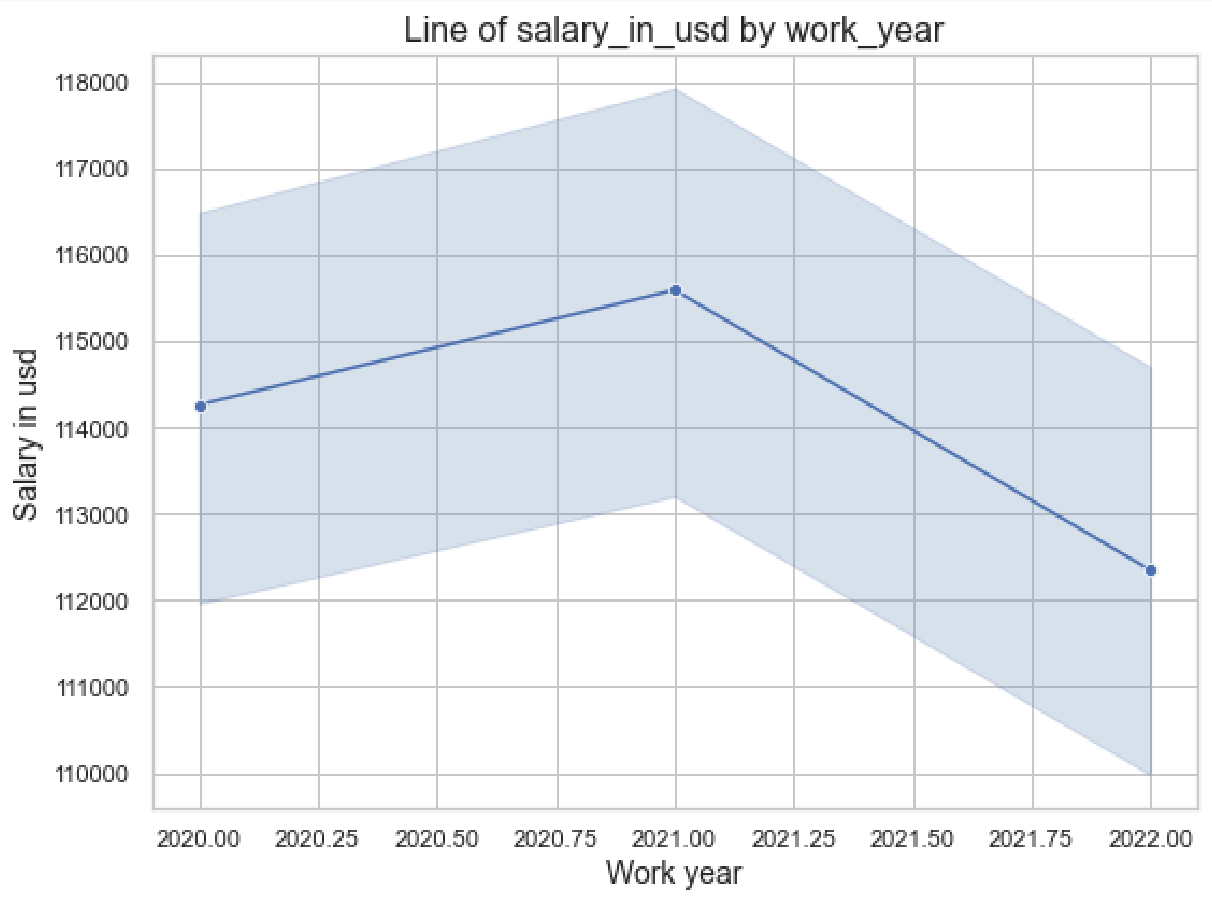
- adjusted fig size
- capitalize
- translated input to graph
- instructions

```
df = pd.read_csv('data_science_job.csv')
fig_size = (10, 6)
def create_custom_graph(data, x_column, y_column, graph_type):

    sns.set_theme(style="whitegrid")

    plt.figure(figsize=fig_size)
    # Title and labels
    title = f"{graph_type.capitalize()} of {y_column} by {x_column}"
    #.capitalize so user doesnt have to worry about exact input
    plt.title(title, fontsize=16)
    plt.xlabel(x_column.replace("_", " ").capitalize(), fontsize=14)
    #replace makes the variables easier for users to read from corn_soup to
    plt.ylabel(y_column.replace("_", " ").capitalize(), fontsize=14)

    # Create the plot based on the graph type
    if graph_type == 'boxplot':
        sns.boxplot(data=data, x=x_column, y=y_column, palette="muted")
    elif graph_type == 'barplot':
        sns.barplot(data=data, x=x_column, y=y_column, palette="muted")
    elif graph_type == 'scatter':
        sns.scatterplot(data=data, x=x_column, y=y_column, palette="muted")
    elif graph_type == 'line':
        sns.lineplot(data=data, x=x_column, y=y_column, marker="o")
    else:
        raise ValueError(f"Unsupported graph type: {graph_type}")
    filename = f"{graph_type}_{x_column}_vs_{y_column}.png"
    plt.savefig(filename)
    print(f"Graph saved as: {filename}")
    # Adjust layout for tight packing
    plt.tight_layout()
    plt.show()
```

```
    else:
        raise ValueError(f"Unsupported graph type: {graph_
    filename = f"{graph_type}_{x_column}_vs_{y_column}.png
    plt.savefig(filename)
    print(f"Graph saved as: {filename}")
    # Adjust layout for tight packing
    plt.tight_layout()
    plt.show()

def print_instructions():
    print("Instructions:")
    print("1. For boxplots and barplots, choose categorica
    print("2. For scatterplots and lineplots, choose numer
    print("3. The graph type should correspond to the kind
print_instructions()
# User inputs for the x and y variables and graph type
x_column = input("Choose your x variable from: 'work_year'
y_column = input("Choose your y variable from: 'work_year'
graph_type = input("Enter the graph type ('boxplot', 'scat

# Create the graph with the inputs
create_custom_graph(df, x_column, y_column, graph_type)
```

Graphs output

```
Instructions:
1. For boxplots and barplots, choose categorical variables for the x-axis and
numerical ones for the y-axis.
2. For scatterplots and lineplots, choose numerical variables on both axes
(e.g., 'work_year' vs 'salary_in_usd').
3. The graph type should correspond to the kind of data you're working with.
Choose your x variable from: 'work_year', 'job_title', 'job_category',
'salary_currency', 'salary', 'salary_in_usd', 'employee_residence',
'experience_level', 'employment_type', 'work_setting', 'company_location',
'company_size' work_year
Choose your y variable from: 'work_year', 'job_title', 'job_category',
'salary_currency', 'salary', 'salary_in_usd', 'employee_residence',
'experience_level', 'employment_type', 'work_setting', 'company_location',
'company_size' salary_in_usd
Enter the graph type ('boxplot', 'scatter', 'barplot', 'line'): line
Graph saved as: line_work_year_vs_salary_in_usd.png
```



Graph examples

Cons: Funky data /variables leads to weird graphs

Questions?

Thank you