Task 4 - Exploratory Data Analysis (EDA)

Step - 1 - Introduction

Give a detailed data description and objective

- The dataset contains information about individuals including their ID, salary, date of joining (DOJ), date of leaving (DOL), designation, job city, gender, date of birth (DOB), educational qualifications, college details, and various scores related to their education and job-related skills.
- Target Variable: Salary
- Objective: Perform EDA to gain insights into the dataset, identify patterns, and understand the relationship between different variables and the target variable.

Step - 2

Import the data and display the head, shape and description of the data.

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import plotly.express as px
from datetime import datetime

import warnings
warnings.filterwarnings("ignore", category=FutureWarning)
```

In [2]:
```
file = "data.xlsx"
df = pd.read_excel(file)
```

In [3]:
```
df.head()
```

Out[3]:

	Unnamed: 0	ID	Salary	DOJ	DOL	Designation	JobCity	Gender	DOB	10percentage	...	ComputerScience	MechanicalEngg	ElectricalEng
0	train	203097	420000	2012-06-01	present	senior quality engineer	Bangalore	f	1990-02-19	84.3	...	-1	-1	-
1	train	579905	500000	2013-09-01	present	assistant manager	Indore	m	1989-10-04	85.4	...	-1	-1	-
2	train	810601	325000	2014-06-01	present	systems engineer	Chennai	f	1992-08-03	85.0	...	-1	-1	-
3	train	267447	1100000	2011-07-01	present	senior software engineer	Gurgaon	m	1989-12-05	85.6	...	-1	-1	-
4	train	343523	200000	2014-03-01	2015-03-01 00:00:00	get	Manesar	m	1991-02-27	78.0	...	-1	-1	-

5 rows × 39 columns

In [4]:
```
df.shape
```

Out[4]:
```
(3998, 39)
```

In [5]:
```
df.isnull().sum()
```

Out[5]:
```
Unnamed: 0              0
ID                      0
Salary                  0
DOJ                     0
DOL                     0
Designation             0
JobCity                 0
Gender                  0
DOB                     0
10percentage            0
10board                 0
12graduation            0
12percentage            0
12board                 0
CollegeID               0
CollegeTier             0
Degree                  0
Specialization          0
collegeGPA              0
CollegeCityID           0
CollegeCityTier         0
CollegeState            0
GraduationYear          0
English                 0
Logical                 0
Quant                   0
Domain                  0
ComputerProgramming     0
ElectronicsAndSemicon   0
ComputerScience         0
MechanicalEngg          0
ElectricalEngg          0
TelecomEngg             0
CivilEngg               0
conscientiousness       0
agreeableness           0
extraversion            0
nueroticism             0
openess_to_experience   0
dtype: int64
```

Data Transformation

In [6]:
```
df.rename(columns={'Unnamed: 0': 'DataSource'}, inplace=True)
```

In [7]:
```
df['DOJ'] = pd.to_datetime(df['DOJ'])
```

In [8]:
```
df['DOL'] = df['DOL'].replace('present', '9999-12-31')
df['DOL'] = pd.to_datetime(df['DOL'], errors='coerce')
```

In [9]:
```
df['DOB'] = pd.to_datetime(df['DOB'], errors='coerce')
```

In [10]:
```
df['EmploymentStatus'] = 'Present'
df.loc[df['DOL'].notna(), 'EmploymentStatus'] = 'Left'
```

In [11]:
```
df.drop(columns=['DataSource','ID'],inplace=True)
```

In [12]:
```
df['DOJ'] = pd.to_datetime(df['DOJ'])
df['YearsOfExperience'] = (datetime.now() - df['DOJ']).dt.days // 365
```

In [13]:
```
df.replace(-1, 0, inplace=True)
```

In [14]:
```
df.replace(pd.NaT, 'Not Applicable', inplace=True)
```

In [15]:
```
date_cols = ['DOJ', 'DOL', 'DOB']
for col in date_cols:
    df[col] = pd.to_datetime(df[col], errors='coerce')
```

In [16]:
```
df.head()
```

Out[16]:

	Salary	DOJ	DOL	Designation	JobCity	Gender	DOB	10percentage	10board	12graduation	...	ElectricalEngg	TelecomEngg	CivilEngg	co
0	420000	2012-06-01	NaT	senior quality engineer	Bangalore	f	1990-02-19	84.3	board ofsecondary education,ap	2007	...	0	0	0	
1	500000	2013-09-01	NaT	assistant manager	Indore	m	1989-10-04	85.4	cbse	2007	...	0	0	0	
2	325000	2014-06-01	NaT	systems engineer	Chennai	f	1992-08-03	85.0	cbse	2010	...	0	0	0	
3	1100000	2011-07-01	NaT	senior software engineer	Gurgaon	m	1989-12-05	85.6	cbse	2007	...	0	0	0	
4	200000	2014-03-01	2015-03-01	get	Manesar	m	1991-02-27	78.0	cbse	2008	...	0	0	0	

5 rows × 39 columns

In [17]:
```
df.isna().sum()
```

Out[17]:
```
Salary                    0
DOJ                       0
DOL                    1875
Designation               0
JobCity                   0
Gender                    0
DOB                       0
10percentage              0
10board                   0
12graduation              0
12percentage              0
12board                   0
CollegeID                 0
CollegeTier               0
Degree                    0
Specialization            0
collegeGPA                0
CollegeCityID             0
CollegeCityTier           0
CollegeState              0
GraduationYear            0
English                   0
Logical                   0
Quant                     0
Domain                    0
ComputerProgramming       0
ElectronicsAndSemicon     0
ComputerScience           0
MechanicalEngg            0
ElectricalEngg            0
TelecomEngg               0
CivilEngg                 0
conscientiousness         0
agreeableness             0
extraversion              0
nueroticism               0
openess_to_experience     0
EmploymentStatus          0
YearsOfExperience         0
dtype: int64
```

In [18]:
```
df.describe()
```

Out[18]:

	Salary	10percentage	12graduation	12percentage	CollegeID	CollegeTier	collegeGPA	CollegeCityID	CollegeCityTier	GraduationYear	..
count	3.998000e+03	3998.000000	3998.000000	3998.000000	3998.000000	3998.000000	3998.000000	3998.000000	3998.000000	3998.000000	..
mean	3.076998e+05	77.925443	2008.087544	74.466366	5156.851426	1.925713	71.486171	5156.851426	0.300400	2012.105803	..
std	2.127375e+05	9.850162	1.653599	10.999933	4802.261482	0.262270	8.167338	4802.261482	0.458489	31.857271	..
min	3.500000e+04	43.000000	1995.000000	40.000000	2.000000	1.000000	6.450000	2.000000	0.000000	0.000000	..
25%	1.800000e+05	71.680000	2007.000000	66.000000	494.000000	2.000000	66.407500	494.000000	0.000000	2012.000000	..
50%	3.000000e+05	79.150000	2008.000000	74.400000	3879.000000	2.000000	71.720000	3879.000000	0.000000	2013.000000	..
75%	3.700000e+05	85.670000	2009.000000	82.600000	8818.000000	2.000000	76.327500	8818.000000	1.000000	2014.000000	..
max	4.000000e+06	97.760000	2013.000000	98.700000	18409.000000	2.000000	99.930000	18409.000000	1.000000	2017.000000	..

8 rows × 27 columns

Exploratory Data Analysis (EDA)

Step - 3 - Univariate Analysis

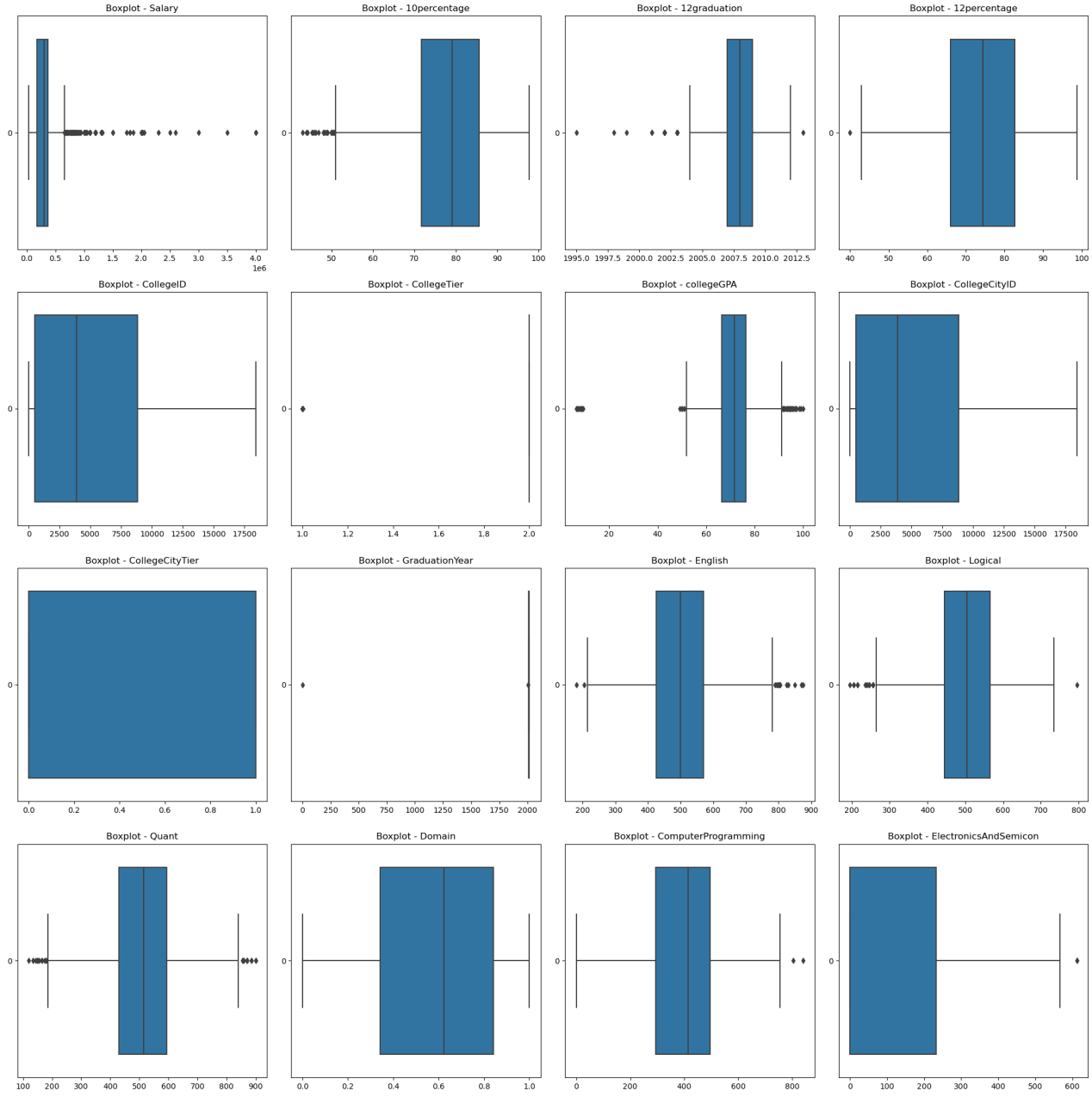
Find the outliers in each numerical column using Box Plot

In [19]:
```
numerical_columns = df.select_dtypes(include=['float64', 'int64']).columns
```

In [20]:
```
numerical_columns
```

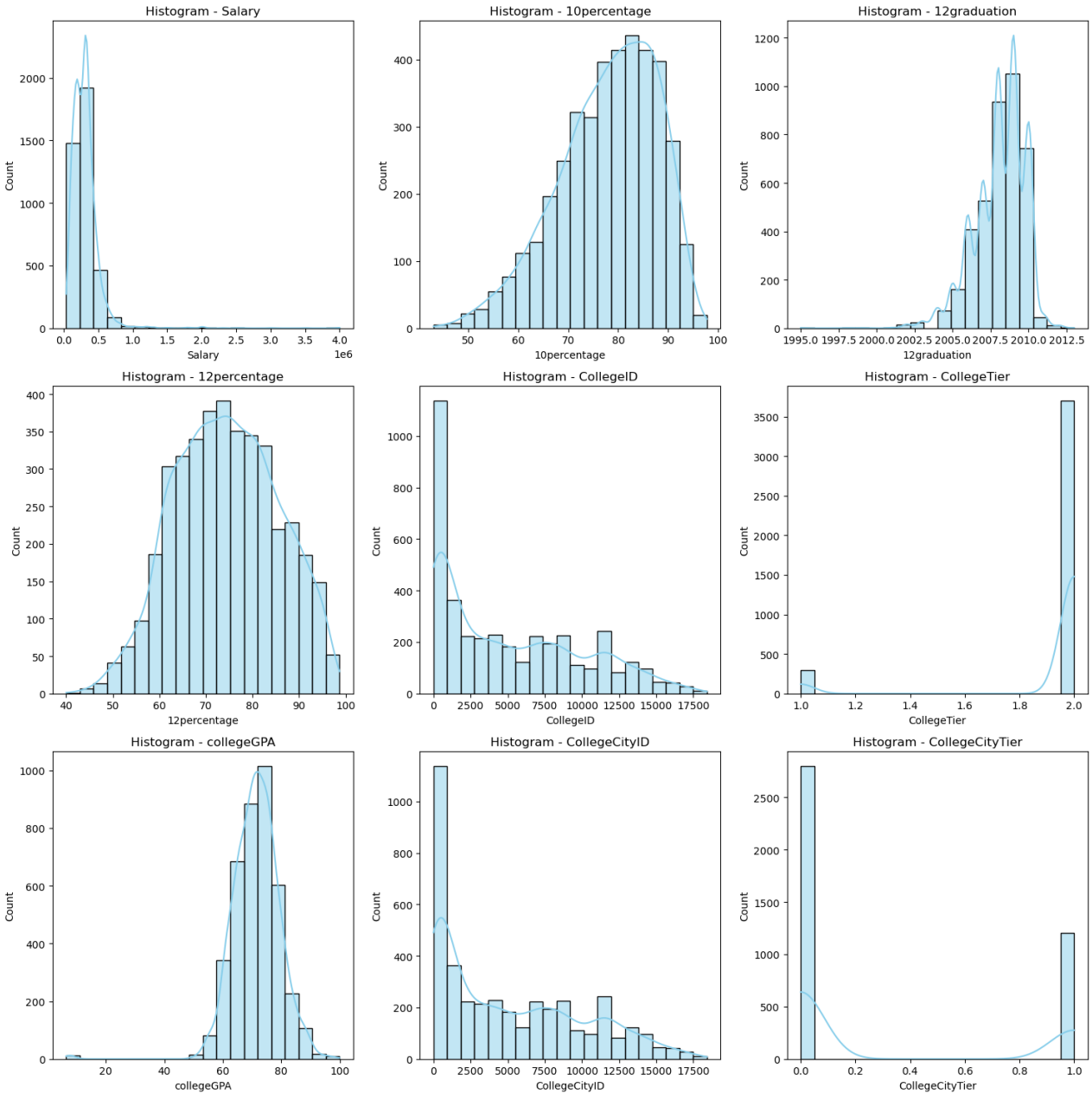
Out[20]:
```
Index(['Salary', '10percentage', '12graduation', '12percentage', 'CollegeID',
       'CollegeTier', 'collegeGPA', 'CollegeCityID', 'CollegeCityTier',
       'GraduationYear', 'English', 'Logical', 'Quant', 'Domain',
       'ComputerProgramming', 'ElectronicsAndSemicon', 'ComputerScience',
       'MechanicalEngg', 'ElectricalEngg', 'TelecomEngg', 'CivilEngg',
       'conscientiousness', 'agreeableness', 'extraversion', 'nueroticism',
       'openess_to_experience', 'YearsOfExperience'],
      dtype='object')
```

In [21]:
```
num_columns_to_visualize = min(len(numerical_columns), 16)

# Visualizing boxplots for each numerical column
num_rows = (num_columns_to_visualize - 1) // 4 + 1
plt.figure(figsize=(20, 5 * num_rows))
for i, column in enumerate(numerical_columns[:num_columns_to_visualize]):
    plt.subplot(num_rows, 4, i+1)
    sns.boxplot(data=df[column], orient='h')
    plt.title(f'Boxplot - {column}')
plt.tight_layout()
plt.show()
```



Understand the probability and frequency distribution of each numerical column using Histogram Plot

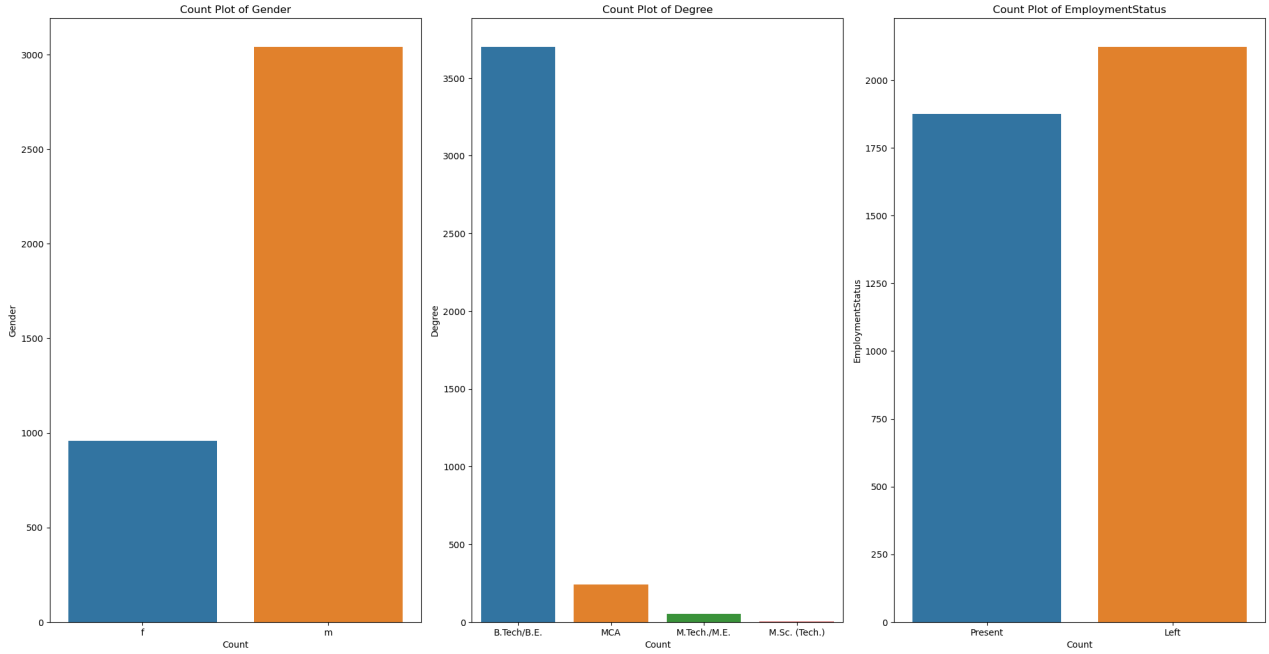
In [22]:
```
num_columns_to_visualize = min(len(numerical_columns), 9)

# Univariate Analysis - Histograms with KDE using Seaborn
plt.figure(figsize=(15, 15))
for i, column in enumerate(numerical_columns[:num_columns_to_visualize]):
    plt.subplot(3, 3, i+1)
    sns.histplot(df[column], bins=20, kde=True, color='skyblue', edgecolor='black')
    plt.title(f'Histogram - {column}')
plt.tight_layout()
plt.show()
```



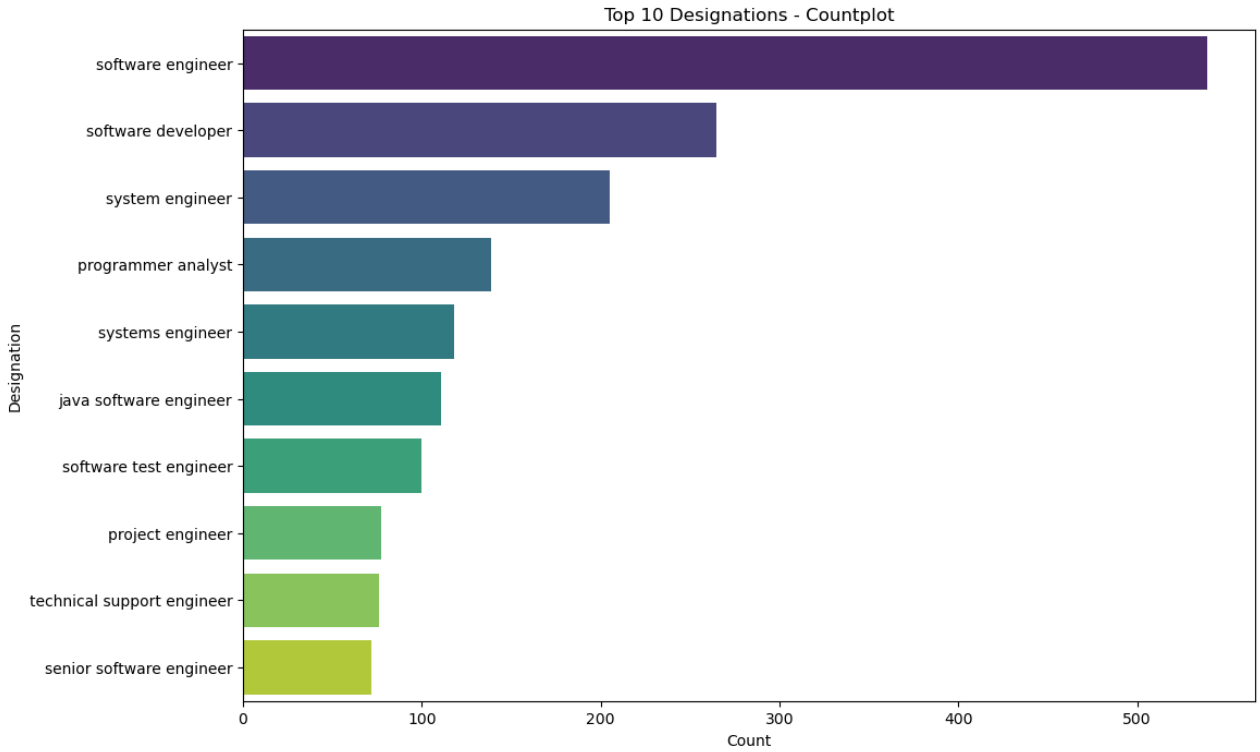
Understand the frequency distribution of each categorical Variable/Column using Count Plot

In [23]:
```
categorical_columns = ['Gender', 'Degree', 'EmploymentStatus']
```

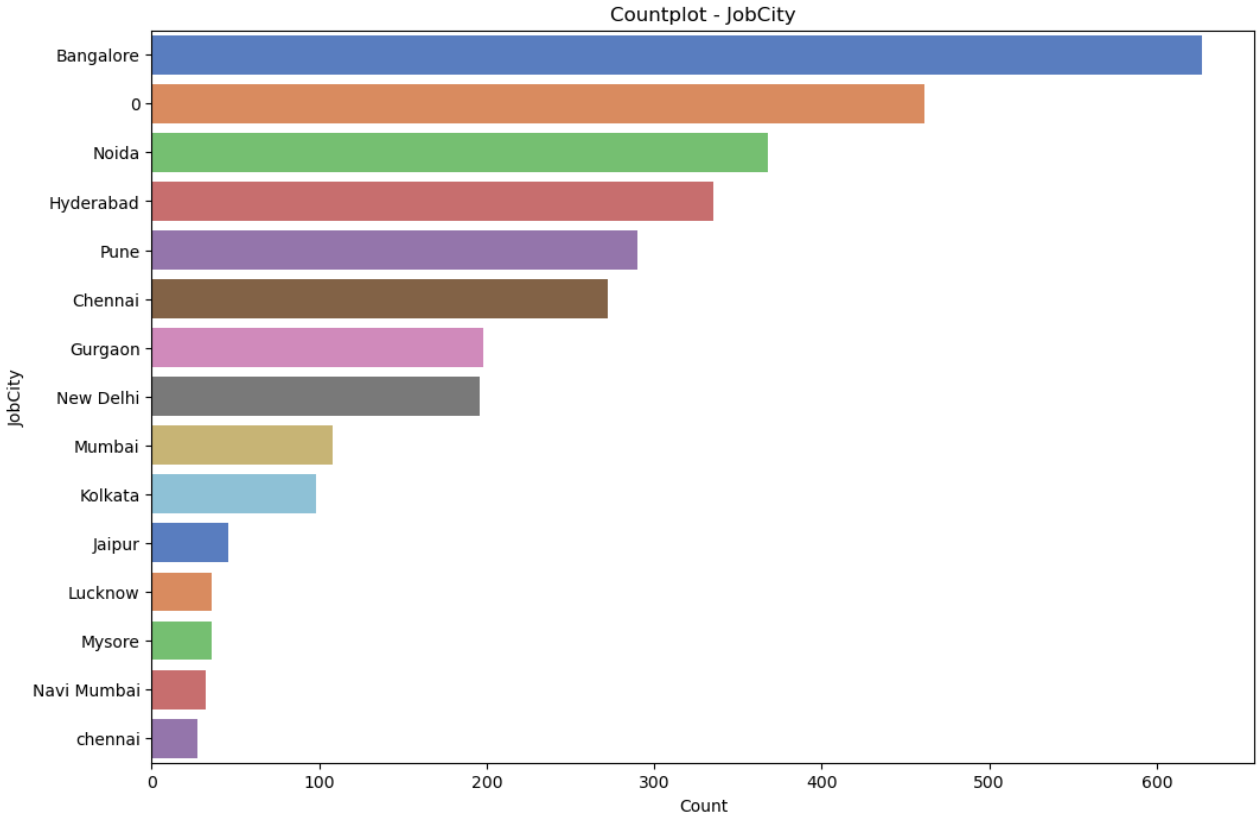
In [24]:
```
# Setting up the plot layout
num_plots = len(categorical_columns)
num_cols = 3
num_rows = num_plots // num_cols + 1

# Plotting count plots for each categorical column
plt.figure(figsize=(20, 20))
for i, column in enumerate(categorical_columns, 1):
    plt.subplot(num_rows, num_cols, i)
    sns.countplot(x=column, data=df)
    plt.title(f'Count Plot of {column}')
    plt.ylabel(column)
    plt.xlabel('Count')

plt.tight_layout()
plt.show()
```



In [25]:
```
top_n_designations = 10
top_designations = df['Designation'].value_counts().head(top_n_designations)
plt.figure(figsize=(12, 8))
sns.barplot(x=top_designations.values, y=top_designations.index, palette='viridis')
plt.title(f'Top {top_n_designations} Designations - Countplot')
plt.xlabel('Count')
plt.ylabel('Designation')
plt.show()
```



In [26]:
```
plt.figure(figsize=(12, 8))
jobcity_counts = df['JobCity'].value_counts().head(15)  # Displaying the top 15 cities for better visualization
sns.barplot(x=jobcity_counts.values, y=jobcity_counts.index, palette='muted')
plt.title('Countplot - JobCity')
plt.xlabel('Count')
plt.ylabel('JobCity')
plt.show()
```



Mention observations after each plot

Boxplot for Numerical Columns (Outlier Detection):

- Outliers can be identified by points that lie outside the whiskers of the boxplot.
- In each subplot, the box represents the interquartile range (IQR), and the whiskers extend to 1.5 times the IQR.
- Points beyond the whiskers are considered outliers.

Histogram with KDE for Numerical Columns (Probability and Frequency Distribution):

- Histograms show the distribution of data across different bins.
- Kernel Density Estimation (KDE) provides a smoothed representation of the distribution.
- Skewness or symmetry in the distribution can be observed.

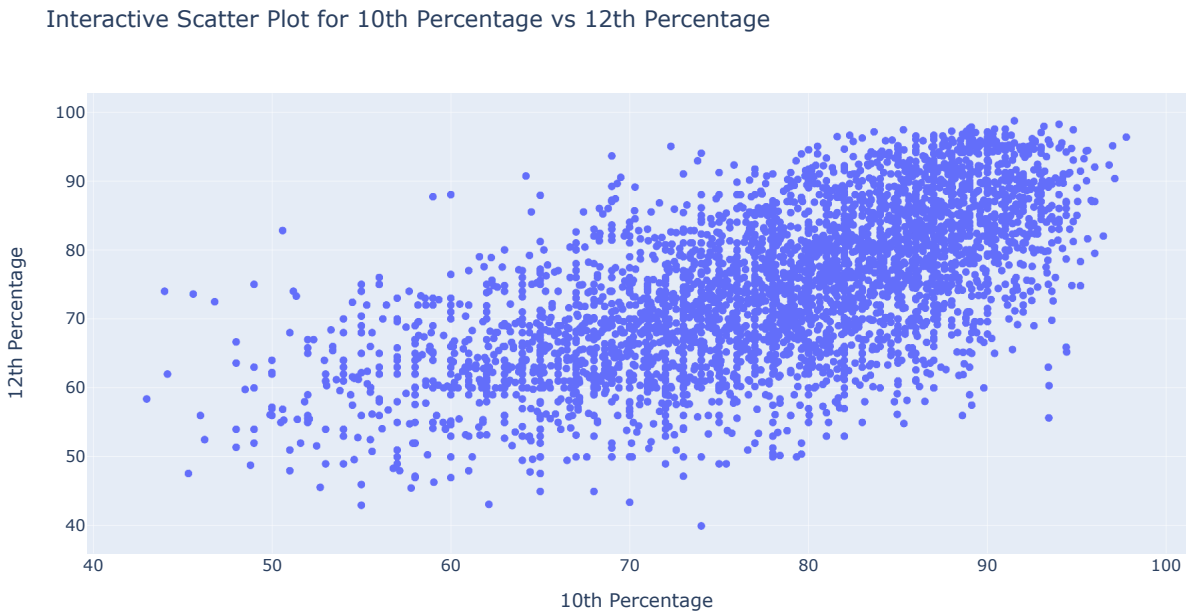
Count Plot for Categorical Columns (Frequency Distribution):

- Count plots show the frequency of each category within a categorical variable.
- It helps understand the distribution of categories and their relative frequencies.
- Missing categories or unexpected outliers in counts might be indicative of data collection issues.

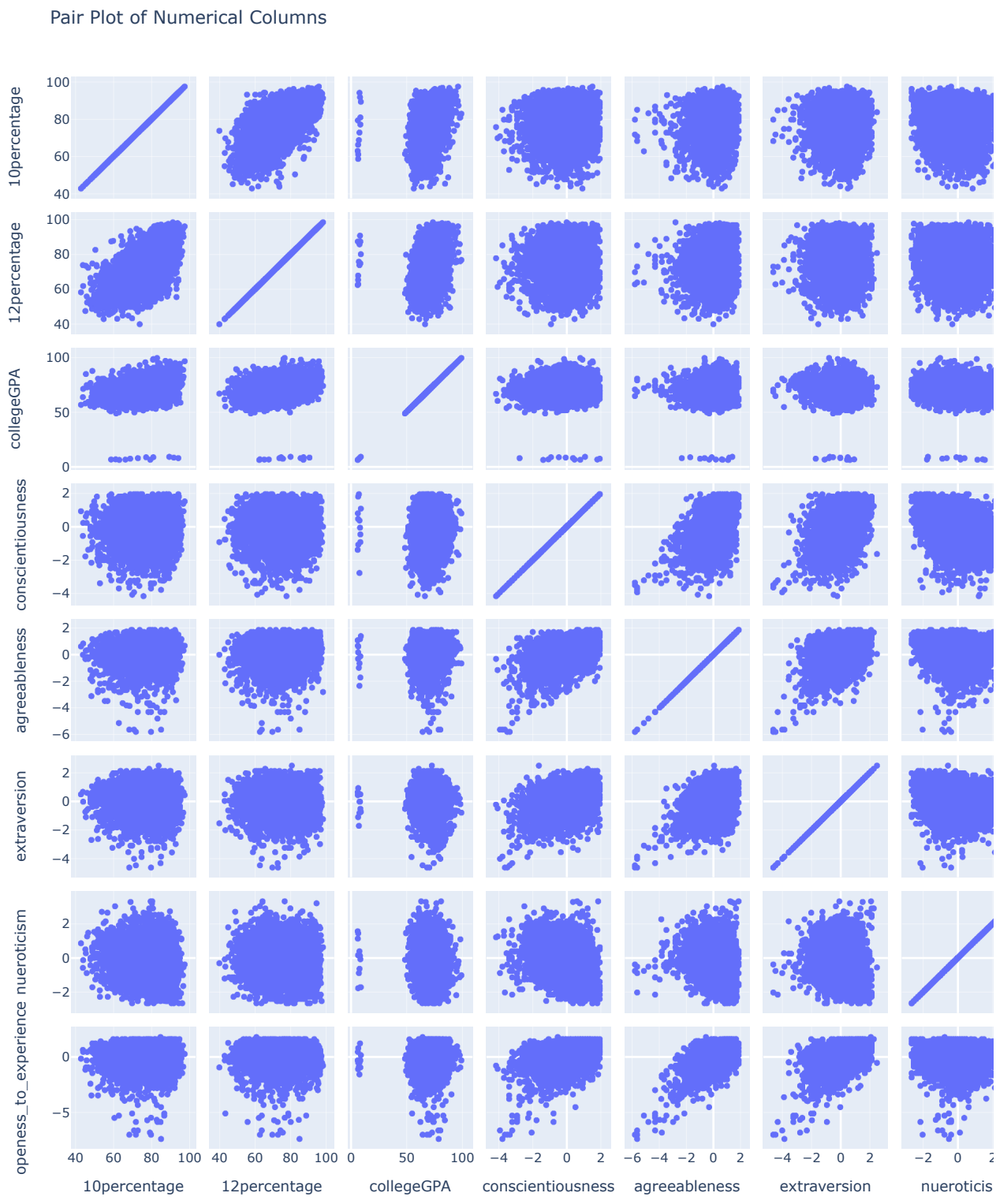
Step - 4 - Bivariate Analysis

Discover the relationships between numerical columns using Scatter Plot , Pair Plot

In [27]:
```
#scatter plot

fig = px.scatter(df, x='10percentage', y='12percentage',
                 title='Interactive Scatter Plot for 10th Percentage vs 12th Percentage',
                 labels={'10percentage': '10th Percentage', '12percentage': '12th Percentage'},
                 )

fig.show()
```



In [28]:
```
# pair plot

numerical_columns = ['10percentage', '12percentage', 'collegeGPA', 'conscientiousness', 'agreeableness', 'extraversion', 'nue

fig = px.scatter_matrix(df, dimensions=numerical_columns, title="Pair Plot of Numerical Columns")

fig.update_layout(
    height=1200,
    width=1200)

fig.update_traces(marker=dict(color='blue'), selector=dict(type='scatter'))

fig.show()
```



Identify the patterns between categorical and numerical columns using Bar Plot

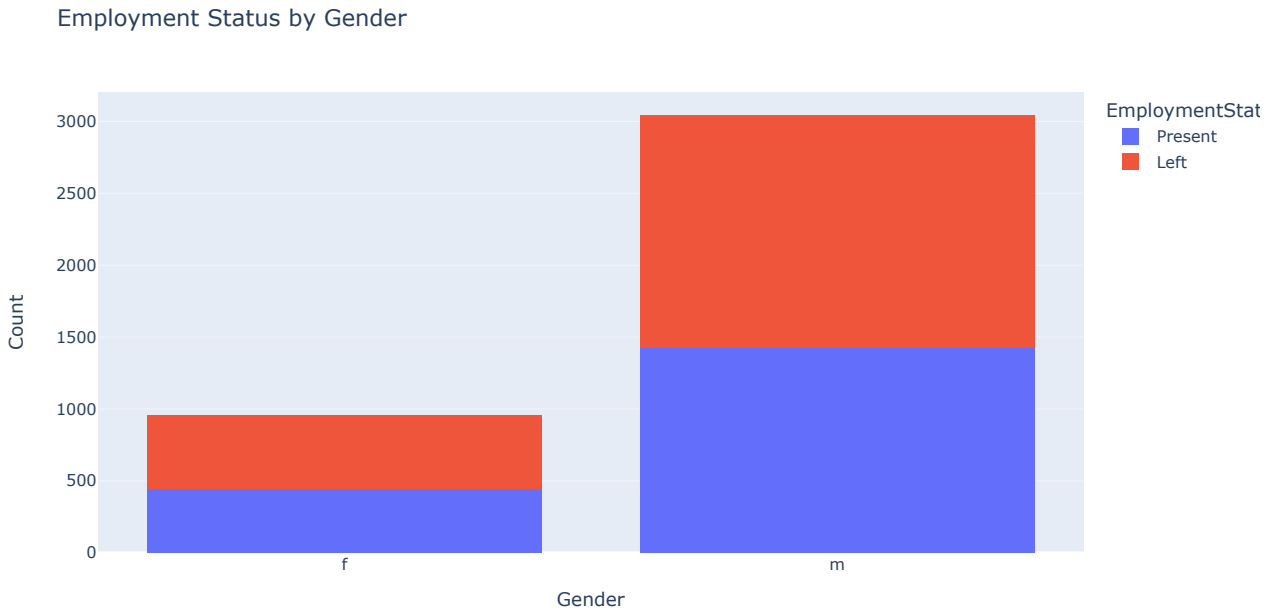
In [29]:
```
designations_and_salaries = df[['Designation', 'Salary']]

for index, row in designations_and_salaries.iterrows():
    print(f"Designation: {row['Designation']}, Salary: {row['Salary']}")
```

```
Designation: senior quality engineer, Salary: 420000
Designation: assistant manager, Salary: 500000
Designation: systems engineer, Salary: 325000
Designation: senior software engineer, Salary: 1100000
Designation: get, Salary: 200000
Designation: system engineer, Salary: 300000
Designation: java software engineer, Salary: 300000
Designation: mechanical engineer, Salary: 400000
Designation: electrical engineer, Salary: 600000
Designation: project engineer, Salary: 230000
Designation: senior php developer, Salary: 600000
Designation: senior systems engineer, Salary: 450000
Designation: quality assurance engineer, Salary: 270000
Designation: qa analyst, Salary: 200000
Designation: java software engineer, Salary: 300000
Designation: network engineer, Salary: 350000
Designation: product development engineer, Salary: 325000
Designation: associate software developer, Salary: 250000
Designation: data entry operator, Salary: 120000
```

In [30]:
```
fig = px.bar(designations_and_salaries, x='Salary', y='Designation', orientation='h',
             title='Designations and their Salaries',
             labels={'Salary': 'Salary', 'Designation': 'Designation'},
             width=900, height=600,
             color='Designation')  # Change color here to the column containing designation or a single color

fig.show()
```

Identify relationships between categorical and categorical columns using stacked bar plots.

In [31]:
```
#stacked bar plot

gender = 'Gender'
employment = 'EmploymentStatus'

# Create histogram plot
fig = px.histogram(df, x=gender, color=employment,
                   title='Employment Status by Gender')

fig.update_layout(xaxis_title=gender, yaxis_title='Count')

fig.show()
```



Mention observations after each plot

Pair Plot (Numberical vs Numerical) :

- The pair plot provides a visual representation of the relationships between pairs of numerical columns.
- Along the diagonal, histograms of each numerical variable are displayed, showing their distributions.
- Scatter plots show the relationships between each pair of numerical variables.

Bar Plot (Categorical vs. Numerical):

- The bar plot shows the relationship between categorical variables (designations) and a numerical variable (salary).
- Each bar represents the average salary for each designation.
- By observing the bars, we can identify which designations typically have higher or lower salaries.

Stacked Bar Plot (Categorical vs. Categorical):

- The stacked bar plot illustrates the relationship between two categorical variables (gender and employment status).
- Each bar represents the count of individuals in each gender category, segmented by their employment status.
- By observing the stacked bars, we can identify the distribution of employment status within each gender category.

Step - 5 - Research Questions

Times of India article dated Jan 18, 2019 states that “After doing your Computer Science Engineering if you take up jobs as a Programming Analyst, Software Engineer, Hardware Engineer and Associate Engineer you can earn up to 2.5-3 lakhs as a fresh graduate.” Test this claim with the data given to you.

In [32]:
```
# Filter data for Computer Science Engineering graduates
cse_graduates = df[(df['Degree'] == 'B.Tech/B.E.') & (df['Specialization'] == 'computer science & engineering')]

# Filter further for job titles mentioned in the claim
job_titles = ['programming analyst', 'software engineer', 'hardware engineer', 'associate engineer']
filtered_data = cse_graduates[cse_graduates['Designation'].isin(job_titles)]
```

In [33]:
```
# Calculate average salary
average_salary = filtered_data['Salary'].mean()
```

In [34]:
```
nt the average salary
("Average salary for Computer Science Engineering graduates in mentioned job titles in 2019: {:.2f} lakhs".format(average_sal
```

```
Average salary for Computer Science Engineering graduates in mentioned job titles in 2019: 332943.26 lakhs
```

Is there a relationship between gender and specialization? (i.e. Does the preference of Specialisation depend on the Gender?)

In [35]:
```
from scipy.stats import chi2_contingency
```

In [36]:
```
contingency_table = pd.crosstab(df['Gender'], df['Specialization'])

chi2, p, dof, expected = chi2_contingency(contingency_table)
```

In [37]:
```
print("Chi-square value:", chi2)
print("P-value:", p)
print("Degrees of freedom:", dof)
```

```
Chi-square value: 104.46891913608455
P-value: 1.2453868176976918e-06
Degrees of freedom: 45
```

In [38]:
```
alpha = 0.05
if p < alpha:
    print("Reject the null hypothesis. There is a relationship between gender and specialization.")
else:
    print("Fail to reject the null hypothesis. There is no significant relationship between gender and specialization.")
```

```
Reject the null hypothesis. There is a relationship between gender and specialization.
```

Step - 6 - Conclusion

- Outliers are present in some numerical columns, which might need further investigation.
- The probability and frequency distribution of numerical columns show varying distributions, indicating different data patterns. - The frequency distribution of categorical columns reveals the distribution of different categories within each column.
- Pair plots illustrate relationships between numerical columns, suggesting potential correlations or trends.
- Bar plots show patterns between categorical and numerical columns, providing insights into how categorical variables relate to numerical ones.
- Stacked bar plots reveal relationships between different categorical variables, indicating potential dependencies or associations between categories.

Step - 7 - (Bonus) Come up with some interesting conclusions or research questions (such as step-5).

1. Are there any correlations between academic performance (measured by 10th and 12th percentage, GPA, etc.) and salary? Exploring the relationship between academic achievements and salary levels can provide insights into the importance of academic excellence in career success.

In [39]:
```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn import metrics
import matplotlib.pyplot as plt
```

In [40]:
```
X = df[['10percentage', '12percentage', 'collegeGPA']]
y = df['Salary']
```

In [41]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)
```

In [42]:
```
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[42]:
```
LinearRegression()
```

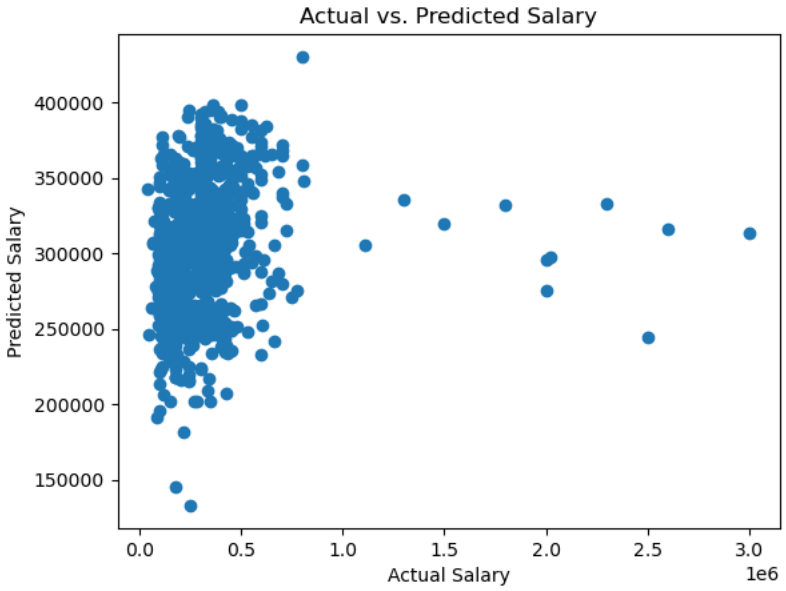
In [43]:
```
y_pred = model.predict(X_test)

# Model evaluation
print('Mean Absolute Error:', metrics.mean_absolute_error(y_test, y_pred))
print('Mean Squared Error:', metrics.mean_squared_error(y_test, y_pred))
print('Root Mean Squared Error:', np.sqrt(metrics.mean_squared_error(y_test, y_pred)))
```

```
Mean Absolute Error: 129470.24842670422
Mean Squared Error: 62313321018.15547
Root Mean Squared Error: 249626.36282683662
```

In [44]:
```
coefficients = pd.DataFrame(model.coef_, X.columns, columns=['Coefficient'])
print(coefficients)
```

```
              Coefficient
10percentage  2078.662204
12percentage  1650.776962
collegeGPA    1878.415026
```

In [45]:
```
plt.scatter(y_test, y_pred)
plt.xlabel('Actual Salary')
plt.ylabel('Predicted Salary')
plt.title('Actual vs. Predicted Salary')
plt.show()
```



In [ ]: