SALARY PREDICTION of data professions

PROBLEM STATEMENT

- **Highlight the variability in salaries based on experience, job role, and performance.**

- **Importance of accurate salary predictions for job seekers and employers.**

Step 1: Exploratory Data Analysis (EDA)

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
#to ignore warnings
import warnings
warnings.filterwarnings('ignore')
```

Import Python Libraries

```python
file_path = '/content/Salary Prediction of Data Professions.csv'
# Read the CSV file into a pandas DataFrame
df = pd.read_csv(file_path)
```

Read CSV file

Step 1: Exploratory Data Analysis (EDA)

```
# Display the first few rows of the DataFrame
df.head()
```

	FIRST NAME	LAST NAME	SEX	DOJ	CURRENT DATE	DESIGNATION	AGE	SALARY	UNIT	LEAVES USED	LEAVES REMAINING	RATINGS	PAST EXP
0	TOMASA	ARMEN	F	5-18-2014	01-07-2016	Analyst	21.0	44570	Finance	24.0	6.0	2.0	0
1	ANNIE	NaN	F	NaN	01-07-2016	Associate	NaN	89207	Web	NaN	13.0	NaN	7

First few rows

```
df.tail()
```

	FIRST NAME	LAST NAME	SEX	DOJ	CURRENT DATE	DESIGNATION	AGE	SALARY	UNIT	LEAVES USED	LEAVES REMAINING	RATINGS	PAST EXP
2634	KATHERINE	ALSDON	F	6-28-2011	01-07-2016	Senior Manager	36.0	185977	Management	15.0	15.0	5.0	10
2635	LOUISE	ALTARAS	F	1-14-2014	01-07-2016	Analyst	23.0	45758	IT	17.0	13.0	2.0	0

Last few rows

Shape

```
df.shape
```

```
(2639, 13)
```

Summary Statistics.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2639 entries, 0 to 2638
Data columns (total 13 columns):
 #   Column            Non-Null Count  Dtype
---  ------            --------------  -----
 0   FIRST NAME        2639 non-null   object
 1   LAST NAME         2637 non-null   object
 2   SEX               2639 non-null   object
 3   DOJ               2638 non-null   object
 4   CURRENT DATE      2639 non-null   object
 5   DESIGNATION       2639 non-null   object
 6   AGE               2636 non-null   float64
 7   SALARY            2639 non-null   int64
 8   UNIT              2639 non-null   object
 9   LEAVES USED       2636 non-null   float64
 10  LEAVES REMAINING  2637 non-null   float64
 11  RATINGS           2637 non-null   float64
 12  PAST EXP          2639 non-null   int64
```

```
df.describe()
```

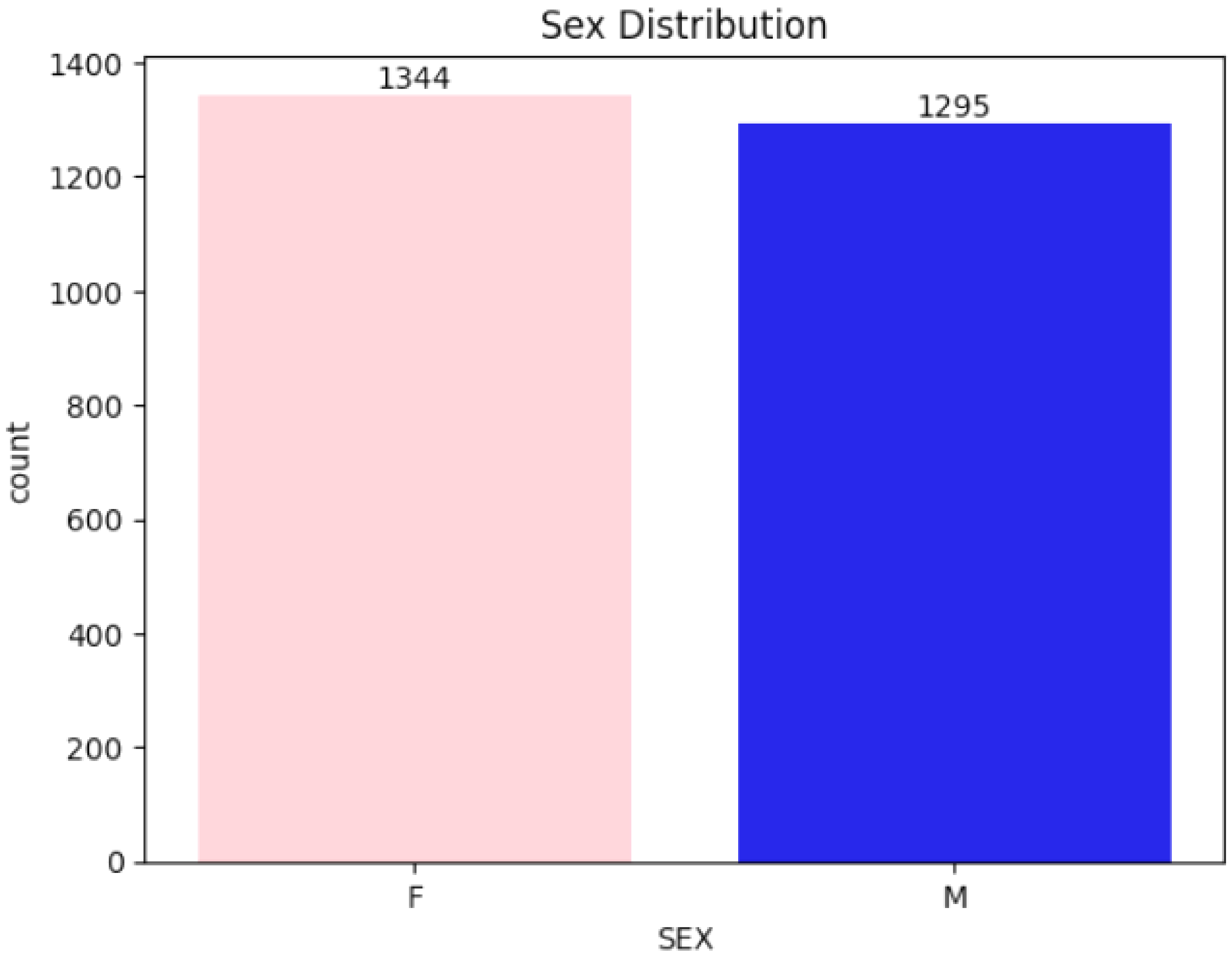
	AGE	SALARY	LEAVES USED	LEAVES REMAINING	RATINGS	PAST EXP
count	2636.000000	2639.000000	2636.000000	2637.000000	2637.000000	2639.000000
mean	24.756449	58136.678287	22.501517	7.503223	3.486159	1.566881
std	3.908228	36876.956944	4.604469	4.603193	1.114933	2.728416
min	21.000000	40001.000000	15.000000	0.000000	2.000000	0.000000
25%	22.000000	43418.000000	19.000000	4.000000	2.000000	0.000000
50%	24.000000	46781.000000	22.000000	8.000000	3.000000	1.000000
75%	25.000000	51401.500000	26.000000	11.000000	4.000000	2.000000
max	45.000000	388112.000000	30.000000	15.000000	5.000000	23.000000

