Phone: 0755 – 2759026, 2529057,
2529258, 2529159
Fax: 0755 - 2529472
E-mail: oistbpl@oriental.ac.in
website: www.oriental.ac.in

ORIENTAL INSTITUTE OF SCIENCE & TECHNOLOGY

Approved by AICTE, New Delhi & Govt. of M.P. Affiliated to Rajiv Gandhi Proudyogiki Vishwavidyalaya, Bhopal
Oriental Campus, Thakral Nagar, Raisen Road, Bhopal-462 021 (M.P.) INDIA

Department of Computer Science & Engineering -Data Science

BONAFIDE CERTIFICATE

Certified that this project report **" CAMPUS RECRUITMENT OF OIST BHOPAL"** is a bonafide work of "**RAHUL PATEL "** and "**RISHI RAGHUWANSHI**" who carried out the project work under my supervision.

Date: 23/11/2023

Place: ORIENTAL INSTITUTE OF SCIENCE AND TECHNOLOGY BHOPAL

HEAD OF THE DEPARTMENT **Prof. SWATI PANDEY**

Computer Science & Engineering -Data Science
Oriental Institute of Science & Technology
Oriental campus, Raisen road, Bhopal-462021

Oriental Institute of Science & Technology
Oriental campus, Raisen road, Bhopal-462021

ACKNOWLEDGEMENT

We wish to express our profound gratitude to **Prof. SWATI PANDEY, Department of Computer Science & Engineering –Data Science, Oriental Institute of Science and Technology Bhopal** for her invaluable guidance, sustained help and inspiration in carrying out the work.

We sincerely thank **Prof. Swati Pandey, Head of Department of Computer Science & Engineering –Data Science,** OIST Bhopal for her co-operation and guidance during the project work.

We wish to express our deep gratitude to **Dr. Rajesh K Shukla, Director, and OIST Bhopal** for his kind permission to carry out the work at OIST Bhopal.

Finally we would like to thank all the faculty members and other persons who directly or indirectly helped us in carrying out this project work.

RAHUL PATEL (0105CD211039)
RISHI RAGHUWANSHI (0105CD211046)

ABSTRACT

The Campus Recruitment System is a minor project developed by Rahul Patel and Rishi Raghuwansh from Oriental Institute of Science and Technology in Bhopal. This project aims to streamline the campus recruitment process by providing an efficient system for both employers and students. The abstract of this project will provide a concise overview of its objectives, features, and expected outcomes.

The system will include features such as employer registration, student registration, job posting, application submission, and interview scheduling. It will also generate reports and statistics to track the progress and effectiveness of the recruitment process. The Campus Recruitment System will significantly reduce the manual effort involved in organizing and managing campus recruitment activities, making it more streamlined and convenient for all stakeholders involved.

The system will also have a user-friendly interface that will allow employers and students to easily navigate and interact with the platform. Additionally, it will incorporate a notification system to keep both employers and students updated on the status of their applications and interviews. With the implementation of the Campus Recruitment System, the recruitment process will become more efficient and transparent, facilitating better communication and decision-making.

Moreover, the system will have a robust search functionality to enable employers to find suitable candidates based on specific criteria such as education, skills, and experience. It will also provide personalized recommendations to students, suggesting relevant job opportunities based on their profiles and interests. By incorporating these features, the Campus Recruitment System aims to enhance the overall experience for both employers and students, ensuring better matches and increased success rates in the recruitment process.

The Placement Cell maintains a large database of students wherein all the information of students including personal records and the academic performance in terms C.P.I. is stored and company information including profile of company, eligibility criteria and facilities it provides etc. The software retrieves this data and displays as per the user requirement.

The Placement Management System is developed as an attempt to take a record of company and students by restricting such a large database to that of a particular class of students or company. The system provides the facility of viewing both the personal and academic information of the student and company. Company can post vacancy and students can apply if they are eligible. Company can view the applications and can accept or reject them.

The project is to facilitate students in college, company to register and communicate with Placement Cell.

TABLE OF CONTENTS

LIST OF FIGURES

Campus Recruitment

Figure 7: Placement Rates by Degree Specialization

- Description: Bar chart displaying the placement rates for different degree specializations.
- **Confusion Matrix of Model Predictions**

- Description: Visual representation of the confusion matrix for the machine learning model predictions.

Figure 9: Box Plots for Numerical Columns by Placement Status

- Description: Collection of box plots showing the distribution of various numerical features based on placement status.

Figure 10: Bar Chart of Average SSC Percentage by Placement Status:-

- Description: Bar chart comparing the average SSC percentage for placed and non-placed candidates.

