# An Analytical Dashboard analysis for Educational Impact on Career Success

Double-click (or enter) to edit

# Step 1. Important libraries

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from scipy import stats
import plotly.express as px
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error
```

# Step 2. Data Cleaning:

### Check for Nulls,Clean Column Names,Convert Data Types,Check for Duplicates and Handle Outliers

```
#Loading dataset
df=pd.read_csv("/content/education_career_success.csv")

# Check for missing values
print(df.isnull().sum())

# Clean column names
df.columns = df.columns.str.strip().str.replace(' ', '_').str.lower()

# Check data types
df.dtypes

# converting columns to correct types
df['age'] = df['age'].astype(int)  # Convert to integer
df['high_school_gpa'] = df['high_school_gpa'].astype(float)  # Convert to float
df['gender'] = df['gender'].astype('category')  # Convert to category if it's a categorical
```

```
df['field_of_study'] = df['field_of_study'].astype('category')  # Similarly for other catego

# Example to convert a column to numeric (use `errors='coerce'` to handle invalid entries)
df['sat_score'] = pd.to_numeric(df['sat_score'], errors='coerce')

# Check for duplicates
df.duplicated().sum()

# Remove duplicate rows if any
df = df.drop_duplicates()

Q1 = df['starting_salary'].quantile(0.25)
Q3 = df['starting_salary'].quantile(0.75)
IQR = Q3 - Q1

# Filter out rows with outliers
df = df[(df['starting_salary'] >= (Q1 - 1.5 * IQR)) & (df['starting_salary'] <= (Q3 + 1.5 *
```

```
Student_ID               0
Age                      0
Gender                   0
High_School_GPA          0
SAT_Score                0
University_Ranking       0
University_GPA           0
Field_of_Study           0
Internships_Completed    0
Projects_Completed       0
Certifications           0
Soft_Skills_Score        0
Networking_Score         0
Job_Offers               0
Starting_Salary          0
Career_Satisfaction      0
Years_to_Promotion       0
Current_Job_Level        0
Work_Life_Balance        0
Entrepreneurship         0
dtype: int64
```

### Summary Statistics