Step 2: Data Visualization

Sex Distribution

BAR CHART SHOWING SEX DISTRIBUTION

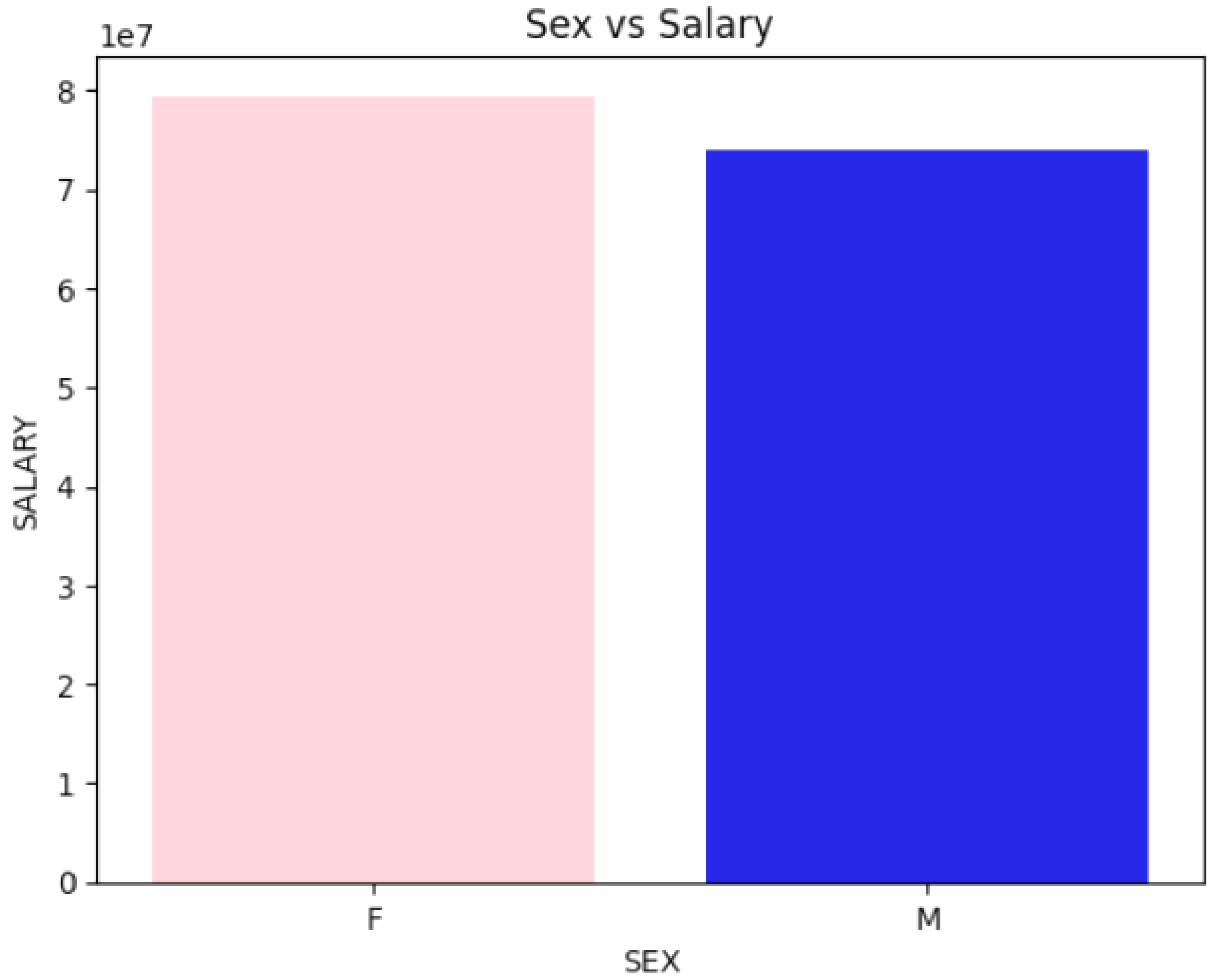
```
colors = {'M': 'blue', 'F': 'pink'}
# Create the bar plot
ax = sns.countplot(data=df, x='SEX', palette=colors)
ax.set_title('Sex Distribution')
for bars in ax.containers:
    ax.bar_label(bars)
```

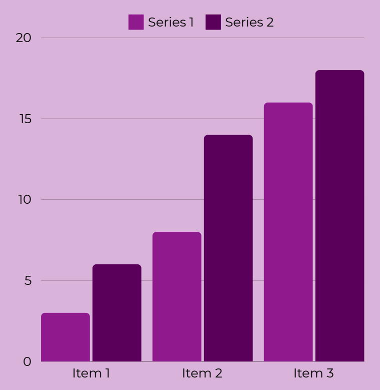


BAR CHART SHOWING SEX VS SALARY

```
colors = {'M': 'blue', 'F': 'pink'}
salary_gen = df.groupby(['SEX'], as_index=False)['SALARY'].sum().
sort_values(by='SALARY', ascending=False)
sns.barplot(x = 'SEX',y= 'SALARY' ,data = salary_gen, palette = colors).
set_title('Sex vs Salary')
```

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



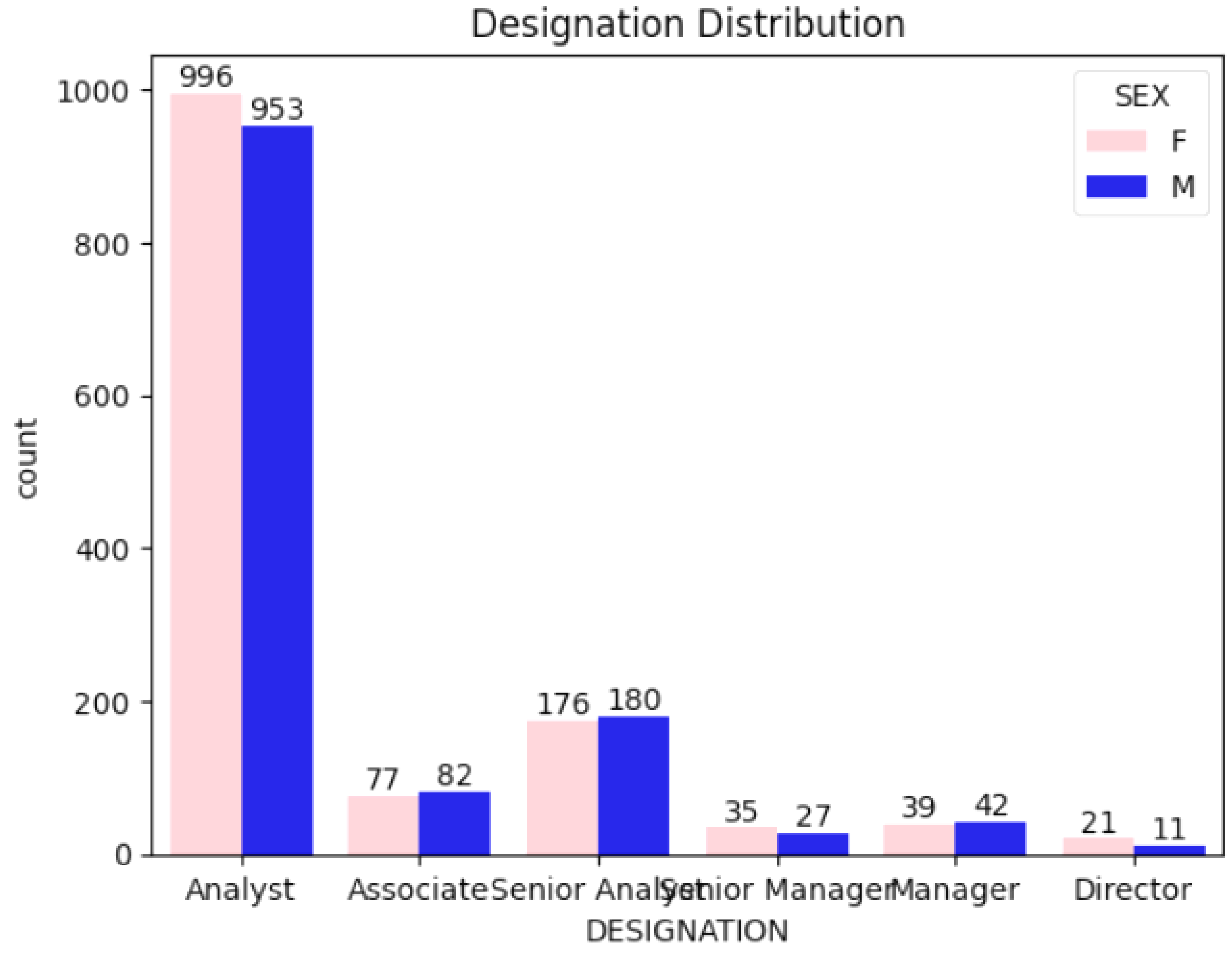


Step 2: Data Visualization

Designation Distribution

DESIGNATION DISTRIBUTION BY SEX

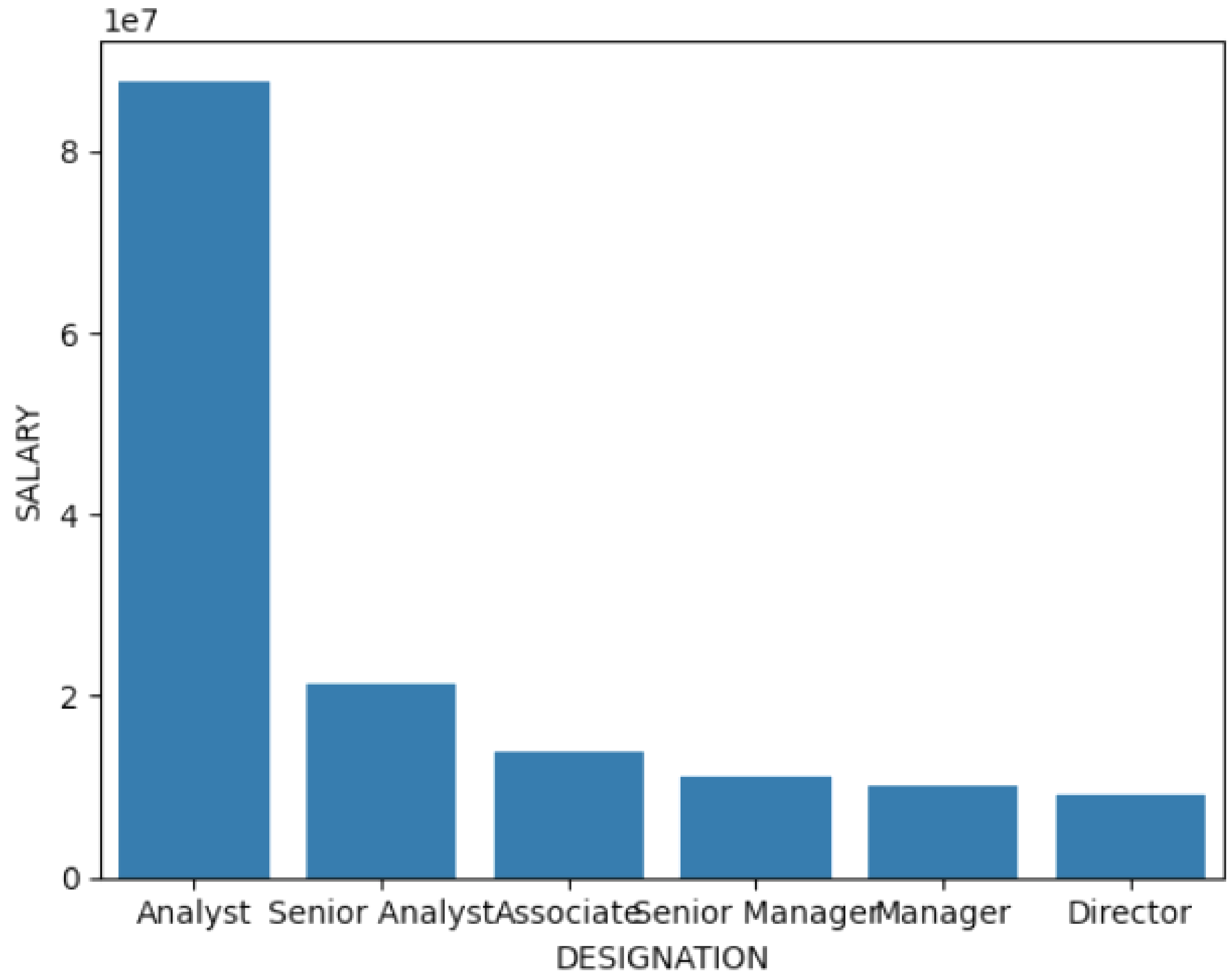
```python
colors = {'M': 'blue', 'F': 'pink'}
ax = sns.countplot(data = df, x = 'DESIGNATION',
                   hue = 'SEX',palette = colors)
ax.set_title('Designation Distribution')
for bars in ax.containers:
    ax.bar_label(bars)
```

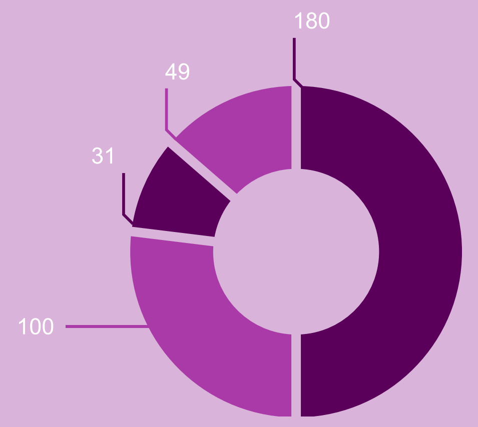


SALARY VS DESIGNATION

```python
# Salary vs Designation
salary_desg = df.groupby(['DESIGNATION'], as_index=False)['SALARY'].sum().
sort_values(by='SALARY', ascending=False)
sns.barplot(x = 'DESIGNATION',y= 'SALARY' ,data = salary_desg)
```

```
<Axes: xlabel='DESIGNATION', ylabel='SALARY'>
```





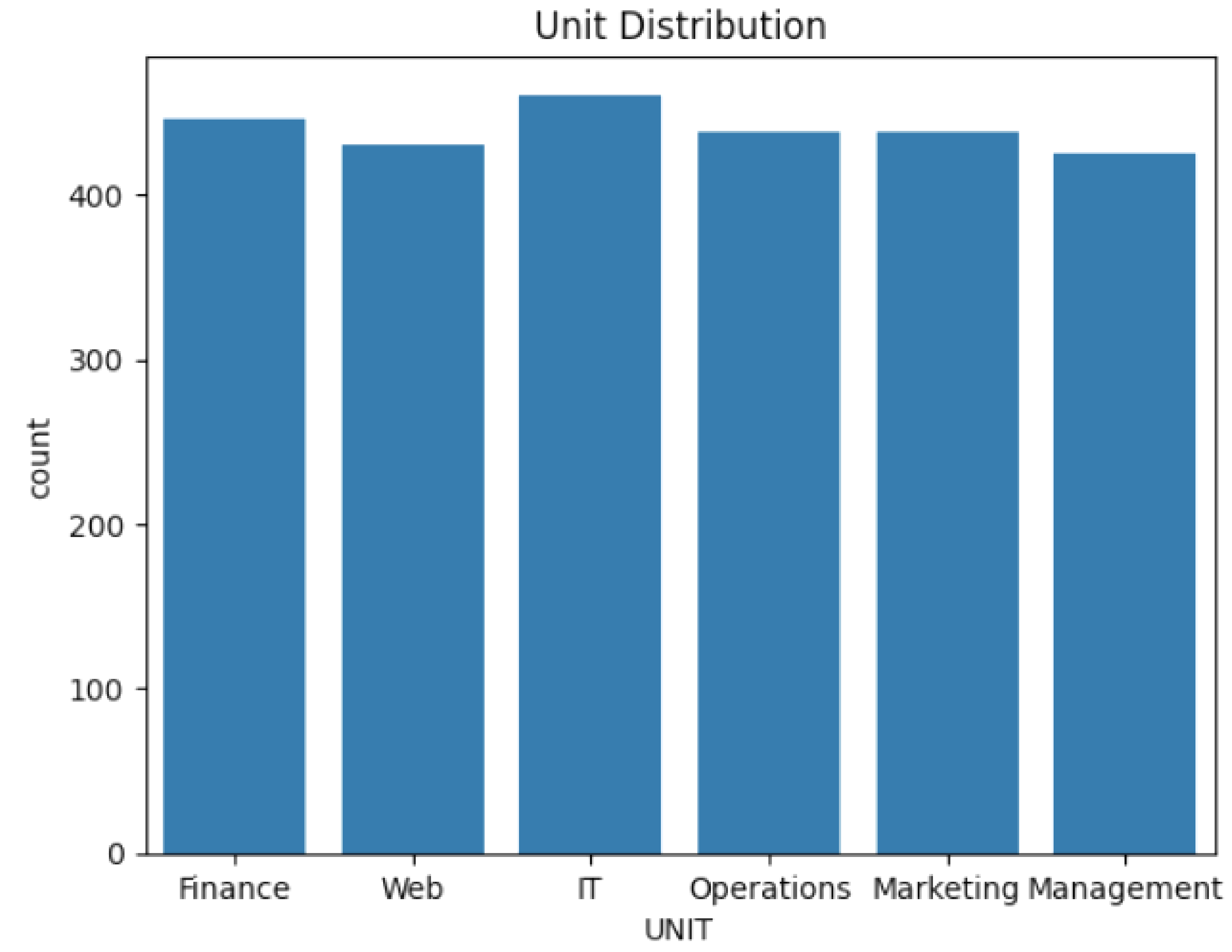
Step 2: Data Visualization

Unit Distribution

UNIT DISTRIBUTION

```
sns.countplot(data=df, x='UNIT').set_title('Unit Distribution')
```

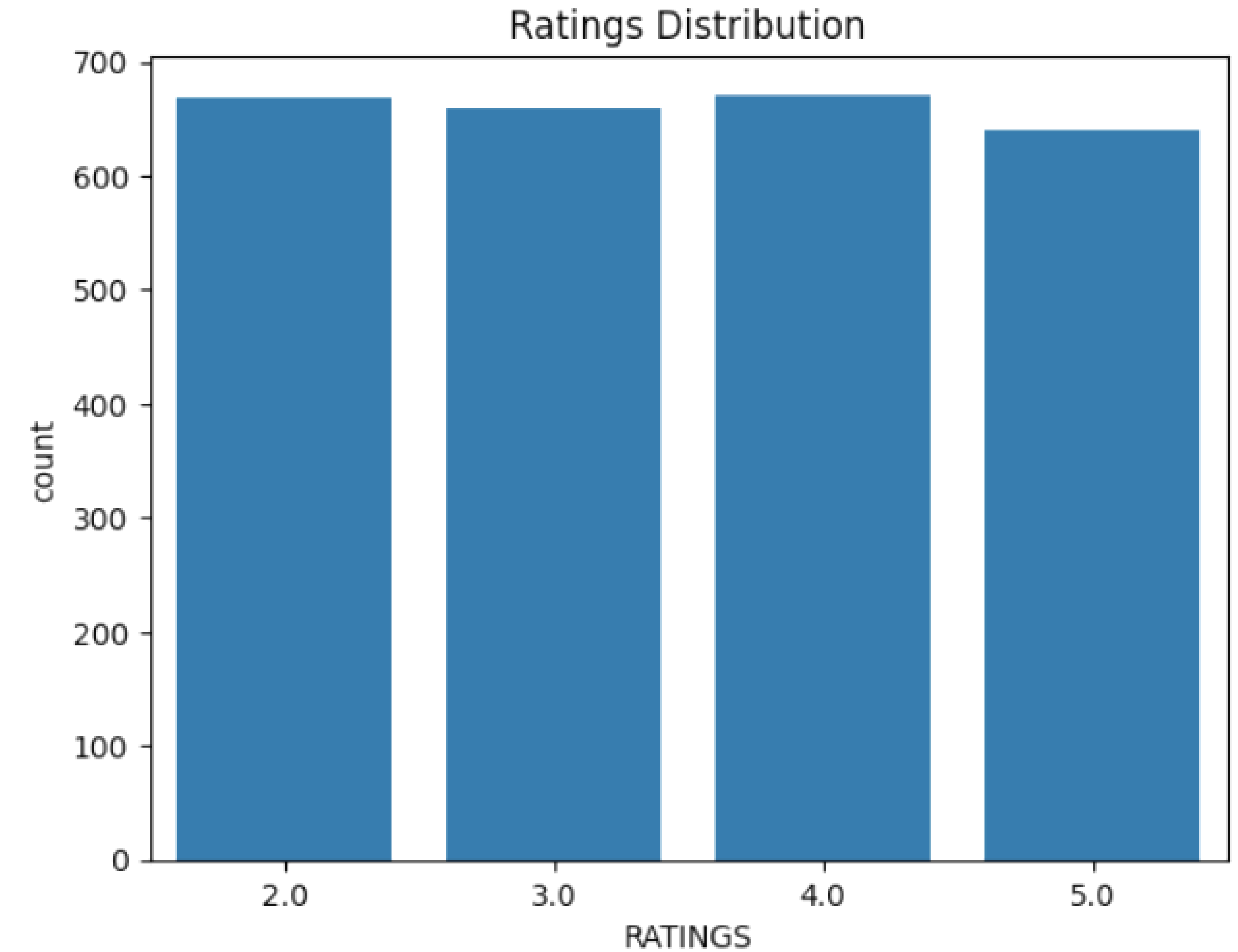
```
Text(0.5, 1.0, 'Unit Distribution')
```

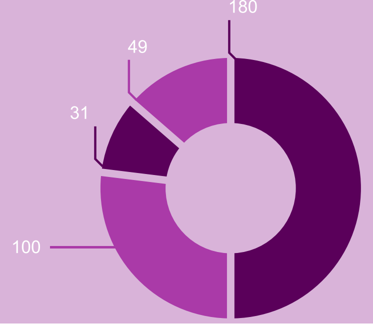


RATINGS DISTRIBUTION

```
sns.countplot(data=df, x='RATINGS').set_title('Ratings Distribution')
```

```
Text(0.5, 1.0, 'Ratings Distribution')
```





Step 3: Data Preprocessing

Missing Values Calculation

```
df.isnull().sum()
```

```
FIRST NAME           0
LAST NAME            2
SEX                  0
DOJ                  1
CURRENT DATE         0
DESIGNATION          0
AGE                  3
SALARY               0
UNIT                 0
LEAVES USED          3
LEAVES REMAINING     2
RATINGS              2
PAST EXP             0
dtype: int64
```

Hndling Missing Values

```
[ ]  df.drop(columns=['LAST NAME'], inplace=True)

[ ]  df['DOJ'].fillna(df['DOJ'].mode()[0], inplace=True)

[ ]  df['AGE'].fillna(df['AGE'].median(), inplace=True)

[ ]  df['LEAVES USED'].fillna(30-df['LEAVES REMAINING'], inplace=True)

[ ]  df['LEAVES REMAINING'].fillna(30-df['LEAVES USED'], inplace=True)

[ ]  df['RATINGS'].fillna(df['RATINGS'].median(), inplace=True)
```

```
# Confirm no missing values remain
print("\nMissing Values After Imputation:")
print(df.isnull().sum())
```

```
Missing Values After Imputation:
FIRST NAME          0
SEX                 0
DOJ                 0
CURRENT DATE        0
DESIGNATION         0
AGE                 0
SALARY              0
UNIT                0
LEAVES USED         0
LEAVES REMAINING    0
RATINGS             0
PAST EXP            0
dtype: int64
```

Filling Null Values in LAST NAME, DOJ, AGE, LEAVES USED, LEAVES REMAINING, RATINGS

Step 3: Data Preprocessing

Step1: Encode Categorical Variables

```python
print(df['UNIT'].unique())
print(df['DESIGNATION'].unique())
print(df['SEX'].unique())

# One-Hot Encoding without dropping the first category
df = pd.get_dummies(df, columns=['SEX', 'DESIGNATION', 'UNIT'], drop_first=False

# Display the resulting DataFrame
print(df.head())
```

```
['Finance' 'Web' 'IT' 'Operations' 'Marketing' 'Management']
['Analyst' 'Associate' 'Senior Analyst' 'Senior Manager' 'Manager'
 'Director']
['F' 'M']
  FIRST NAME        DOJ CURRENT DATE   AGE  SALARY  LEAVES USED  \
0     TOMASA  5-18-2014   01-07-2016  21.0   44570         24.0
1      ANNIE 10-19-2013   01-07-2016  24.0   89207         17.0
2      OLIVE  7-28-2014   01-07-2016  21.0   40955         23.0
3     CHERRY 04-03-2013   01-07-2016  22.0   45550         22.0
4       LEON 11-20-2014   01-07-2016  24.0   43161         27.0

   LEAVES REMAINING  RATINGS  PAST EXP  SEX_F  ...  DESIGNATION_Director  \
0               6.0      2.0         0      1  ...                     0
1              13.0      3.0         7      1  ...                     0
2               7.0      3.0         0      1  ...                     0
3               8.0      3.0         0      1  ...                     0
4               3.0      3.0         3      0  ...                     0

   DESIGNATION_Manager  DESIGNATION_Senior Analyst  \
0                    0                           0
1                    0                           0
2                    0                           0
3                    0                           0
4                    0                           0
```

One-Hot Encoding: Convert categorical variables into one-hot encoding to simplify processing.

```
   DESIGNATION_Senior Manager  UNIT_Finance  UNIT_IT  UNIT_Management  \
0                           0             1        0                0
1                           0             0        0                0
2                           0             1        0                0
3                           0             0        1                0
4                           0             0        0                0

   UNIT_Marketing  UNIT_Operations  UNIT_Web
0               0                0         0
1               0                0         1
2               0                0         0
3               0                0         0
4               0                1         0

[5 rows x 23 columns]
```

Step 4: Model Development & Evaluation

Train various regression models: Linear Regression, Decision Trees, Random Forests, Gradient Boosting and Evaluate using Metrics

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import GradientBoostingRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.preprocessing import StandardScaler


df.drop(columns=['FIRST NAME','DOJ','CURRENT DATE'], inplace=True)


# Features and target variable
X = df.drop(columns=['SALARY'])
y = df['SALARY']

# Split data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Step 4: Model Development & Evaluation

LINEAR REGRESSION

```
# Linear Regression Model
linear_reg = LinearRegression()
linear_reg.fit(X_train, y_train)
y_pred_lr = linear_reg.predict(X_test)

# Model Evaluation
print("Linear Regression:")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, y_pred_lr))}")
print(f"MAE: {mean_absolute_error(y_test, y_pred_lr)}")
print(f"R-squared: {r2_score(y_test, y_pred_lr)}")
```

```
Linear Regression:
RMSE: 9787.762726268194
MAE: 4636.401683800093
R-squared: 0.9423076375836923
```

Step 4: Model Development & Evaluation

DECISION TREE

```
# Decision Tree Model
dt_reg = DecisionTreeRegressor()
dt_reg.fit(X_train, y_train)
y_pred_dt = dt_reg.predict(X_test)

# Model Evaluation
print("\nDecision Tree:")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, y_pred_dt))}")
print(f"MAE: {mean_absolute_error(y_test, y_pred_dt)}")
print(f"R-squared: {r2_score(y_test, y_pred_dt)}")
```

```
Decision Tree:
RMSE: 12174.388340461179
MAE: 5442.899621212121
R-squared: 0.9107422861084971
```

Step 4: Model Development & Evaluation

RANDOM FOREST

```
# Random Forest Model
rf_reg = RandomForestRegressor()
rf_reg.fit(X_train, y_train)
y_pred_rf = rf_reg.predict(X_test)

# Model Evaluation
print("\nRandom Forest:")
print(f"RMSE: {np.sqrt(mean_squared_error(y_test, y_pred_rf))}")
print(f"MAE: {mean_absolute_error(y_test, y_pred_rf)}")
print(f"R-squared: {r2_score(y_test, y_pred_rf)}")
```

```
Random Forest:
RMSE: 10545.036499927288
MAE: 4717.079515038781
R-squared: 0.9330350371434462
```

Step 4: Model Development & Evaluation

GRADIENT BOOSTING

```python
# Gradient Boosting Model
gb_reg = GradientBoostingRegressor()
gb_reg.fit(X_train, y_train)
y_pred_gb = gb_reg.predict(X_test)

# Model Evaluation
print("\nGradient Boosting:")
print(f"RMSE: {np.sqrt(mean_squared_error(y_te Loading... _gb))}")
print(f"MAE: {mean_absolute_error(y_test, y_pred_gb)}")
print(f"R-squared: {r2_score(y_test, y_pred_gb)}")
```

```
Gradient Boosting:
RMSE: 11720.98800070862
MAE: 4792.66656544245
R-squared: 0.9172667851809078
```

Step 5: Model SELECTION

MODEL PERFORMANCE:

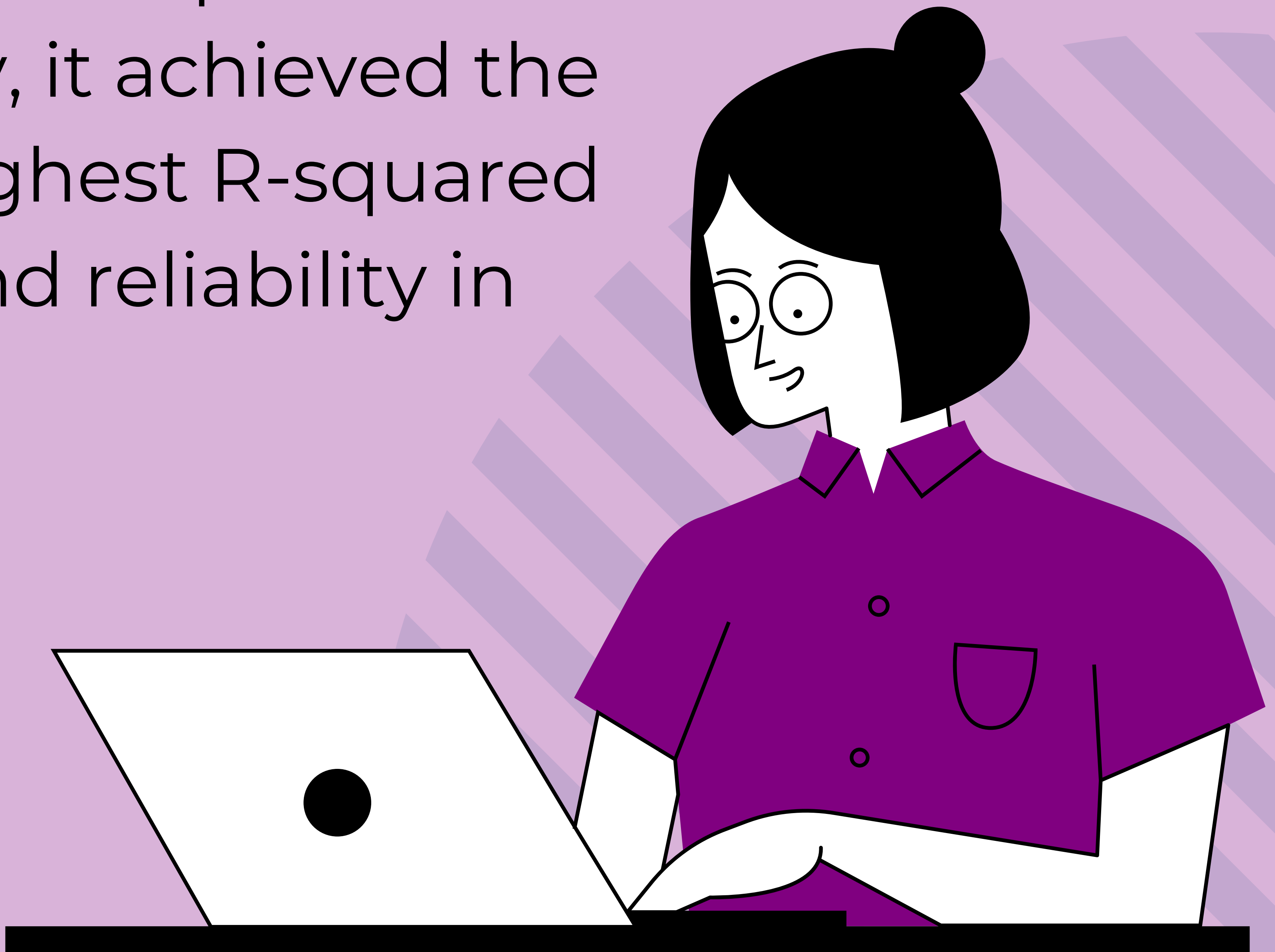
MODEL	RMSE	MAE	R-SQUARED
Linear Regression	9787.76	4636.4	0.942
Decision Tree Regressor	12174.38	5442.89	0.91
Random Forest Regressor	10545.03	4717.07	0.933
Gradient Boosting	11720.98	4792.66	0.917

Step 5: Model SELECTION

SELECTED MODEL: LINEAR REGRESSION

REASON:

This model was selected based on its superior performance across the evaluation metrics. Specifically, it achieved the lowest RMSE and MAE, along with the highest R-squared value (0.942), indicating both accuracy and reliability in predictions.



Step 6: Model Saving and Feature Importance

```python
import pickle
with open('linear_regression_model.pkl', 'wb') as file:
    pickle.dump(linear_reg, file)
```

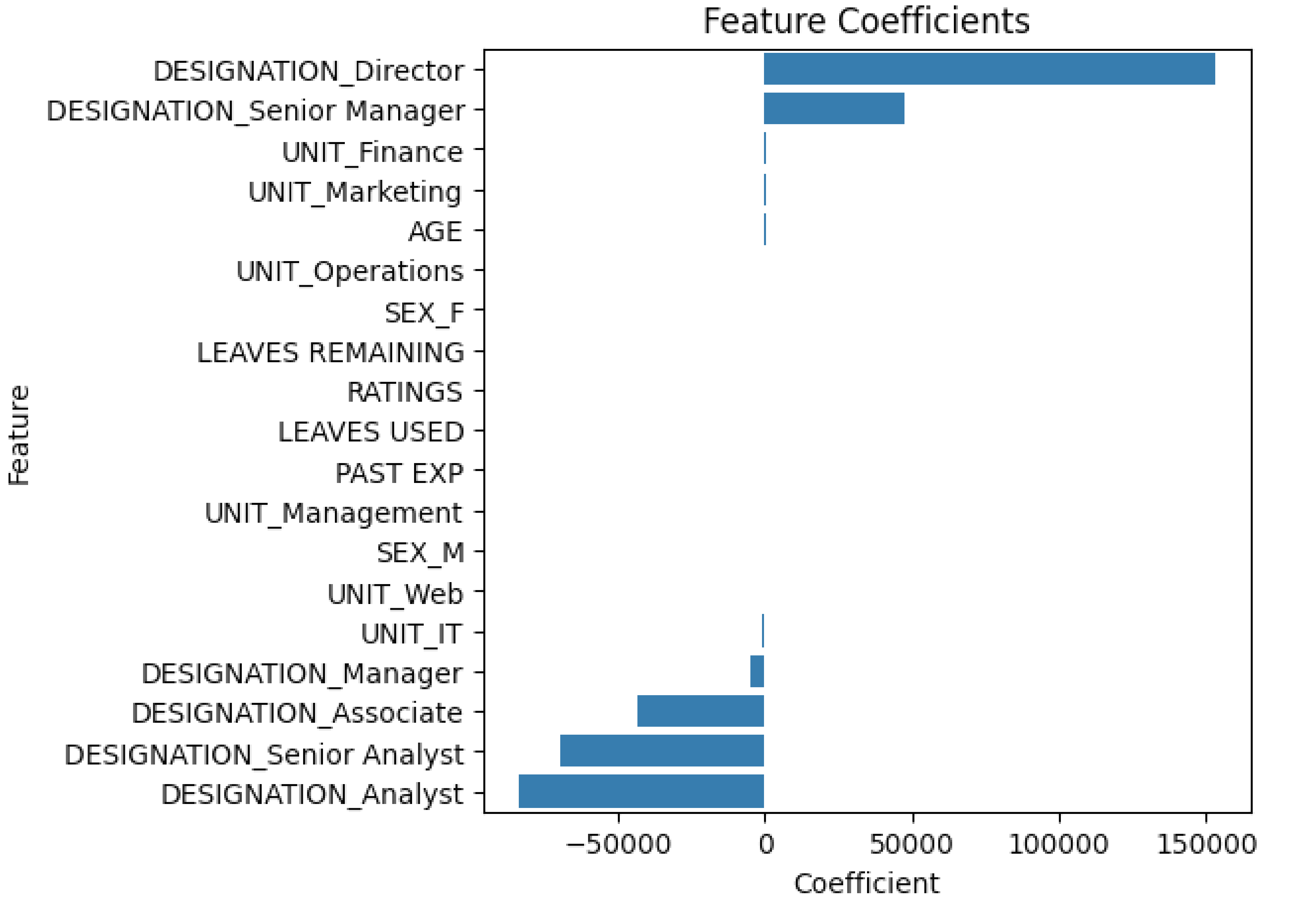
Model Saving

```python
# Linear Regression Model
linear_reg = LinearRegression()
linear_reg.fit(X_train, y_train)

# Get coefficients
coefficients = linear_reg.coef_

# Create a DataFrame to display coefficients
coefficients_df = pd.DataFrame(coefficients, index=X.columns, columns=['Coefficient'])
coefficients_df = coefficients_df.sort_values(by='Coefficient', ascending=False)

# Display the coefficients
print("Coefficients:")
print(coefficients_df)

# Plot coefficients
plt.figure(figsize=(12, 8))
sns.barplot(x=coefficients_df['Coefficient'], y=coefficients_df.index)
plt.xlabel('Coefficient')
plt.ylabel('Feature')
plt.title('Feature Coefficients')
plt.show()
```

Feature Importance

Step 6: Model Saving and Feature Importance

```
Coefficients:
                                Coefficient
DESIGNATION_Director          153314.565871
DESIGNATION_Senior Manager     47499.545994
UNIT_Finance                     393.751858
UNIT_Marketing                   336.450964
AGE                              253.401240
UNIT_Operations                  162.480550
SEX_F                            135.424512
LEAVES REMAINING                  12.780092
RATINGS                           -6.245792
LEAVES USED                      -12.780092
PAST EXP                        -106.741197
UNIT_Management                 -111.599899
SEX_M                           -135.424512
UNIT_Web                        -209.747110
UNIT_IT                         -571.336362
DESIGNATION_Manager            -4567.305057
DESIGNATION_Associate         -43325.846738
DESIGNATION_Senior Analyst    -69492.365462
DESIGNATION_Analyst           -83428.594608
```



Step 7: Model Deployment

Salary Prediction

Age:
24

Leaves Used:
10

Leaves Remaining:
20

Ratings:
5

Past Experience:
2

Sex:
Male

Designation:
Associate

Unit:
Management

Predict Salary

Predicted Salary: $85,320.20

CONCLUSION

The Linear Regression model emerged as the best performer among the models evaluated, exhibiting the lowest RMSE and MAE, and the highest R-squared value. This indicates its superior accuracy and ability to explain variance in salary data. Features like age, past experience, and job role were found to be significant predictors of salary. Moving forward, continuous monitoring and potential feature engineering could further enhance the model's performance.

THANK YOU

SONAL SONARGHARE

sonalsonarghare30@gmail.com

A.P SHAH INSTITUTE OF TECHNOLOGY