In [68]:
```
# William Barker
# DSC680
# Project 1

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# downloading my dataset
df = pd.read_csv('salary_prediction_data.csv', encoding = "ISO-8859-1")
df
```

Out[68]:

	Education	Experience	Location	Job_Title	Age	Gender	Salary
0	High School	8	Urban	Manager	63	Male	84620.053665
1	PhD	11	Suburban	Director	59	Male	142591.255894
2	Bachelor	28	Suburban	Manager	61	Female	97800.255404
3	High School	29	Rural	Director	45	Male	96834.671282
4	PhD	25	Urban	Analyst	26	Female	132157.786175
...	...	...	...	...	...	...	...
995	High School	8	Suburban	Analyst	25	Female	64683.389864
996	High School	24	Urban	Engineer	30	Female	74468.205020
997	Master	18	Rural	Analyst	44	Male	98207.026024
998	Bachelor	27	Suburban	Director	31	Female	108544.922720
999	High School	25	Urban	Director	41	Female	71077.000066

1000 rows × 7 columns

In [69]:
```
# checking the unique values
education_values = df['Education'].unique()
education_values
```

Out[69]:
```
array(['High School', 'PhD', 'Bachelor', 'Master'], dtype=object)
```

In [70]:
```
# checking the unique values
location_values = df['Location'].unique()
location_values
```

Out[70]:
```
array(['Urban', 'Suburban', 'Rural'], dtype=object)
```

In [71]:
```
# checking the unique values
job_values = df['Job_Title'].unique()
job_values
```

Out[71]:
```
array(['Manager', 'Director', 'Analyst', 'Engineer'], dtype=object)
```

In [72]:
```
# mapping the unique values to numbers
df['Education'] = df['Education'].map({'High School': 1, 'Bachelor': 2, 'Master': 3, 'PhD'
df
```

Out[72]:

	Education	Experience	Location	Job_Title	Age	Gender	Salary
0	1	8	Urban	Manager	63	Male	84620.053665
1	4	11	Suburban	Director	59	Male	142591.255894
2	2	28	Suburban	Manager	61	Female	97800.255404
3	1	29	Rural	Director	45	Male	96834.671282
4	4	25	Urban	Analyst	26	Female	132157.786175
...	...	...	...	...	...	...	...
995	1	8	Suburban	Analyst	25	Female	64683.389864
996	1	24	Urban	Engineer	30	Female	74468.205020
997	3	18	Rural	Analyst	44	Male	98207.026024
998	2	27	Suburban	Director	31	Female	108544.922720
999	1	25	Urban	Director	41	Female	71077.000066

1000 rows × 7 columns

In [73]:

```
# mapping the unique values to numbers
df['Location'] = df['Location'].map({'Urban': 1, 'Suburban': 2, 'Rural': 3})
df
```

Out[73]:

	Education	Experience	Location	Job_Title	Age	Gender	Salary
0	1	8	1	Manager	63	Male	84620.053665
1	4	11	2	Director	59	Male	142591.255894
2	2	28	2	Manager	61	Female	97800.255404
3	1	29	3	Director	45	Male	96834.671282
4	4	25	1	Analyst	26	Female	132157.786175
...	...	...	...	...	...	...	...
995	1	8	2	Analyst	25	Female	64683.389864
996	1	24	1	Engineer	30	Female	74468.205020
997	3	18	3	Analyst	44	Male	98207.026024
998	2	27	2	Director	31	Female	108544.922720
999	1	25	1	Director	41	Female	71077.000066

1000 rows × 7 columns

In [74]:

```
# mapping the unique values to numbers
df['Job_Title'] = df['Job_Title'].map({'Analyst': 1, 'Engineer': 2, 'Manager': 3, 'Directo
df
```

Out[74]:

	Education	Experience	Location	Job_Title	Age	Gender	Salary
0	1	8	1	3	63	Male	84620.053665
1	4	11	2	4	59	Male	142591.255894
2	2	28	2	3	61	Female	97800.255404

	Education	Experience	Location	Job_Title	Age	Gender	Salary
3	1	29	3	4	45	Male	96834.671282
4	4	25	1	1	26	Female	132157.786175
...	...	...	...	...	...	...	...
995	1	8	2	1	25	Female	64683.389864
996	1	24	1	2	30	Female	74468.205020
997	3	18	3	1	44	Male	98207.026024
998	2	27	2	4	31	Female	108544.922720
999	1	25	1	4	41	Female	71077.000066

1000 rows × 7 columns

In [75]:

```
# mapping the unique values to numbers
df['Gender'] = df['Gender'].map({'Male': 1, 'Female': 2})
df
```

Out[75]:

