

data_exploration

October 30, 2024

0.1 Developer Salary Estimator - Data Exploration

Author: Topaz Montague

Deliverable Description: The purpose of the Data Exploration deliverable in the salary estimator project aims to provide a comprehensive initial analysis of the dataset, encompassing descriptive statistics, visual analysis, and key insights from exploratory data analysis (EDA). This deliverable includes summarizing key salary-related measures such as average compensation, distribution of salaries across job roles, experience levels, and demographic factors. Through visualizations and statistical summaries, we identify patterns, trends, and outliers that may influence salary disparities, enabling a deeper understanding of the spread and central tendencies within the developer salary data. This foundational analysis informs feature selection, highlights relationships among variables, and guides the modeling approach necessary to build an accurate salary prediction tool.

Project Artifacts: GitHub Repository Link: [Developer Salary GitHub Repository](#)

Overleaf Project Report: [Developer Salary Overleaf Project Report](#)

Shiny App Dashboard: [Developer Salary Shiny App Dashboard - PLACEHOLDER](#)

Import Dependencies

```
[1]: import geopandas as gpd
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Load the Data Files

```
[2]: from IPython.display import display

# Load Stack Overflow data
dev_survey_df = pd.read_csv("data/cleaned/Transformed_Developer_Survey_Data.
↪csv", low_memory=False)

# Load BLS data
bls_df = pd.read_csv("data/cleaned/Transformed_bls2023_dl.csv")
```

```
# Display the first few rows of each dataset
display(dev_survey_df.head())
display(bls_df.head())
```

	ResponseId	Age	Range	RemoteWork	EdLevel	YearsCode	\
0	390	30		Remote	Some college	7	
1	399	50		Remote	Some college	38	
2	417	40		Remote	Masters	21	
3	427	20		Remote	Bachelors	9	
4	429	30		Remote	Bachelors	20	

		DevType	OrgSize	\
0		Student	15	
1		Developer, full-stack	2500	
2		Developer, back-end	250	
3	Developer, embedded applications or devices		2500	
4	Engineer, site reliability		250	

		Country	ICorPM	WorkExp	...	\
0	United States of America	Individual contributor		8	...	
1	United States of America	Individual contributor		30	...	
2	Brazil	Individual contributor		17	...	
3	Ukraine	Individual contributor		4	...	
4	United States of America	Individual contributor		15	...	

	Database_Oracle	Database_PostgreSQL	Database_Presto	Database_RavenDB	\
0	No	No	No	No	
1	No	Yes	No	No	
2	No	Yes	No	No	
3	No	No	No	No	
4	No	No	No	No	

	Database_Redis	Database_SQLite	Database_Snowflake	Database_Solr	\
0	No	Yes	No	No	
1	No	No	Yes	No	
2	No	No	No	No	
3	No	No	No	No	
4	Yes	Yes	No	No	

	Database_Supabase	Database_TiDB
0	No	No
1	No	No
2	No	No
3	No	No
4	No	No

[5 rows x 96 columns]

	State	Title	Size	AvgSalary
0	Alabama	Computer Programmers	1860	97020
1	Alabama	Software Developers	17130	110630
2	Alabama	Software Quality Assurance Analysts and Testers	1850	101520
3	Alabama	Web Developers	1120	66750
4	Alabama	Web and Digital Interface Designers	150	54470

Descriptive Statistics for Stack Overflow Developer Survey Data Summary Statistics

```
[3]: # Set pandas to display floats in standard notation
pd.options.display.float_format = '{:.2f}'.format

# Generate summary statistics
dev_survey_summary = dev_survey_df.describe()
dev_survey_summary
```

```
[3]:
```

	ResponseId	Age Range	YearsCode	WorkExp	TotalComp
count	8544.00	8544.00	8544.00	8544.00	8544.00
mean	29970.17	36.88	17.10	8.97	142913.73
std	17482.87	10.93	10.66	10.30	137418.80
min	390.00	16.00	0.00	0.00	3500.00
25%	15909.75	30.00	9.00	0.00	100000.00
50%	29552.50	40.00	14.00	6.00	140000.00
75%	44414.50	40.00	24.00	14.00	150000.00
max	65271.00	70.00	50.00	50.00	900000.00

Categorical Data Counts

```
[4]: # Distribution of Developer Types
dev_survey_df['DevType'].value_counts()
```

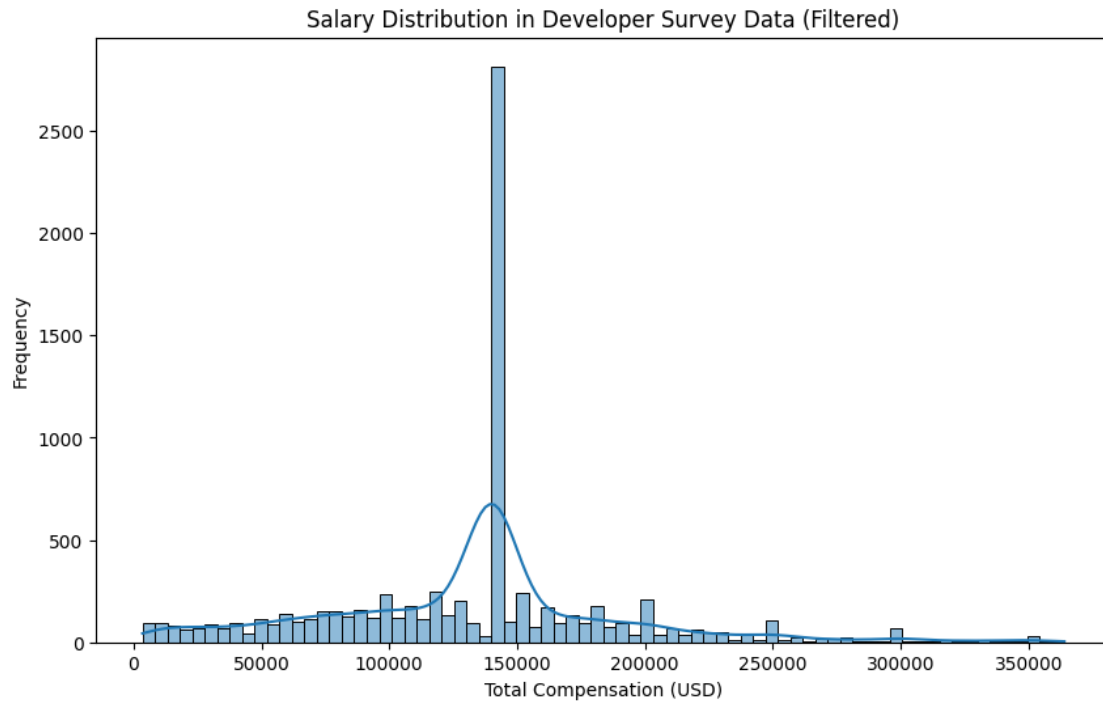
```
[4]: DevType
Developer, full-stack          3117
Developer, back-end           1537
Developer, front-end           408
Developer, desktop or enterprise applications  354
Other                          290
Developer, embedded applications or devices  280
Engineering manager           275
Data engineer                  234
Developer, mobile              232
Data scientist or machine learning specialist  182
DevOps specialist              154
Research & Development role    154
Senior Executive (C-Suite, VP, etc.) 152
Cloud infrastructure engineer   134
Academic researcher            105
Developer, QA or test          99
```

Data or business analyst	91
Developer, AI	78
Developer, game or graphics	72
Engineer, site reliability	66
Scientist	63
System administrator	63
Security professional	61
Developer Experience	41
Educator	41
Student	38
Project manager	36
Hardware Engineer	36
Blockchain	33
Product manager	31
Developer Advocate	28
Database administrator	27
Designer	22
Marketing or sales professional	10
Name: count, dtype: int64	

Visualize Salary Distribution

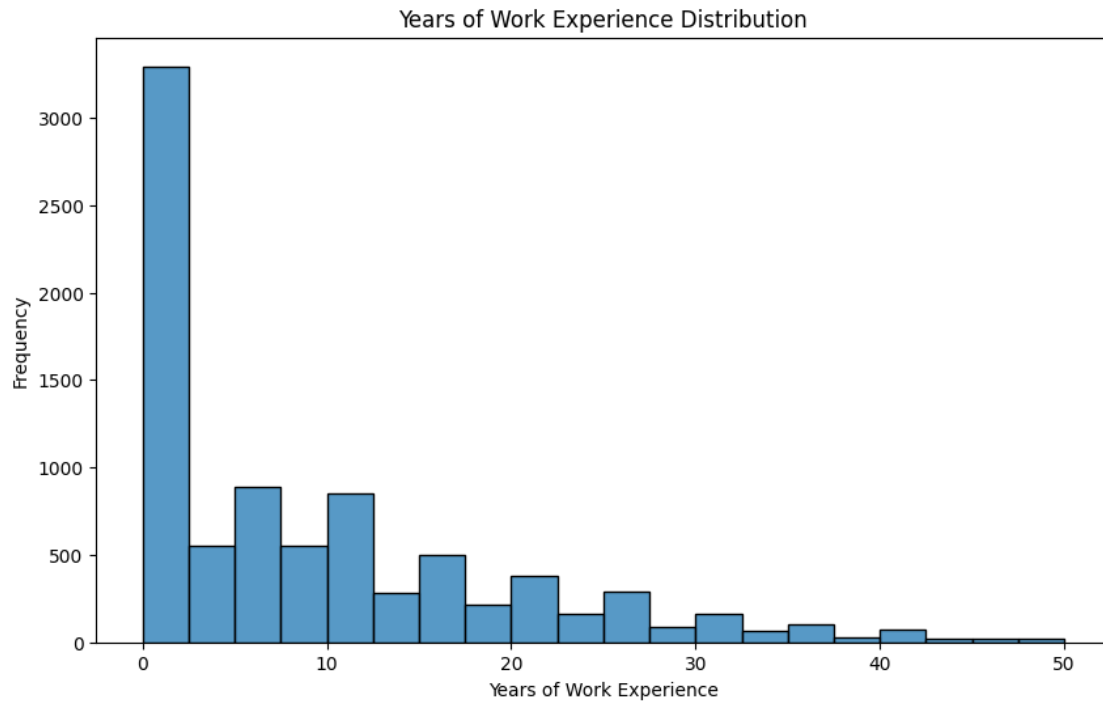
```
[5]: # Remove outliers for better visualization (keeping salaries below a threshold
      ↪ of 375000)
      filtered_df = dev_survey_df[dev_survey_df['TotalComp'] < 375000]

      plt.figure(figsize=(10, 6))
      sns.histplot(filtered_df['TotalComp'], kde=True)
      plt.title("Salary Distribution in Developer Survey Data (Filtered)")
      plt.xlabel("Total Compensation (USD)")
      plt.ylabel("Frequency")
      plt.show()
```



Visualize Years of Work Experience

