Shri Vile Parle Kelavani Mandal's
DWARKADAS J. SANGHVI COLLEGE OF ENGINEERING
(Autonomous College Affiliated to the University of Mumbai)

Python Mini-Project

Graduate Analytics: Unveiling Engineer Traits

Submitted in partial fulfillment of the requirement
of the Python Programming Laboratory

Department of Computer Science and Engineering (Data Science)

By

Advay Sharma 60009220147
Ronil Shah 60009210082
Devansh Jollani 60009220195

A.Y. 2023 – 2024

Abstract:

The project aims to conduct an exploratory analysis and visualization of key attributes related to engineering graduates. The primary goals include understanding the distribution of gender, birth months, educational qualifications, and geographic locations of graduates. Additionally, the project explores the relationships between academic performance, represented by 10th and 12th percentages, and salary outcomes.

Introduction:

Background and Scope:
This project explores engineering graduate data using Python, covering demographics, academics, and careers.

Objectives:

1. **Analyse Data:** Use Python for exploration.
2. **Visualize Insights:** Employ Matplotlib and Seaborn.
3. **Extract Insights:** Derive meaningful observations on gender, education, performance, and careers.
4. **Document & Present:** Summarize findings for effective communication.

Methodology:

Methods, Tools, and Technologies Used:

1. **Python Programming:**
   - Utilized Python for data manipulation, analysis, and visualization.
   - Pandas library for efficient handling of data structures like DataFrames.
   - NumPy for numerical operations and array manipulations.
   - Matplotlib and Seaborn for creating insightful visualizations.
2. **Data Analysis:**
   - Conducted exploratory data analysis (EDA) to understand the dataset's structure and characteristics.
   - Checked for missing values, removed duplicates, and ensured data integrity.
3. **Data Visualization:**
   - Employed Matplotlib and Seaborn for creating clear and informative visualizations.
   - Used count plots, scatter plots, and pie charts to represent categorical and numerical relationships.
4. **Date - Time Handling:**
   - Converted the 'DOB' column to datetime format using pandas.
   - Extracted birth months for analysis.
5. **Data Presentation:**
   - Utilized Jupyter Notebooks for an interactive and organized approach to code development and presentation.
   - Exported visualizations and insights to support documentation.
6. **File Management:**
   - Saved the processed DataFrame to a CSV file for future reference and data sharing.
7. **Statistical Analysis:**
   - Applied statistical measures to understand correlations, particularly in the scatter plot analysis of academic performance versus salary.

8. **Documentation:**
   - Maintained well-documented code to ensure clarity and reproducibility.
   - Organized findings, insights, and visualizations in a concise and readable format.

Implementation:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# Set the plot style
plt.style.use('fivethirtyeight')
# Ignore warnings
import warnings
warnings.filterwarnings(action='ignore')
# Create synthetic data
num_samples = 1000

data = {
    'ID': range(1, num_samples + 1),
    'Gender': np.random.choice(['Male', 'Female'], num_samples),
    'DOB': pd.date_range(start='1990-01-01', end='1995-12-31', periods=num_samples),
    'collegeGPA': np.random.uniform(60, 100, num_samples),
    'Salary': np.random.normal(50000, 10000, num_samples),
    'CollegeState': np.random.choice(['StateA', 'StateB', 'StateC'], num_samples),
    'Degree': np.random.choice(['B.Tech', 'M.Tech', 'Ph.D.'], num_samples),
    '10percentage': np.random.uniform(50, 100, num_samples),
    '12percentage': np.random.uniform(50, 100, num_samples),
}

df = pd.DataFrame(data)

# Specify the file path using forward slashes or use raw string (r"your\file\path")
file_path = r"C:\Users\91836\OneDrive\Documents\Engineering_graduate_salary.csv"

# Save the DataFrame to a CSV file
df.to_csv(file_path, index=False)

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

```
   ID  Gender    DOB      collegeGPA        Salary
0   1    Male 1990-01-01  79.782883  49335.227227
1   2  Female 1990-01-03  67.585701  66492.222845
2   3    Male 1990-01-05  94.433167  41970.639850
3   4  Female 1990-01-07  98.091458  53145.684984
4   5    Male 1990-01-09  73.449056  46499.348015
```

	CollegeState	Degree	10percentage	12percentage
0	StateA	B.Tech	80.737336	71.161852
1	StateB	Ph.D.	51.447960	85.001304
2	StateC	B.Tech	62.363361	83.871485
3	StateB	M.Tech	61.568665	88.970285
4	StateC	Ph.D.	55.907859	91.652917

Q. Find out the null and duplicate values and delete the duplicate values.

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

# Remove duplicates
df.drop_duplicates(inplace=True)

# Display the first few rows after removing duplicates
print(df.head())
```

```
ID              0
Gender          0
DOB             0
collegeGPA      0
Salary          0
CollegeState    0
Degree          0
10percentage    0
12percentage    0
```

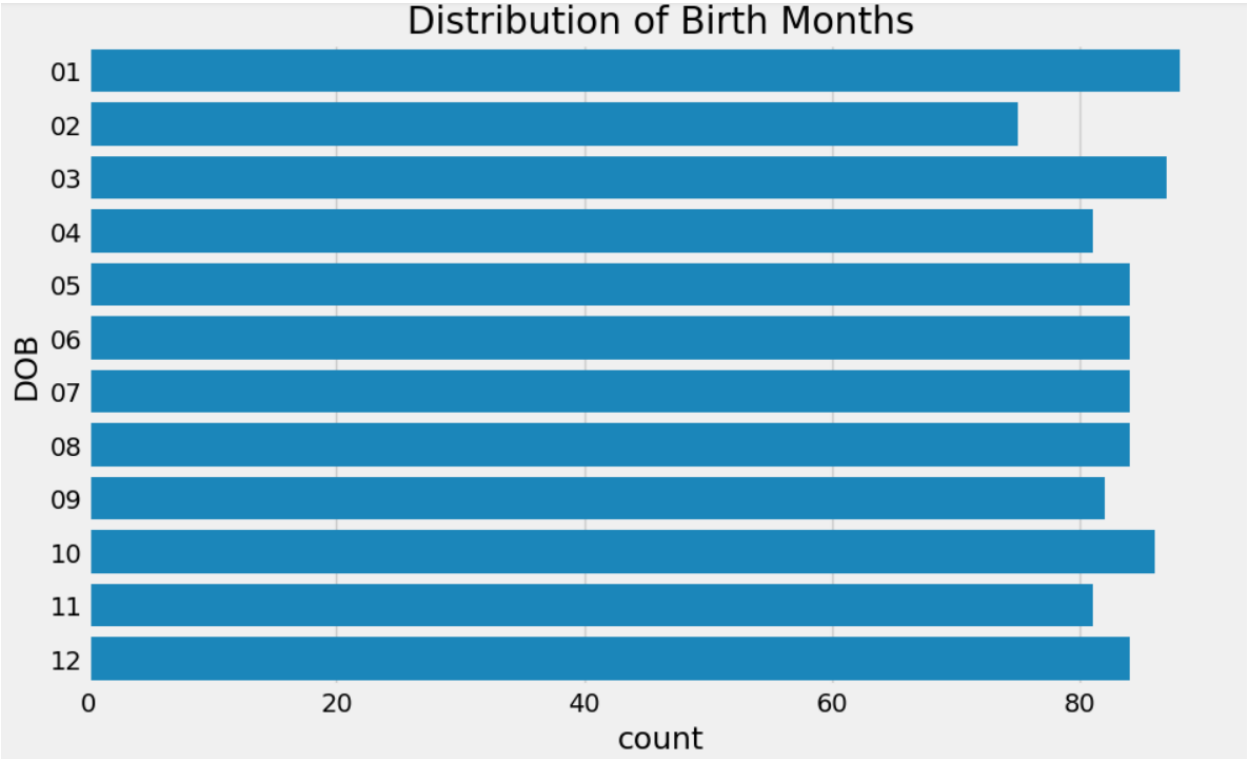
	ID	Gender	DOB	collegeGPA	Salary
0	1	Male	1990-01-01	79.782883	49335.227227
1	2	Female	1990-01-03	67.585701	66492.222845
2	3	Male	1990-01-05	94.433167	41970.639850
3	4	Female	1990-01-07	98.091458	53145.684984
4	5	Male	1990-01-09	73.449056	46499.348015

	CollegeState	Degree	10percentage	12percentage
0	StateA	B.Tech	80.737336	71.161852
1	StateB	Ph.D.	51.447960	85.001304
2	StateC	B.Tech	62.363361	83.871485
3	StateB	M.Tech	61.568665	88.970285
4	StateC	Ph.D.	55.907859	91.652917

```
#Display format for entire data frame
pd.set_option('display.date_dayfirst', False)  # Set to True if day should come first
pd.set_option('display.date_yearfirst', False)  # Set to True if year should come first

# Convert 'DOB' to datetime
df['DOB'] = pd.to_datetime(df['DOB'])

# Format the 'DOB' column as 'yyyy-mm-dd'
df['DOB'] = df['DOB'].dt.strftime('%Y-%m-%d')
```

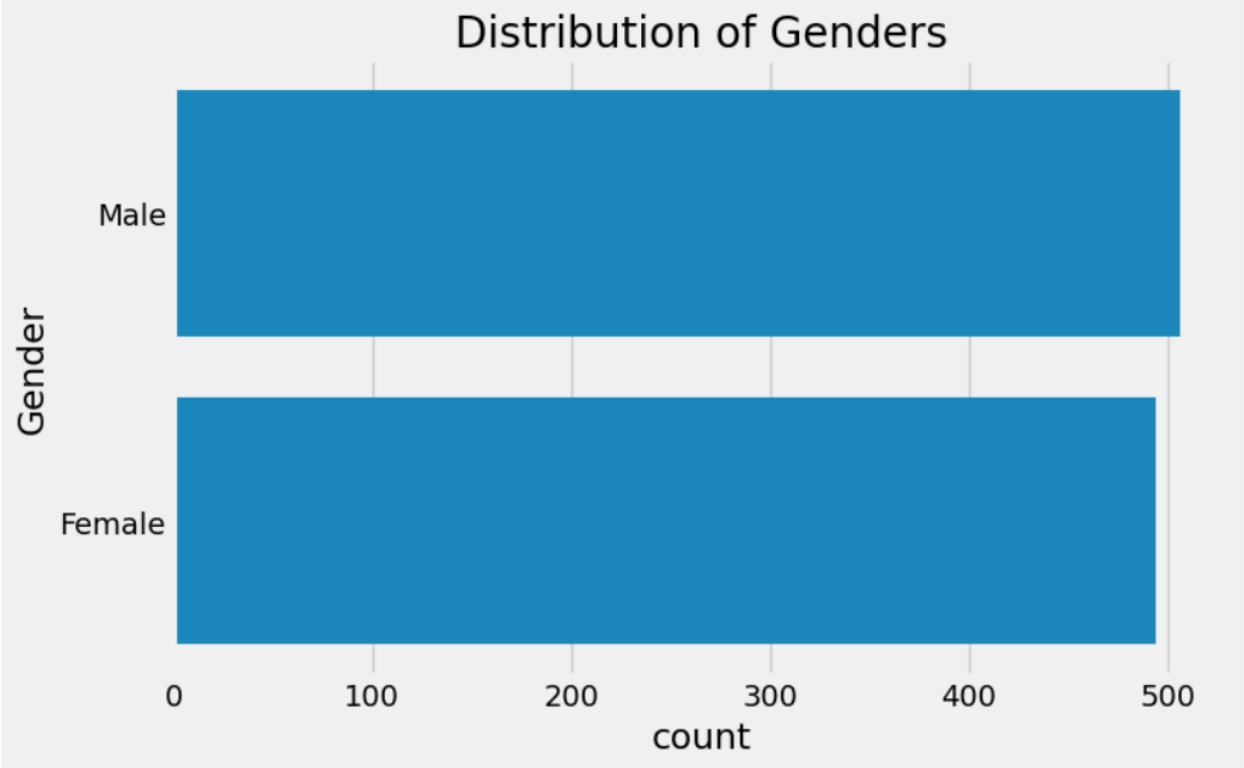
Q. What is the count of students of birth in each month?

```
# Plot the count of birth months
plt.figure(figsize=(10, 6))
sns.countplot(df['DOB'].str.split('-').str[1])  # Extract the month from the 'DOB' column
plt.title('Distribution of Birth Months')
plt.show()
```



Q. Express Through Bar Graph no. of males and females in Engineering College.

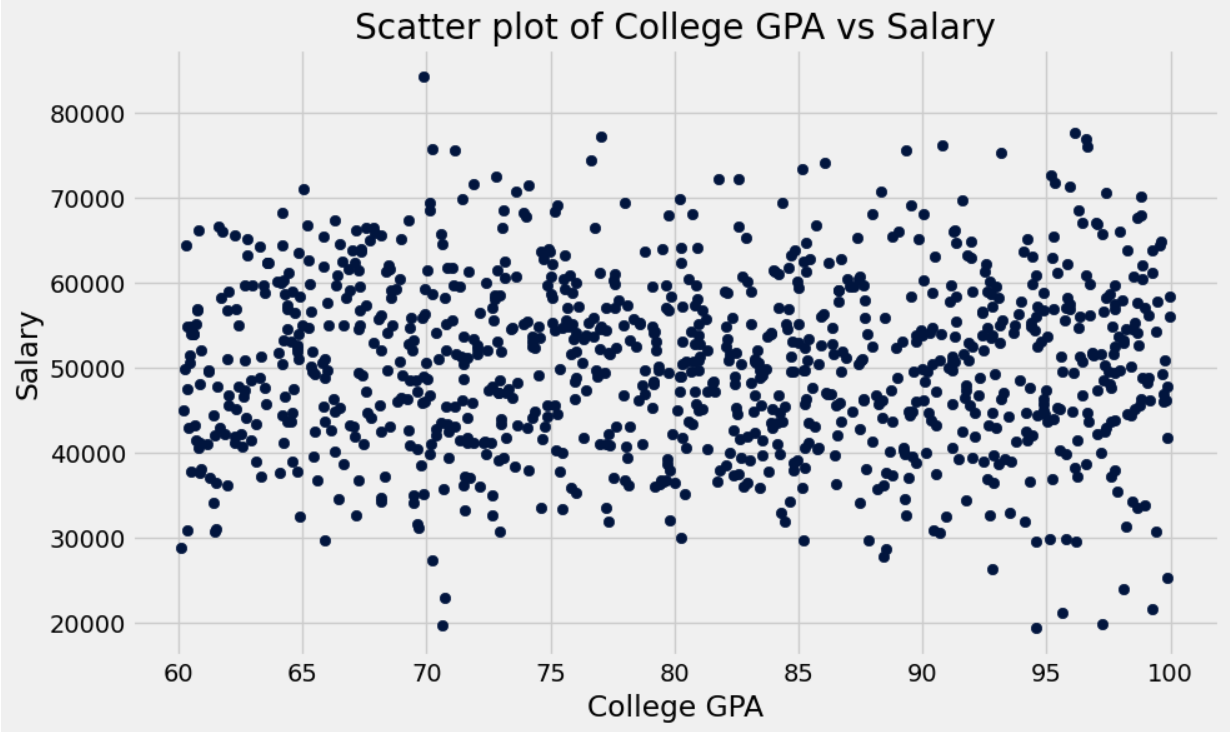
```
# Plot the count of genders
plt.figure(figsize=(8, 5))
sns.countplot(df['Gender'])
plt.title('Distribution of Genders')
plt.show()

# Display the count of each gender
print(df['Gender'].value_counts())
```




```
Gender
Male      506
Female    494
Name: count, dtype: int64
```

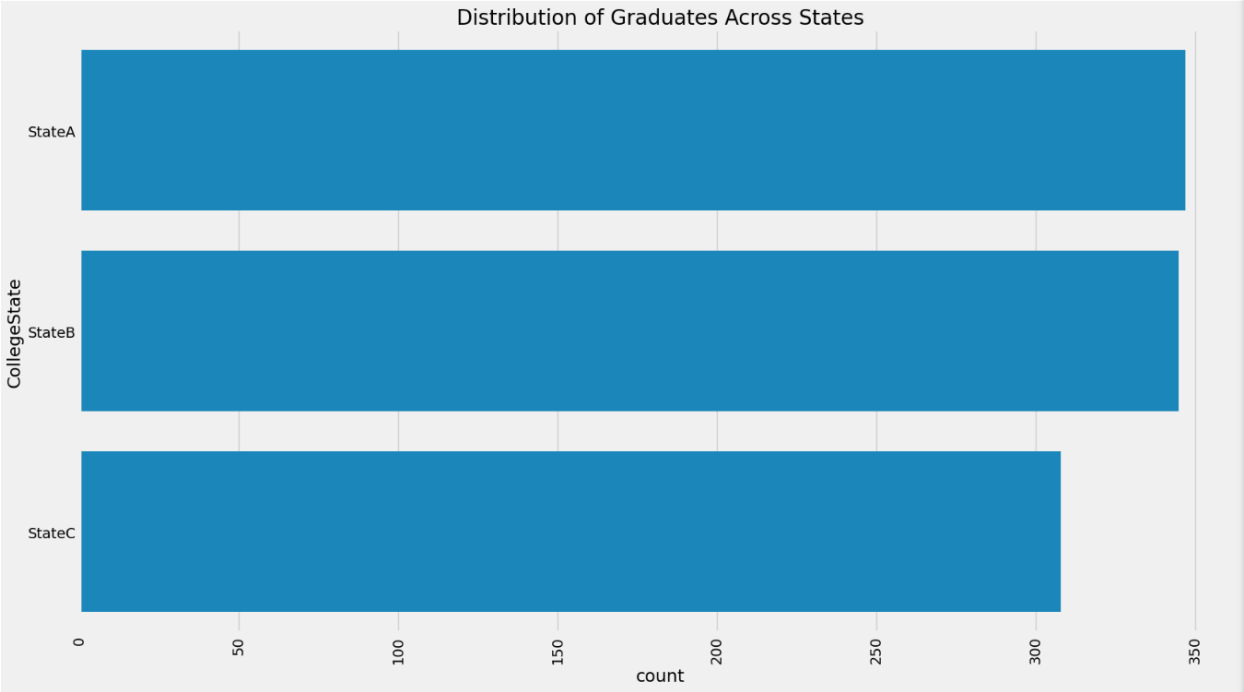
Q. Express through Scatter Plot Engineering College CGPA vs Salary.

```
# Scatter plot of college GPA vs Salary
plt.figure(figsize=(10, 6))
plt.scatter(x=df['collegeGPA'], y=df['Salary'], color='#01153E')
plt.xlabel('College GPA')
plt.ylabel('Salary')
plt.title('Scatter plot of College GPA vs Salary')
plt.show()
```



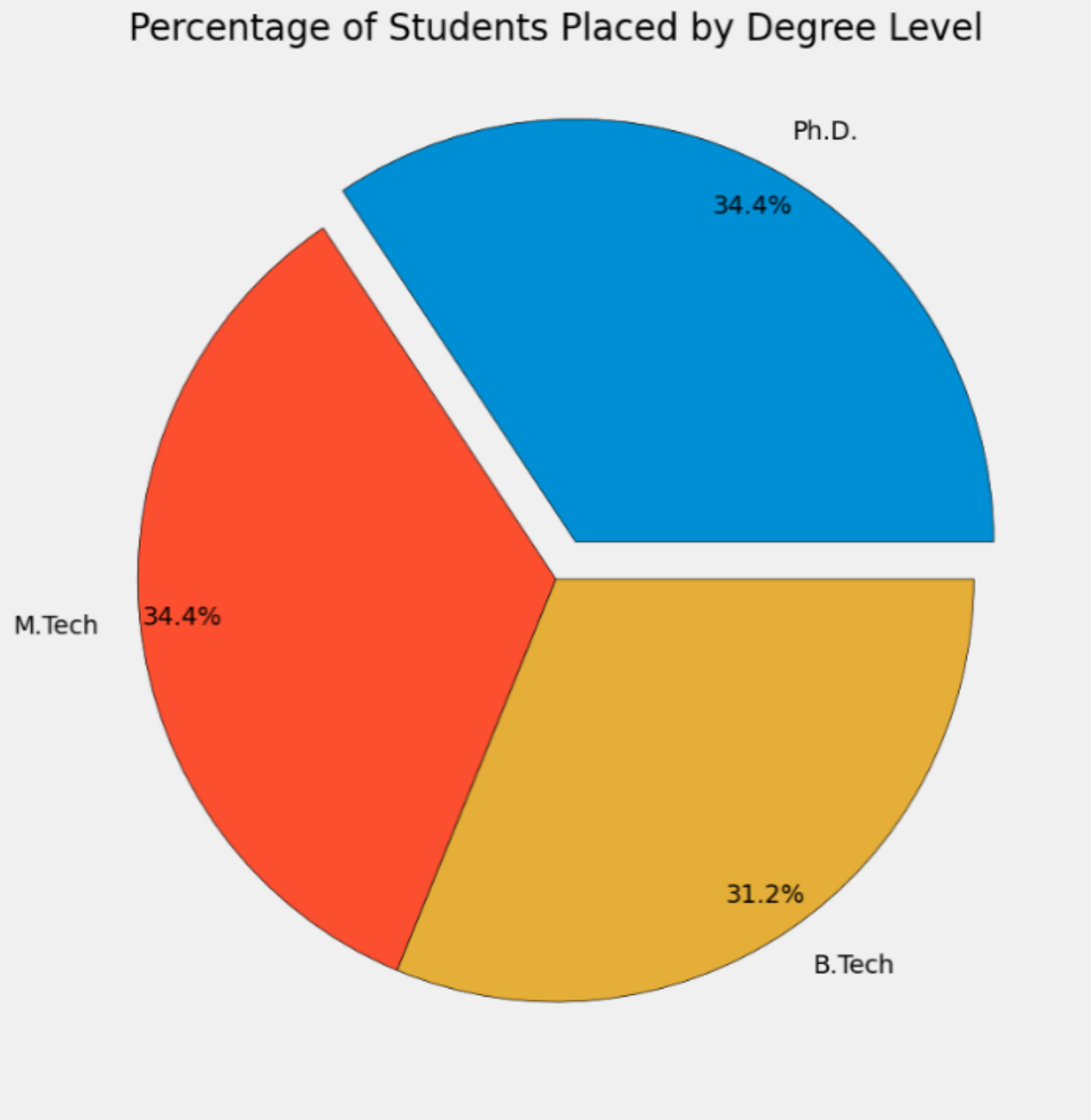
Q. College Management wants to know that where the maximum students from which state.

```
# Plot the count of graduates from each state
plt.figure(figsize=(18, 10))
sns.countplot(df['CollegeState'])
plt.xticks(rotation=90)
plt.title('Distribution of Graduates Across States')
plt.show()
```



Q. Percentage of Students placed from each branch.

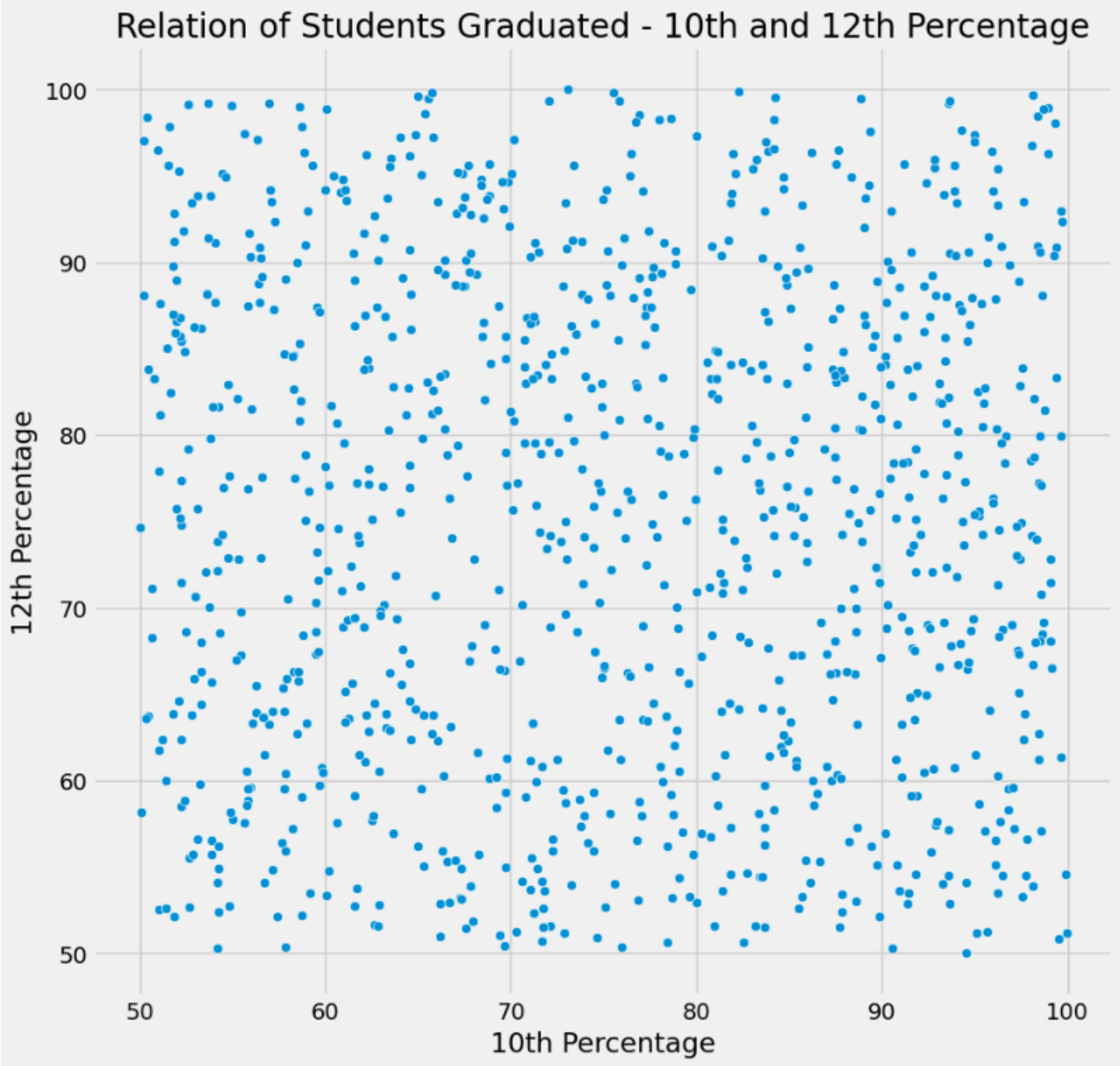
```
#percentage of students are placed from which level
index = df['Degree'].value_counts().index
values = df['Degree'].value_counts()

# Set explode to have the same length as the number of wedges
explode = [0.1] + [0] * (len(index) - 1)
plt.figure(figsize=(18, 10))
plt.pie(values, labels=index, autopct="%1.1f%%", wedgeprops={'edgecolor': 'black'}, explode=explode, pctdistance=0.9)
plt.title('Percentage of Students Placed by Degree Level')
plt.show()
print(df['Degree'].value_counts())
```




```
Degree
Ph.D.      344
M.Tech     344
B.Tech     312
Name: count, dtype: int64
```

Q. Show Relation of Students Graduated and their 10th and 12th percentage.

```
#Relation of students graduated and their 10th and 12th percentage using Scatter plot
plt.figure(figsize=(10, 10))
sns.scatterplot(x="10percentage", y="12percentage", data=df)
plt.xlabel('10th Percentage')
plt.ylabel('12th Percentage')
plt.title('Relation of Students Graduated - 10th and 12th Percentage')
plt.show()
```



Results and Discussion:

This Python project generates and analyzes synthetic data on engineering graduates. It explores demographics, academic performance, and career outcomes, providing visualizations for birth months, gender distribution, state-wise representation, degree-level placement, and correlations between college GPA, salary, and academic performance. The project demonstrates proficiency in data manipulation, analysis, and visualization using Python and relevant libraries.

Conclusion:

The project reveals diverse demographic patterns and academic qualifications among engineering graduates. A moderate correlation between college GPA and salary, along with insights into geographic distribution and degree-level placement, provides a comprehensive overview of the characteristics and potential career trajectories of engineering graduates.