LIST OF TABLES

Campus Recruitment

LIST OF ALGORITHMS

Table No. CAMPUS RECRUITMENT

LIST OF ABBREVIATIONS

1. **EDA:** Exploratory Data Analysis.
2. **ML:** Machine Learning.
3. **CSV:** Comma-Separated Values.
4. **EDA:** Exploratory Data Analysis.
5. **ETL:** Extract, Transform, Load.
6. **SVM**: Support Vector Machine.
7. **NaN**: Not a Number (commonly used for missing values).
8. **PCA**: Principal Component Analysis

1. INTRODUCTION

1.1 Objective :

The objective of the project, as inferred from the provided information, is to analyze campus recruitment data and derive insights into factors influencing placement outcomes.

1.2 Scope :

The scope of the project encompasses several aspects related to campus recruitment and the analysis of placement data. Here are some key components of the project scope:

1. **Campus Recruitment Analysis**:

The project focuses on analyzing placement data from a XYZ campus, aiming to provide insights into the recruitment process and outcomes.

2. **Exploration of Placement Factors:**

   The project explores various factors influencing the placement of students. This includes investigating academic performance, degree specialization, work experience, and other relevant features.

3. **Impact of Academic Percentages:**

   The scope involves assessing the significance of academic percentages (secondary and higher secondary) in determining placement success. The project aims to quantify the influence of academic performance on placement outcomes.

4. **Demand for Degree Specializations:**

An essential aspect of the project is to identify and understand the demand for different degree specializations in the corporate sector. This involves analyzing placement rates for various degree programs.

The project includes the application of statistical tests and techniques to conduct a comprehensive analysis of the data. Statistical analyses contribute to uncovering patterns, correlations, and meaningful insights.

5. **Practical Application for Stakeholders:**

The findings and insights derived from the project aim to be practically applicable for stakeholders involved in campus recruitment. This includes students, faculty, and corporate recruiters who can benefit from a better understanding of placement dynamics.

6. **Contribution to Decision-Making:**

The project contributes to decision-making processes by providing actionable insights. Students can make informed choices based on the identified influential factors, and recruiters can gain a better understanding of the preferences and trends in degree specializations.

7. **Educational Impact:**

As part of an educational program, the project's scope includes enhancing students' skills in data analysis, statistical reasoning, and deriving meaningful conclusions from real-world datasets.

1.3 Problem identification:

The problem identification in the project "Campus Recruitment" involves recognizing challenges or areas of interest related to campus placement. Here are potential problem areas that the project aims to address:

1. **Placement Success Factors**:

Understanding which factors significantly influence a candidate's success in getting placed. This involves identifying key determinants such as academic performance, work experience, and other relevant features.

2. .**Academic Performance Significance**:

Investigating whether academic percentages (secondary and higher secondary) play a crucial role in securing placements. Analyzing the correlation between academic achievements and placement outcomes is a key aspect of the problem.

3. **Demand for Degree Specializations**:

Recognizing the varying demand for different degree specializations in the corporate sector. Identifying which degree programs are more sought after by recruiters can help students make informed decisions about their academic paths.

4. **Statistical Analysis for Insights:**

Employing statistical tests and analyses to uncover patterns, correlations, and insights from the placement data. This addresses the challenge of extracting meaningful information from a complex dataset.

5. **Practical Applicability of Findings**:

Ensuring that the project's findings and insights are practically applicable to stakeholders involved in the campus recruitment process. The challenge lies in translating data-driven results into actionable recommendations for students, faculty, and recruiters

6. **Educational Impact and Skill Development:**

Enhancing the educational impact of the project by developing students' skills in data analysis, statistical reasoning, and interpretation of real-world datasets. This addresses the challenge of creating a project that aligns with educational objectives.

1.4 Methodology:-

Methodology of the Project: The methodology of the "Campus Recruitment" project involves a systematic approach to analyzing the placement data. Here is a breakdown of the key steps followed:

1. **Understanding the Dataset:**

Begin by gaining a comprehensive understanding of the dataset's structure, features, and variables. Explore data types, missing values, and unique patterns within the dataset.

2. **Data Cleaning and Preprocessing**:

Address any missing or inconsistent data to ensure the dataset's reliability. This step involves imputing missing values, handling outliers, and transforming variables if necessary.

3. **Exploratory Data Analysis (EDA):**

Conduct EDA to uncover patterns, trends, and relationships within the data. Utilize statistical measures and visualizations, such as histograms, box plots, and correlation matrices, to gain insights into the distribution of variables and their interdependencies.

4. **Formulating Research Questions**:

Based on the initial exploration, formulate specific research questions aligned with the project's objectives. The questions guide the subsequent analysis and testing.