	Education	Experience	Location	Job_Title	Age	Gender	Salary
0	1	8	1	3	63	1	84620.053665
1	4	11	2	4	59	1	142591.255894
2	2	28	2	3	61	2	97800.255404
3	1	29	3	4	45	1	96834.671282
4	4	25	1	1	26	2	132157.786175
...	...	...	...	...	...	...	...
995	1	8	2	1	25	2	64683.389864
996	1	24	1	2	30	2	74468.205020
997	3	18	3	1	44	1	98207.026024
998	2	27	2	4	31	2	108544.922720
999	1	25	1	4	41	2	71077.000066

1000 rows × 7 columns

In [76]:

```
# checking for Nan values
missing_values = df.isnull().sum()
missing_values
```

Out[76]:

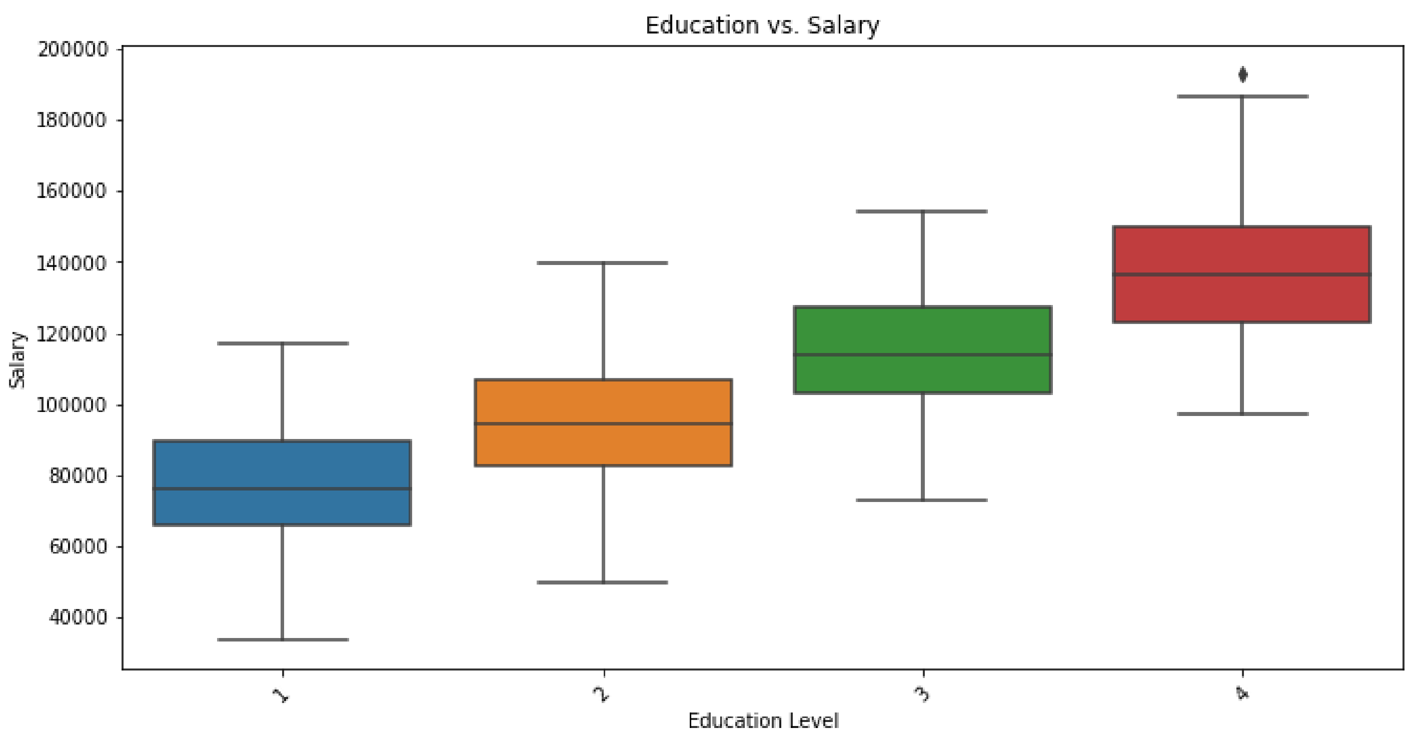
```
Education     0
Experience    0
Location      0
Job_Title     0
Age           0
Gender        0
Salary        0
dtype: int64
```

In [77]:

```
# Scatter plot for Experience vs. Salary
plt.figure(figsize=(10, 6))
sns.scatterplot(data=df, x='Experience', y='Salary')
```