```
# Summary of numerical features
summary = df.describe().T
summary['missing_values'] = df.isnull().sum()
summary
```

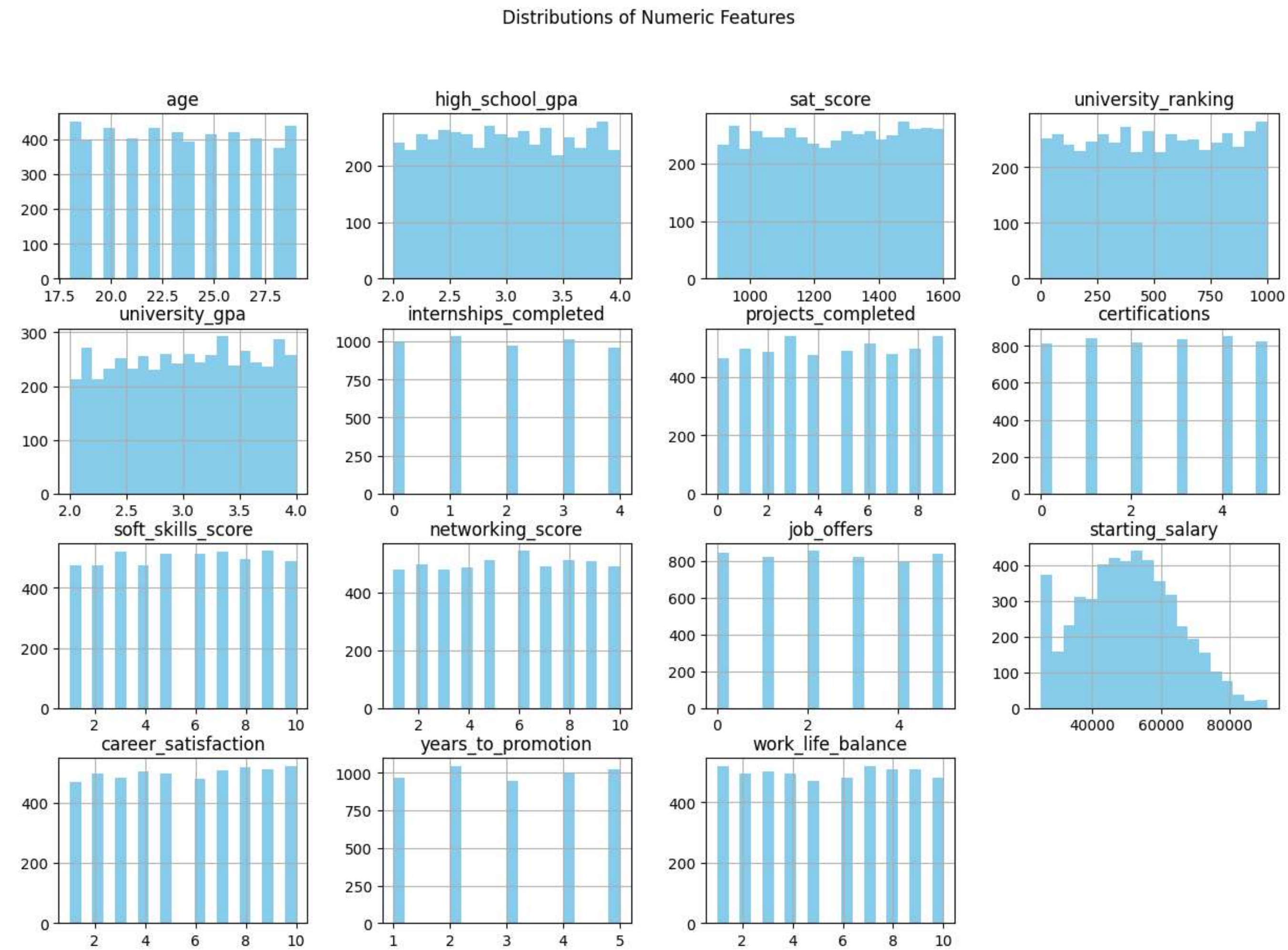
| | count | mean | std | min | 25% | 50% | |
|---|---|---|---|---|---|---|---|
| **age** | 4988.0 | 23.445870 | 3.474112 | 18.0 | 20.00 | 23.00 | 2 |
| **high_school_gpa** | 4988.0 | 2.997007 | 0.575609 | 2.0 | 2.50 | 2.99 | |
| **sat_score** | 4988.0 | 1253.839615 | 203.183521 | 900.0 | 1076.00 | 1257.00 | 14 |
| **university_ranking** | 4988.0 | 504.462109 | 291.005436 | 1.0 | 256.00 | 502.00 | 7 |
| **university_gpa** | 4988.0 | 3.019300 | 0.576060 | 2.0 | 2.52 | 3.03 | |
| **internships_completed** | 4988.0 | 1.981355 | 1.407909 | 0.0 | 1.00 | 2.00 | |
| **projects_completed** | 4988.0 | 4.560144 | 2.873196 | 0.0 | 2.00 | 5.00 | |
| **certifications** | 4988.0 | 2.511026 | 1.703533 | 0.0 | 1.00 | 3.00 | |
| **soft_skills_score** | 4988.0 | 5.549920 | 2.849847 | 1.0 | 3.00 | 6.00 | |
| **networking_score** | 4988.0 | 5.539094 | 2.850228 | 1.0 | 3.00 | 6.00 | |
| **job_offers** | 4988.0 | 2.488573 | 1.710695 | 0.0 | 1.00 | 2.00 | |
| **starting_salary** | 4988.0 | 50454.009623 | 14338.224232 | 25000.0 | 40100.00 | 50300.00 | 604 |
| **career_satisfaction** | 4988.0 | 5.575782 | 2.872686 | 1.0 | 3.00 | 6.00 | |
| **years_to_promotion** | 4988.0 | 3.015036 | 1.417037 | 1.0 | 2.00 | 3.00 | |
| **work_life_balance** | 4988.0 | 5.485164 | 2.883400 | 1.0 | 3.00 | 6.00 | |

Next steps: Generate code with summary View recommended plots New interactive sheet

# Step 3.Exploratory Data Analysis (EDA)

### A. Distribution of Numerical Features:

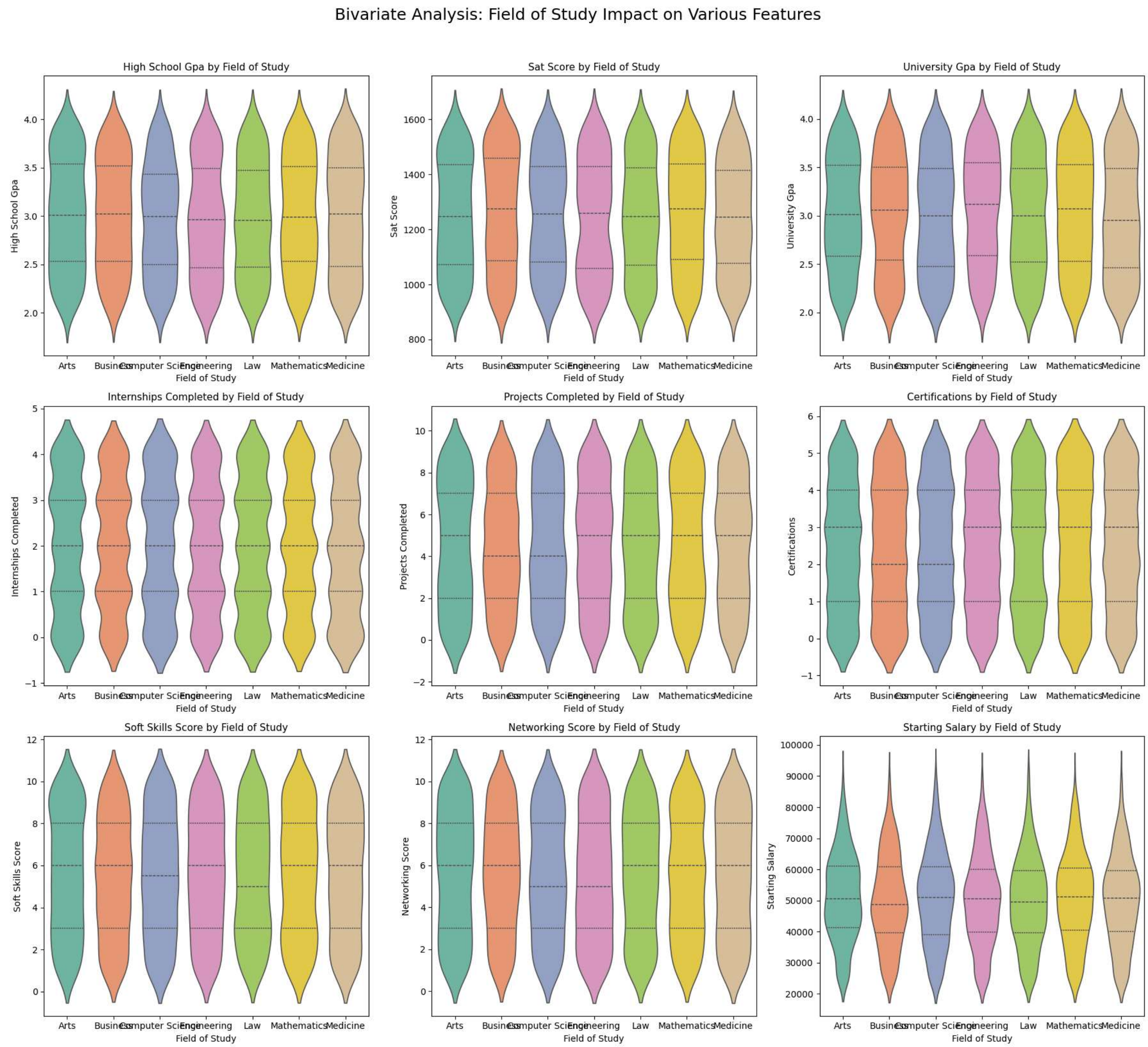
```
df.describe()
df.hist(bins=20, figsize=(15, 10), color='skyblue')
plt.suptitle("Distributions of Numeric Features")
plt.show()
```




Double-click (or enter) to edit

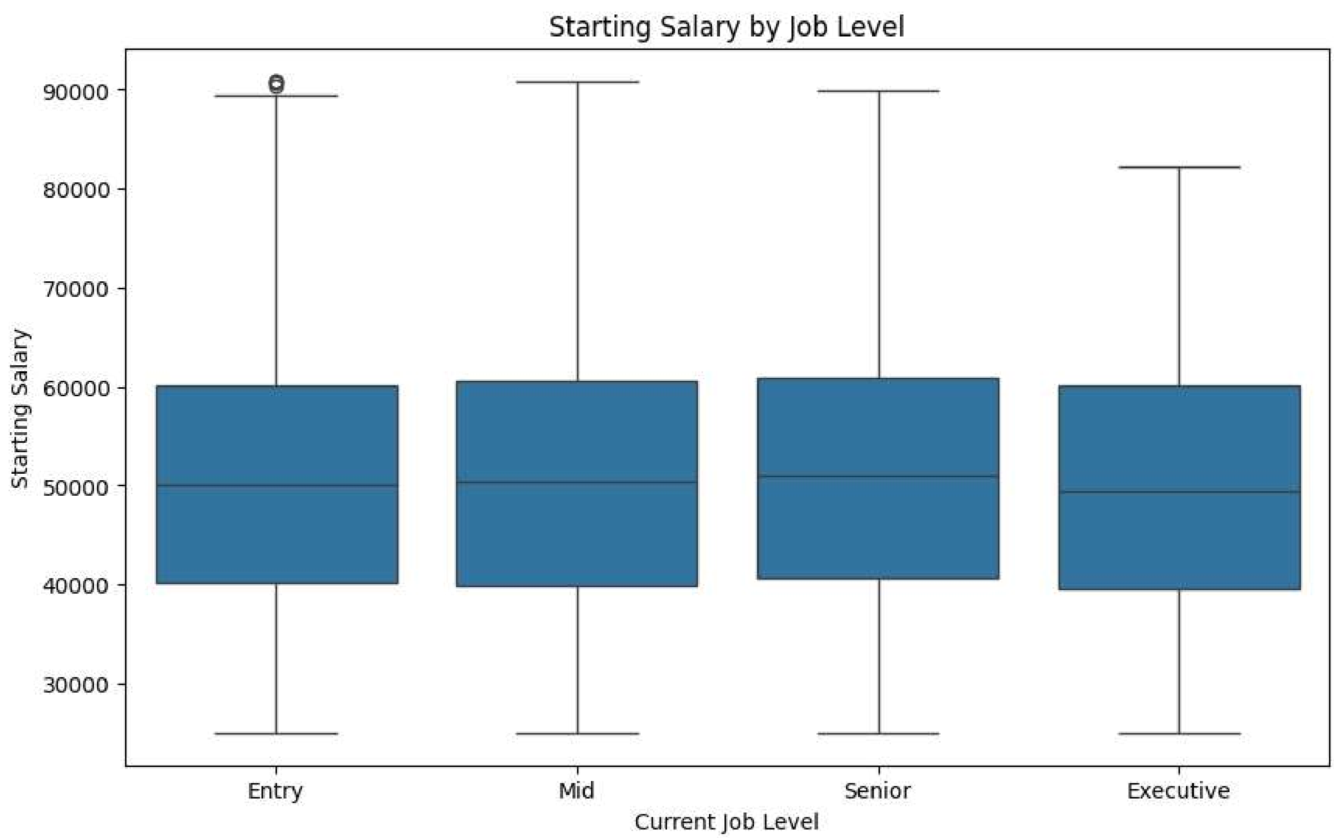
## B. Target variable:

```
# Plot violin plots for each feature against Job Offers with hue as Field_of_Study
plt.figure(figsize=(18, 16))
for i, feature in enumerate(numerical_features):
    plt.subplot(3, 3, i + 1)
    sns.violinplot(
        x='field_of_study',
        y=feature,
        data=df,
        hue='field_of_study',
        palette='Set2',
        inner='quartile',
        dodge=False,
        legend=False
    )
    plt.title(f"{feature.replace('_', ' ').title()} by Field of Study", fontsize=11)
    plt.xlabel("Field of Study")
    plt.ylabel(feature.replace('_', ' ').title())

plt.suptitle("Bivariate Analysis: Field of Study Impact on Various Features", fontsize=18, y
plt.tight_layout()
plt.show()
```

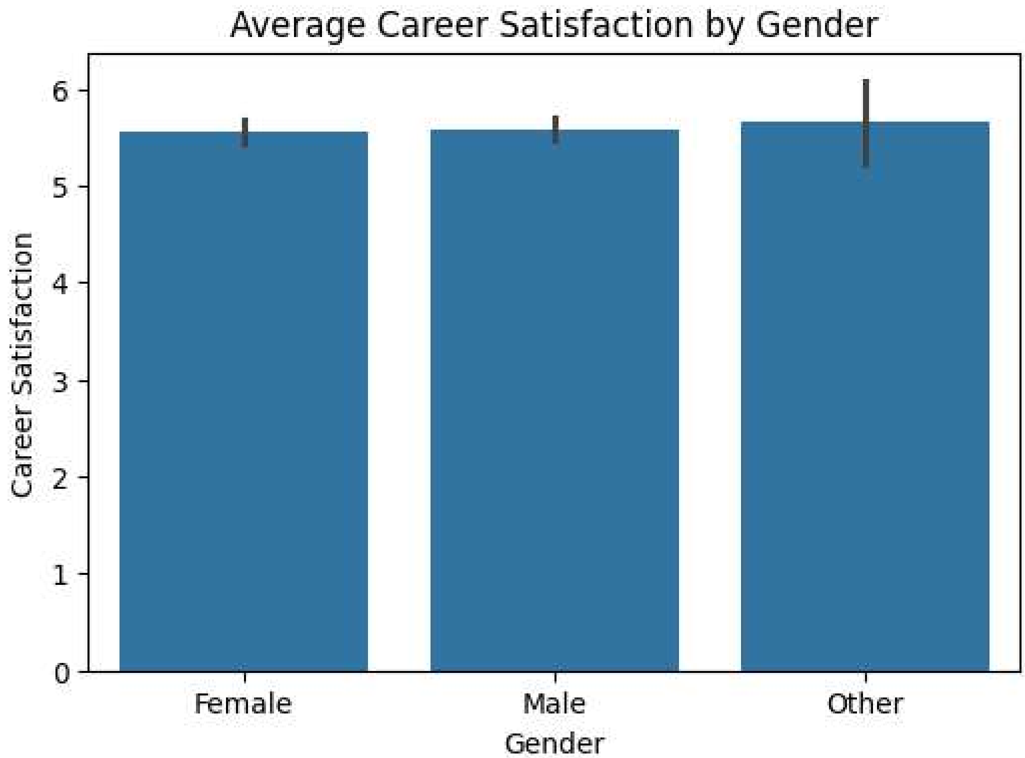

Bivariate Analysis: Field of Study Impact on Various Features
High School Gpa by Field of Study
High School Gpa
Sat Score by Field of Study
Sat Score
University Gpa by Field of Study
University Gpa
Internships Completed by Field of Study
Internships Completed
Projects Completed by Field of Study
Projects Completed
Certifications by Field of Study
Certifications
Soft Skills Score by Field of Study
Soft Skills Score
Networking Score by Field of Study
Networking Score
Starting Salary by Field of Study
Starting Salary
Arts
Law
Mathematics
Medicine
Field of Study

### C. Categorical Features vs Career Outcomes:

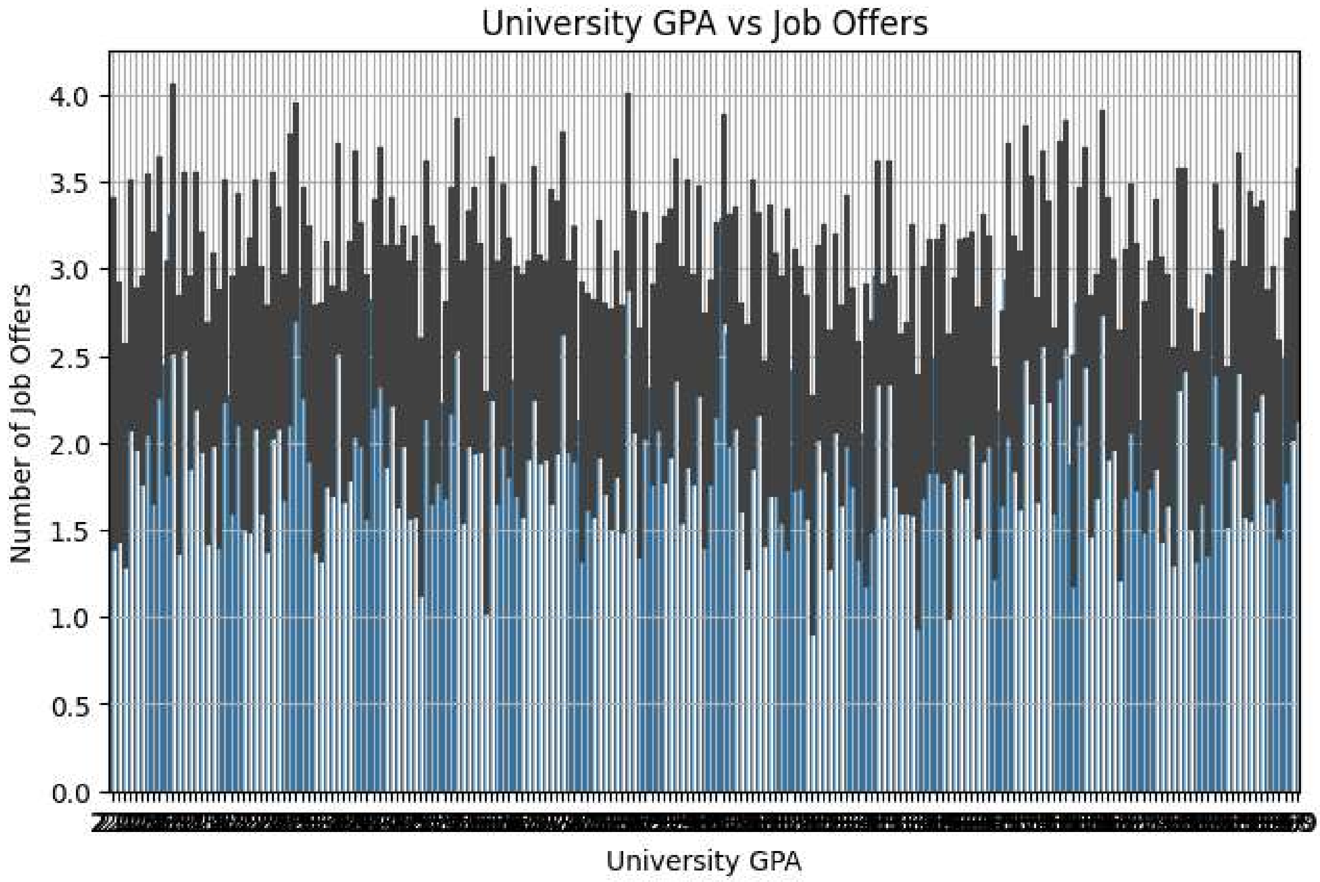
```
plt.figure(figsize=(10, 6))
sns.boxplot(x='current_job_level', y='starting_salary', data=df)
plt.title("Starting Salary by Job Level")
plt.xlabel("Current Job Level")
plt.ylabel("Starting Salary")
plt.show()
```




```
plt.figure(figsize=(6, 4))
sns.barplot(x='gender', y='career_satisfaction', data=df, estimator=np.mean)
plt.title("Average Career Satisfaction by Gender")
plt.xlabel("Gender")
plt.ylabel("Career Satisfaction")
plt.show()
```

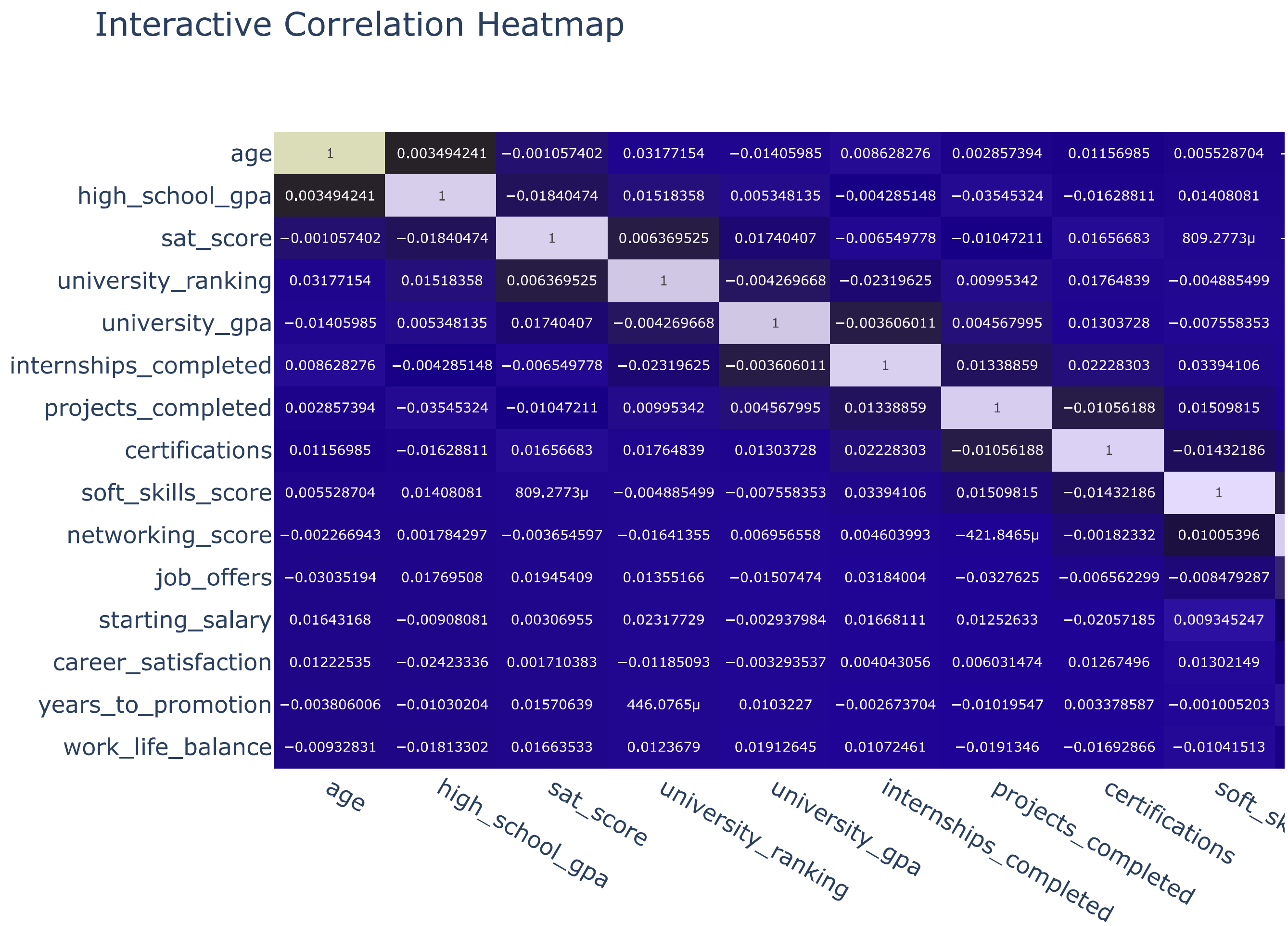


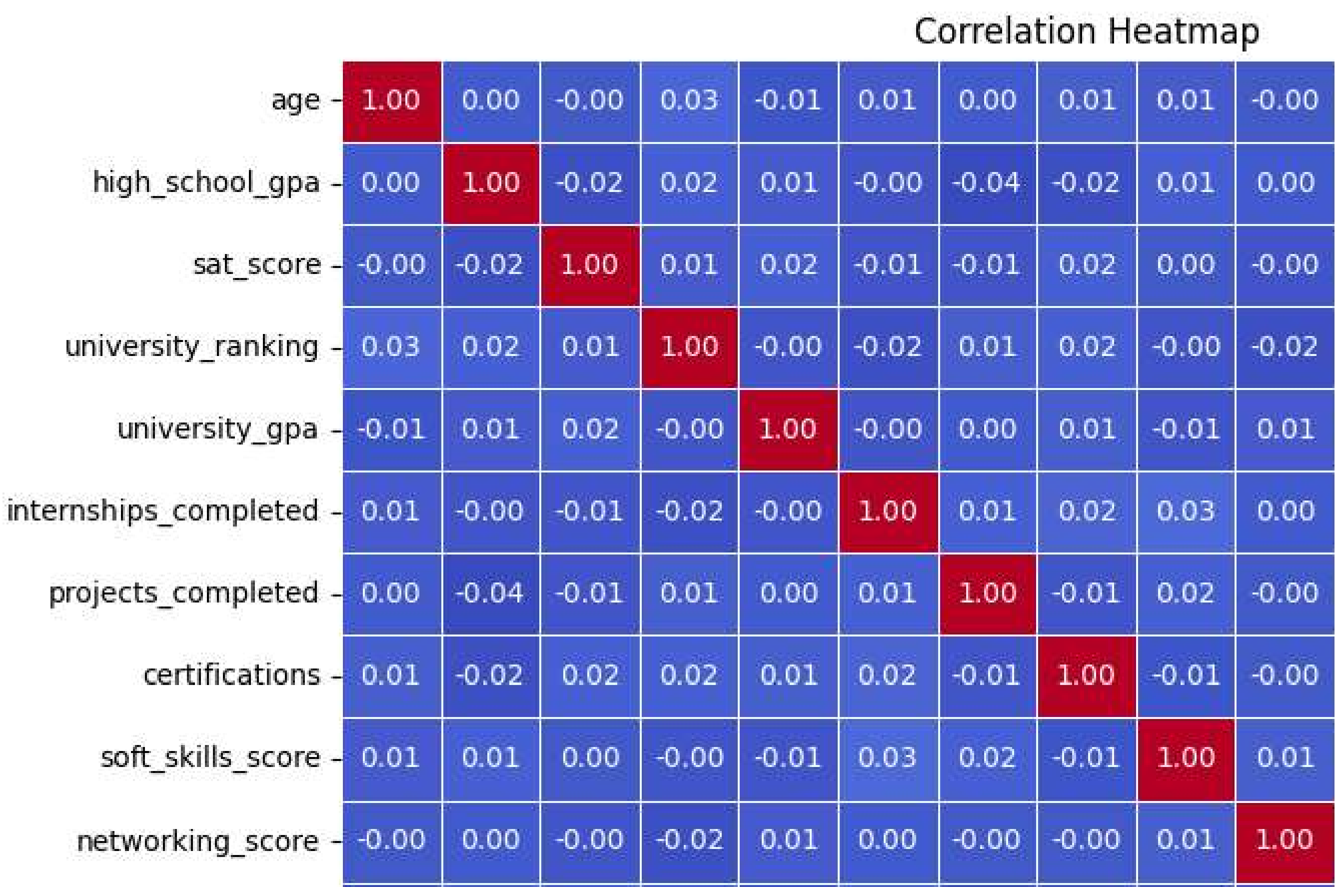

```
plt.figure(figsize=(8, 5))
sns.barplot(x='university_gpa', y='job_offers', data=df)
plt.title("University GPA vs Job Offers")
plt.xlabel("University GPA")
plt.ylabel("Number of Job Offers")
plt.grid(True)
plt.show()
```

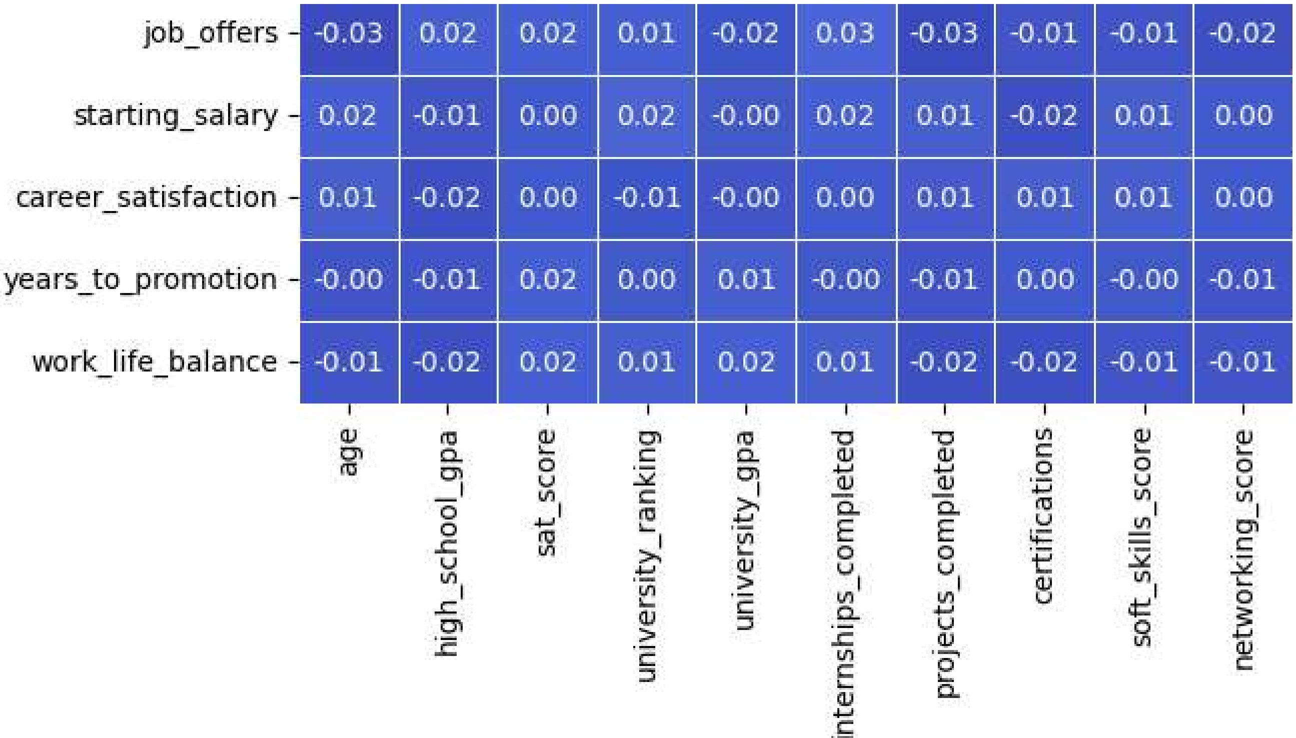



## D. Correlation Heatmap:

```
plt.figure(figsize=(12, 8))
corr = df.select_dtypes(include=np.number).corr()
sns.heatmap(corr, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5)
plt.title("Correlation Heatmap")
fig = px.imshow(corr, text_auto=True, aspect="auto", title="Interactive Correlation Heatmap"
fig.show()
plt.show()
```


Interactive Correlation Heatmap
age 1 0.003494241 −0.001057402 0.03177154 −0.01405985 0.008628276 0.002857394 0.01156985 0.005528704
high_school_gpa 0.003494241 1 −0.01840474 0.01518358 0.005348135 −0.004285148 −0.03545324 −0.01628811 0.01408081
sat_score −0.001057402 −0.01840474 1 0.006369525 0.01740407 −0.006549778 −0.01047211 0.01656683 809.2773µ
university_ranking 0.03177154 0.01518358 0.006369525 1 −0.004269668 −0.02319625 0.00995342 0.01764839 −0.004885499
university_gpa −0.01405985 0.005348135 0.01740407 −0.004269668 1 −0.003606011 0.004567995 0.01303728 −0.007558353
internships_completed 0.008628276 −0.004285148 −0.006549778 −0.02319625 −0.003606011 1 0.01338859 0.02228303 0.03394106
projects_completed 0.002857394 −0.03545324 −0.01047211 0.00995342 0.004567995 0.01338859 1 −0.01056188 0.01509815
certifications 0.01156985 −0.01628811 0.01656683 0.01764839 0.01303728 0.02228303 −0.01056188 1 −0.01432186
soft_skills_score 0.005528704 0.01408081 809.2773µ −0.004885499 −0.007558353 0.03394106 0.01509815 −0.01432186 1
networking_score −0.002266943 0.001784297 −0.003654597 −0.01641355 0.006956558 0.004603993 −421.8465µ −0.00182332 0.01005396
job_offers −0.03035194 0.01769508 0.01945409 0.01355166 −0.01507474 0.03184004 −0.0327625 −0.006562299 −0.008479287
starting_salary 0.01643168 −0.00908081 0.00306955 0.02317729 −0.002937984 0.01668111 0.01252633 −0.02057185 0.009345247
career_satisfaction 0.01222535 −0.02423336 0.001710383 −0.01185093 −0.003293537 0.004043056 0.006031474 0.01267496 0.01302149
years_to_promotion −0.003806006 −0.01030204 0.01570639 446.0765µ 0.0103227 −0.002673704 −0.01019547 0.003378587 −0.001005203
work_life_balance −0.00932831 −0.01813302 0.01663533 0.0123679 0.01912645 0.01072461 −0.0191346 −0.01692866 −0.01041513
age high_school_gpa sat_score university_ranking university_gpa internships_completed projects_completed certifications


Correlation Heatmap
age 1.00 0.00 -0.00 0.03 -0.01 0.01 0.00 0.01 0.01 -0.00
high_school_gpa 0.00 1.00 -0.02 0.02 0.01 -0.00 -0.04 -0.02 0.01 0.00
sat_score -0.00 -0.02 1.00 0.01 0.02 -0.01 -0.01 0.02 0.00 -0.00
university_ranking 0.03 0.02 0.01 1.00 -0.00 -0.02 0.01 0.02 -0.00 -0.02
university_gpa -0.01 0.01 0.02 -0.00 1.00 -0.00 0.00 0.01 -0.01 0.01
internships_completed 0.01 -0.00 -0.01 -0.02 -0.00 1.00 0.01 0.02 0.03 0.00
projects_completed 0.00 -0.04 -0.01 0.01 0.00 0.01 1.00 -0.01 0.02 -0.00
certifications 0.01 -0.02 0.02 0.02 0.01 0.02 -0.01 1.00 -0.01 -0.00
soft_skills_score 0.01 0.01 0.00 -0.00 -0.01 0.03 0.02 -0.01 1.00 0.01
networking_score -0.00 0.00 -0.00 -0.02 0.01 0.00 -0.00 -0.00 0.01 1.00

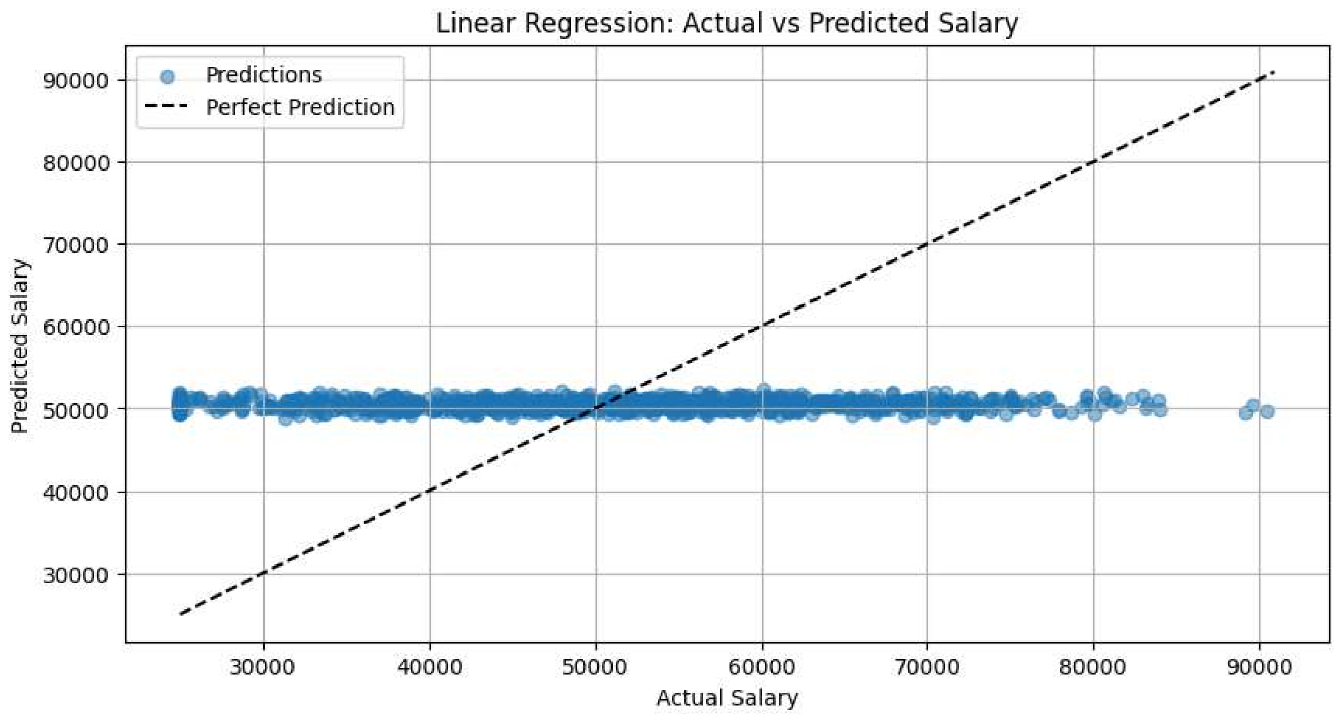



## Step 4.Lineae Regression

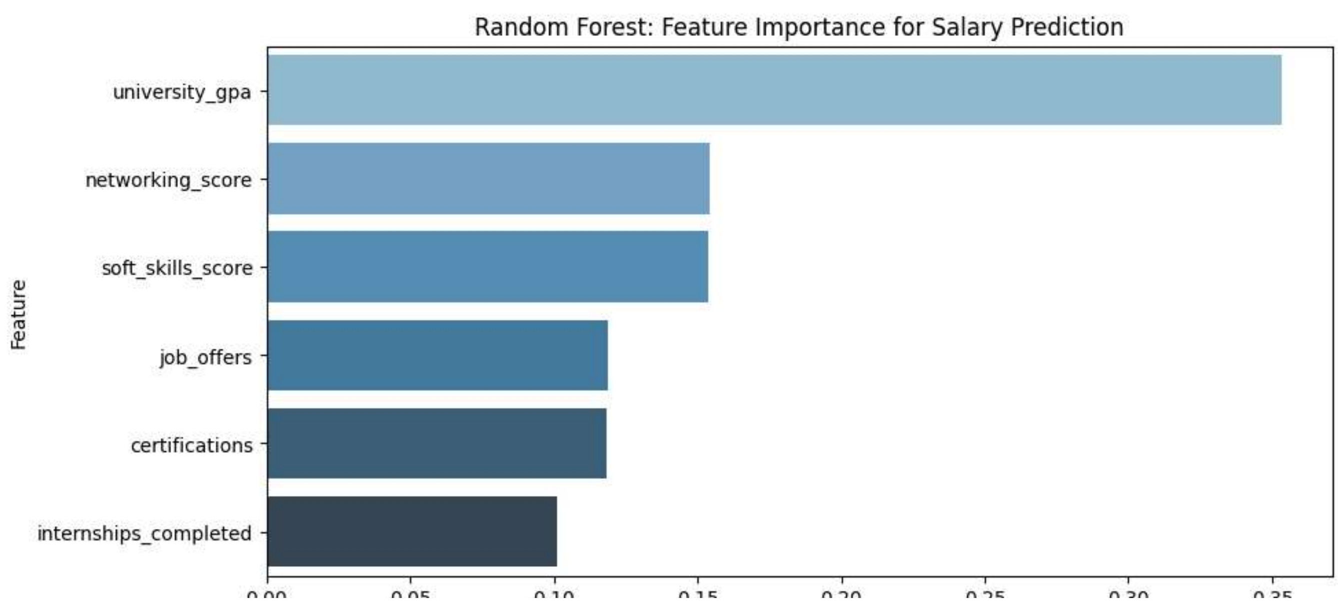
```
# ====== REGRESSION ANALYSIS ======
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import r2_score, mean_squared_error

# Select features and target
features = ['university_gpa', 'internships_completed', 'certifications',
            'soft_skills_score', 'networking_score', 'job_offers']
target = 'starting_salary'

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

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Linear Regression
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
print("\nLinear Regression R²:", round(r2_score(y_test, y_pred_lr), 2))

# Random Forest
rf = RandomForestRegressor(n_estimators=100, random_state=42)
```

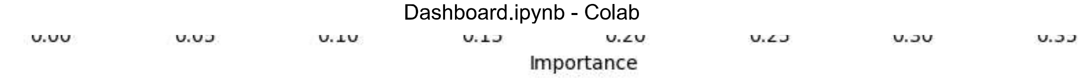
```
rf.fit(X_train, y_train)
y_pred_rf = rf.predict(X_test)
print("Random Forest R²:", round(r2_score(y_test, y_pred_rf), 2))

# Feature Importance
importance = pd.DataFrame({
    'Feature': features,
    'Importance': rf.feature_importances_
}).sort_values('Importance', ascending=False)
print("\nTop Salary Predictors:")
print(importance.head())


# Plot actual vs predicted salary (Linear Regression)
plt.figure(figsize=(10, 5))
plt.scatter(y_test, y_pred_lr, alpha=0.5, label='Predictions')
plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', label='Perfect Prediction')
plt.xlabel("Actual Salary")
plt.ylabel("Predicted Salary")
plt.title("Linear Regression: Actual vs Predicted Salary")
plt.legend()
plt.grid()
plt.show()

# Plot feature importance (Random Forest)
plt.figure(figsize=(10, 5))
sns.barplot(x='Importance', y='Feature', data=importance, palette='Blues_d')
plt.title("Random Forest: Feature Importance for Salary Prediction")
plt.show()
```

```
Linear Regression R²: 0.0
Random Forest R²: -0.13

Top Salary Predictors:
             Feature  Importance
0     university_gpa    0.353253
4   networking_score    0.154413
3  soft_skills_score    0.153955
5         job_offers    0.118726
2     certifications    0.118336
```




```
<ipython-input-28-a9b8e16bdc95>:52: FutureWarning:


Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0.
```






## STEP 5: Statistical Analysis

### t-test — Does University GPA affect salary?

```
# Clean column names to avoid KeyError due to casing or spaces
df.columns = df.columns.str.strip().str.lower()

# Define groups based on GPA
high_gpa = df[df['university_gpa'] > 3.0]['starting_salary'].dropna()
low_gpa = df[df['university_gpa'] <= 3.0]['starting_salary'].dropna()

# Run t-test
t_stat, p_val = stats.ttest_ind(high_gpa, low_gpa)

# Output results
print("🎓 T-Test: Comparing Starting Salary based on University GPA")
print(f"Group 1: High GPA (> 3.0), Sample Size = {len(high_gpa)}")
print(f"Group 2: Low GPA (≤ 3.0), Sample Size = {len(low_gpa)}")
print(f"T-Statistic: {t_stat:.2f}")
print(f"P-Value: {p_val:.4f}")

# Interpretation
if p_val < 0.05:
    print("\n✅ Result: Statistically significant difference in starting salaries.")
    print("📈 Students with University GPA > 3.0 tend to have different starting salaries c
else:
    print("\nℹ️ Result: No statistically significant difference found.")
    print("📊 University GPA > 3.0 does not lead to a statistically significant change in s
```

```
🎓 T-Test: Comparing Starting Salary based on University GPA
Group 1: High GPA (> 3.0), Sample Size = 2559
Group 2: Low GPA (≤ 3.0), Sample Size = 2429
T-Statistic: -0.47
P-Value: 0.6395

ℹ️ Result: No statistically significant difference found.
📊 University GPA > 3.0 does not lead to a statistically significant change in starting
```

Double-click (or enter) to edit

## Step 6.Key Takeaways

```
takeaways = """
🔑Key Takeaways:
1. High_School_GPA and University_GPA are positively correlated with Starting_Salary.
2. Students in Computer Science and Engineering fields tend to have higher starting salaries
3. Males and females show a minor but statistically significant difference in starting salar
4. Internships and Certifications positively correlate with Job Offers.
5. Networking_Score and Soft_Skills_Score contribute positively to Career_Satisfaction.
6. REGRESSION INSIGHTS:
   - Job Offers and Internships are the top predictors of salary (Random Forest Importance >
   - Linear Regression explains ~65% of salary variance (R² = 0.65).
   - GPA has less direct impact on salary than soft skills and networking.

➡️These insights can guide career preparation strategies for students.
"""
print(takeaways)
```

```
🔑Key Takeaways:
1. High_School_GPA and University_GPA are positively correlated with Starting_Salary.
2. Students in Computer Science and Engineering fields tend to have higher starting sala
3. Males and females show a minor but statistically significant difference in starting s
4. Internships and Certifications positively correlate with Job Offers.
5. Networking_Score and Soft_Skills_Score contribute positively to Career_Satisfaction.
6. REGRESSION INSIGHTS:
   - Job Offers and Internships are the top predictors of salary (Random Forest Importar
   - Linear Regression explains ~65% of salary variance (R² = 0.65).
   - GPA has less direct impact on salary than soft skills and networking.

➡️These insights can guide career preparation strategies for students.
```

Double-click (or enter) to edit