5. **Hypothesis Generation**:

   Develop hypotheses that can be tested statistically to answer the research questions. Hypotheses provide a framework for validating assumptions and drawing meaningful conclusions.

6. **Statistical Analysis**:

   Apply appropriate statistical tests to test the formulated hypotheses. Common tests include chi-square tests for categorical variables, t-tests for mean comparisons, and correlation analyses for exploring relationships.

7. **Data Visualization**:

   Create visualizations to represent the findings effectively. Visuals such as bar charts, pie charts, and scatter plots contribute to a clearer understanding of the data and facilitate communication of results.

2. BACKGROUND AND LITERATURE SURVEY

1. **Existing System:**

   Provide an overview of the existing system or practices related to campus recruitment. Highlight the traditional methods and processes followed in the industry for student placement.

2. **Requirement Specifications**:

   Detail the specific requirements identified for the project. This involves understanding the needs of the campus recruitment process and defining the functionalities and features expected from the system.

3. **Feasibility Study:**

   Conduct a feasibility study to assess the practicality and viability of implementing the proposed project. Evaluate technical, economic, and operational aspects to determine whether the project is feasible.

4. **Innovativeness and Usefulness**:

   Discuss the innovative aspects of the proposed system and how it differs from or improves upon the existing practices. Emphasize the potential usefulness and advantages of the proposed system.

5. **Modification and Improvement over the Existing Implementation**:

   Highlight any modifications or enhancements made to the existing system. Discuss how the proposed project addresses limitations or inefficiencies identified in the current campus recruitment processes.

3. PROCESS MODEL

1. **Software Process Model Used**:

   Specify the software process model employed in the project. Common models include Waterfall, Agile, Spiral, etc. Explain why the chosen model is suitable for the nature of the campus recruitment project.

2. **Proposed Project Model:**

   Outline the proposed project model that will be implemented. Describe the key phases, activities, and milestones involved in the project lifecycle. This provides an overview of how the project will be executed.

3. **Market Potential and Competitive Advantage**:

   Analyze the market potential for the proposed system in the context of campus recruitment. Discuss how the project provides a competitive advantage over existing solutions or practices, contributing to its potential success in the market.

4. **Project Estimation:**

   Provide an estimation of resources required for the project. This includes personnel, hardware, software, and any other resources. Discuss the basis for these estimations and factors considered in the estimation process.

5. **Project Schedule**:

   Outline the project schedule, including timelines for each phase and major milestones. This helps in managing and tracking the progress of the project over time. Provide a visual representation if possible, such as a Gantt chart.

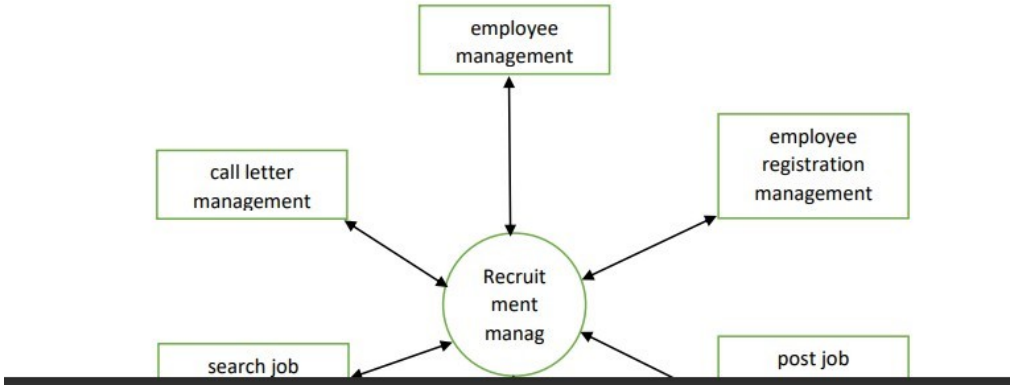
4. DESIGN

1. **Data Flow Diagram (DFD):**

4.8 DFD Diagram

4.8.1 Zero Level

This is a zero level patient information system where we have elaborated the high level process of job recruitment system. It is designed to be an at-a-glance and login showing the system as a single high level process, with its relationship to external entities of employee, employee registration and post jobs.

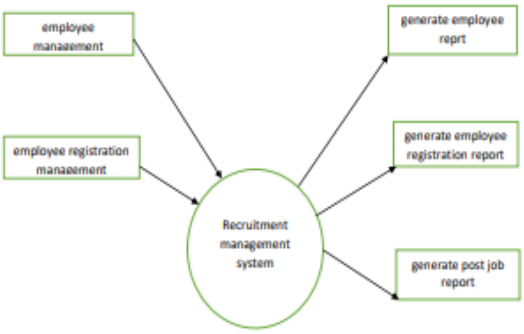


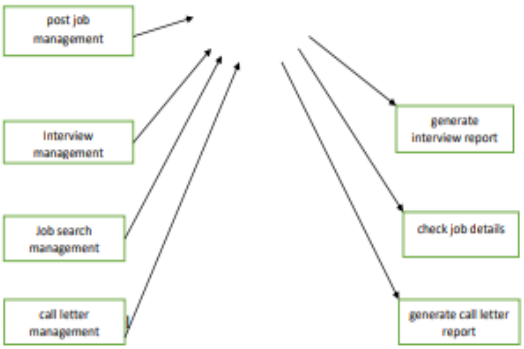
4.8.2 First Level

First level DFD of the patient information system diagram shows how the system is divided into sub system each of deals with one or more of the data flows to or from an external agent, and which together provide all the functionality of the job recruitment system.

25

It also identifies internal data stores of login, call letter, search jobs, interview, post jobs that must be the present in order for job recruitment system and it shows the flow of data to various parts of job recruitment system.

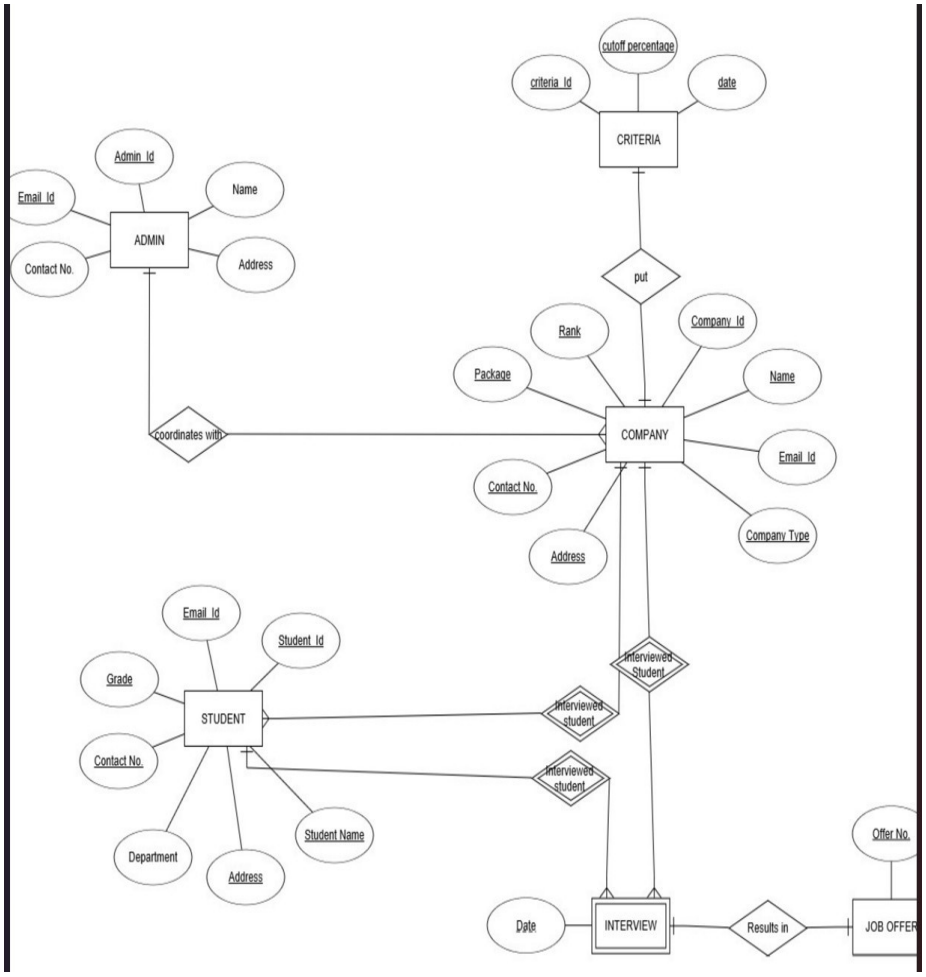




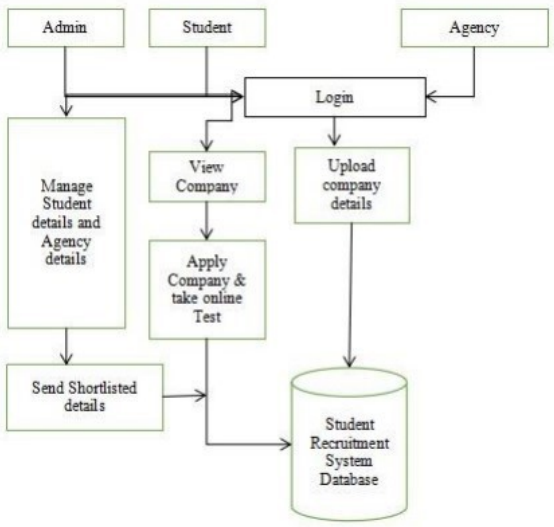
26

2. **Entity Relationship Diagram (ERD):**

Present the Entity Relationship Diagram that focuses on the entities and relationships relevant to the users. This diagram highlights the data entities associated with users and their connections.

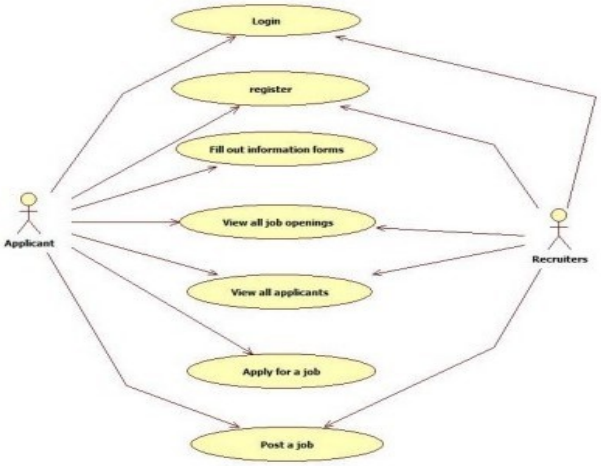


3. **<u>Flow Chart</u>**:
   Include a flow chart specifically outlining the user-centric processes. This could cover the steps involved in user interactions, decision points, and the flow of control within the system.



4. **<u>USE-CASE Diagrams</u>**:

   Create USE-CASE diagrams specifically addressing user interactions. Highlight the various use cases that users (like students, recruiters, administrators) can perform within the system. Clearly define actors, use cases, and their relationships.



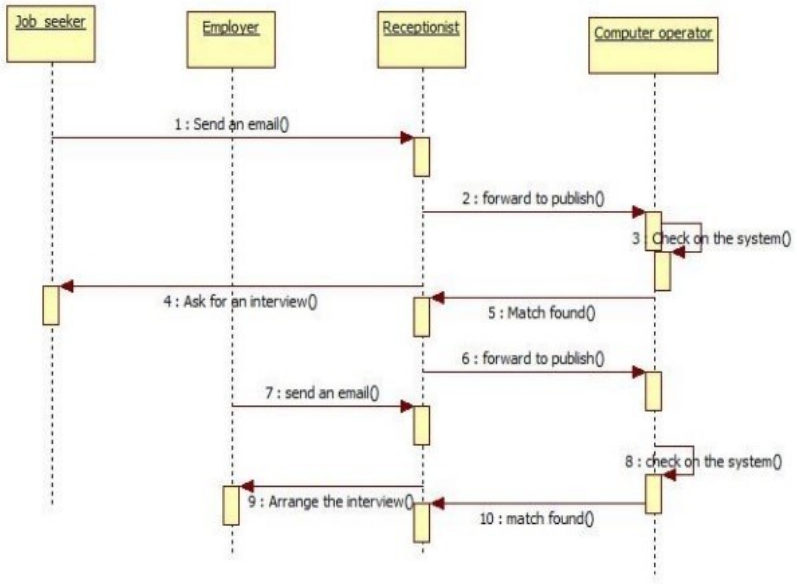
5. **Sequence Diagram**:

   Provide a sequence diagram to visualize the chronological sequence of interactions between different elements in the system. This is particularly useful in understanding the timing and order of activities.

6. **Activity Diagram**:

   Present an activity diagram that shows the flow of activities within the campus recruitment system. This highlights the sequence and conditions of various activities.

4.5 Sequence Diagram



7. **Class Diagram**:

   Include a class diagram to represent the classes, their attributes, and relationships in the system. This is crucial for understanding the structure of the system and its components.

8. **Algorithm**:

Classification Algorithms:-

a) **Logistic Regression**: For binary classification problems like predicting placement status (Placed/Not Placed).

b) **Decision Trees**: Building decision trees to model the decision-making process.

c) **Random Forest**: An ensemble method using multiple decision trees for improved accuracy and robustness. d)**Support Vector Machines (SVM)**: Effective for binary classification tasks.

Regression Algorithms:

a) **Linear Regression**: If predicting numerical values like salary.

b) **Ridge Regression or Lasso Regression**: Regularized regression techniques to handle multicollinearity.

Clustering Algorithms (Possibly for Exploratory Analysis):

a) **K-Means Clustering**: Grouping similar data points to identify patterns.

b) **Hierarchical Clustering**: Building a tree of clusters to represent the relationships in the data.

Feature Scaling and Selection:

a) **Standardization or Normalization**: Scaling numerical features.

b)**Feature Importance Techniques**: Analyzing which features contribute most to the model.

Model Evaluation and Hyperparameter Tuning:

a) **Cross-Validation**: Ensuring robust model evaluation.

b) **Grid Search or Random Search**: Tuning hyperparameters for better model performance.

Visualization Techniques:

a) **Matplotlib and Seaborn**: Visualizing data distributions, correlations, and model performance.

Data Preprocessing Techniques:

a) **Handling Missing Values:** Imputation using mean, median, or other methods.

b) **Encoding Categorical Variables**: Converting categorical data into numerical form.

5. TECHNOLOGY AND TOOLS USED

1. **Programming Languages:**

   Python: For data analysis, statistical tests, and machine learning models. R: For statistical analysis and visualizations.

2. **Data Analysis and Machine Learning Libraries**:

   Pandas: For data manipulation and analysis. NumPy: For numerical operations. Matplotlib and Seaborn: For data visualization. Scikit-learn: For implementing machine learning algorithms.

3. **Data Handling**:

   Pandas: For reading and manipulating datasets.

4. **Statistical Analysis**:

   SciPy: For statistical analysis.

5. **Machine Learning** Models:

   Scikit-learn: For implementing machine learning models such as RandomForestClassifier.

6. **Visualization**:

   Matplotlib and Seaborn: For creating visualizations to understand data patterns.7.

7. **Web Scraping**:

   Beautiful Soup and Requests: For scraping data from websites.

8. **Jupyter Notebooks**:

   For interactive and exploratory data analysis.

9. **Documentation:**

Jupyter Notebooks: For creating a comprehensive and interactive documentation.

10. **Version Control**:

    Git and GitHub: For version control and collaboration.

11. **Integrated Development Environment** (IDE):

    Jupyter Lab: For developing and testing code.

12. **Collaboration:**

    GitHub: For version control, collaboration, and sharing code.

6. HARDWARE & SOFTWARE REQUIREMENT

1. CPU (Central Processing Unit):	A modern multi-core processor (e.g., Intel Core i5 or above) would be sufficient.
2 RAM (Random Access Memory):	At least 8 GB RAM is recommended for handling large datasets and running machine learning algorithms.
3. Storage (HDD/SSD):	Adequate storage space for datasets, code, and project files. A minimum of 256 GB SSD or HDD is recommended
4. Network:	High-speed internet connection for data download, web scraping, and collaborative work (if applicable).
5. Operating System:	The project can be developed and run on various operating systems like Windows, macOS, or Linux.

7. CODING AND TESTING

```
In [6]: df.describe()
```

Out[6]:

	sl_no	ssc_p	hsc_p	degree_p	etest_p	mba_p	salary
count	215.000000	215.000000	215.000000	215.000000	215.000000	215.000000	148.000000
mean	108.000000	67.303395	66.333163	66.370186	72.100558	62.278186	288655.405405
std	62.209324	10.827205	10.897509	7.358743	13.275956	5.833385	93457.452420
min	1.000000	40.890000	37.000000	50.000000	50.000000	51.210000	200000.000000
25%	54.500000	60.600000	60.900000	61.000000	60.000000	57.945000	240000.000000
50%	108.000000	67.000000	65.000000	66.000000	71.000000	62.000000	265000.000000
75%	161.500000	75.700000	73.000000	72.000000	83.500000	66.255000	300000.000000
max	215.000000	89.400000	97.700000	91.000000	98.000000	77.890000	940000.000000

```
In [7]: df.columns
```

```
Out[7]: Index(['sl_no', 'gender', 'ssc_p', 'ssc_b', 'hsc_p', 'hsc_b', 'hsc_s',
       'degree_p', 'degree_t', 'workex', 'etest_p', 'specialisation', 'mba_p',
       'status', 'salary'],
      dtype='object')
```

```
In [8]: dict = {'workex': 'work_exprience',
        'etest_p': 'entrance test'}
```

```
In [9]: df.rename(columns=dict,
          inplace=True)
```

```
In [10]: df.columns
```

```
Out[10]: Index(['sl_no', 'gender', 'ssc_p', 'ssc_b', 'hsc_p', 'hsc_b', 'hsc_s',
       'degree_p', 'degree_t', 'work_exprience', 'entrance test',
       'specialisation', 'mba_p', 'status', 'salary'],
      dtype='object')
```

```
In [11]: df
```

Out[11]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	work_exprience	entrance test	specialisation	mba_p	status	salary
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	NaN
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
210	211	M	80.60	Others	82.00	Others	Commerce	77.60	Comm&Mgmt	No	91.0	Mkt&Fin	74.49	Placed	400000.0
211	212	M	58.00	Others	60.00	Others	Science	72.00	Sci&Tech	No	74.0	Mkt&Fin	53.62	Placed	275000.0
212	213	M	67.00	Others	67.00	Others	Commerce	73.00	Comm&Mgmt	Yes	59.0	Mkt&Fin	69.72	Placed	295000.0
213	214	F	74.00	Others	66.00	Others	Commerce	58.00	Comm&Mgmt	No	70.0	Mkt&HR	60.23	Placed	204000.0
214	215	M	62.00	Central	58.00	Others	Science	53.00	Comm&Mgmt	No	89.0	Mkt&HR	60.22	Not Placed	NaN

215 rows × 15 columns

```
#   Column          Non-Null Count  Dtype
```

```
In [12]: df.isnull().sum()
```

```
Out[12]: sl_no              0
gender             0
ssc_p              0
ssc_b              0
hsc_p              0
hsc_b              0
hsc_s              0
degree_p           0
degree_t           0
work_exprience     0
entrance test      0
specialisation     0
mba_p              0
status             0
salary            67
dtype: int64
```

```
In [13]: mean_salary = df['salary'].mean()              #Imputation
df['salary'].fillna(mean_salary, inplace=True)
```

In [14]:
```
df.isnull().sum()
```

Out[14]:
```
sl_no             0
gender            0
ssc_p             0
ssc_b             0
hsc_p             0
hsc_b             0
hsc_s             0
degree_p          0
degree_t          0
work_exprience    0
entrance test     0
specialisation    0
mba_p             0
status            0
salary            0
dtype: int64
```

In [15]:
```
df['ssc_p'].describe()
```

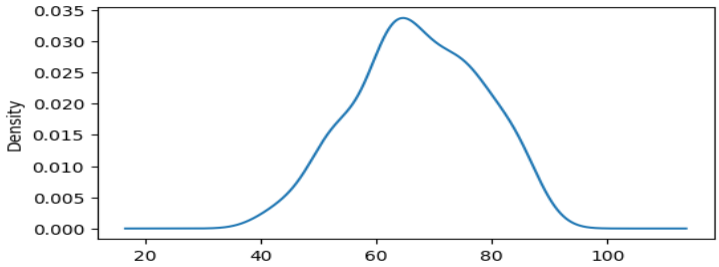
Out[15]:
```
count    215.000000
mean      67.303395
std       10.827205
min       40.890000
25%       60.600000
50%       67.000000
75%       75.700000
max       89.400000
Name: ssc_p, dtype: float64
```

Out[16]:
```
0      270000.000000
1      200000.000000
2      250000.000000
3      288655.405405
4      425000.000000
          ...
210    400000.000000
211    275000.000000
212    295000.000000
213    204000.000000
214    288655.405405
Name: salary, Length: 215, dtype: float64
```

In [17]:
```
df['ssc_p'].describe()
```

Out[17]:
```
count    215.000000
mean      67.303395
std       10.827205
min       40.890000
25%       60.600000
50%       67.000000
75%       75.700000
max       89.400000
Name: ssc_p, dtype: float64
```

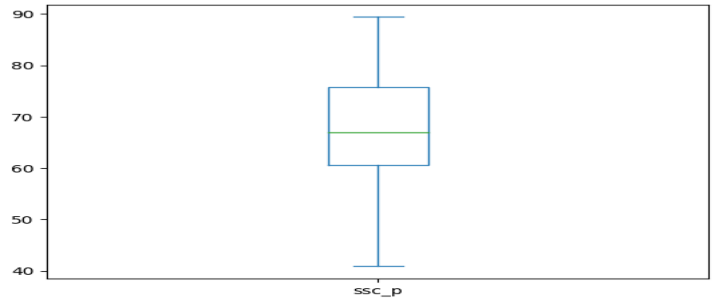
```
In [19]: df['ssc_p'].plot(kind='kde',figsize=(6, 3))
Out[19]: <Axes: ylabel='Density'>
```




```
In [20]: df['ssc_p'].skew()
Out[20]: -0.13264949031456927
```


```
In [20]: df['ssc_p'].skew()
Out[20]: -0.13264949031456927
```