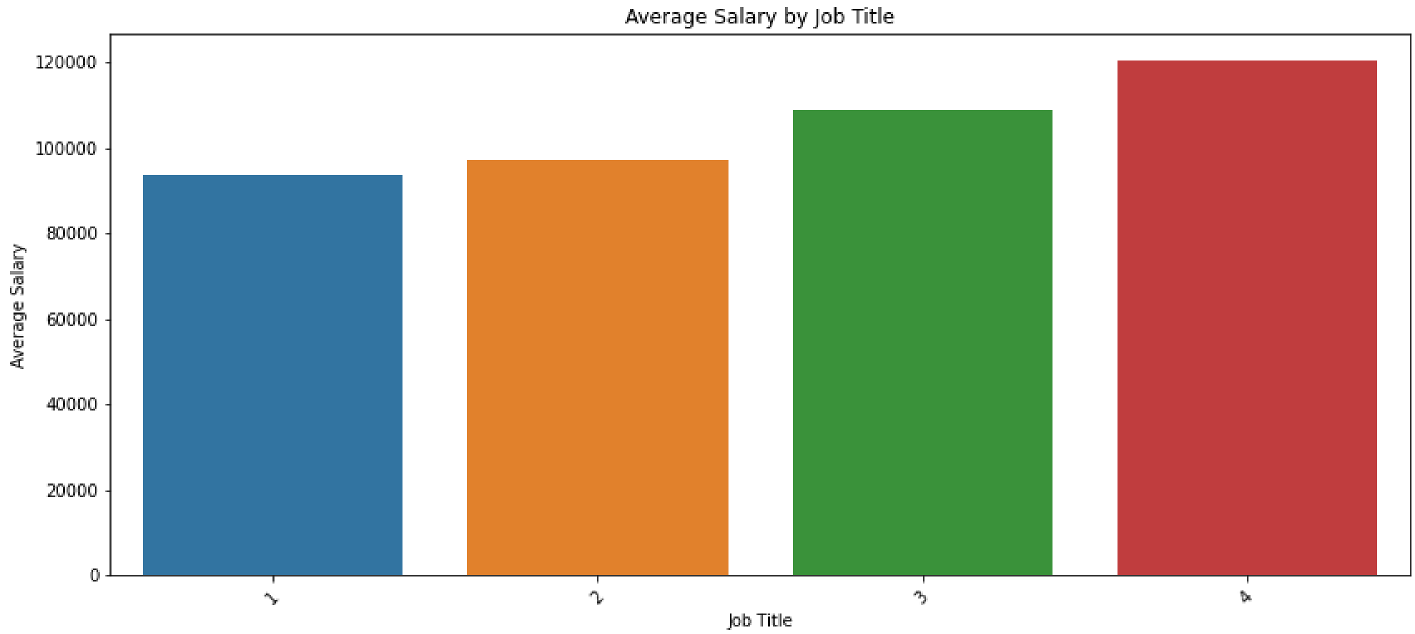
```
plt.title('Experience vs. Salary')
plt.xlabel('Experience (Years)')
plt.ylabel('Salary')
plt.show()
```

In [78]:
```
# Box plot for Education vs. Salary
plt.figure(figsize=(12, 6))
sns.boxplot(data=df, x='Education', y='Salary')
plt.title('Education vs. Salary')
plt.xlabel('Education Level')
plt.ylabel('Salary')
plt.xticks(rotation=45)
plt.show()
```



In [79]:

```
# Bar plot for Job Title vs. Average Salary
plt.figure(figsize=(14, 6))
average_salary_by_job_title = df.groupby('Job_Title')['Salary'].mean().reset_index()
sns.barplot(data=average_salary_by_job_title, x='Job_Title', y='Salary')
plt.title('Average Salary by Job Title')
plt.xlabel('Job Title')
plt.ylabel('Average Salary')
plt.xticks(rotation=45)
plt.show()
```



In [80]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score

# Selecting features and target variable
features = ['Education', 'Experience', 'Location', 'Job_Title', 'Age', 'Gender']
target = 'Salary'

X = df[features]
y = df[target]
```

In [81]:
```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

In [82]:
```
from sklearn.linear_model import LinearRegression

# Initialize the linear regression model
lr_model = LinearRegression()

# Train the linear regression model
lr_model.fit(X_train, y_train)

# Predict on the test set using linear regression
lr_y_pred = lr_model.predict(X_test)

# Calculate evaluation metrics for linear regression
lr_mse = mean_squared_error(y_test, lr_y_pred)
lr_mae = mean_absolute_error(y_test, lr_y_pred)
```

```python
lr_r2 = r2_score(y_test, lr_y_pred)

print("Linear Regression Metrics:")
print(f"Mean Squared Error: {lr_mse}")
print(f"Mean Absolute Error: {lr_mae}")
print(f"R-squared: {lr_r2}")
```

```
Linear Regression Metrics:
Mean Squared Error: 104532131.71307541
Mean Absolute Error: 8247.815038380479
R-squared: 0.8719795075733404
```

In [83]:

```python
from sklearn.ensemble import RandomForestRegressor

# Initialize the random forest model
rf_model = RandomForestRegressor(random_state=42)

# Train the random forest model
rf_model.fit(X_train, y_train)

# Predict on the test set using random forest
rf_y_pred = rf_model.predict(X_test)

# Calculate evaluation metrics for random forest
rf_mse = mean_squared_error(y_test, rf_y_pred)
rf_mae = mean_absolute_error(y_test, rf_y_pred)
rf_r2 = r2_score(y_test, rf_y_pred)

print("Random Forest Metrics:")
print(f"Mean Squared Error: {rf_mse}")
print(f"Mean Absolute Error: {rf_mae}")
print(f"R-squared: {rf_r2}")
```

```
Random Forest Metrics:
Mean Squared Error: 127929799.23398338
Mean Absolute Error: 9208.641830556251
R-squared: 0.8433243862381729
```

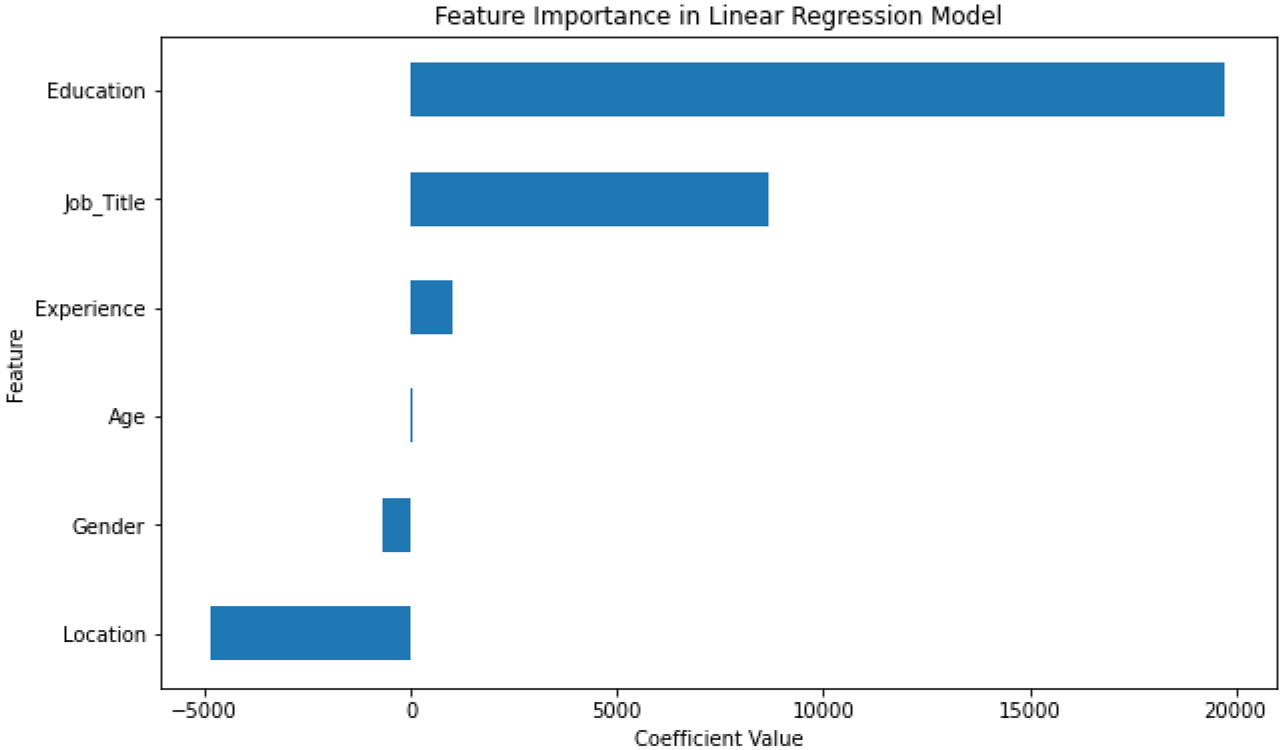
In [84]:

```python
# finding the average of the salary column
average_salary = df['Salary'].mean()

print(f"The average salary is: {average_salary}")
```

```
The average salary is: 105558.40423878135
```

In [85]:

```python
# Extract the coefficients
coefficients = lr_model.coef_
feature_importance = pd.Series(coefficients, index=features)

# Plot the feature importance
plt.figure(figsize=(10, 6))
feature_importance.sort_values().plot(kind='barh')
plt.title('Feature Importance in Linear Regression Model')
plt.xlabel('Coefficient Value')
plt.ylabel('Feature')
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: