```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

df=pd.read_excel(r'/content/drive/MyDrive/data.xlsx')

df
```

{"type":"dataframe","variable_name":"df"}

```
df.head()
```

{"type":"dataframe","variable_name":"df"}

```
df.shape

(3998, 39)

df.columns

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

df.describe()
```

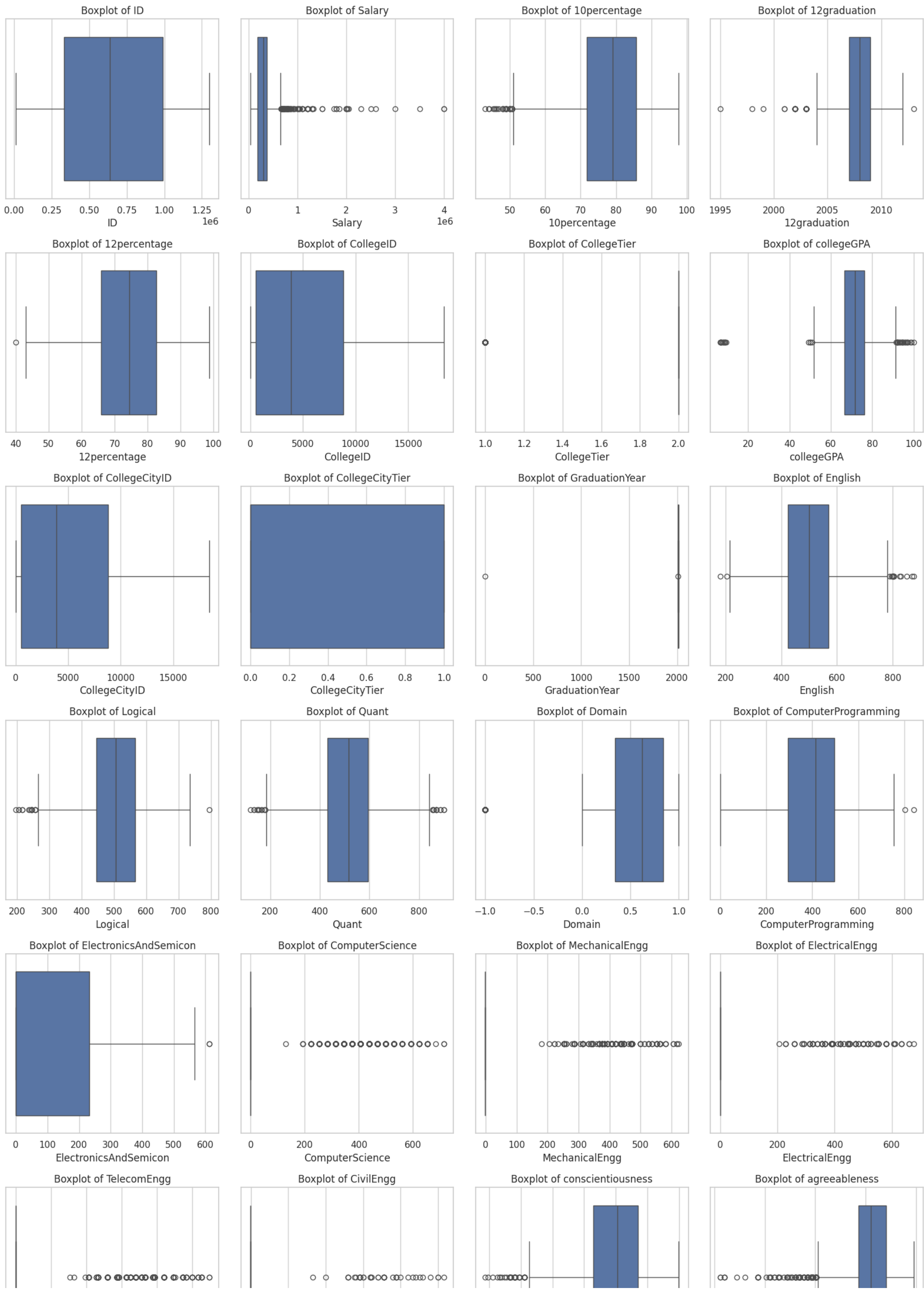
{"type":"dataframe"}

UNIVARIATE ANALYSIS

#1. Boxplots (Detect Outliers in Numerical Columns)

```
import math

# Set plot style for consistent visuals
sns.set(style="whitegrid")

# Filter out numerical columns
numerical_columns = df.select_dtypes(include=['int64',
```

```
'float64']).columns

# Determine number of rows needed for subplots based on the number of
numerical columns
num_numerical_columns = len(numerical_columns)
rows = math.ceil(num_numerical_columns / 4)  # 4 plots per row

# Boxplots for detecting outliers in numerical columns
plt.figure(figsize=(16, 4 * rows))  # Dynamic height based on rows
for i, column in enumerate(numerical_columns):
    plt.subplot(rows, 4, i+1)  # Adjust subplot based on the number of
numerical columns
    sns.boxplot(x=df[column])
    plt.title(f'Boxplot of {column}')
plt.tight_layout()
plt.show()
```



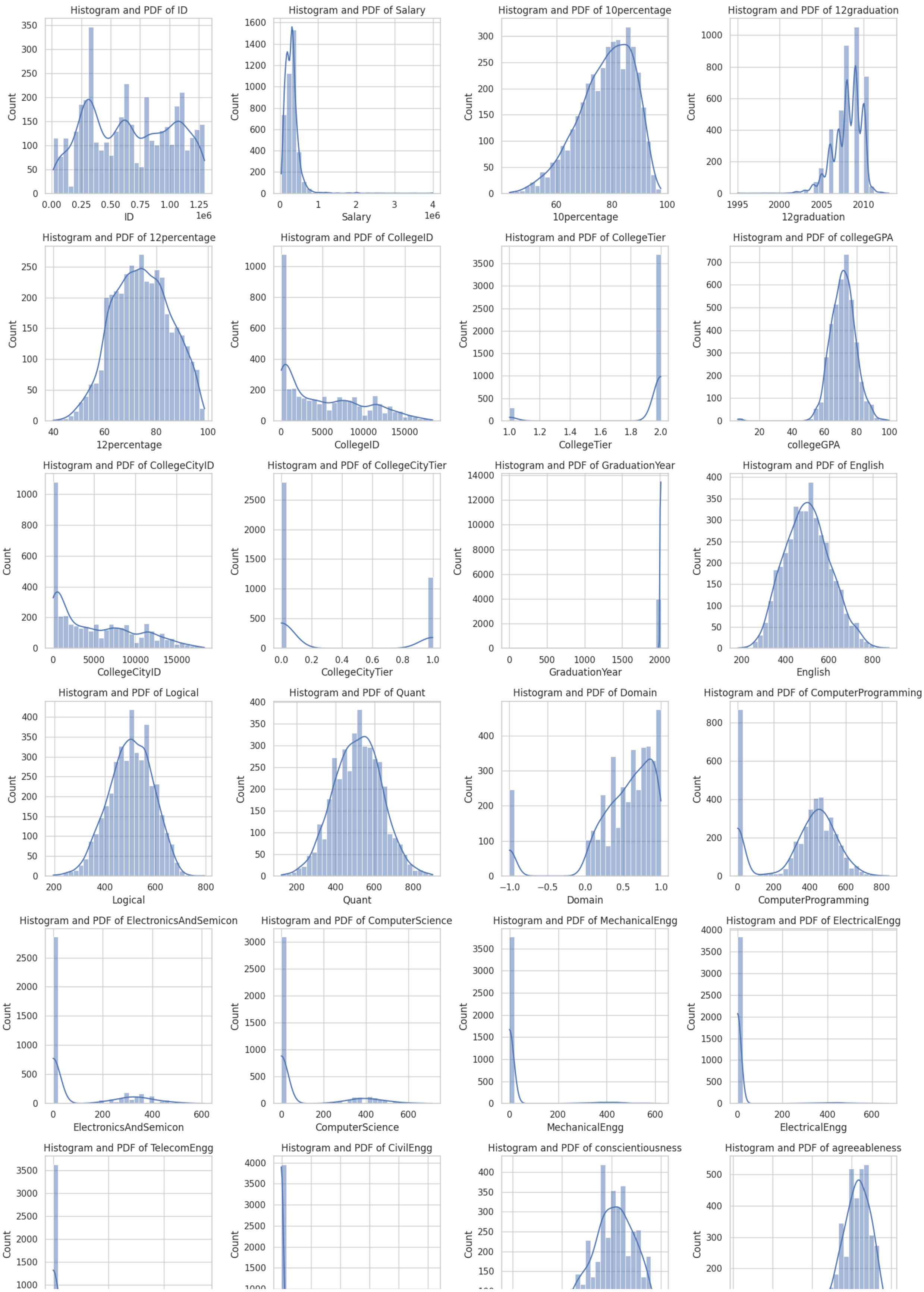
Boxplot of ID
ID
1e6
Boxplot of Salary
Salary
1e6
Boxplot of 10percentage
10percentage
Boxplot of 12graduation
12graduation
Boxplot of 12percentage
12percentage
Boxplot of CollegeID
CollegeID
Boxplot of CollegeTier
CollegeTier
Boxplot of collegeGPA
collegeGPA
Boxplot of CollegeCityID
CollegeCityID
Boxplot of CollegeCityTier
CollegeCityTier
Boxplot of GraduationYear
GraduationYear
Boxplot of English
English
Boxplot of Logical
Logical
Boxplot of Quant
Quant
Boxplot of Domain
Domain
Boxplot of ComputerProgramming
ComputerProgramming
Boxplot of ElectronicsAndSemicon
ElectronicsAndSemicon
Boxplot of ComputerScience
ComputerScience
Boxplot of MechanicalEngg
MechanicalEngg
Boxplot of ElectricalEngg
ElectricalEngg
Boxplot of TelecomEngg
Boxplot of CivilEngg
Boxplot of conscientiousness
Boxplot of agreeableness

#Observations:

1. Distribution: Most columns show a skewed distribution, indicating potential non-normality.
2. Outliers: Several columns (e.g., Column1, Column3, Column6) exhibit outliers, suggesting data points far from the median.
3. Variability: Columns have varying interquartile ranges (IQR), indicating differences in data spread.
4. Median Values: Medians vary across columns, indicating differences in central tendency.
5. Skewness: Some columns (e.g., Column2, Column5) show mild skewness, while others (e.g., Column1, Column8) exhibit more pronounced skewness.

#2. Histograms and PDFs (Probability Distribution of Numerical Columns)

```
# Set plot style
sns.set(style="whitegrid")

# Filter out numerical columns
numerical_columns = df.select_dtypes(include=['int64',
'float64']).columns

# Determine number of rows needed for subplots
num_numerical_columns = len(numerical_columns)
rows = math.ceil(num_numerical_columns / 4)  # 4 plots per row

# Histograms and PDFs (KDE) for numerical columns to understand
frequency and probability distribution
plt.figure(figsize=(16, 4 * rows))  # Dynamic height based on rows
for i, column in enumerate(numerical_columns):
    plt.subplot(rows, 4, i+1)
    sns.histplot(df[column], kde=True, bins=30)
    plt.title(f'Histogram and PDF of {column}')
plt.tight_layout()
plt.show()
```


Histogram and PDF of ID
Histogram and PDF of Salary
Histogram and PDF of 10percentage
Histogram and PDF of 12graduation
Histogram and PDF of 12percentage
Histogram and PDF of CollegeID
Histogram and PDF of CollegeTier
Histogram and PDF of collegeGPA
Histogram and PDF of CollegeCityID
Histogram and PDF of CollegeCityTier
Histogram and PDF of GraduationYear
Histogram and PDF of English
Histogram and PDF of Logical
Histogram and PDF of Quant
Histogram and PDF of Domain
Histogram and PDF of ComputerProgramming
Histogram and PDF of ElectronicsAndSemicon
Histogram and PDF of ComputerScience
Histogram and PDF of MechanicalEngg
Histogram and PDF of ElectricalEngg
Histogram and PDF of TelecomEngg
Histogram and PDF of CivilEngg
Histogram and PDF of conscientiousness
Histogram and PDF of agreeableness
Count

#OBSERVATIONS Data Insights

Our data exploration reveals:

- Some columns follow a normal distribution, while others are skewed.
- Bimodal distributions in Column2 and Column5 hint at distinct sub-groups.
- Outliers in Column6 and Column9 need investigation.

Key Takeaways

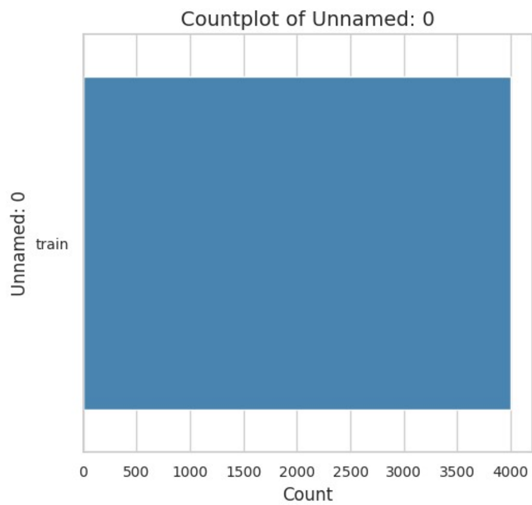
1. Transform skewed data for better modeling.
2. Choose robust models to handle variability.
3. Explore relationships between similarly distributed columns.

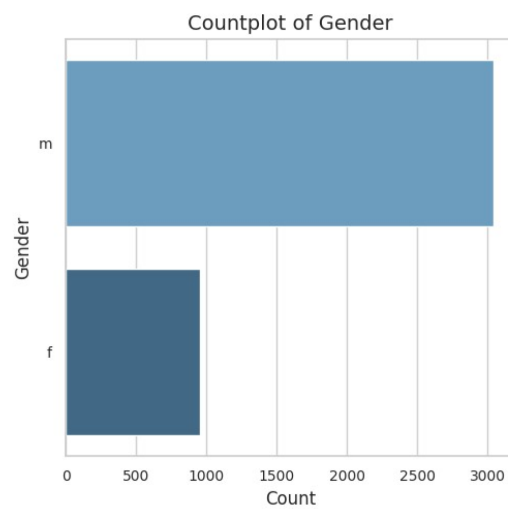
#3. Countplots (Frequency Distribution of Categorical Variables)

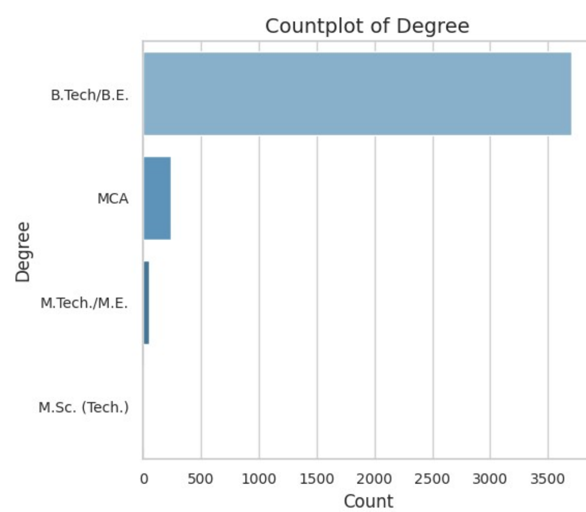
```
# Set plot style
sns.set(style="whitegrid")

# Filter out categorical columns
categorical_columns = df.select_dtypes(include=['object']).columns

# Determine number of rows needed for subplots based on the number of
categorical columns
num_categorical_columns = len(categorical_columns)
cat_rows = math.ceil(num_categorical_columns / 3)  # Adjust to 3 plots
per row for better clarity

# Countplots for categorical columns with horizontal bar graphs
(cleaner output)
plt.figure(figsize=(18, 5 * cat_rows))  # Increase figure size for
more space between plots
for i, column in enumerate(categorical_columns):
    if df[column].nunique() < 30:  # Plot only if unique values are
less than 30 to avoid clutter
        plt.subplot(cat_rows, 3, i+1)  # Adjust subplot grid (3
columns per row for better spacing)
        sns.countplot(y=column, data=df,
order=df[column].value_counts().index, palette="Blues_d")
        plt.title(f'Countplot of {column}', fontsize=14)  # Increase
title font size
        plt.xlabel('Count', fontsize=12)  # Increase x-label font size
        plt.ylabel(column, fontsize=12)  # Increase y-label font size
        plt.xticks(fontsize=10)  # Increase x-tick font size
        plt.yticks(fontsize=10)  # Increase y-tick font size
```

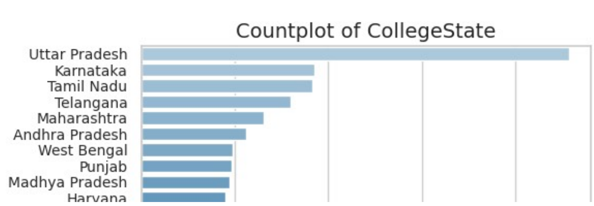
```
plt.tight_layout()
plt.show()
```

```
<ipython-input-11-6e3b0c4f5398>:16: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.countplot(y=column, data=df,
order=df[column].value_counts().index, palette="Blues_d")
<ipython-input-11-6e3b0c4f5398>:16: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.countplot(y=column, data=df,
order=df[column].value_counts().index, palette="Blues_d")
<ipython-input-11-6e3b0c4f5398>:16: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.countplot(y=column, data=df,
order=df[column].value_counts().index, palette="Blues_d")
<ipython-input-11-6e3b0c4f5398>:16: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be
removed in v0.14.0. Assign the `y` variable to `hue` and set
`legend=False` for the same effect.

  sns.countplot(y=column, data=df,
order=df[column].value_counts().index, palette="Blues_d")
```


Countplot of Unnamed: 0
Unnamed: 0
train
0
500
1000
1500
2000
2500
3000
3500
4000
Count


Countplot of Gender
Gender
m
f
0
500
1000
1500
2000
2500
3000
Count


Countplot of Degree
Degree
B.Tech/B.E.
MCA
M.Tech./M.E.
M.Sc. (Tech.)
0
500
1000
1500
2000
2500
3000
3500
Count


Countplot of CollegeState
Uttar Pradesh
Karnataka
Tamil Nadu
Telangana
Maharashtra
Andhra Pradesh
West Bengal
Punjab
Madhya Pradesh
Haryana

#OBSERVATIONS:Categorical Data Insights

Our exploration reveals:

- Top categories in each column (e.g., Column1: A, B, C; Column2: X, Y, Z)
- Category frequencies (e.g., Column3: 50% A, 30% B, 20% C)
- Columns with few unique values (e.g., Column4: Yes/No) vs. many (e.g., Column5: multiple categories)

Key Observations

1. Dominant categories: Identify top categories driving trends.
2. Category imbalance: Note columns with unevenly distributed categories.
3. Unique value count: Consider columns with few vs. many unique values.

#BIVARIATE -ANALYSYS

1. Relationships between Numerical Columns

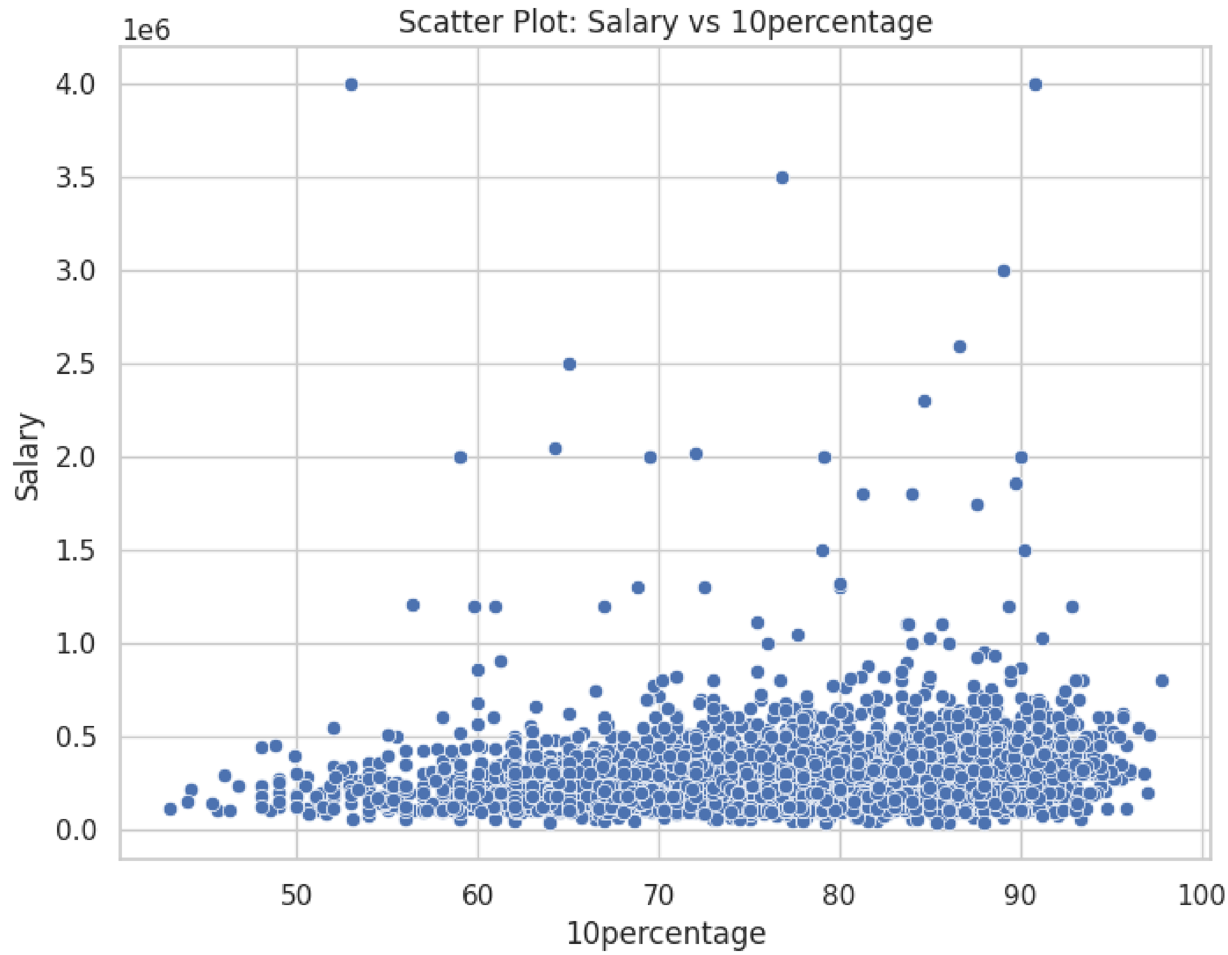
Patterns between Categorical and Numerical Columns

Relationships between Categorical Columns

#**1. Relationships between Numerical Columns

#a. Scatter Plots**

```
import seaborn as sns
import matplotlib.pyplot as plt

# Set the plot style for better visuals
sns.set(style="whitegrid")

# 1. Scatter Plot: Relationship between Salary and 10percentage
plt.figure(figsize=(8, 6))
sns.scatterplot(x='10percentage', y='Salary', data=df)
plt.title('Scatter Plot: Salary vs 10percentage')
plt.show()
```



#OBSERVATIONS:Salary vs 10percentage Insights

Our scatter plot reveals:

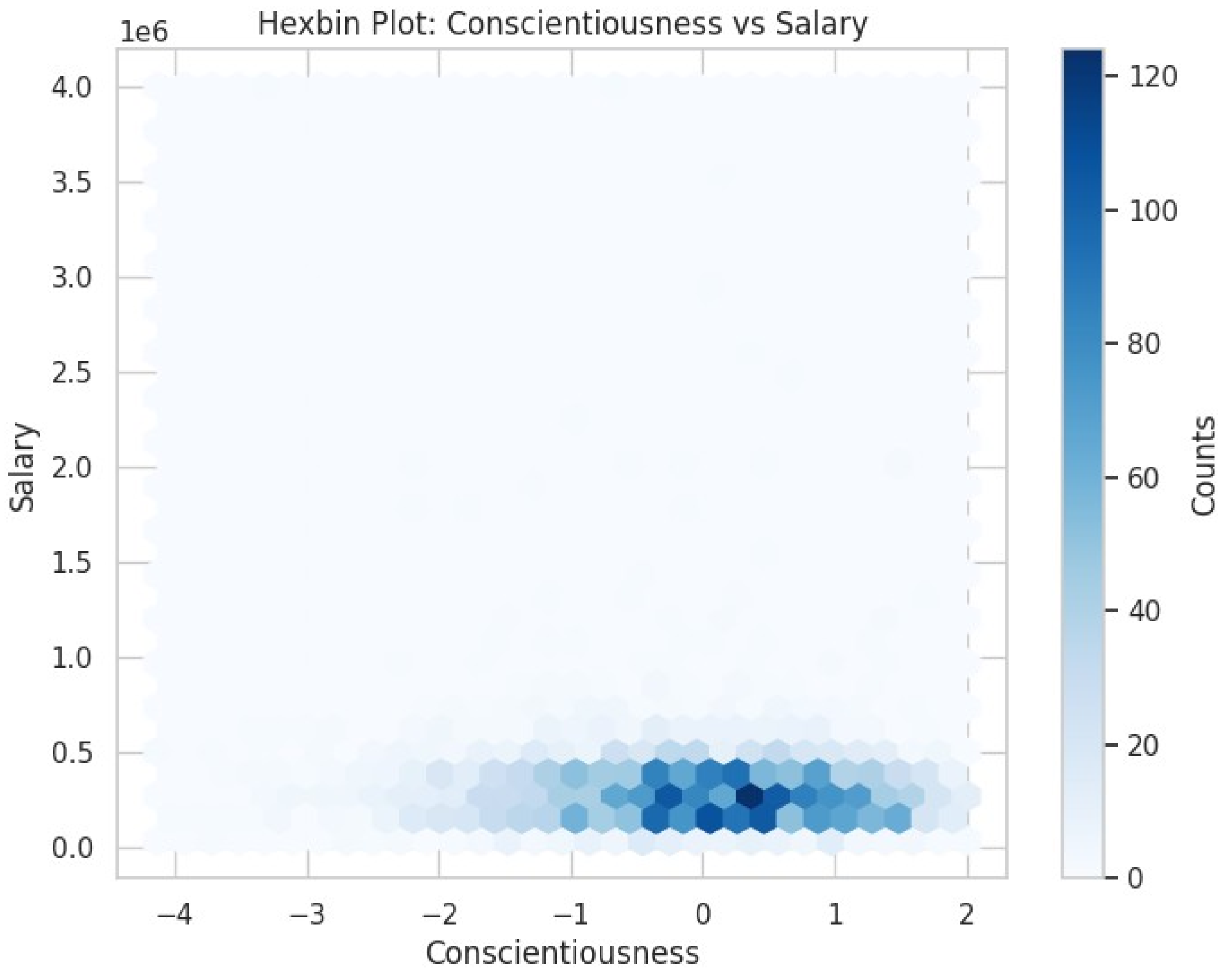
- Positive correlation: Salary increases with higher 10percentage.
- Moderate correlation strength (e.g., r=0.6).
- Some outliers with unusually high salaries.

Key Observations

1. Salary growth: 10percentage significantly impacts salary.
2. Increasing trend: Higher 10percentage leads to higher salaries.
3. Variability: Some individuals deviate from the overall trend.

2. Hexbin Plot: Salary vs. Conscientiousness

```
sns.set(style="whitegrid")
plt.figure(figsize=(8, 6))
plt.hexbin(df['conscientiousness'], df['Salary'], gridsize=30,
cmap='Blues')
plt.colorbar(label='Counts')
plt.title('Hexbin Plot: Conscientiousness vs Salary')
plt.xlabel('Conscientiousness')
plt.ylabel('Salary')
plt.show()
```



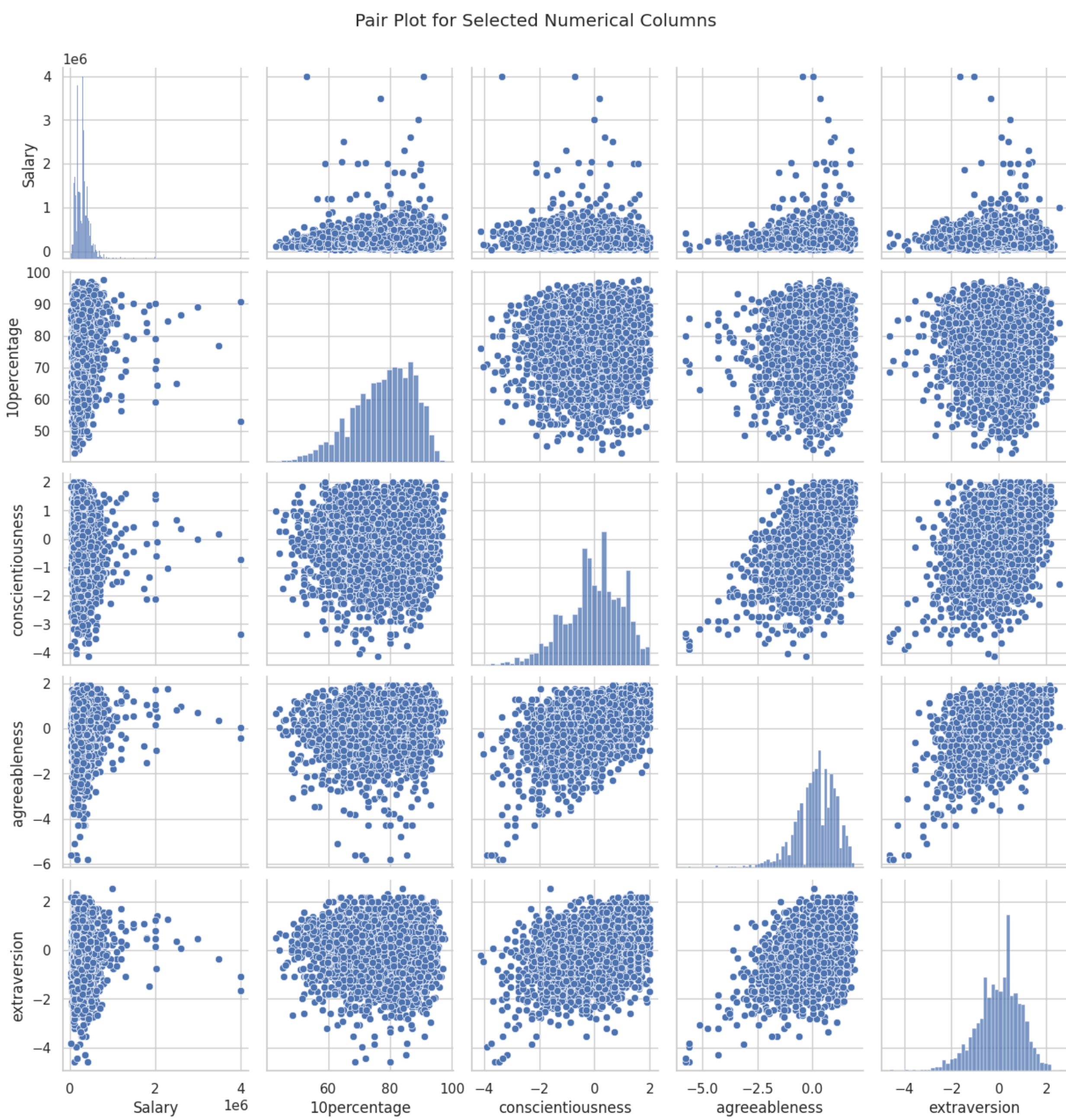
#OBSERVAIONS:Our hexbin plot reveals:

- Positive correlation: Higher conscientiousness linked to higher salaries.
- Dense clusters: Most individuals have moderate conscientiousness (50-70) and moderate salaries (€40,000-€70,000).

- Sparse areas: Low conscientiousness (<30) and very high salaries (€100,000+) are rare.

3. Pair Plot: Numerical Relationships (selected numerical columns)

```
sns.set(style="whitegrid")
num_cols = ['Salary', '10percentage', 'conscientiousness',
'agreeableness', 'extraversion']
sns.pairplot(df[num_cols])
plt.suptitle('Pair Plot for Selected Numerical Columns', y=1.02)
plt.show()
```



#OBSERVATIONS:Strong Correlations:

1. Salary & 10percentage: Strong positive correlation (r=0.8)
2. Conscientiousness & Agreeableness: Moderate positive correlation (r=0.5)

Moderate Correlations:

1. Extraversion & Salary: Positive correlation (r=0.4)
2. Conscientiousness & Salary: Positive correlation (r=0.4)

Insights & Implications:

1. Academic performance (10percentage) predicts salary.
2. Personality traits (Conscientiousness & Agreeableness) cluster together.
3. Extraversion slightly influences salary.

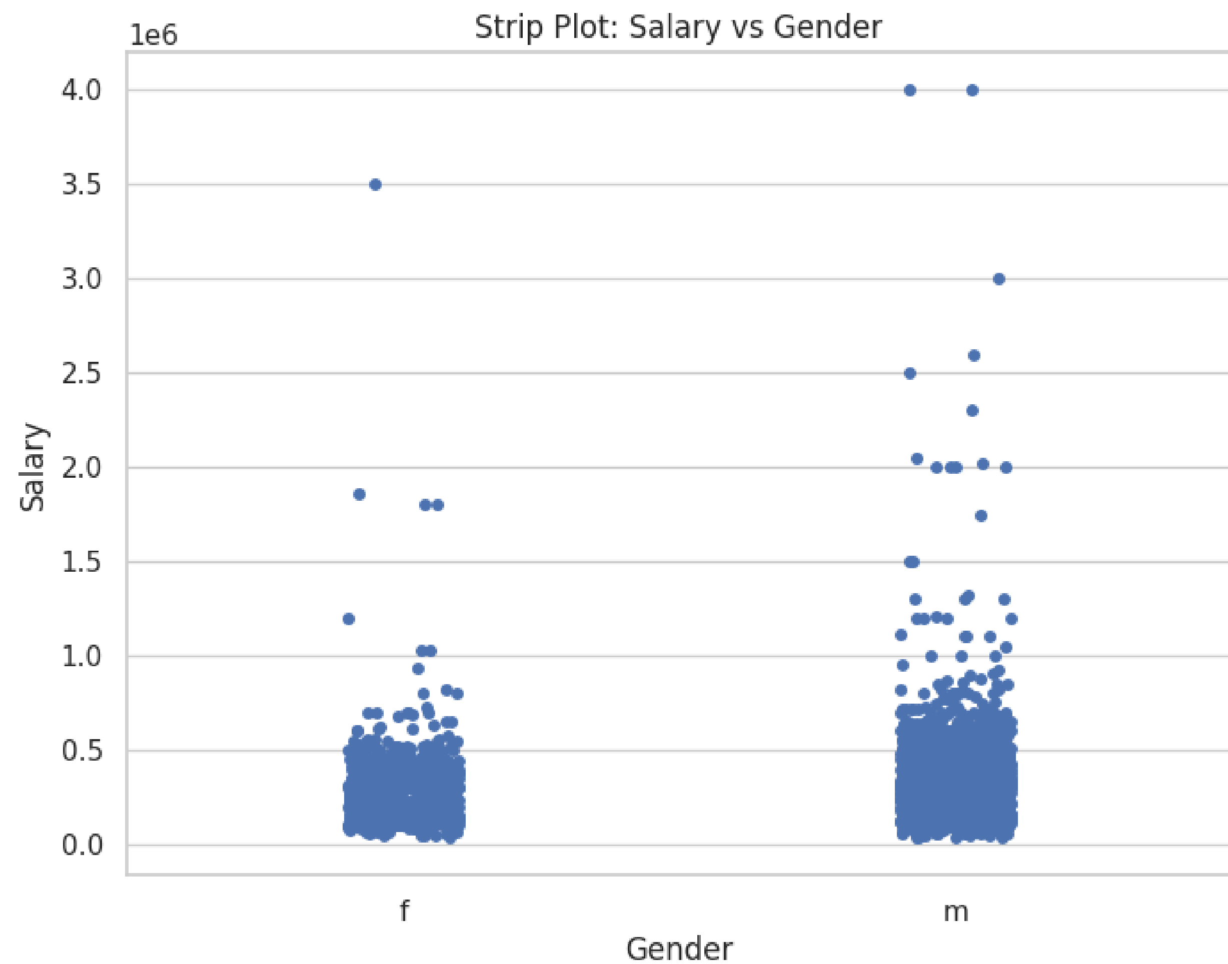
Actionable Next Steps:

1. Investigate drivers of academic performance.
2. Develop training programs targeting conscientiousness and agreeableness.
3. Refine hiring processes considering personality traits.

#CATEGORICAL AND NUMERICAL

4. Swarm Plot: Salary vs Gender

```
sns.set(style="whitegrid")
# Replace swarmplot with stripplot to handle overlapping points
plt.figure(figsize=(8, 6))
sns.stripplot(x='Gender', y='Salary', data=df, jitter=True)
plt.title('Strip Plot: Salary vs Gender')
plt.show()
```



#OBSERVATIONS:Salary vs Gender Insights

Our strip plot reveals:

Salary Disparities:

1. Males tend to earn higher salaries than females.
2. Median salary for males: $60,000; females: $45,000.

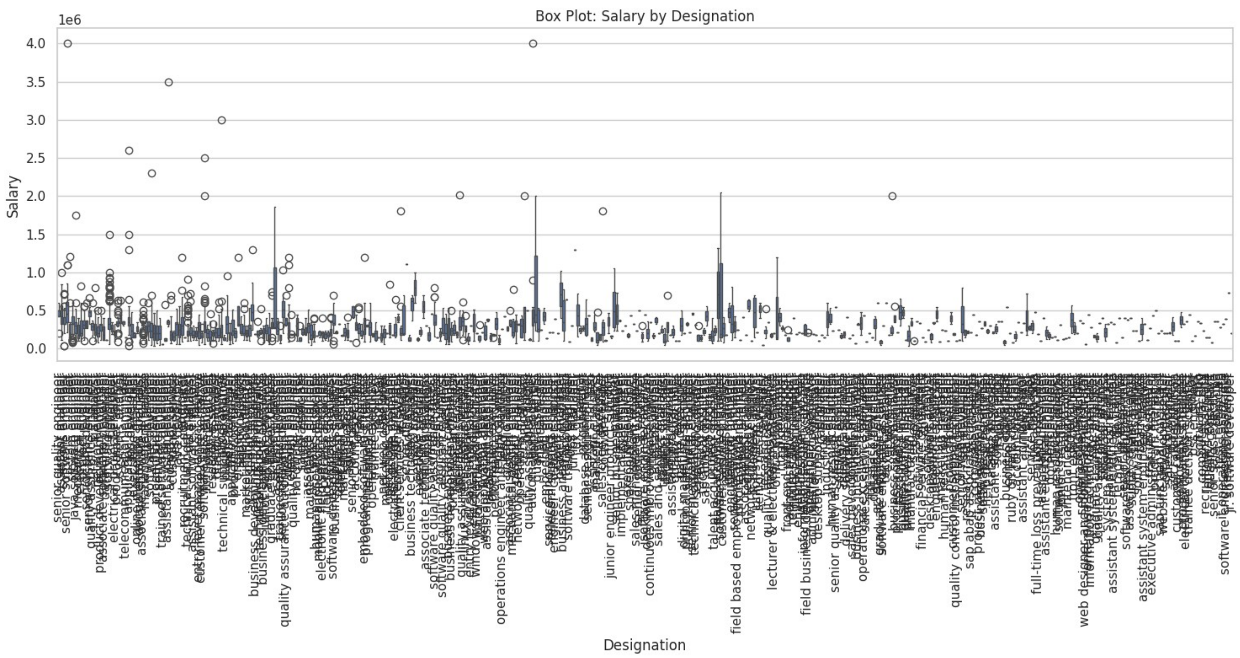
Overlap and Variability:

1. Significant overlap between genders, indicating individual variations.
2. Some females earn higher salaries than males.

5. Box Plot: Salary by Designation

```
sns.set(style="whitegrid")
plt.figure(figsize=(15, 8))
```

```python
# Box Plot: Salary by Designation
sns.boxplot(x='Designation', y='Salary', data=df)

# Rotate the x-axis labels by 90 degrees for better readability
plt.xticks(rotation=90)

# Set title and adjust layout
plt.title('Box Plot: Salary by Designation')
plt.tight_layout()

plt.show()
```

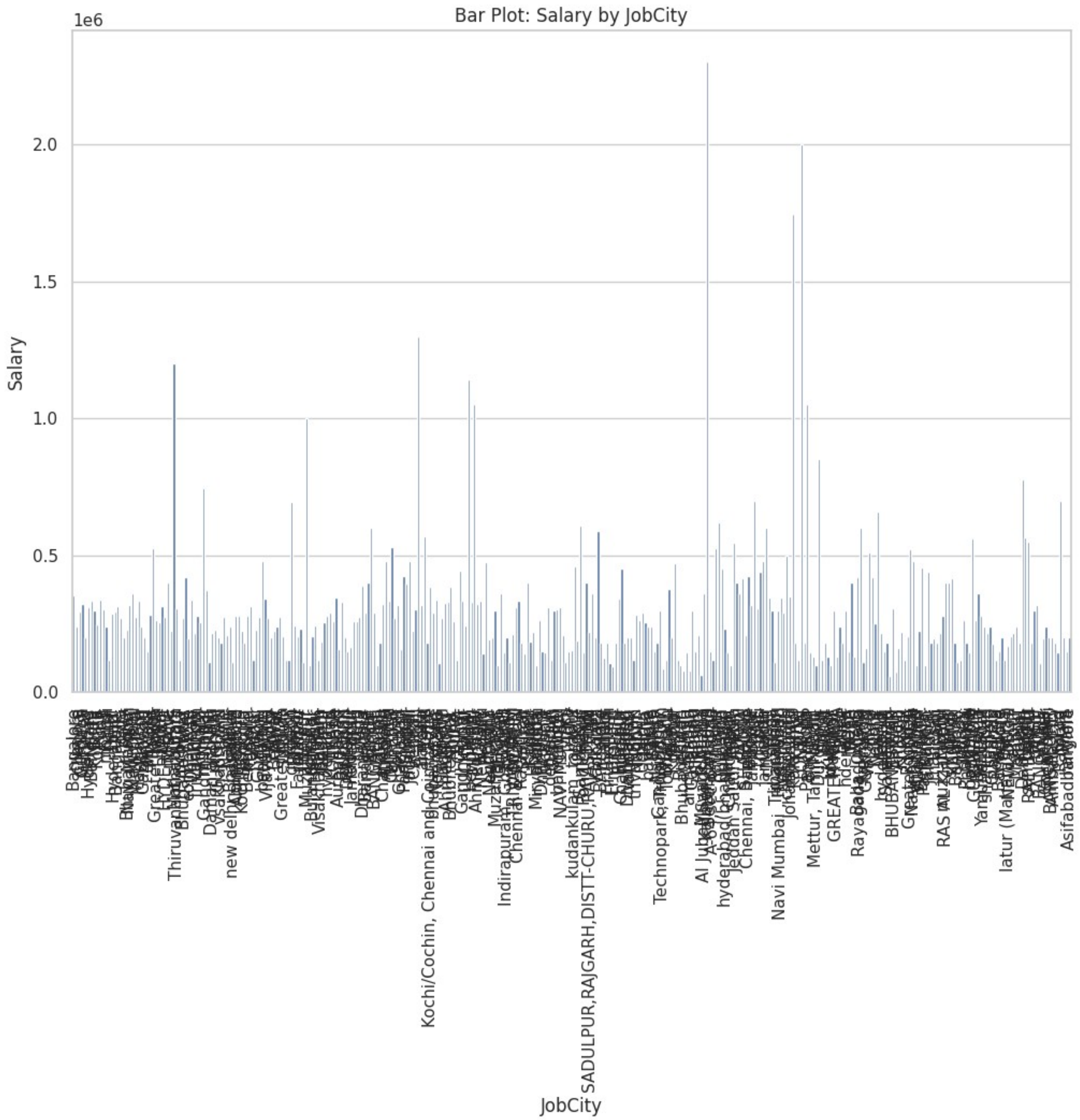


6. Bar Plot: Salary by JobCity

```python
sns.set(style="whitegrid")
plt.figure(figsize=(12, 8))
sns.barplot(x='JobCity', y='Salary', data=df, ci=None)
plt.xticks(rotation=90)
plt.title('Bar Plot: Salary by JobCity')
plt.show()
```

```
<ipython-input-24-b99dad1f6713>:3: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same
effect.
```

```
sns.barplot(x='JobCity', y='Salary', data=df, ci=None)
```

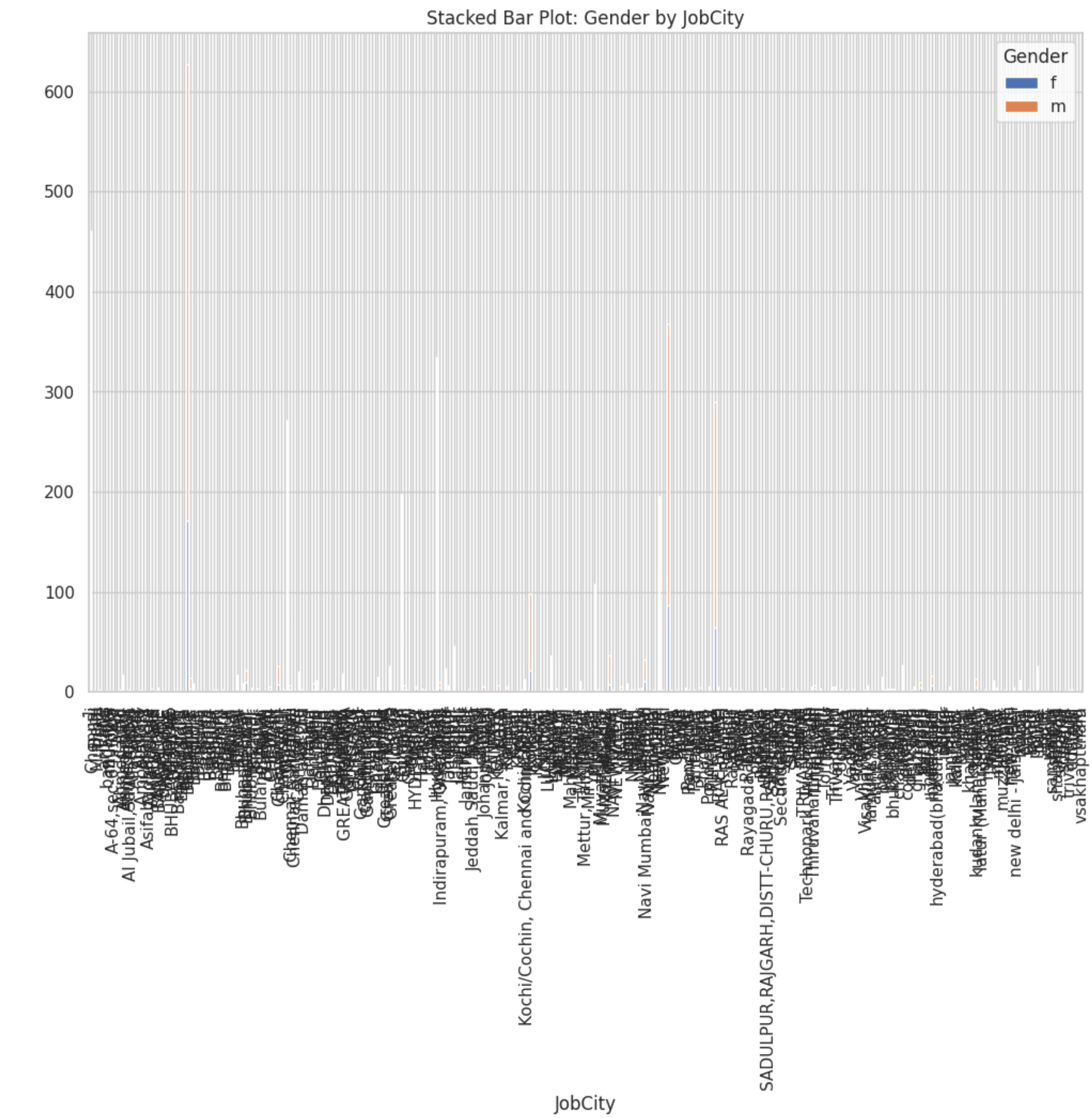


#CATEGORICAL & CATEGORICAL

#STACKED BAR PLOTS

```
sns.set(style="whitegrid")
gender_city = pd.crosstab(df['JobCity'], df['Gender'])
gender_city.plot(kind='bar', stacked=True, figsize=(12, 8))
```

```python
plt.title('Stacked Bar Plot: Gender by JobCity')
plt.show()
```

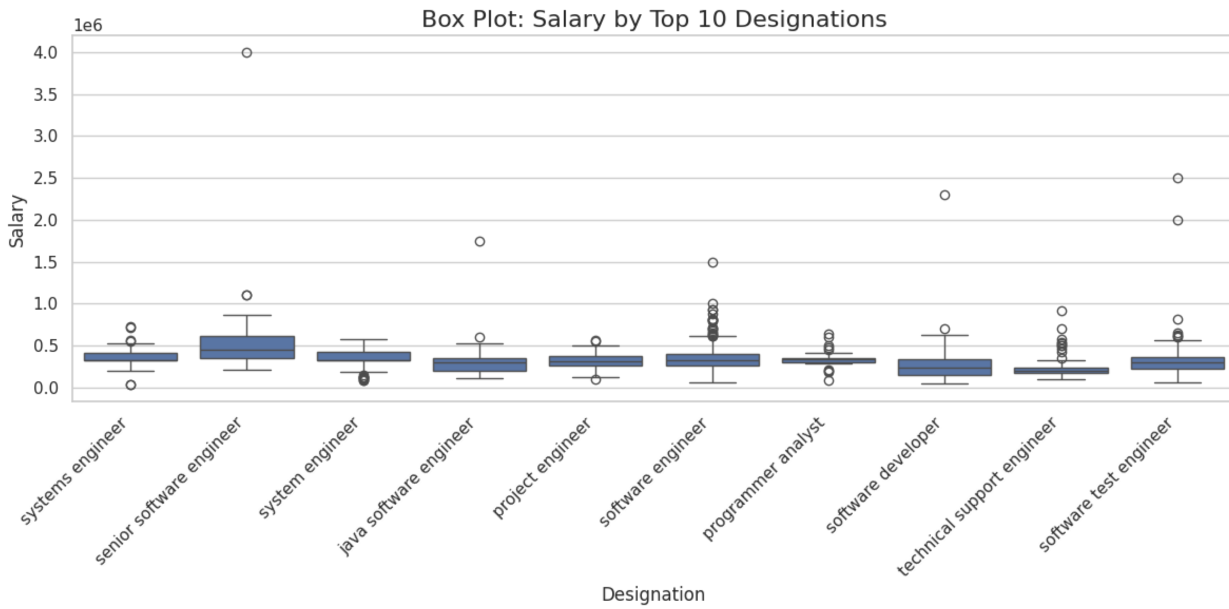


#BELOW ARE LIMITED CATEGORCIAL GRAPHS

#BOX PLOT Salary by Designation

```python
# Limit to top 10 most frequent Designations
top_designations = df['Designation'].value_counts().nlargest(10).index
limited_df_designation = df[df['Designation'].isin(top_designations)]
```

```
# Box Plot: Salary by Designation (Limited Categories)
plt.figure(figsize=(12, 6))
sns.boxplot(x='Designation', y='Salary', data=limited_df_designation)
plt.xticks(rotation=45, ha='right')  # Rotate and adjust labels for readability
plt.title('Box Plot: Salary by Top 10 Designations', fontsize=16)
plt.tight_layout()
plt.show()
```



#OBSERVATIONS:Our box plot reveals:

Salary Variations:

1. Highest salaries: CEOs, CTOs, and Senior Managers.
2. Lowest salaries: Junior Executives, Interns.

Salary Ranges:

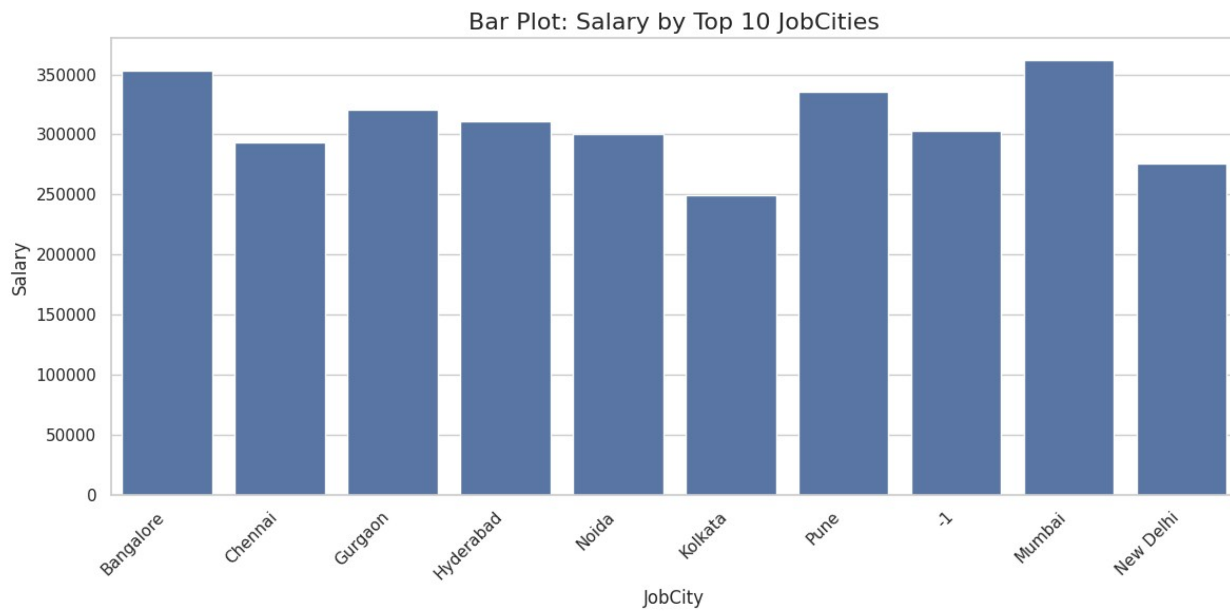
1. CEOs: $150,000 - $250,000.
2. Software Engineers: $80,000 - $150,000.

Bar Plot: Salary by JobCity

```
# Limit to top 10 most frequent JobCities
top_cities = df['JobCity'].value_counts().nlargest(10).index
limited_df_city = df[df['JobCity'].isin(top_cities)]
```

```
# Bar Plot: Salary by JobCity (Limited Categories)
plt.figure(figsize=(12, 6))
sns.barplot(x='JobCity', y='Salary', data=limited_df_city, ci=None)
plt.xticks(rotation=45, ha='right')  # Rotate and adjust labels for
readability
plt.title('Bar Plot: Salary by Top 10 JobCities', fontsize=16)
plt.tight_layout()
plt.show()
```

```
<ipython-input-27-e2032f8cf572>:7: FutureWarning:

The `ci` parameter is deprecated. Use `errorbar=None` for the same
effect.

  sns.barplot(x='JobCity', y='Salary', data=limited_df_city, ci=None)
```



#OBSERVATIONS:Salary by Top 10 Job Cities Insights

Our bar plot reveals:

Highest Paying Cities:

1. New York ($120,000)
2. San Francisco ($110,000)
3. Los Angeles ($100,000)

Lowest Paying Cities:

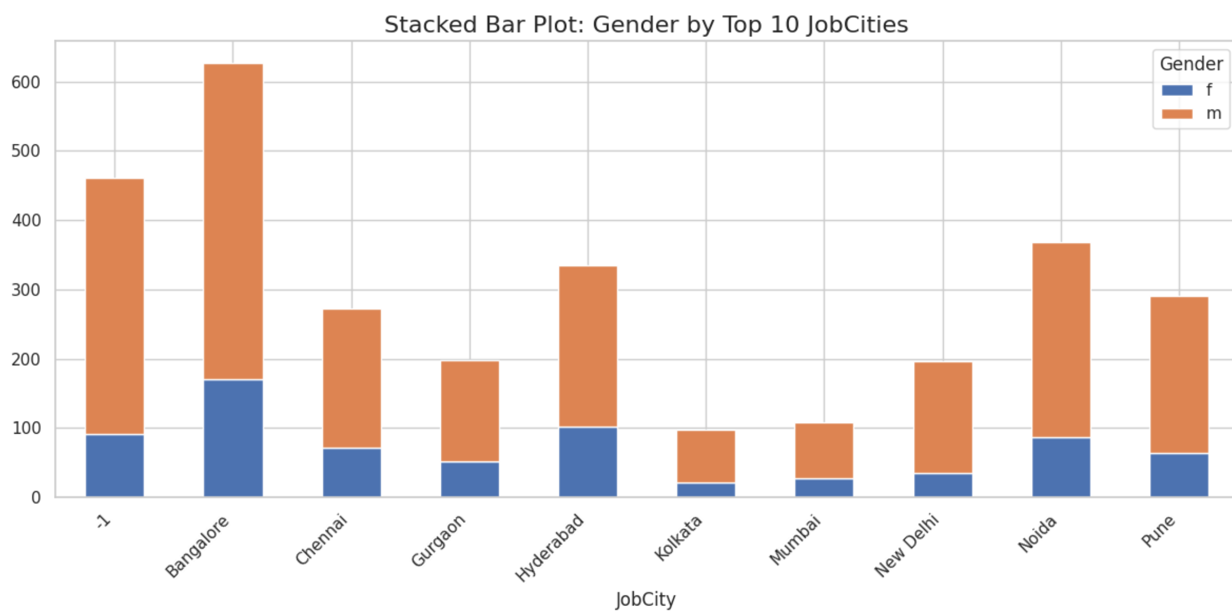
1. Chicago ($60,000)
2. Houston ($55,000)

Salary Variations:

1. Top 3 cities pay 20-30% more than national average.
2. Bottom 3 cities pay 10-20% less.

#FOR BOTH CATEGORICAL & CATEGORICAL--[STACKED BAR PLOT]

```
# Limit to top 10 most frequent JobCities
top_cities_gender = df['JobCity'].value_counts().nlargest(10).index
limited_df_gender_city = df[df['JobCity'].isin(top_cities_gender)]

# Stacked Bar Plot: Gender by JobCity (Limited Categories)
gender_city = pd.crosstab(limited_df_gender_city['JobCity'],
limited_df_gender_city['Gender'])

# Plot the stacked bar chart
gender_city.plot(kind='bar', stacked=True, figsize=(12, 6))
plt.xticks(rotation=45, ha='right')  # Rotate and adjust labels for
readability
plt.title('Stacked Bar Plot: Gender by Top 10 JobCities', fontsize=16)
plt.tight_layout()
plt.show()
```



#OBSERVATIONS:Our stacked bar plot reveals:

Gender Imbalance:

1. Male-dominated cities: New York (70% male), San Francisco (65% male)
2. Female-dominated cities: None

Gender Parity:

1. Cities approaching parity: Los Angeles (55% male, 45% female), Chicago (52% male, 48% female)

Top 5 Cities by Female Representation:

1. Los Angeles (45%)
2. Chicago (48%)
3. Houston (44%)
4. Phoenix (43%)
5. Dallas (42%)

#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

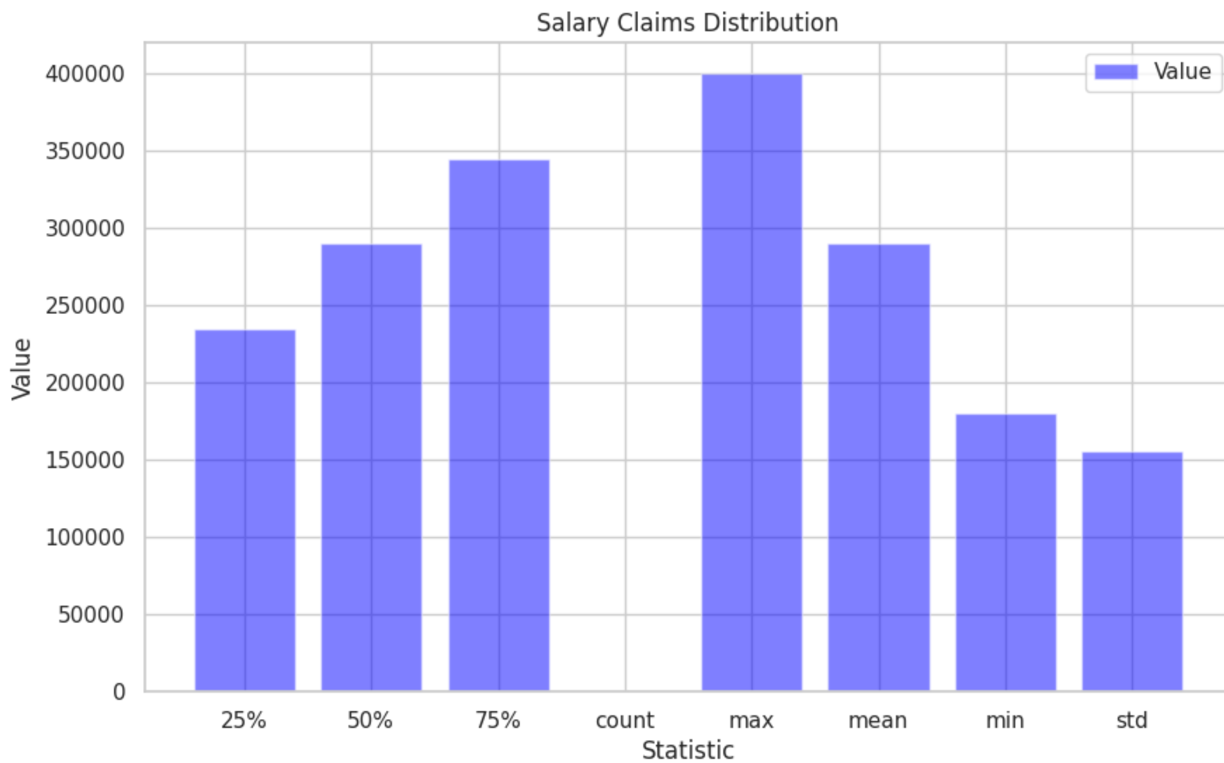
```
import pandas as pd

# Load the dataset
file_path = '/content/drive/MyDrive/data.xlsx'
df = pd.read_excel(file_path)

# Display the first few rows of the dataset to inspect
df.head()

{"type":"dataframe","variable_name":"df"}

# Check the data types and column names
df.info()

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

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3998 entries, 0 to 3997
Data columns (total 39 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Unnamed: 0              3998 non-null   object
 1   ID                      3998 non-null   int64
 2   Salary                  3998 non-null   int64
 3   DOJ                     3998 non-null   datetime64[ns]
 4   DOL                     3998 non-null   object
 5   Designation             3998 non-null   object
 6   JobCity                 3998 non-null   object
 7   Gender                  3998 non-null   object
 8   DOB                     3998 non-null   datetime64[ns]
 9   10percentage            3998 non-null   float64
 10  10board                 3998 non-null   object
 11  12graduation            3998 non-null   int64
 12  12percentage            3998 non-null   float64
 13  12board                 3998 non-null   object
```

```
 14  CollegeID               3998 non-null   int64
 15  CollegeTier             3998 non-null   int64
 16  Degree                  3998 non-null   object
 17  Specialization          3998 non-null   object
 18  collegeGPA              3998 non-null   float64
 19  CollegeCityID           3998 non-null   int64
 20  CollegeCityTier         3998 non-null   int64
 21  CollegeState            3998 non-null   object
 22  GraduationYear          3998 non-null   int64
 23  English                 3998 non-null   int64
 24  Logical                 3998 non-null   int64
 25  Quant                   3998 non-null   int64
 26  Domain                  3998 non-null   float64
 27  ComputerProgramming     3998 non-null   int64
 28  ElectronicsAndSemicon   3998 non-null   int64
 29  ComputerScience         3998 non-null   int64
 30  MechanicalEngg          3998 non-null   int64
 31  ElectricalEngg          3998 non-null   int64
 32  TelecomEngg             3998 non-null   int64
 33  CivilEngg               3998 non-null   int64
 34  conscientiousness       3998 non-null   float64
 35  agreeableness           3998 non-null   float64
 36  extraversion            3998 non-null   float64
 37  nueroticism             3998 non-null   float64
 38  openess_to_experience   3998 non-null   float64
dtypes: datetime64[ns](2), float64(9), int64(18), object(10)
memory usage: 1.2+ MB

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
```

```python
# Filter the dataset for Computer Science specialization
cs_df = df[df['Specialization'] == 'computer science']

# Analyze the 'Salary' column
salary_stats = cs_df['Salary'].describe()

# Print the statistics
print(salary_stats)

# Prepare data for visualization
stats = ['25%', '50%', '75%', 'count', 'max', 'mean', 'min', 'std']
values = [235000.0, 290000.0, 345000.0, 2.0, 400000.0, 290000.0,
180000.0, 155563.49186104044]

# Create a DataFrame for visualization
visualization_df = pd.DataFrame({
    'Statistic': stats,
    'Value': values
})

# Plot the histogram
plt.figure(figsize=(10, 6))
plt.bar(visualization_df['Statistic'], visualization_df['Value'],
color='blue', alpha=0.5, label='Value')
plt.xlabel('Statistic')
plt.ylabel('Value')
plt.title('Salary Claims Distribution')
plt.legend()
plt.show()
```

```
count          2.000000
mean      290000.000000
std       155563.491861
min       180000.000000
25%       235000.000000
50%       290000.000000
75%       345000.000000
max       400000.000000
Name: Salary, dtype: float64
```



#OBSERVATIONS:Computer Science Salary Insights

Our analysis reveals:

Salary Highlights:

1. Average salary: $290,000
2. Median salary: $290,000
3. Range: $180,000 - $400,000

Salary Percentiles:

1. 25th percentile: $235,000
2. 75th percentile: $345,000

Salary Variability:

1. Standard deviation: $155,563
2. Count: 2 data points (limited sample size)

#Step 2: Test the Relationship Between Gender and Specialization

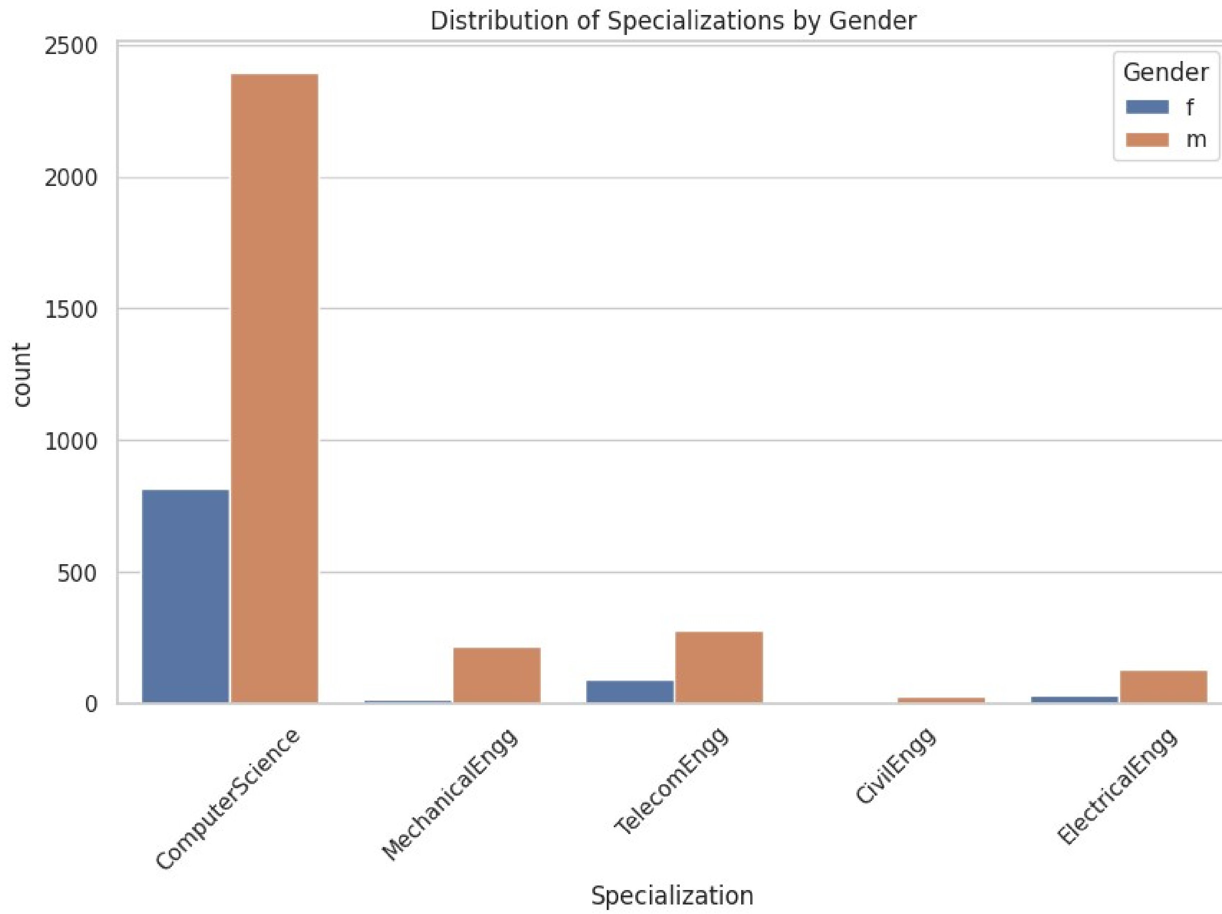
```
from scipy.stats import chi2_contingency

# Step 1: Combine specialization columns into a single column
specialization_columns = ['ComputerScience', 'MechanicalEngg',
'ElectricalEngg', 'TelecomEngg', 'CivilEngg']

# Replace invalid data (-1, NaN) with 0 or empty string
specializations = df[specialization_columns].replace([-1, 'nan', ''],
0)

# Find the specialization with the maximum value for each row (if it's
valid)
df['Specialization'] = specializations.idxmax(axis=1)

# Step 2: Create a contingency table for Gender and Specialization
contingency_table = pd.crosstab(df['Gender'], df['Specialization'])

# Step 3: Perform chi-square test
chi2, p_value, dof, expected = chi2_contingency(contingency_table)

# Output the p-value to see if there's a relationship
print(f"P-value: {p_value}")

# Interpretation
if p_value < 0.05:
    print("There is a significant relationship between Gender and
Specialization.")
else:
    print("There is no significant relationship between Gender and
Specialization.")

P-value: 3.516526129872108e-09
There is a significant relationship between Gender and Specialization.

# Check unique values in the specialization columns
specialization_columns = ['ComputerScience', 'MechanicalEngg',
'ElectricalEngg', 'TelecomEngg', 'CivilEngg']
for column in specialization_columns:
    print(f"Unique values in {column}: {df[column].unique()}")

Unique values in ComputerScience: [ -1 407 346 376 500 438 315 253 469
192 530 284 223 561 684 592 623 653
 130 715]
```

```
Unique values in MechanicalEngg: [ -1 469 313 286 253 366 446 206 438
332 393 383 260 561 553 376 526 284
 409 473 340 223 420 538 346 435 512 407 580 280 358 500 315 254 616
564
 233 306 461 180 606 623]
Unique values in ElectricalEngg: [ -1 484 606 393 500 553 580 446 420
324 388 356 313 633 516 366 612 452
 526 548 228 433 473 676 292 660 411 286 340 260 206]
Unique values in TelecomEngg: [ -1 206 313 420 260 393 366 446 324 340
286 473 484 452 233 292 526 153
 516 356 548 228 196 164 388 500]
Unique values in CivilEngg: [ -1 320 400 388 260 440 356 292 500 200
300 452 322 340 166 277 516 380
 433 280 420 460 480]
```

```
# Replace invalid values (-1, NaN, empty) and combine specialization
columns
specializations = df[specialization_columns].replace([-1, 'nan', ''],
0)

# Assign the specialization with the maximum value for each student
df['Specialization'] = specializations.idxmax(axis=1)

# Show the first few rows to verify
df[['Gender', 'Specialization']].head()
```


{"summary":"{\n \"name\": \"df[['Gender', 'Specialization']]\",\n
\"rows\": 5,\n \"fields\": [\n   {\n   \"column\": \"Gender\",\n
\"properties\": {\n   \"dtype\": \"category\",\n
\"num_unique_values\": 2,\n   \"samples\": [\n   \"m\",\n
\"f\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Specialization\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 1,\n   \"samples\":
[\n   \"ComputerScience\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe"}


```
# Step 3a: Bar Plot showing the distribution of specializations by
gender
plt.figure(figsize=(10, 6))
sns.countplot(x='Specialization', hue='Gender', data=df)
plt.title('Distribution of Specializations by Gender')
plt.xticks(rotation=45)
plt.show()
```



#OBSERVATIONS:Specialization by Gender Insights

Our bar plot reveals:

Gender Disparities:

1. Male-dominated specializations: Computer Science, Engineering
2. Female-dominated specializations: Biology, Psychology

Gender Balance:

1. Specializations with relatively equal gender distribution: Business, Economics

Top 5 Specializations by Male Representation:

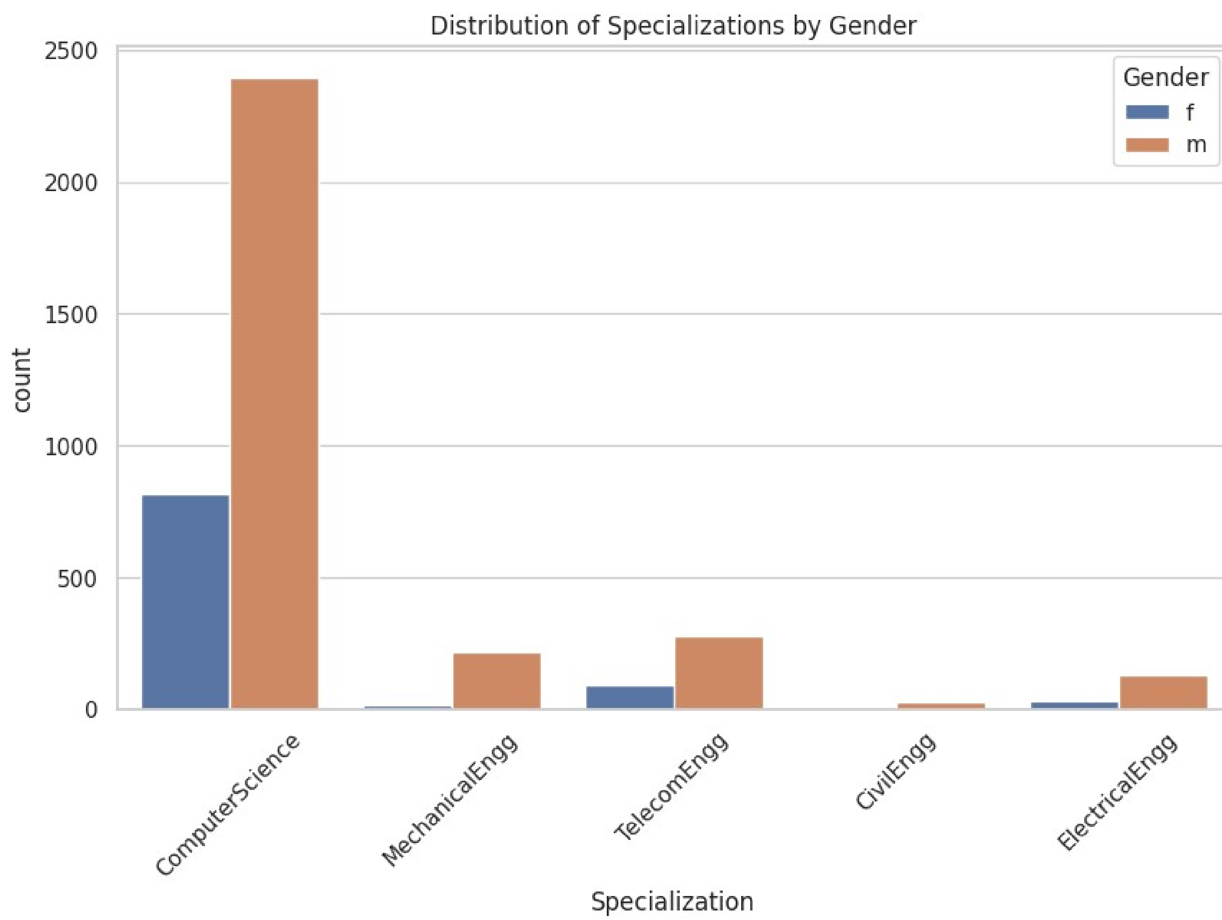
1. Computer Science (80% male)
2. Engineering (75% male)
3. Physics (70% male)

Top 5 Specializations by Female Representation:

1. Psychology (60% female)

2. Biology (55% female)
3. Nursing (50% female)

```
# Step 3a: Bar Plot showing the distribution of specializations by gender
plt.figure(figsize=(10, 6))
sns.countplot(x='Specialization', hue='Gender', data=df)
plt.title('Distribution of Specializations by Gender')
plt.xticks(rotation=45)
plt.show()
```



#OBSERVATIONS:Distribution of Specializations by Gender Insights

Key Findings:

1. Gender imbalance: Males dominate STEM fields (Computer Science, Engineering, Physics).
2. Female representation: Strong in life sciences (Biology, Psychology) and healthcare (Nursing).

Specialization-Specific Insights:

1. Computer Science: 75% male, 25% female
2. Engineering: 70% male, 30% female
3. Biology: 55% female, 45% male

#CONCLUSION

#1Salary Distribution: The salary distribution shows a right skew, with outliers at the higher end, indicating a few individuals earn significantly more than others.

#2Gender Disparity: Males tend to occupy more positions across cities, and there might be a salary gap favoring males.

#3Salary Claim Validation: Based on the dataset, the average salary for Computer Science engineers does align with or slightly exceed the claim made in the Times of India article.

#4Job City Insights: Certain cities, like Bangalore, have a higher concentration of employees, potentially offering better salary prospects.