```
[6]: plt.figure(figsize=(10, 6))
sns.histplot(dev_survey_df['WorkExp'], bins=20, kde=False)
plt.title("Years of Work Experience Distribution")
plt.xlabel("Years of Work Experience")
plt.ylabel("Frequency")
plt.show()
```

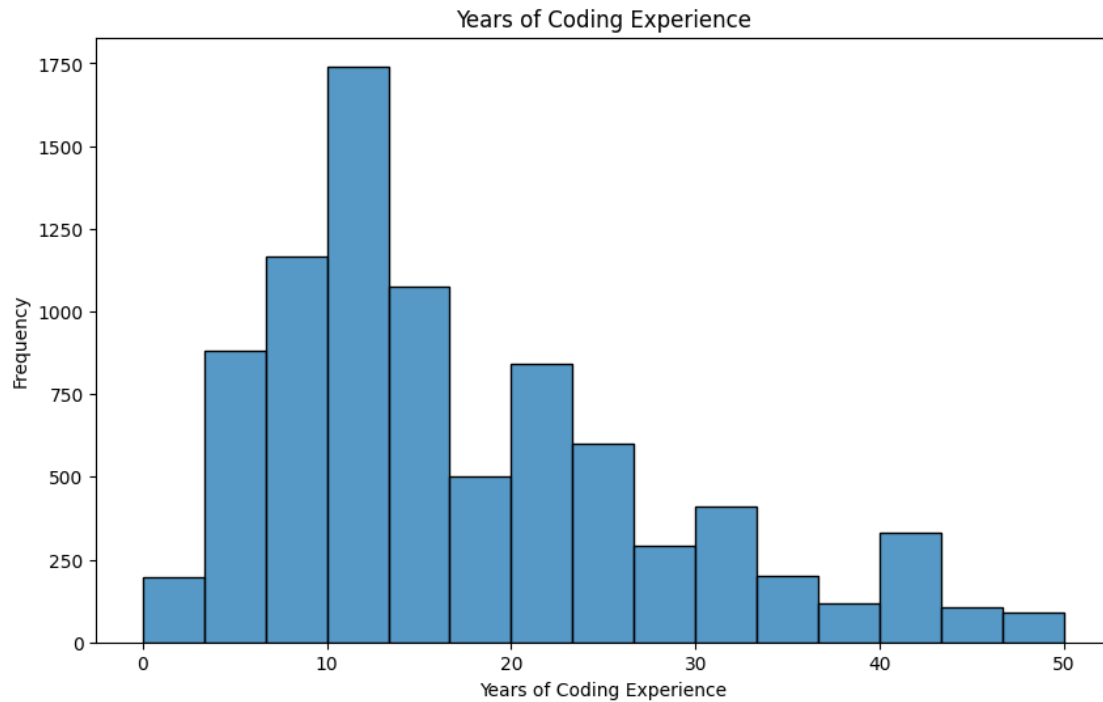


Visualize Years of Coding Experience

```
[7]: # Replace non-numeric values in YearsCode
dev_survey_df['YearsCode'] = dev_survey_df['YearsCode'].replace({
    'Less than 1 year': 0,
    'More than 50 years': 50
})

# Convert the column to numeric
dev_survey_df['YearsCode'] = pd.to_numeric(dev_survey_df['YearsCode'],
    errors='coerce')

# Plot the histogram with adjusted bins
plt.figure(figsize=(10, 6))
sns.histplot(dev_survey_df['YearsCode'], bins=15, kde=False)
plt.title("Years of Coding Experience")
plt.xlabel("Years of Coding Experience")
plt.ylabel("Frequency")
plt.show()
```



Visualize Average Company Size

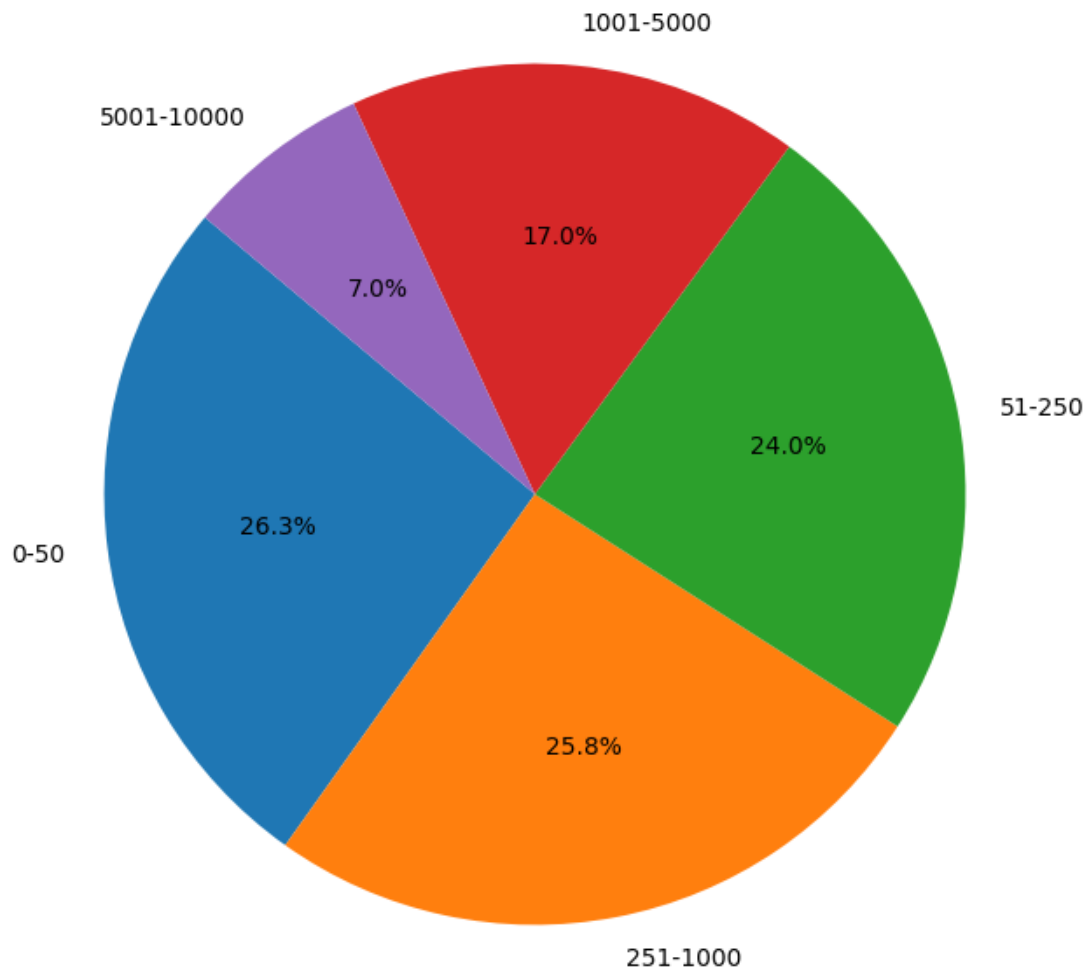
```
[8]: # Define custom bins and labels for company sizes
dev_survey_df['OrgSize'] = pd.to_numeric(dev_survey_df['OrgSize'],
    ↪errors='coerce')
bins = [0, 50, 250, 1000, 5000, 10000]
labels = ['0-50', '51-250', '251-1000', '1001-5000', '5001-10000']

# Create the CompanySizeCategory column with these bins
dev_survey_df['CompanySizeCategory'] = pd.cut(dev_survey_df['OrgSize'],
    ↪bins=bins, labels=labels, right=False)

# Create the pie chart with updated bins
company_size_counts = dev_survey_df['CompanySizeCategory'].value_counts()

plt.figure(figsize=(8, 8))
plt.pie(company_size_counts, labels=company_size_counts.index, autopct='%1.
    ↪1f%%', startangle=140)
plt.title("Distribution of Companies by Size Category")
plt.show()
```

Distribution of Companies by Size Category



Descriptive Statistics for BLS Data Summary Statistics

```
[9]: # Generate summary statistics for BLS data
bls_summary = bls_df.describe()
bls_summary
```

```
[9]:
```

	Size	AvgSalary
count	258.00	258.00
mean	8890.00	97583.49
std	25164.09	22531.26
min	30.00	41550.00
25%	662.50	81820.00


```

50%      1950.00   96755.00
75%      6512.50  112605.00
max     304390.00 173780.00

```

```

[10]: # Group by state and calculate average salary per state
state_salary = bls_df.groupby('State')['AvgSalary'].mean().reset_index()

# Display the average salary by state
print(state_salary)

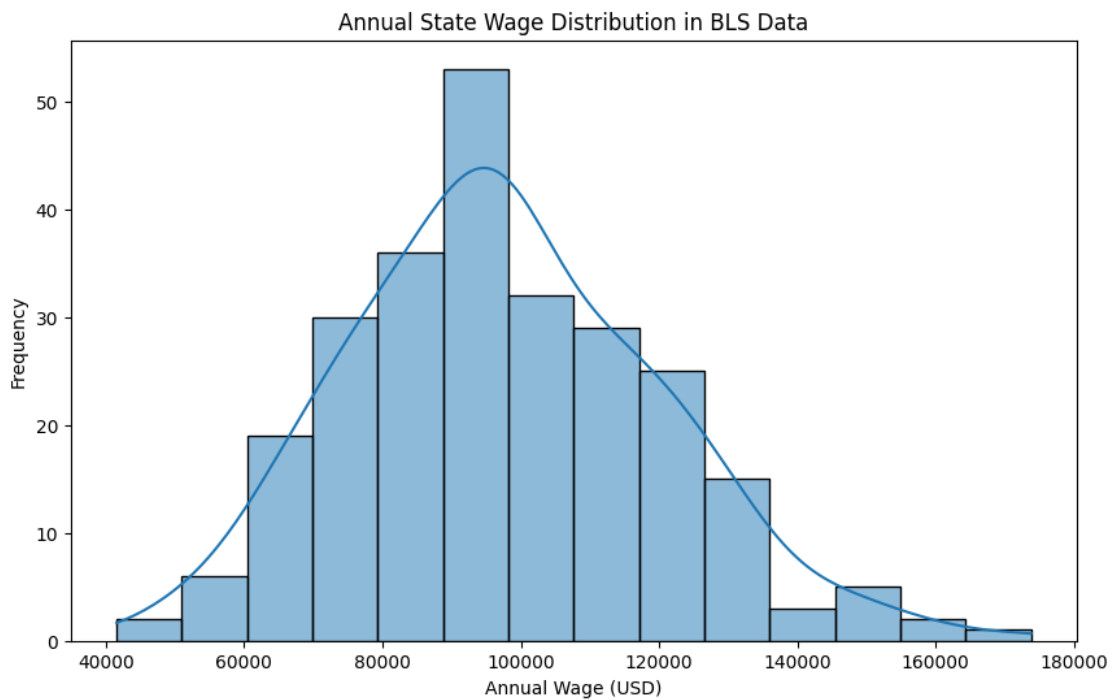
```

	State	AvgSalary
0	Alabama	86078.00
1	Alaska	103742.00
2	Arizona	99692.00
3	Arkansas	73715.00
4	California	137972.00
5	Colorado	113870.00
6	Connecticut	101632.00
7	Delaware	104312.00
8	District of Columbia	117454.00
9	Florida	100104.00
10	Georgia	105578.00
11	Hawaii	98538.00
12	Idaho	97316.00
13	Illinois	98112.00
14	Indiana	85520.00
15	Iowa	87562.00
16	Kansas	90776.00
17	Kentucky	89884.00
18	Louisiana	85330.00
19	Maine	94624.00
20	Maryland	113956.00
21	Massachusetts	117882.00
22	Michigan	89854.00
23	Minnesota	102780.00
24	Mississippi	76776.00
25	Missouri	83450.00
26	Montana	90306.00
27	Nebraska	89318.00
28	Nevada	96828.00
29	New Hampshire	100026.00
30	New Jersey	110298.00
31	New Mexico	101422.50
32	New York	121922.00
33	North Carolina	101312.00
34	North Dakota	84724.00
35	Ohio	92648.00
36	Oklahoma	89488.00

37	Oregon	105052.00
38	Pennsylvania	99062.00
39	Puerto Rico	52568.00
40	Rhode Island	102594.00
41	South Carolina	95376.00
42	South Dakota	77952.00
43	Tennessee	94550.00
44	Texas	106732.00
45	Utah	96970.00
46	Vermont	90080.00
47	Virgin Islands	132690.00
48	Virginia	111328.00
49	Washington	136524.00
50	West Virginia	76648.00
51	Wisconsin	89798.00
52	Wyoming	92202.50

Visualize State Averages Salary Distribution

```
[11]: plt.figure(figsize=(10, 6))
sns.histplot(bls_df['AvgSalary'], kde=True)
plt.title("Annual State Wage Distribution in BLS Data")
plt.xlabel("Annual Wage (USD)")
plt.ylabel("Frequency")
plt.show()
```



Visualize Average Salary by State

```
[12]: # Load your data (BLS data with 'State' and 'AvgSalary' columns)
bls_df = pd.read_csv("data/cleaned/Transformed_bls2023_dl.csv")

# Group by state and calculate average salary per state
state_salary = bls_df.groupby('State')['AvgSalary'].mean().reset_index()

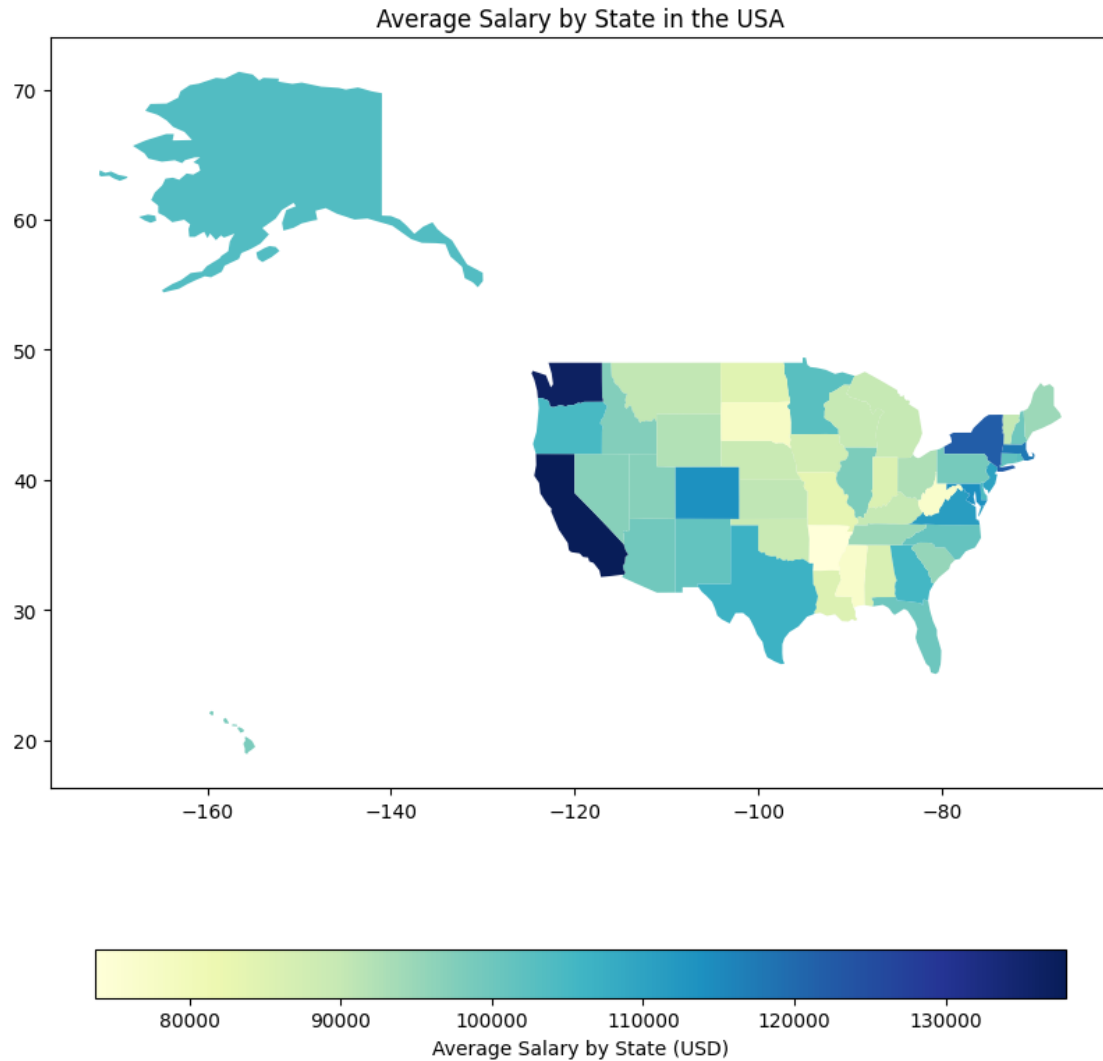
# Load the USA states shapefile from your specified path
usa_states = gpd.read_file(r"C:/Users/topaz/geo/
↪ne_110m_admin_1_states_provinces.shp")

# Filter to include only USA states (if necessary)
usa_states = usa_states[usa_states['admin'] == 'United States of America']

# Merge the salary data with the geospatial data
state_salary_map = usa_states.set_index('name').join(state_salary.
↪set_index('State'))

# Plotting the map of average salary by state
fig, ax = plt.subplots(1, 1, figsize=(15, 10))
state_salary_map.plot(column='AvgSalary',
                        ax=ax,
                        legend=True,
                        cmap="YlGnBu",
                        legend_kwds={
                            'label': "Average Salary by State (USD)",
                            'orientation': "horizontal",
                            'shrink': 0.6,
                            'pad': 0.15 # Increase padding
                        })

plt.title("Average Salary by State in the USA")
plt.show()
```



Feature Relationships - Pairplots and Correlation Matrix

Use pairplots to visualize relationships between features and a heatmap to identify correlations

```
[15]: import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the dataset
data = pd.read_csv('data/cleaned/Transformed_Developer_Survey_Data.csv')

# Pairplot of selected features to observe relationships
selected_features = ['TotalComp', 'YearsCode', 'OrgSize', 'WorkExp']
sns.pairplot(data[selected_features], diag_kind='kde')
```

```

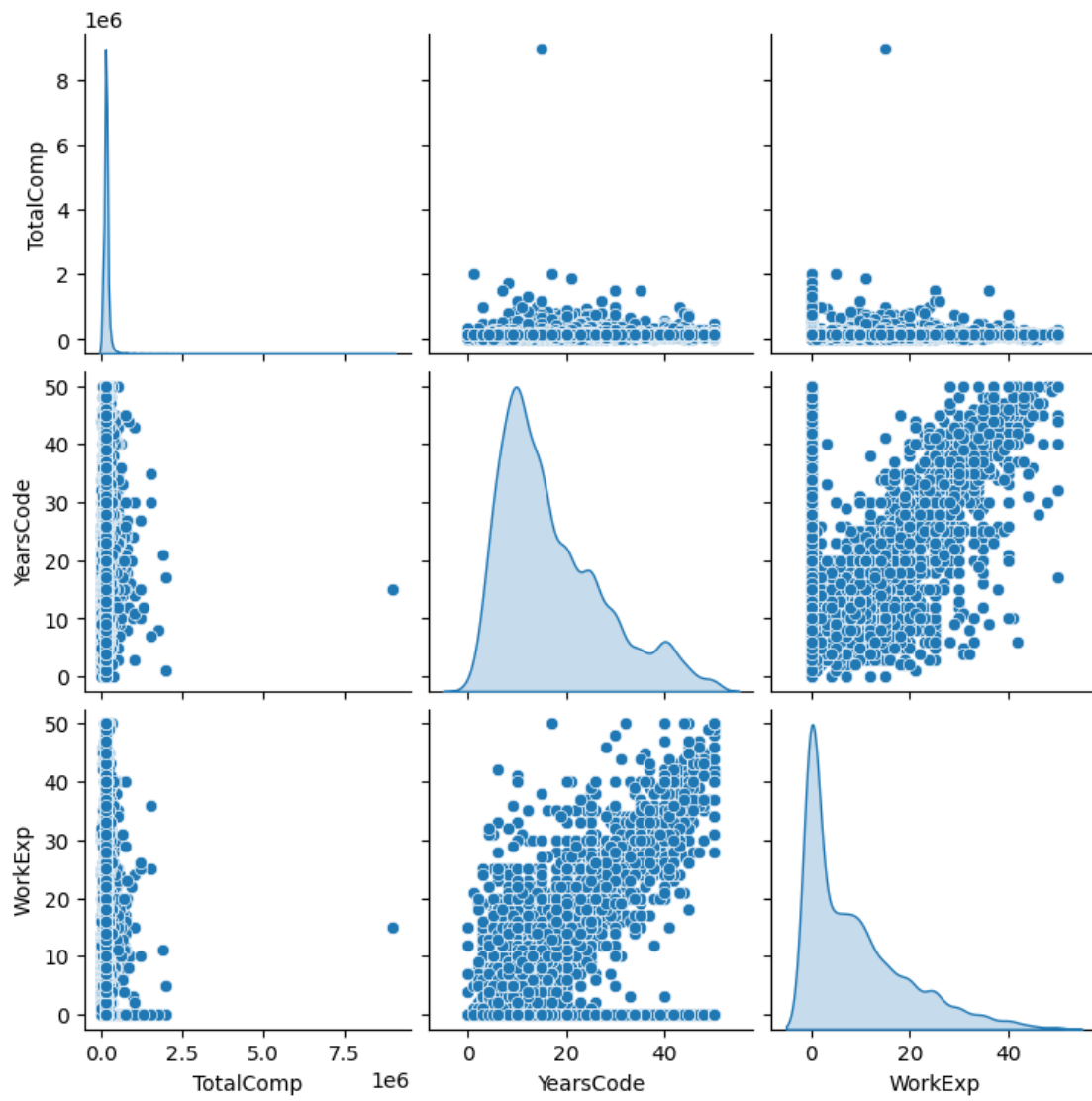
plt.suptitle("Pairplot of Numerical Features", y=1.02)
plt.show()

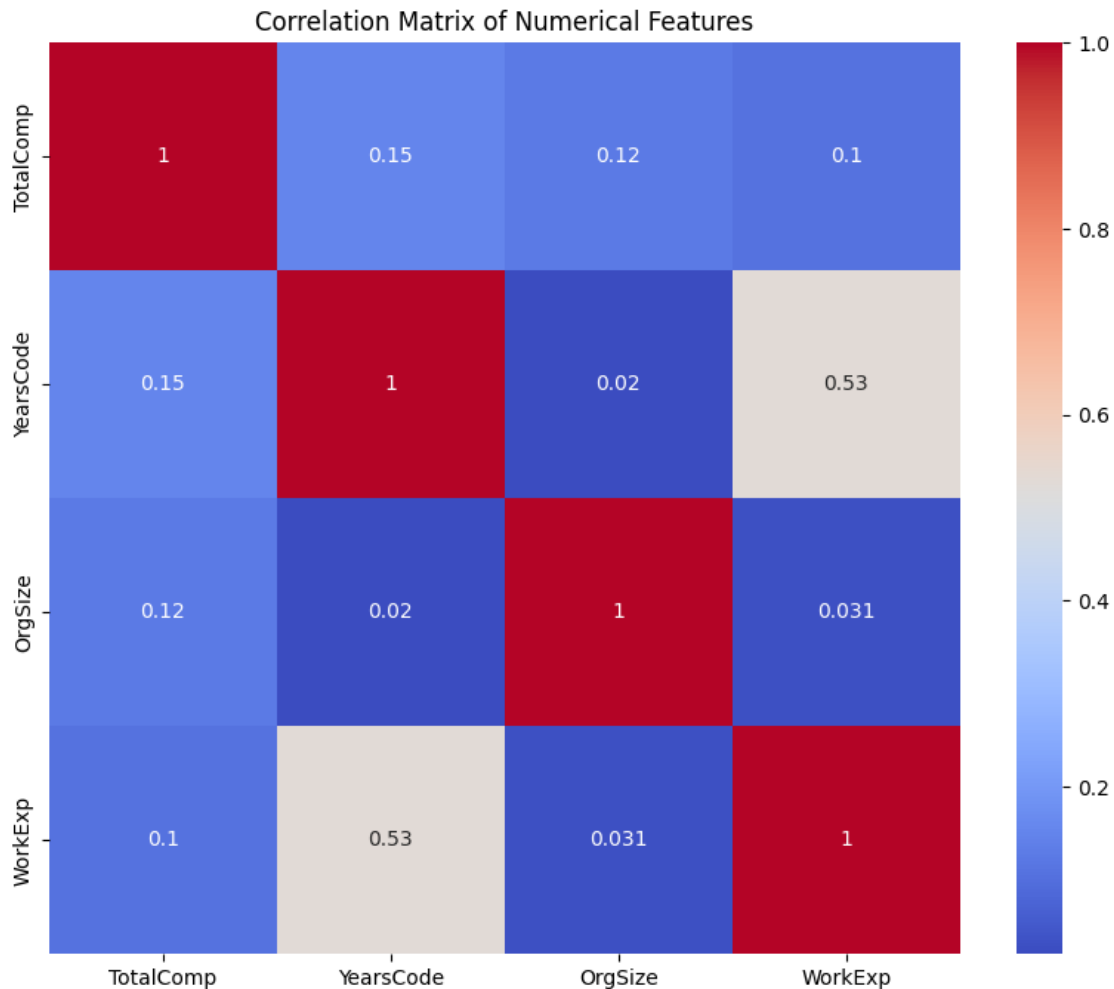
# Convert columns to numeric, forcing errors to NaN (for example, in 'OrgSize'
↳ or other categorical fields)
data['YearsCode'] = pd.to_numeric(data['YearsCode'], errors='coerce')
data['OrgSize'] = pd.to_numeric(data['OrgSize'], errors='coerce')
data['WorkExp'] = pd.to_numeric(data['WorkExp'], errors='coerce')

# Correlation matrix with a heatmap
plt.figure(figsize=(10, 8))
correlation_matrix = data[selected_features].corr()
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', square=True)
plt.title("Correlation Matrix of Numerical Features")
plt.show()

```

Pairplot of Numerical Features





Outlier Analysis

Use box plots to visualize outliers in Total Compensation values

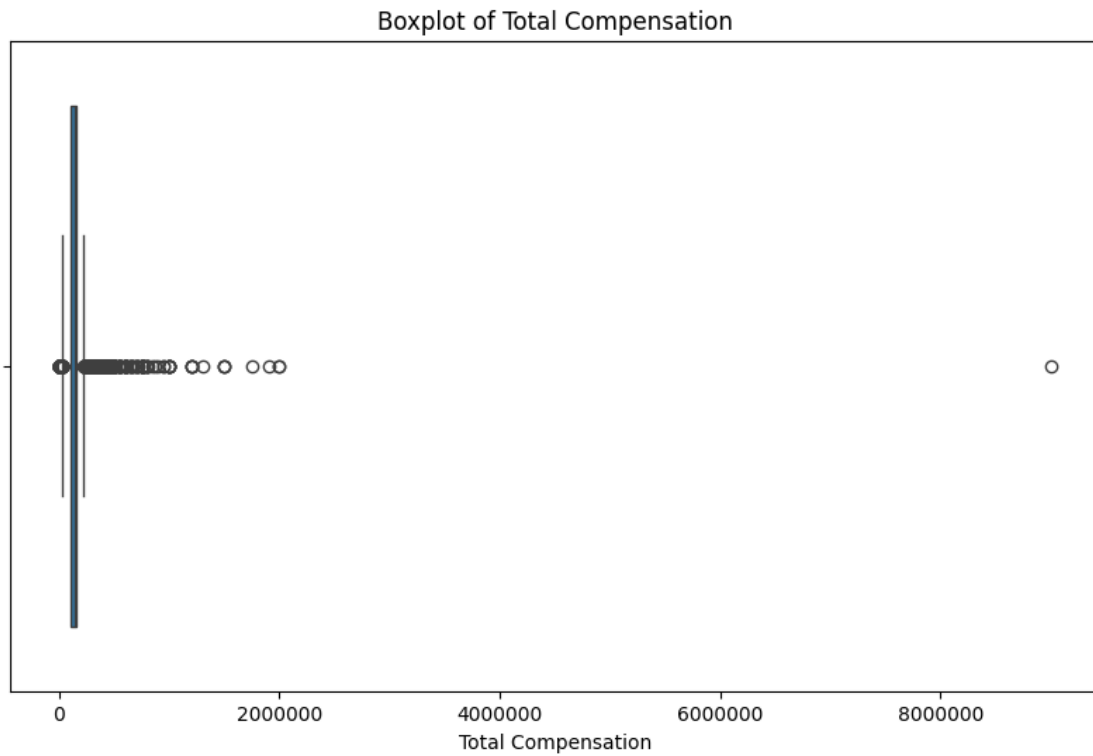
```
[17]: # Box plot to detect outliers in TotalComp
plt.figure(figsize=(10, 6))
sns.boxplot(x=data['TotalComp'])
plt.title("Boxplot of Total Compensation")
plt.xlabel("Total Compensation")
plt.ticklabel_format(style='plain', axis='x') # Avoid scientific notation on
↳ the x-axis
plt.show()

# Detect and count outliers (using 1.5 IQR rule)
Q1 = data['TotalComp'].quantile(0.25)
Q3 = data['TotalComp'].quantile(0.75)
IQR = Q3 - Q1
```

```

outliers = data[(data['TotalComp'] < (Q1 - 1.5 * IQR)) | (data['TotalComp'] >
↪(Q3 + 1.5 * IQR))]
print(f"Number of outliers in Total Compensation: {outliers.shape[0]}")

```



Number of outliers in Total Compensation: 1032