Salary Prediction based on Country and Race

The aim of this project to predict the salary of individuals from varied countires and races based on their demographics such as occupation, age, gender, experience, education, etc. The dataset is taken from Kaggle. The dataset has 32561 rows and 15 columns. The dataset has 8 independent variables and 1 target variable i.e Salary

Aboout the dataset

The dataset consists of a comprehensive collection of salary and demographic information with additional details on years of experience. It offers a valuable resource for studying the relationship between income and various socio-demographic factors. The demographic attributes include age, gender, education, country, and race, providing a diverse range of variables for analysis. Researchers can explore patterns and trends in income distribution across different demographic categories, allowing for insights into potential disparities or variations in earning potential. Moreover, the dataset incorporates the crucial dimension of years of experience, enabling investigations into the impact of professional tenure on salary levels. This aspect adds a dynamic aspect to the analysis, enabling researchers to examine how income varies based on both demographic characteristics and accumulated work experience. The dataset presents a rich opportunity for conducting comprehensive studies on income diversity and understanding the multifaceted factors influencing earning potential in today's workforce.

Data Dictionary

Column	Description
Unnamed: 0	Index
Age	Age of the employee
Education Level	Education level of the employee
Job Title	Job title of the employee
Years of Experience	Years of experience of the employee
Salary	Salary of the employee
Country	Country of the employee
Race	Race of the employee

In [ ]:

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

In [ ]:

```
#loading the data
df = pd.read_csv('Salary_Data_Based_country_and_race.csv')
df.head()
```

Out[ ]:

	Unnamed: 0	Age	Gender	Education Level	Job Title	Years of Experience	Salary	Country	R
0	0	32.0	Male	Bachelor's	Software Engineer	5.0	90000.0	UK	W
1	1	28.0	Female	Master's	Data Analyst	3.0	65000.0	USA	Hispa
2	2	45.0	Male	PhD	Senior Manager	15.0	150000.0	Canada	W
3	3	36.0	Female	Bachelor's	Sales Associate	7.0	60000.0	USA	Hispa
4	4	52.0	Male	Master's	Director	20.0	200000.0	USA	As

Data Preprocessing

In [ ]:

```
#checking the shape of the data
df.shape
```

Out[ ]:

```
(6704, 9)
```

In [ ]:

```
#checking for null/missing values
df.isnull().sum()
```

Out[ ]:

```
Unnamed: 0             0
Age                    2
Gender                 2
Education Level        3
Job Title              2
Years of Experience    3
Salary                 5
Country                0
Race                   0
dtype: int64
```

Since the number of rows with null/missing value is very less as compared to the total number of rows, I will be dropping these rows.

In [ ]:

```
df.dropna(axis=0, inplace=True)
```

In [ ]:

```
#checking for null values
df.isnull().sum()
```

```
Out[ ]: Unnamed: 0            0
        Age                   0
        Gender                0
        Education Level       0
        Job Title             0
        Years of Experience   0
        Salary                0
        Country               0
        Race                  0
        dtype: int64
```

Dropping Unnamed Column beacuse it is just an index column

```
In [ ]: #dropping column
        df.drop(columns = 'Unnamed: 0',axis=1,inplace=True)
```

Checking data type of each column

```
In [ ]: df.dtypes
```

```
Out[ ]: Age                    float64
        Gender                  object
        Education Level         object
        Job Title               object
        Years of Experience    float64
        Salary                 float64
        Country                 object
        Race                    object
        dtype: object
```

Checking for unique values in each column

```
In [ ]: #unique values in each column
        df.nunique()
```

```
Out[ ]: Age                     41
        Gender                   3
        Education Level          7
        Job Title              191
        Years of Experience     37
        Salary                 444
        Country                  5
        Race                    10
        dtype: int64
```

The job title column has 191 different values. It will be very difficult to analyze so many job titles. So, I will group the job titles under similar job domains.

Grouping Job Titles

```
In [ ]: df['Job Title'].unique()
```

```
Out[ ]:  array(['Software Engineer', 'Data Analyst', 'Senior Manager',
                'Sales Associate', 'Director', 'Marketing Analyst',
                'Product Manager', 'Sales Manager', 'Marketing Coordinator',
                'Senior Scientist', 'Software Developer', 'HR Manager',
                'Financial Analyst', 'Project Manager', 'Customer Service Rep',
                'Operations Manager', 'Marketing Manager', 'Senior Engineer',
                'Data Entry Clerk', 'Sales Director', 'Business Analyst',
                'VP of Operations', 'IT Support', 'Recruiter', 'Financial Manager',
                'Social Media Specialist', 'Software Manager', 'Junior Developer',
                'Senior Consultant', 'Product Designer', 'CEO', 'Accountant',
                'Data Scientist', 'Marketing Specialist', 'Technical Writer',
                'HR Generalist', 'Project Engineer', 'Customer Success Rep',
                'Sales Executive', 'UX Designer', 'Operations Director',
                'Network Engineer', 'Administrative Assistant',
                'Strategy Consultant', 'Copywriter', 'Account Manager',
                'Director of Marketing', 'Help Desk Analyst',
                'Customer Service Manager', 'Business Intelligence Analyst',
                'Event Coordinator', 'VP of Finance', 'Graphic Designer',
                'UX Researcher', 'Social Media Manager', 'Director of Operations',
                'Senior Data Scientist', 'Junior Accountant',
                'Digital Marketing Manager', 'IT Manager',
                'Customer Service Representative', 'Business Development Manager',
                'Senior Financial Analyst', 'Web Developer', 'Research Director',
                'Technical Support Specialist', 'Creative Director',
                'Senior Software Engineer', 'Human Resources Director',
                'Content Marketing Manager', 'Technical Recruiter',
                'Sales Representative', 'Chief Technology Officer',
                'Junior Designer', 'Financial Advisor', 'Junior Account Manager',
                'Senior Project Manager', 'Principal Scientist',
                'Supply Chain Manager', 'Senior Marketing Manager',
                'Training Specialist', 'Research Scientist',
                'Junior Software Developer', 'Public Relations Manager',
                'Operations Analyst', 'Product Marketing Manager',
                'Senior HR Manager', 'Junior Web Developer',
                'Senior Project Coordinator', 'Chief Data Officer',
                'Digital Content Producer', 'IT Support Specialist',
                'Senior Marketing Analyst', 'Customer Success Manager',
                'Senior Graphic Designer', 'Software Project Manager',
                'Supply Chain Analyst', 'Senior Business Analyst',
                'Junior Marketing Analyst', 'Office Manager', 'Principal Engineer',
                'Junior HR Generalist', 'Senior Product Manager',
                'Junior Operations Analyst', 'Senior HR Generalist',
                'Sales Operations Manager', 'Senior Software Developer',
                'Junior Web Designer', 'Senior Training Specialist',
                'Senior Research Scientist', 'Junior Sales Representative',
                'Junior Marketing Manager', 'Junior Data Analyst',
                'Senior Product Marketing Manager', 'Junior Business Analyst',
                'Senior Sales Manager', 'Junior Marketing Specialist',
                'Junior Project Manager', 'Senior Accountant', 'Director of Sales',
                'Junior Recruiter', 'Senior Business Development Manager',
                'Senior Product Designer', 'Junior Customer Support Specialist',
                'Senior IT Support Specialist', 'Junior Financial Analyst',
                'Senior Operations Manager', 'Director of Human Resources',
                'Junior Software Engineer', 'Senior Sales Representative',
                'Director of Product Management', 'Junior Copywriter',
                'Senior Marketing Coordinator', 'Senior Human Resources Manager',
                'Junior Business Development Associate', 'Senior Account Manager',
                'Senior Researcher', 'Junior HR Coordinator',
                'Director of Finance', 'Junior Marketing Coordinator',
                'Junior Data Scientist', 'Senior Operations Analyst',
                'Senior Human Resources Coordinator', 'Senior UX Designer',
                'Junior Product Manager', 'Senior Marketing Specialist',
                'Senior IT Project Manager', 'Senior Quality Assurance Analyst',
                'Director of Sales and Marketing', 'Senior Account Executive',
                'Director of Business Development', 'Junior Social Media Manager',
                'Senior Human Resources Specialist', 'Senior Data Analyst',
                'Director of Human Capital', 'Junior Advertising Coordinator',
                'Junior UX Designer', 'Senior Marketing Director',
                'Senior IT Consultant', 'Senior Financial Advisor',
                'Junior Business Operations Analyst',
                'Junior Social Media Specialist',
                'Senior Product Development Manager', 'Junior Operations Manager',
                'Senior Software Architect', 'Junior Research Scientist',
                'Senior Financial Manager', 'Senior HR Specialist',
                'Senior Data Engineer', 'Junior Operations Coordinator',
                'Director of HR', 'Senior Operations Coordinator',
                'Junior Financial Advisor', 'Director of Engineering',
                'Software Engineer Manager', 'Back end Developer',
                'Senior Project Engineer', 'Full Stack Engineer',
                'Front end Developer', 'Front End Developer',
                'Director of Data Science', 'Human Resources Coordinator',
                'Junior Sales Associate', 'Human Resources Manager',
                'Juniour HR Generalist', 'Juniour HR Coordinator',
                'Digital Marketing Specialist', 'Receptionist',
                'Marketing Director', 'Social Media Man', 'Delivery Driver'],
               dtype=object)
```

In [ ]:

```python
def categorize_job_title(job_title):
    job_title = str(job_title).lower()
    if 'software' in job_title or 'developer' in job_title:
        return 'Software/Developer'
    elif 'data' in job_title or 'analyst' in job_title or 'scientist' in job_tit
        return 'Data Analyst/Scientist'
    elif 'manager' in job_title or 'director' in job_title or 'vp' in job_title:
        return 'Manager/Director/VP'
    elif 'sales' in job_title or 'representative' in job_title:
        return 'Sales'
    elif 'marketing' in job_title or 'social media' in job_title:
        return 'Marketing/Social Media'
    elif 'product' in job_title or 'designer' in job_title:
        return 'Product/Designer'
    elif 'hr' in job_title or 'human resources' in job_title:
        return 'HR/Human Resources'
    elif 'financial' in job_title or 'accountant' in job_title:
        return 'Financial/Accountant'
    elif 'project manager' in job_title:
        return 'Project Manager'
    elif 'it' in job_title or 'support' in job_title:
        return 'IT/Technical Support'
    elif 'operations' in job_title or 'supply chain' in job_title:
        return 'Operations/Supply Chain'
    elif 'customer service' in job_title or 'receptionist' in job_title:
        return 'Customer Service/Receptionist'
    else:
        return 'Other'

df['Job Title'] = df['Job Title'].apply(categorize_job_title)
```

In [ ]:

```python
df['Education Level'].unique()
```

Out[ ]:

```
array(["Bachelor's", "Master's", 'PhD', "Bachelor's Degree",
       "Master's Degree", 'High School', 'phD'], dtype=object)
```

In the dataset the education level is represented in two different ways : Bachelor and Bachelor degree, which means same. So I will be grouping it with Bachelor

Grouping Education Level

In [ ]:

```
def group_education(Educaton):

    Educaton = str(Educaton).lower()
    if 'high school' in Educaton:
        return 'High School'
    elif 'bachelor\'s' in Educaton:
        return 'Bachelors'
    elif 'master\'s' in Educaton:
        return 'Masters'
    elif 'phd' in Educaton:
        return 'PhD'

df['Education Level'] = df['Education Level'].apply(group_education)
```

Descriptive Statistics

In [ ]:

```
#descriptive statistics
df.describe()
```

Out[ ]:

	Age	Years of Experience	Salary
count	6698.000000	6698.000000	6698.000000
mean	33.623022	8.095178	115329.253061
std	7.615784	6.060291	52789.792507
min	21.000000	0.000000	350.000000
25%	28.000000	3.000000	70000.000000
50%	32.000000	7.000000	115000.000000
75%	38.000000	12.000000	160000.000000
max	62.000000	34.000000	250000.000000

In [ ]:

```
df.head()
```

Out[ ]:

	Age	Gender	Education Level	Job Title	Years of Experience	Salary	Country	Ra
0	32.0	Male	Bachelors	Software/Developer	5.0	90000.0	UK	Whi
1	28.0	Female	Masters	Data Analyst/Scientist	3.0	65000.0	USA	Hispar
2	45.0	Male	PhD	Manager/Director/VP	15.0	150000.0	Canada	Whi
3	36.0	Female	Bachelors	Sales	7.0	60000.0	USA	Hispar
4	52.0	Male	Masters	Manager/Director/VP	20.0	200000.0	USA	Asi

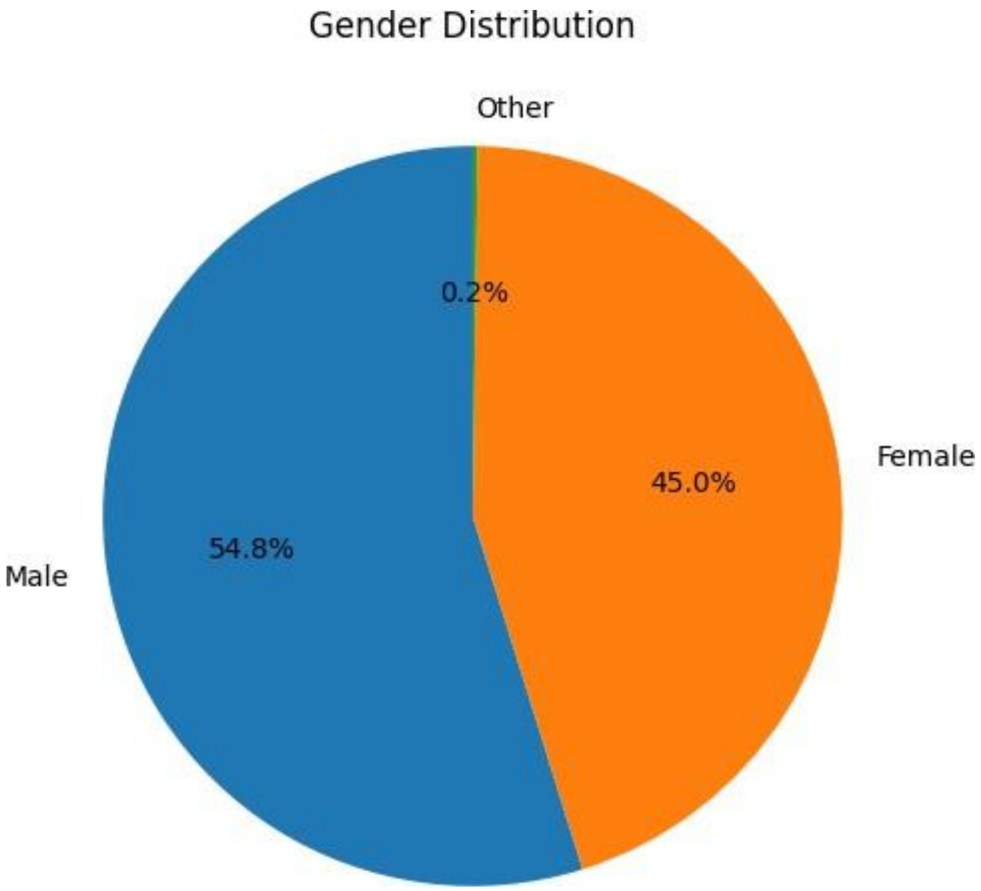
Exploratory Data Analysis

In the exploratory data analysis, I will be looking at the data and try to understand the data. I will begin by looking at the distribution of data across the datset, followed by visualizing the data to understand the relationship between the features and the target variable.

Pie chart for Gender

In [ ]:

```
#pie chart
plt.figure(figsize=(10,6))
plt.pie(df['Gender'].value_counts(), labels=['Male','Female', 'Other'], autopct=
plt.title('Gender Distribution')
plt.show()
```

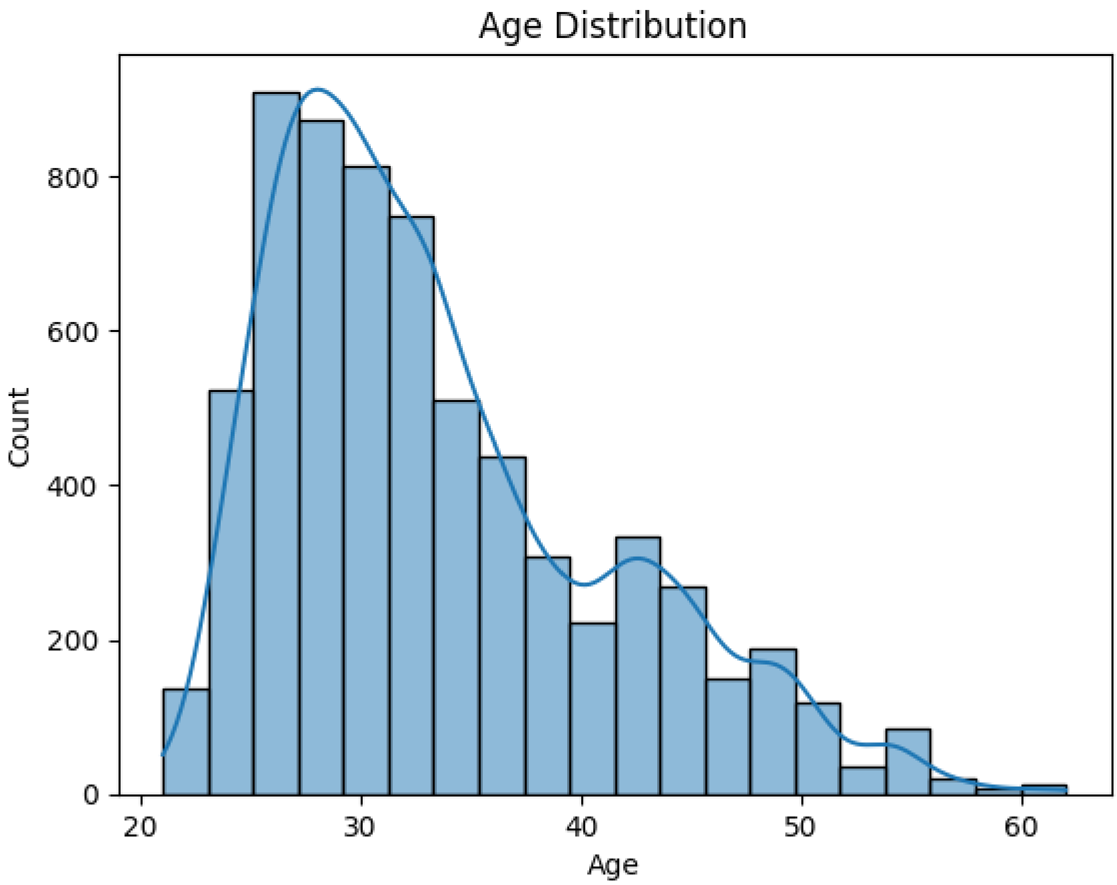


The pie chart shows that majority of the employees are male with 54.8 % on the dataset, followed by females with 45% and 0.2% employees belong to other gender.

Age Distribution

In [ ]:

```
sns.histplot(data=df, x='Age', bins=20, kde=True)
plt.title('Age Distribution')
plt.show()
```

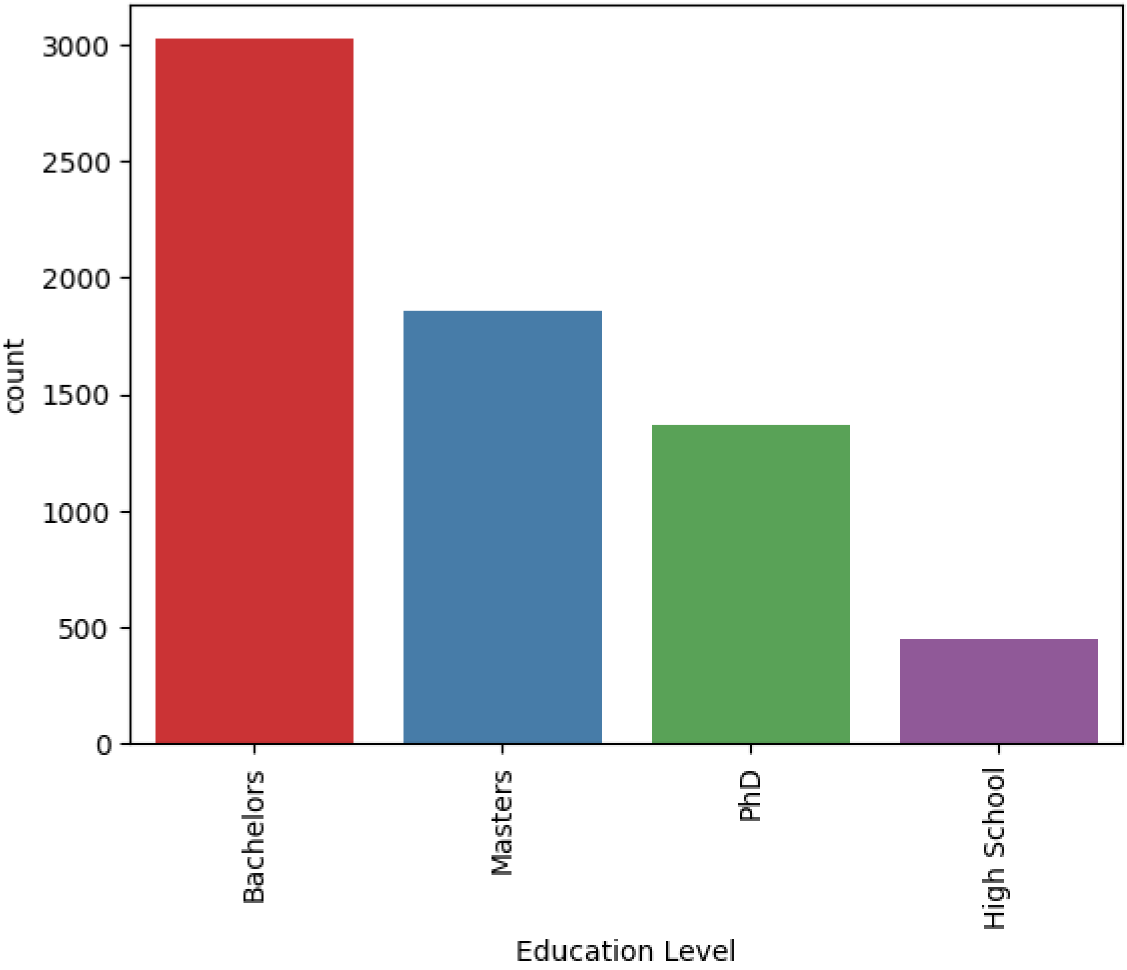


Majority of the employees are in the range of 25 - 35 years of age, which means majority of the employees are young and energetic. There is only minimal number of old employees in the dataset having age more than 55 years.

Education Level

```
In [ ]: sns.countplot(x = 'Education Level', data = df, palette='Set1')
        plt.xticks(rotation=90)
```

```
Out[ ]: (array([0, 1, 2, 3]),
         [Text(0, 0, 'Bachelors'),
          Text(1, 0, 'Masters'),
          Text(2, 0, 'PhD'),
          Text(3, 0, 'High School')])
```

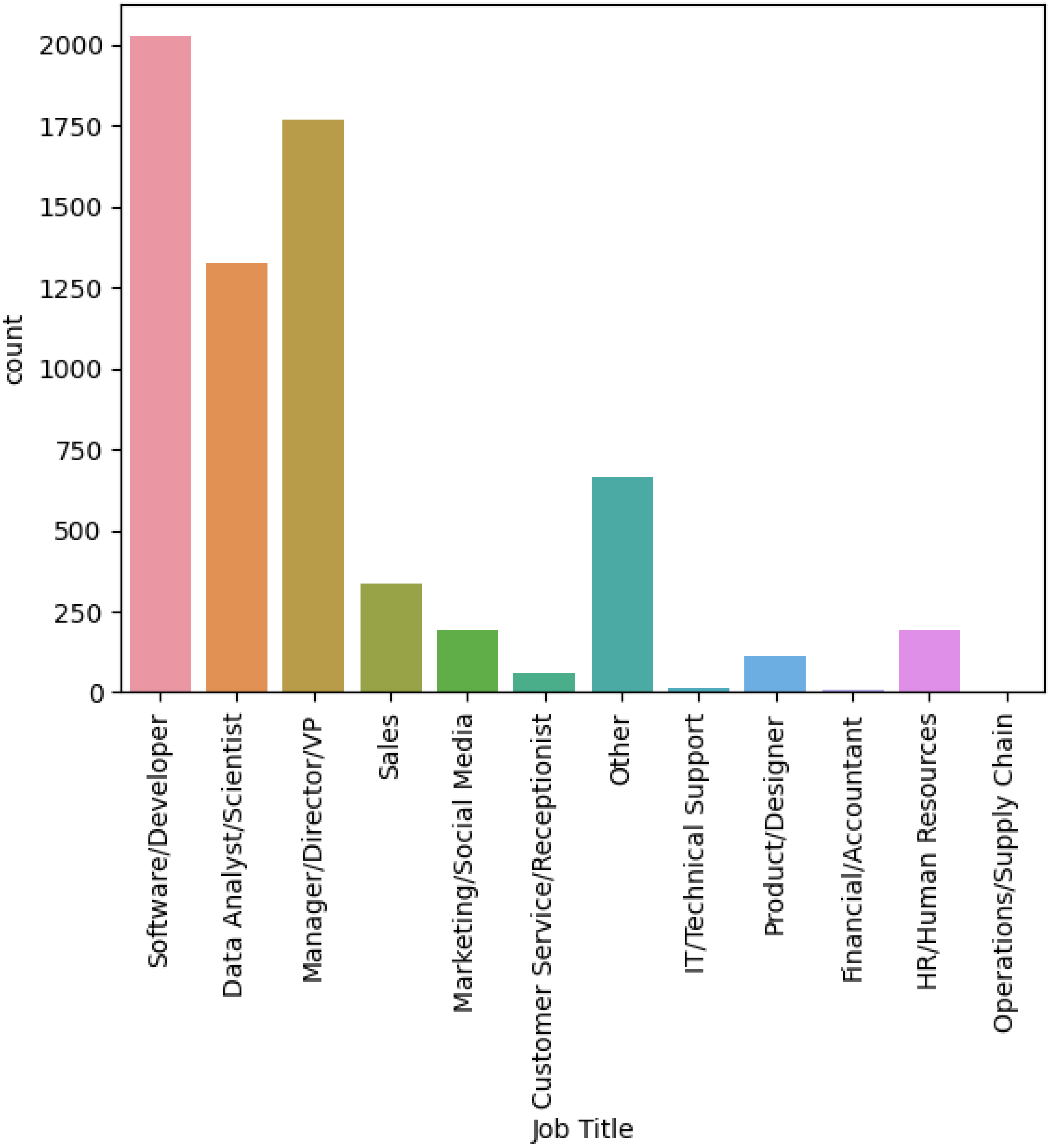


Most of the employees have a Bachelor's degree followed by Master's degree and Doctoral degree. The least number of employees have a High School education. From the graph it is clear that most of the employees started working after graduation, few of them started working after post graduation and very few of them have gone for doctorate. The least number of employees have started working after high school education.

Job Title

In [ ]:
```
sns.countplot(x='Job Title', data = df)
plt.xticks(rotation=90)
```

Out[ ]:
```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11]),
 [Text(0, 0, 'Software/Developer'),
  Text(1, 0, 'Data Analyst/Scientist'),
  Text(2, 0, 'Manager/Director/VP'),
  Text(3, 0, 'Sales'),
  Text(4, 0, 'Marketing/Social Media'),
  Text(5, 0, 'Customer Service/Receptionist'),
  Text(6, 0, 'Other'),
  Text(7, 0, 'IT/Technical Support'),
  Text(8, 0, 'Product/Designer'),
  Text(9, 0, 'Financial/Accountant'),
  Text(10, 0, 'HR/Human Resources'),
  Text(11, 0, 'Operations/Supply Chain')])
```



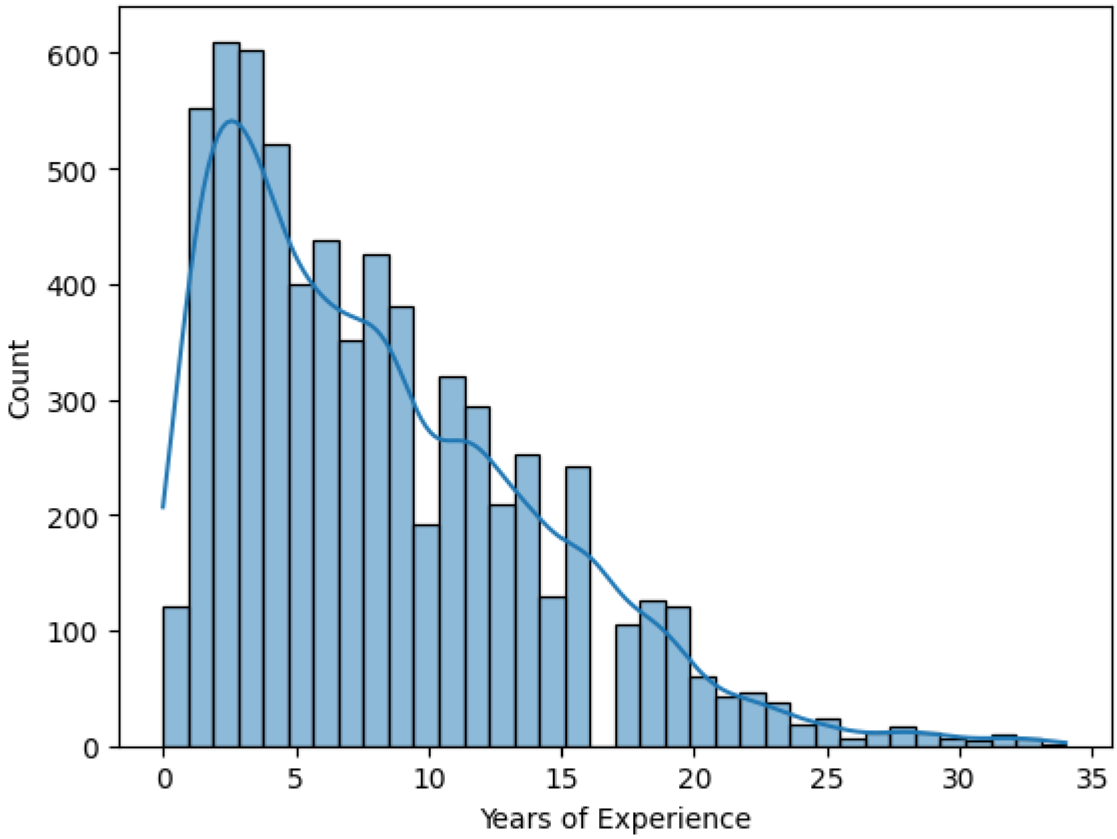
This graph helps us to breakdown the data of job title in a simpler form. From the graph, it is clear that majority of the employees have job titles - Software Developer, Data Analyst/Scientist or Manager/Director/Vp. Few amount of employees have job titles such as sales, marketing/social media, HR, Product Designer and Customer Service. Very few of the eomployees work as a Financial/accountant or operation/supply management.

From this I build a hypothesis that the job titles such as Software Developer, Data Analyst/Scientist and Manager/Director are in more demand as compared to other job titles. It also means that job titles like Financial/accountant or operation/supply management and Customer Service are in less demand and paid comparatively less.

Years of Experience

In [ ]:
```
sns.histplot(x = 'Years of Experience', data = df,kde=True)
```

Out[ ]:
```
<Axes: xlabel='Years of Experience', ylabel='Count'>
```



Most of the employees in the dataset havr experience of 0-7 years in the respective domains in which particularly majority of them have experience between less than 5 years. Moreover the number of employees in the dataset decreases with increasing number of years of experience.

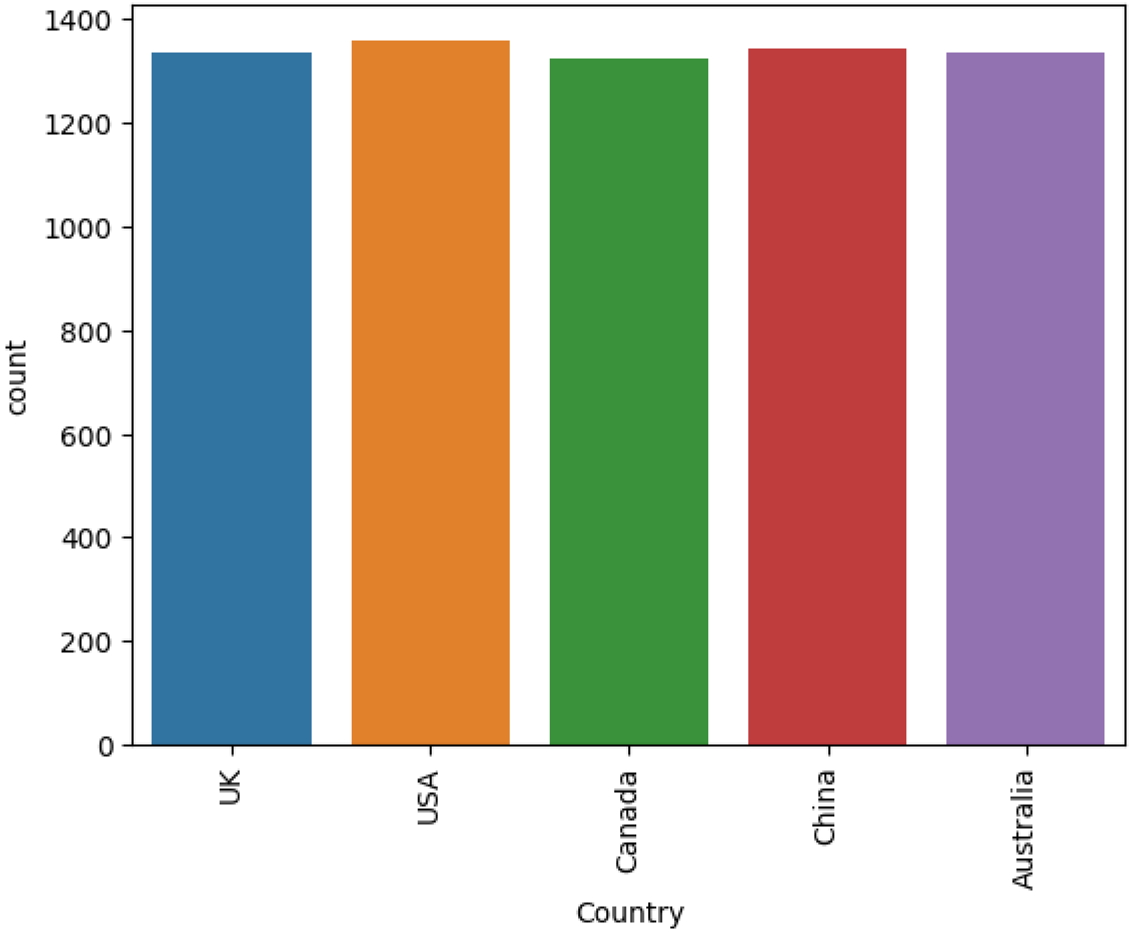
Country

In [ ]:

```
sns.countplot(x='Country', data=df)
plt.xticks(rotation=90)
```

Out[ ]:

```
(array([0, 1, 2, 3, 4]),
 [Text(0, 0, 'UK'),
  Text(1, 0, 'USA'),
  Text(2, 0, 'Canada'),
  Text(3, 0, 'China'),
  Text(4, 0, 'Australia')])
```

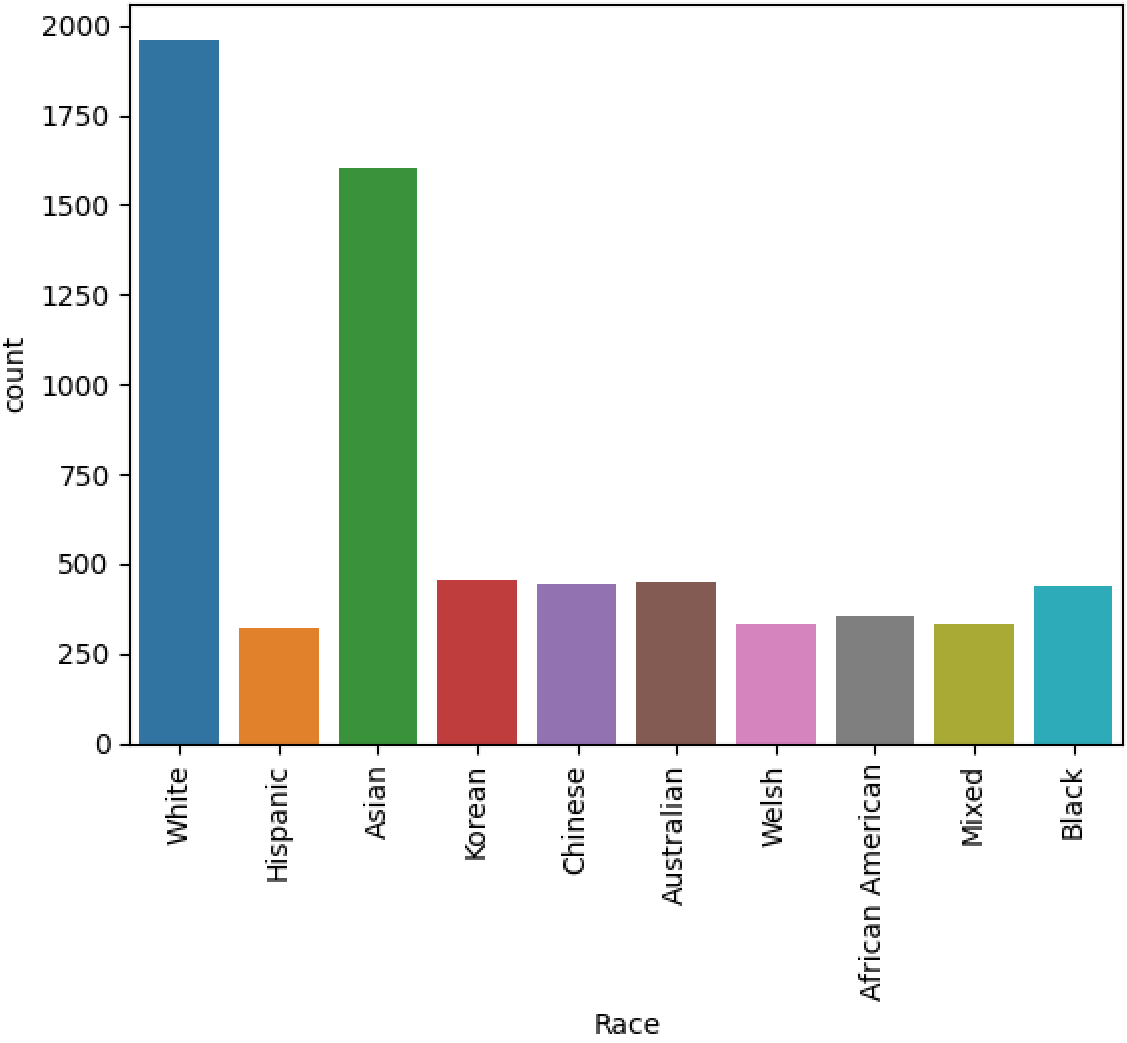


The number of employees from the above 5 countries is nearly same, with a little more in USA.

Racial Distribution

```
In [ ]: sns.countplot(x='Race', data=df)
        plt.xticks(rotation=90)
```

```
Out[ ]: (array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9]),
         [Text(0, 0, 'White'),
          Text(1, 0, 'Hispanic'),
          Text(2, 0, 'Asian'),
          Text(3, 0, 'Korean'),
          Text(4, 0, 'Chinese'),
          Text(5, 0, 'Australian'),
          Text(6, 0, 'Welsh'),
          Text(7, 0, 'African American'),
          Text(8, 0, 'Mixed'),
          Text(9, 0, 'Black')])
```



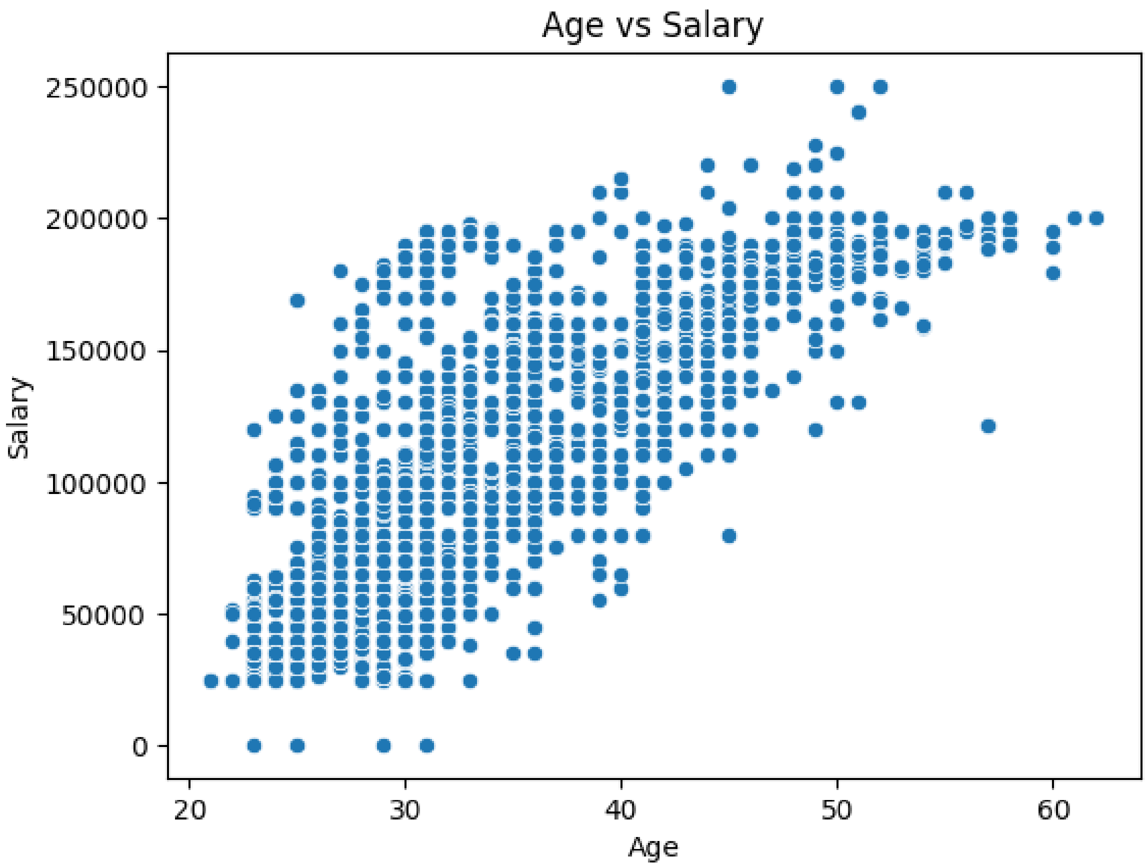
This graph help us to know about the racial distribution in the dataset. From the graph, it is clear that most of the employees are either White or Asian, followed by Korean, Chinese, Australian and Black. Number of employees from Welsh, African American, Mixed and Hispanic race are less as compared to other groups.

From all the above plots and graphs, we can a understanding about the data we are dealing with, its distribution and quantity as well. Now I am gonna explore the realtion of these independent variables with the target Variable i.e. Salary.

Age and Salary

In [ ]:
```
sns.scatterplot(x = 'Age', y='Salary', data=df)
plt.title('Age vs Salary')
```

Out[ ]:
```
Text(0.5, 1.0, 'Age vs Salary')
```



In this scatter plot we see a trend that the salary of the person increases with increse in the age, which is obvious because of promotion and apprisals. However upon closer observation we can find that similar age have multiple salaries, which means there are other factors which decides the salary.

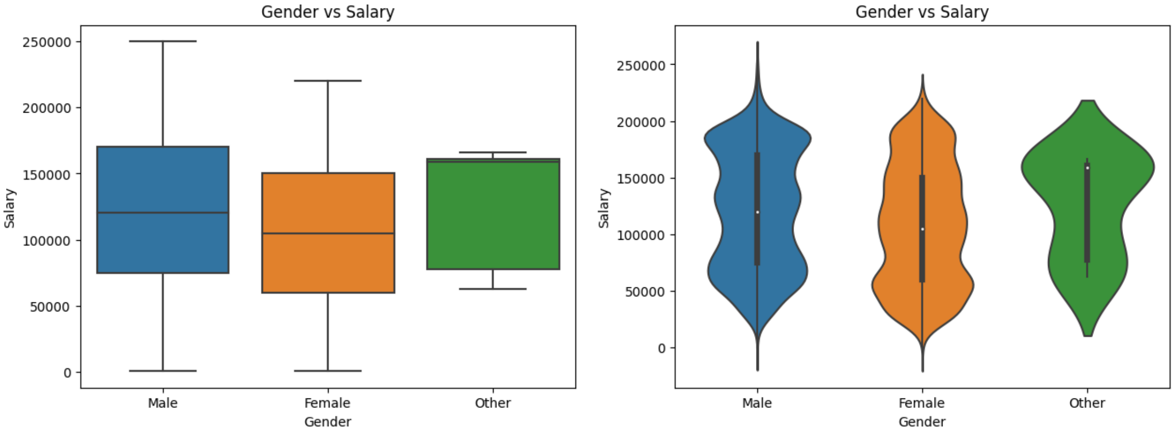
Gender and Salary

In [ ]:

```
fig, ax = plt.subplots(1,2, figsize = (15, 5))
sns.boxplot(x = 'Gender', y='Salary', data = df, ax =ax[0]).set_title('Gender vs
sns.violinplot(x = 'Gender', y='Salary', data = df, ax =ax[1]).set_title('Gender
```

Out[ ]:

```
Text(0.5, 1.0, 'Gender vs Salary')
```



The boxplot and violinplot describes the salary distribution among the three genders. In the boxplot the employees from Other gender has quite high salary as compared to Makes and Females. The other gender employees have a median salary above 150000, followed by males with median salary near 107500 and females with median salary near 100000. The voilin plot visualizes the distribution of salary with respect to the gender, where most of the Other gender employees have salary above 150000. In makes this distribution is concentrated between 50000 and 10000 as well as near 200000. In case of females, there salary distribution is quite spread as compared to other genders with most near 50000.

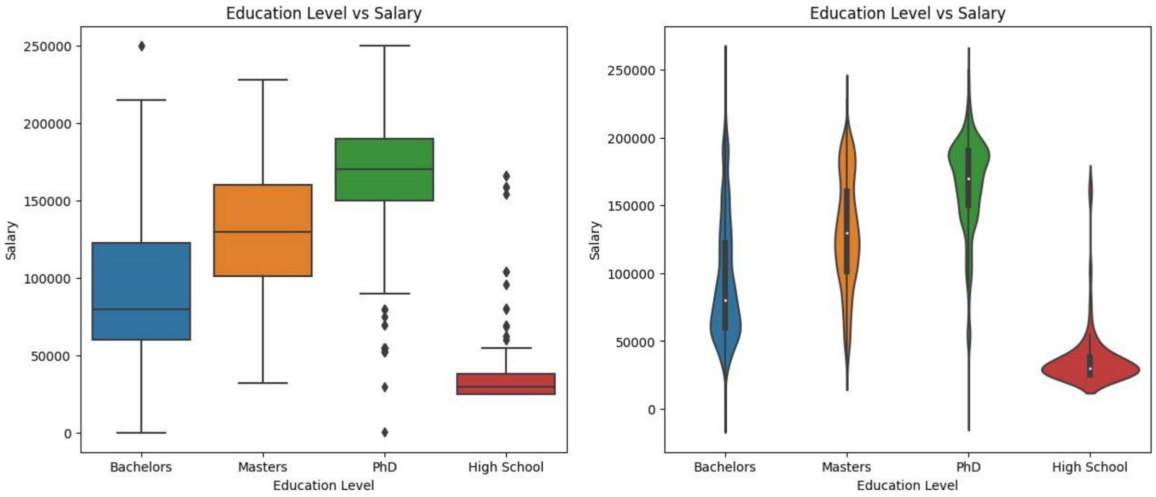
Education Level and Salary

In [ ]:

```
fig,ax = plt.subplots(1,2,figsize=(15,6))
sns.boxplot(x = 'Education Level', y = 'Salary', data = df, ax=ax[0]).set_title(
sns.violinplot(x = 'Education Level', y = 'Salary', data = df, ax=ax[1]).set_tit
```

Out[ ]:

```
Text(0.5, 1.0, 'Education Level vs Salary')
```



The boxplot and violinplot shows the distribution of salary based on the employees education level. The median salary for the Phd holders is highest followed by Masters and bachelors degreee holders, with employees with no degree having the lowest median salary. In the violinplot the phd scholars have distribution near 200000, whereas Masters degree holders have a very sleak distribution where the salary distribution is spread from 100k to 150k, The Bachelors degree holders have a salary distribution near 50000 whereas the employees with no degree have a salary distribution near 40k-45k.

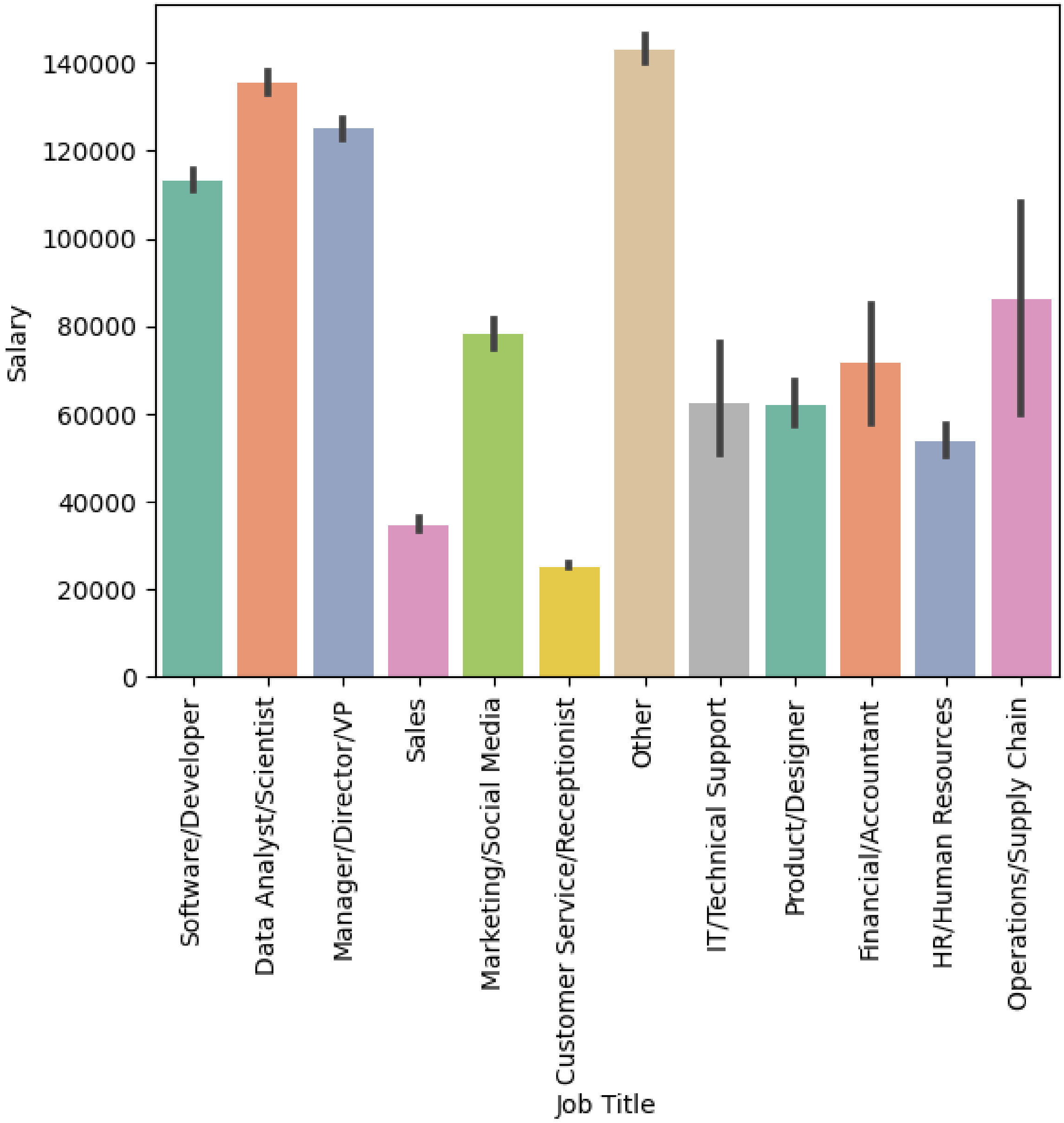
From these graph, I assume that the employees with higher education level have higher salary than the employees with lower education level.

Job Title and Salary

In [ ]:

```
sns.barplot(x = 'Job Title', y = 'Salary', data = df, palette = 'Set2')
plt.xticks(rotation = 90)
```

```
Out[ ]:  (array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11]),
          [Text(0, 0, 'Software/Developer'),
           Text(1, 0, 'Data Analyst/Scientist'),
           Text(2, 0, 'Manager/Director/VP'),
           Text(3, 0, 'Sales'),
           Text(4, 0, 'Marketing/Social Media'),
           Text(5, 0, 'Customer Service/Receptionist'),
           Text(6, 0, 'Other'),
           Text(7, 0, 'IT/Technical Support'),
           Text(8, 0, 'Product/Designer'),
           Text(9, 0, 'Financial/Accountant'),
           Text(10, 0, 'HR/Human Resources'),
           Text(11, 0, 'Operations/Supply Chain')])
```



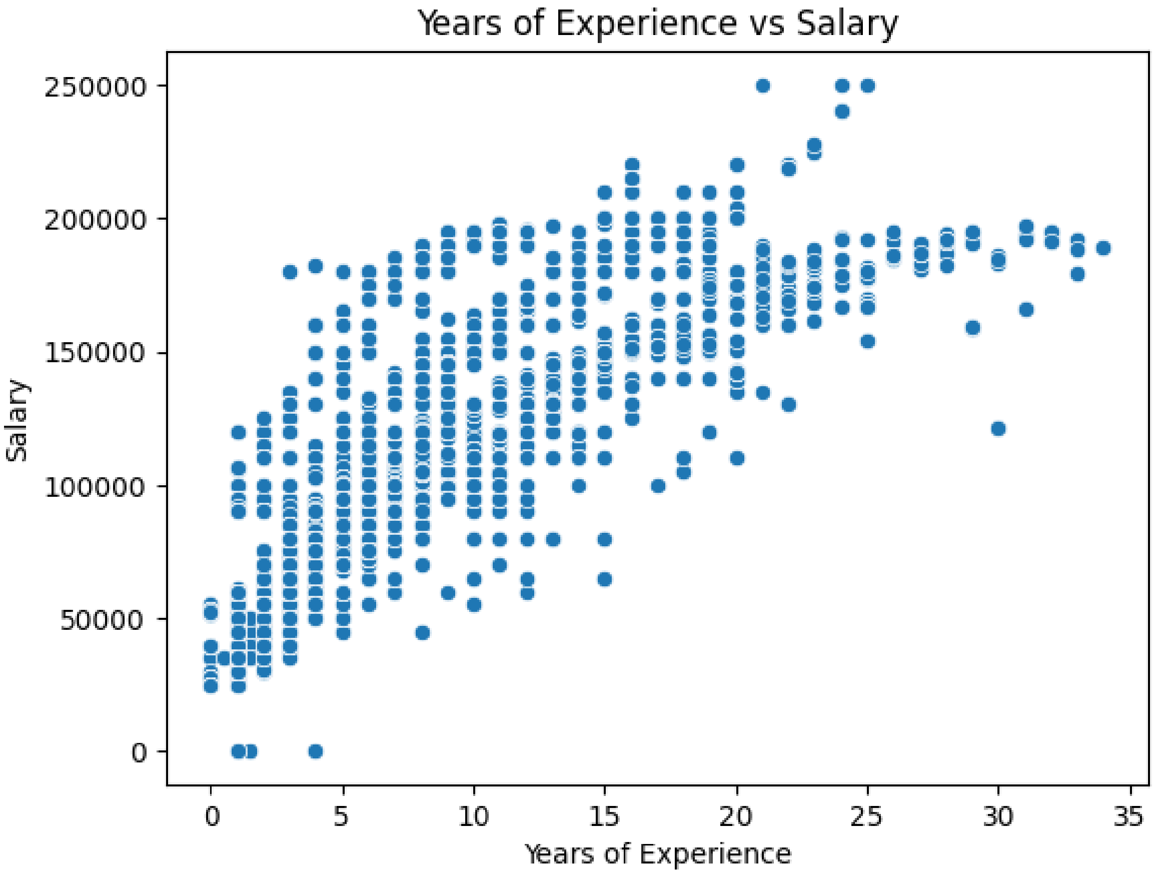
This graph falsifies my previous hypothesis regarding the demand and paywith respect to job titles. In this graph, 'Other' category job titles have higher salary than those titles which assumed to be in high demand and pay. In contrast to previous Job title graph, this graph shows that there is no relation between the job title distribution and salary. The job titles which gave high salary are found to be less in number.

However the hypothesis is true about the Job titles such as Software Developer, Data analyst/scuentust and Manager/Director/VP. These job titles are found to be in high demand and pay. But in contrast to that the job titles such as Operation/Supply chain, HR, Financial/Accountant and Marketing/Social Media are found have much more salary as assumed.

Experience and Salary

```
In [ ]: sns.scatterplot(x= 'Years of Experience', y  = 'Salary', data = df).set_title('Y
```

```
Out[ ]: Text(0.5, 1.0, 'Years of Experience vs Salary')
```

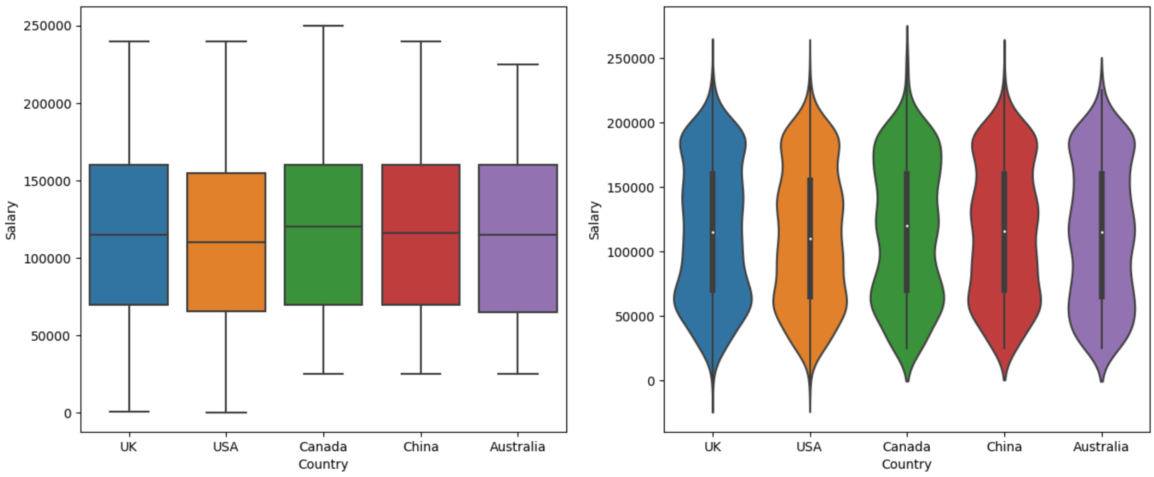


From this scaaterplot, it is clear that on the whole, the salary of the employees is increasing with the years of experience. However, on closer look we can see that similar experience have different salaries. This is because the salary is also dependent on other factors like job title, age, gender education level as discussed earlier.

Country and Salary

```
In [ ]: fig,ax = plt.subplots(1,2,figsize=(15,6))
        sns.boxplot(x = 'Country', y = 'Salary', data = df, ax=ax[0])
        sns.violinplot(x = 'Country', y = 'Salary', data = df, ax=ax[1])
```

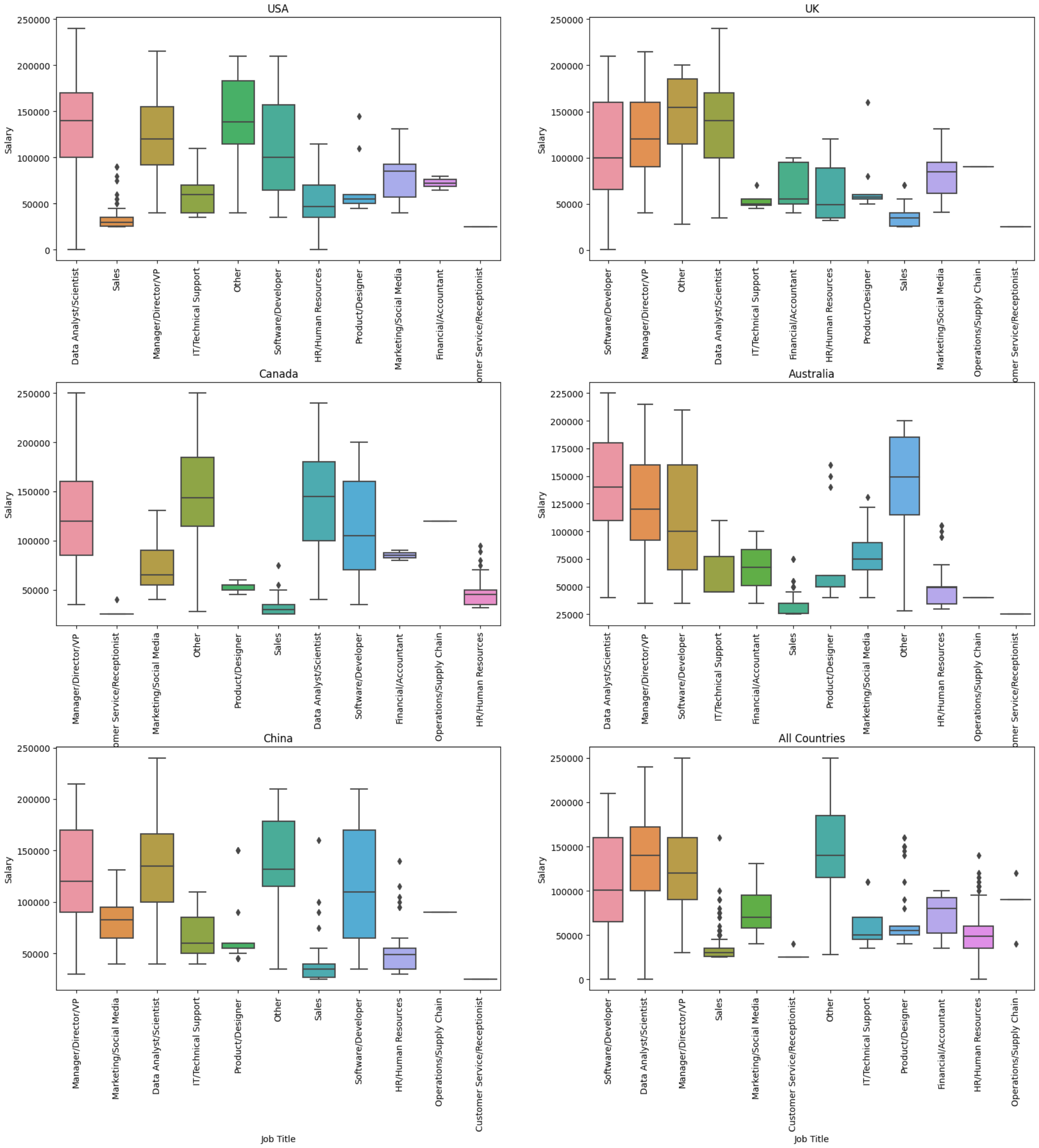
```
Out[ ]: <Axes: xlabel='Country', ylabel='Salary'>
```



Both the boxplot and violinplot shows very similar insight about the salary across all the countiries even in the violinplot distribution. However, there is very small variation in median salary in USA, which is slighlty less as compared to other countries.

Since, the we cannot get much information about the salary with respect to the countries. So, I will plot the job title vs salary graph for each country, so that we can get a overview of job title vs salary for each country.

```
In [ ]: fig,ax = plt.subplots(3,2,figsize=(20,20))
        plt.subplots_adjust(hspace=0.5)
        sns.boxplot(x = 'Job Title', y = 'Salary', data = df[df['Country'] == 'USA'], ax
        ax[0,0].tick_params(axis='x', rotation=90)
        sns.boxplot(x = 'Job Title', y = 'Salary', data = df[df['Country'] == 'UK'], ax
        ax[0,1].tick_params(axis='x', rotation=90)
        sns.boxplot(x = 'Job Title', y = 'Salary', data = df[df['Country'] == 'Canada'],
        ax[1,0].tick_params(axis='x', rotation=90)
        sns.boxplot(x = 'Job Title', y = 'Salary', data = df[df['Country'] == 'Australia
        ax[1,1].tick_params(axis='x', rotation=90)
        sns.boxplot(x = 'Job Title', y = 'Salary', data = df[df['Country'] == 'China'],
        ax[2,0].tick_params(axis='x', rotation=90)
        sns.boxplot(x = 'Job Title', y = 'Salary', data = df, ax = ax[2,1]).set_title('A
        ax[2,1].tick_params(axis='x', rotation=90)
```

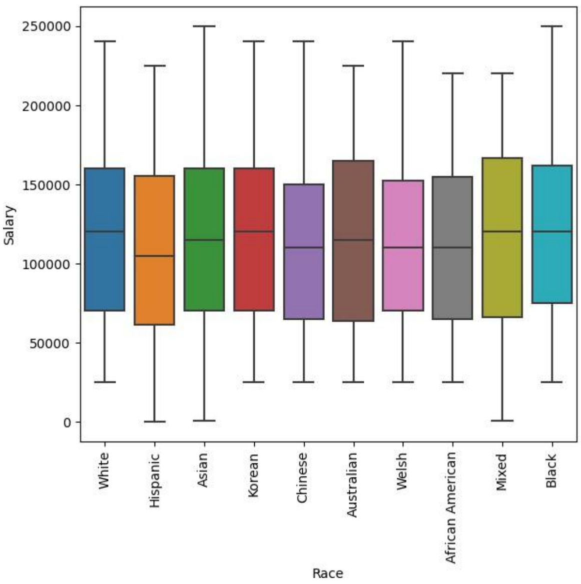


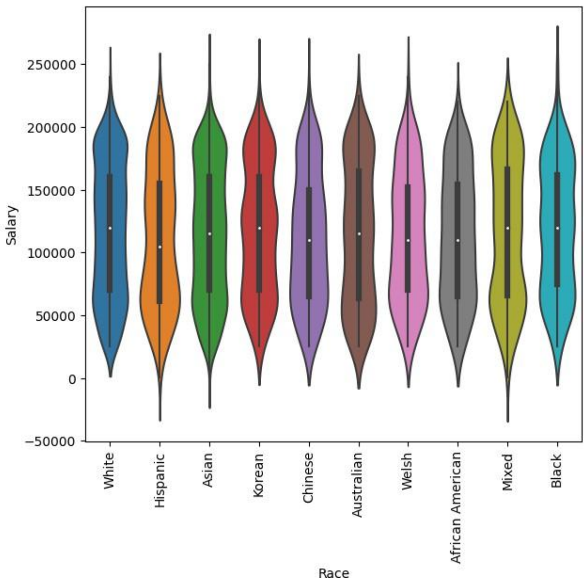
After observing all these plots, I conclude that the Job Titles such as Softwarre Developer, Manager/Director/VP and Data Analyst/Scientist hare in high demand as well as receive much higer salary than other job titles, excluding the Job Titles that come under 'Other' category. The job titles such as Operation/Supply Chain, Customer Service/Receptionist, Product Designer and sales are in low demand and have low salary.

Race and Salary

In [ ]:

```
fig,ax = plt.subplots(1,2,figsize=(15,6))
sns.boxplot(x = 'Race', y = 'Salary', data = df, ax = ax[0])
ax[0].tick_params(axis='x', rotation=90)
sns.violinplot(x = 'Race', y ='Salary', data = df, ax = ax[1])
ax[1].tick_params(axis='x', rotation=90)
```





The employees from the races - Australian, Mixed, Blacks and White have the highest median salary, followed by Asian, Korean and Chinese with lowest median salary in employees from hispanic race. Looking at the violinplot the salary distribution is more concentrated after 150k in white, australian, black and mixed race. Whereas the hispanic has more concentration near 75k

Data Preprocessing 2

Label encoding to categorical features

In [ ]:

```
from sklearn.preprocessing import LabelEncoder
features = ['Gender','Country','Education Level','Job Title', 'Race']
le = LabelEncoder()
for feature in features:
    le.fit(df[feature].unique())
    df[feature] = le.transform(df[feature])
    print(feature, df[feature].unique())
```

```
Gender [1 0 2]
Country [3 4 1 2 0]
Education Level [0 2 3 1]
Job Title [11  1  5 10  6  0  8  4  9  2  3  7]
Race [9 5 1 6 4 2 8 0 7 3]
```

Normalization

In [ ]:

```
#normalizing the continuous variables
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df[['Age', 'Years of Experience', 'Salary']] = scaler.fit_transform(df[['Age', '
```

In [ ]:

```
df.head()
```

Out[ ]:

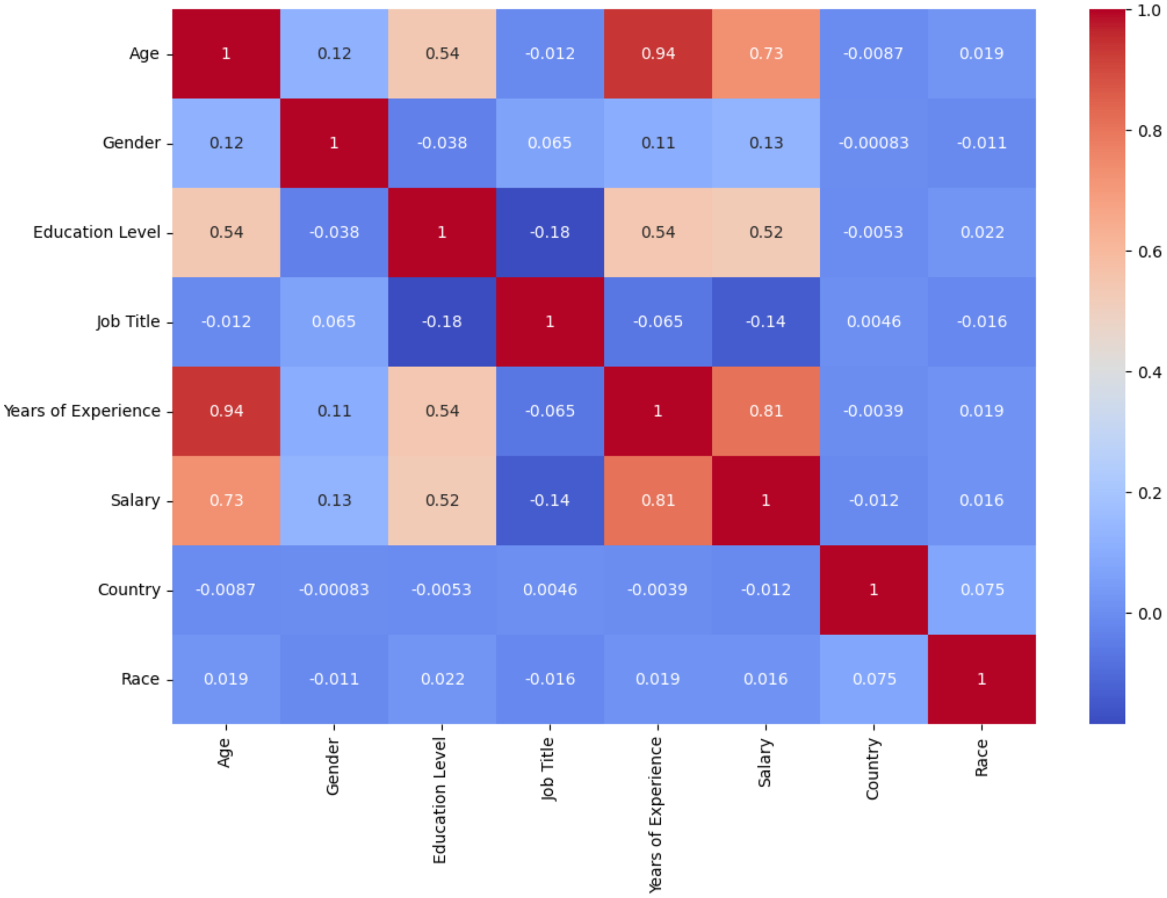
	Age	Gender	Education Level	Job Title	Years of Experience	Salary	Country	Race
0	-0.213129	1	0	11	-0.510769	-0.479849	3	9
1	-0.738393	0	2	1	-0.840811	-0.953461	4	5
2	1.493980	1	3	5	1.139440	0.656819	1	9
3	0.312135	0	0	10	-0.180727	-1.048183	4	5
4	2.413192	1	2	5	1.964544	1.604042	4	1

Coorelation Matrix Heatmap

In [ ]:

```
#coorelation heatmap
plt.figure(figsize=(12,8))
sns.heatmap(df.corr(),annot=True,  cmap='coolwarm')
```

Out[ ]: `<Axes: >`



In this coorelation matrix, there are three major coorealtions.

- Salary and Age
- Salary and Years of Experience
- Years of Experience and Age

The coorelation salary with age and years of experience is already explored in the above plots. The coorelation between the years of experience and age is obvious as the person ages the experience will be more.

Train Test Split

In [ ]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(df.drop('Salary', axis=1), d
```

Salary Prediction

I will be using the following models:

- Decision Tree Regressor
- Random Forest Regressor

Decision Tree Regressor

In [ ]:
```
from sklearn.tree import DecisionTreeRegressor

#createing the decision tree gressor object
dtree = DecisionTreeRegressor()
```

Hypertuning the model

In [ ]:
```
from sklearn.model_selection import GridSearchCV

#defining the parameters for the grid search
parameters = {'max_depth' :[2,4,6,8,10],
              'min_samples_split' :[2,4,6,8],
              'min_samples_leaf' :[2,4,6,8],
              'max_features' :['auto','sqrt','log2'],
              'random_state' :[0,42]}
#creating the grid search object
grid_search  =  GridSearchCV(dtree,parameters,cv=5,scoring='neg_mean_squared_error

#fit the grid search object to the training data
grid_search.fit(X_train,y_train)

#print the best parameters
print(grid_search.best_params_)
```

```
{'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_spl
it': 8, 'random_state': 42}
```

Building the model on best parameters

In [ ]:
```
dtree = DecisionTreeRegressor(max_depth = 10, max_features = 'auto', min_samples
dtree
```

Out[ ]:

```
▾                    DecisionTreeRegressor
DecisionTreeRegressor(max_depth=10, max_features='auto', min_samples_le
af=2,
                      min_samples_split=8, random_state=42)
```

In [ ]:

```
#fitting the training data
dtree.fit(X_train,y_train)
```

Out[ ]:

```
▾                    DecisionTreeRegressor
DecisionTreeRegressor(max_depth=10, max_features='auto', min_samples_le
af=2,
                      min_samples_split=8, random_state=42)
```

In [ ]:

```
#training accuracy
dtree.score(X_train,  y_train)
```

Out[ ]:

```
0.9656459784687974
```

In [ ]:

```
#predicting the salary of an employee
d_pred = dtree.predict(X_test)
```

Evaluating the Decision Tree Regressor Model

In [ ]:

```
dft = pd.DataFrame({'Actual': y_test, 'Predicted': d_pred})
dft.reset_index(drop=True, inplace=True)
dft.head(10)
```

Out[ ]:

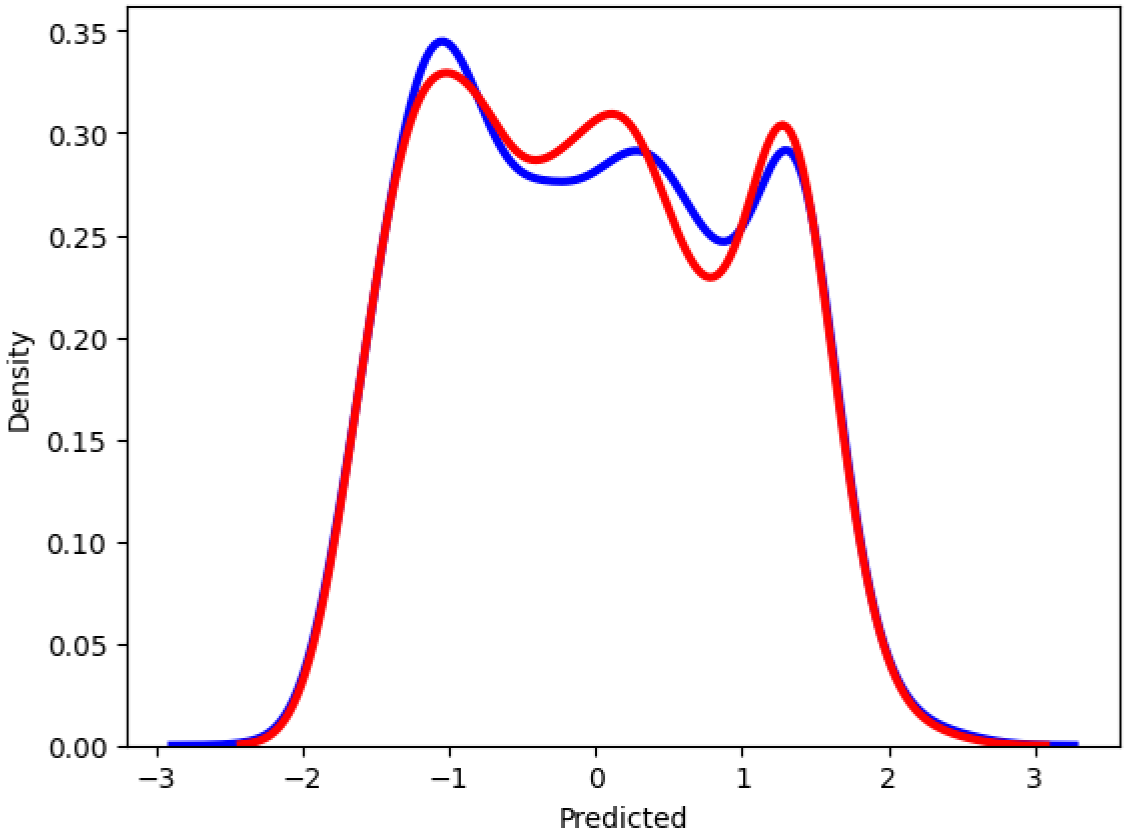
	Actual	Predicted
0	0.656819	0.678470
1	-0.745659	-0.688434
2	-0.290405	-0.290405
3	-1.048183	-1.036343
4	-0.669294	-0.610093
5	1.414598	1.494747
6	-0.820850	-0.715794
7	-1.142906	-1.122777
8	1.509320	1.554189
9	0.277930	0.287811

In [ ]:

```
ax = sns.distplot(dft['Actual'], color = 'blue', hist = False, kde = True, kde_k
sns.distplot(  dft['Predicted'], color = 'red', ax=ax, hist = False, kde = True,
```

Out[ ]:

```
<Axes: xlabel='Predicted', ylabel='Density'>
```



The blue shows the distribution count for actual values and the red line shows the distribution count for predicted values. The predicted values are close to the actual values and ther curve coincides with the actual values curve. This shows that the model is a good fit.

In [ ]:

```
from sklearn.metrics import r2_score, mean_squared_error, mean_absolute_error
print("R2 Score: ", r2_score(y_test, d_pred))
print("Mean Squared Error: ", mean_squared_error(y_test, d_pred))
print("Mean Absolute Error: ", mean_absolute_error(y_test, d_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, d_pred)))
```

```
R2 Score:  0.9323013355107719
Mean Squared Error:  0.06928069008068977
Mean Absolute Error:  0.13812719621413622
RMSE: 0.2632122529075912
```

Random Forest Regressor

In [ ]:

```
from sklearn.ensemble import RandomForestRegressor
#creating random forest regressor object
rfg = RandomForestRegressor()
```

In [ ]:

```
#trainig the model
rfg.fit(X_train, y_train)
```

Out[ ]:

```
▾ RandomForestRegressor
RandomForestRegressor()
```

In [ ]:
```
#training accuracy
rfg.score(X_train, y_train)
```

Out[ ]:
```
0.9881489086015691
```

In [ ]:
```
#predicitng salary of the employee
r_pred = rfg.predict(X_test)
```

Evaluating Random Forest Regressor Model

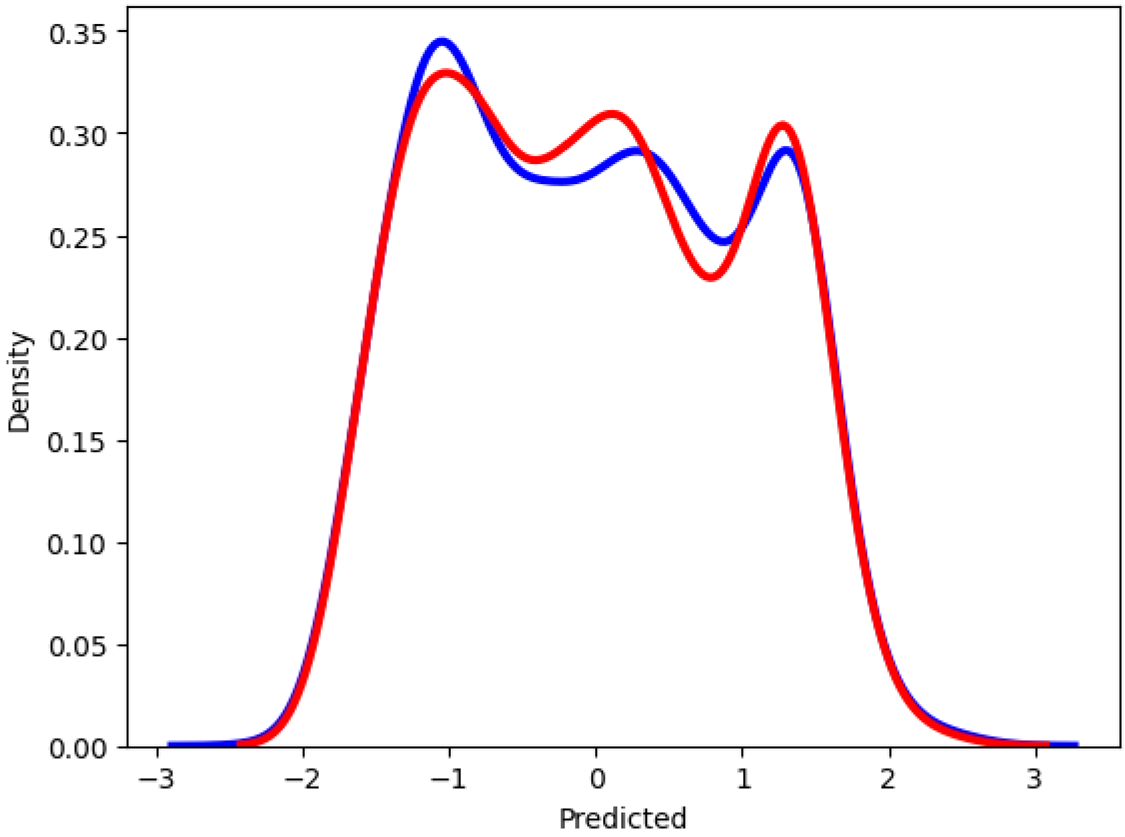
In [ ]:
```
dfr = pd.DataFrame({'Actual': y_test, 'Predicted': r_pred})
dfr.reset_index(drop=True, inplace=True)
dfr.head(10)
```

Out[ ]:

	Actual	Predicted
0	0.656819	0.648206
1	-0.745659	-0.716941
2	-0.290405	-0.288510
3	-1.048183	-1.049699
4	-0.669294	-0.637562
5	1.414598	1.501506
6	-0.820850	-0.813651
7	-1.142906	-1.113062
8	1.509320	1.541334
9	0.277930	0.306604

In [ ]:
```
ax = sns.distplot(dft['Actual'], color = 'blue', hist = False, kde = True, kde_k
sns.distplot(  dft['Predicted'], color = 'red', ax=ax, hist = False, kde = True,
```

Out[ ]:
```
<Axes: xlabel='Predicted', ylabel='Density'>
```



The blue shows the distribution count for actual values and the red line shows the distribution count for predicted values. The predicted values are close to the actual values and ther curve coincides with the actual values curve. This shows that the model is a good fit.

In [ ]:

```
print("R2 Score: ", r2_score(y_test, r_pred))
print("Mean Squared Error: ", mean_squared_error(y_test, r_pred))
print("Mean Absolute Error: ", mean_absolute_error(y_test, r_pred))
print('RMSE:', np.sqrt(mean_squared_error(y_test, r_pred)))
```

```
R2 Score:  0.946740751192265
Mean Squared Error:  0.05450384491951317
Mean Absolute Error:  0.11418652633630026
RMSE: 0.23346058536616662
```

Conclusion

From the exploratory data analysis, I have concluded that the salary of the employees is dependent upon the following factors:

1. **Years of Experience**
2. **Job Title**
3. **Education Level**

Employees with greater years of experience, having job title such as Data analyst/scientist, Software Developer or Director/Manager/VP and having a Master's or Doctoral degree are more likely to have a higher salary.

Coming to the machine learning models, I have used regressor models - Decision Tree Regressor and Random Forest Regressor for predicting the salary. The Random Forest Regressor has performed well with the accuracy of 94.6%