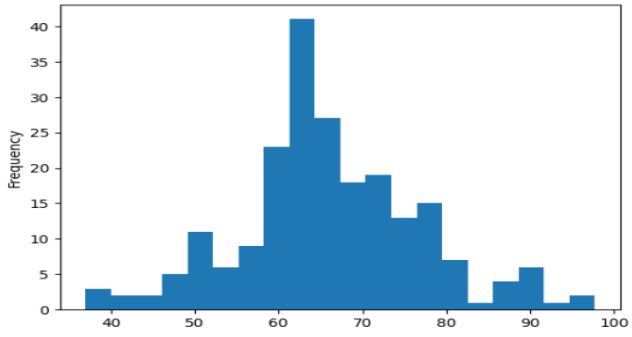

```
In [21]: df['ssc_p'].plot(kind='box')
Out[21]: <Axes: >
```



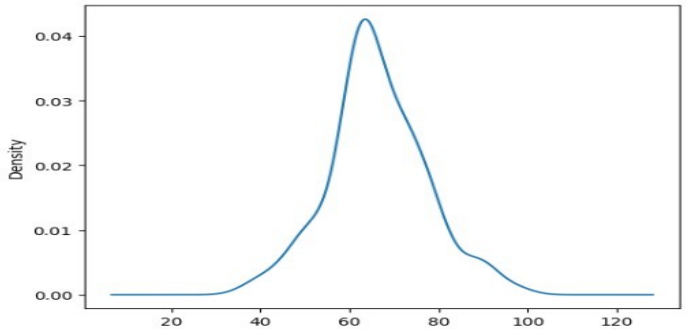

```
In [22]: df['ssc_p'].isnull().sum()
Out[22]: 0
```


```
In [22]: df['ssc_p'].isnull().sum()
Out[22]: 0
```


```
In [23]: df['hsc_p'].describe()
Out[23]: count    215.000000
         mean      66.333163
         std       10.897509
         min       37.000000
         25%       60.900000
         50%       65.000000
         75%       73.000000
         max       97.700000
         Name: hsc_p, dtype: float64

In [24]: df['hsc_p'].plot(kind='hist',bins=20)
Out[24]: <Axes: ylabel='Frequency'>
```



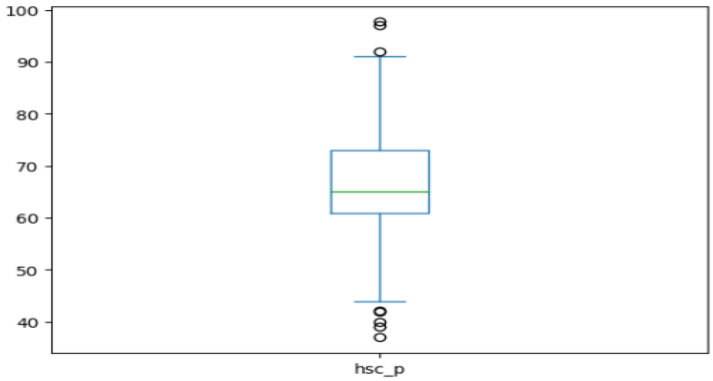
```
In [25]: df['hsc_p'].plot(kind='kde',figsize=(6,5))
Out[25]: <Axes: ylabel='Density'>
```




```
In [26]: df['hsc_p'].skew()
Out[26]: 0.16363913146416115
```

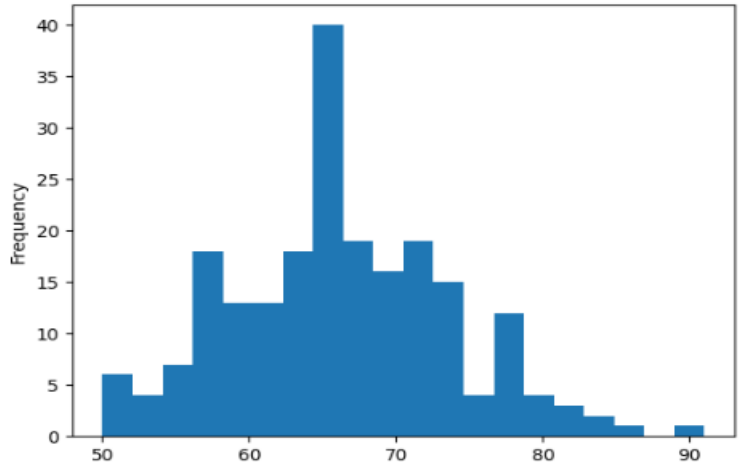
perfectly symetrical distribution

```
In [27]: df['hsc_p'].plot(kind='box')
Out[27]: <Axes: >
```

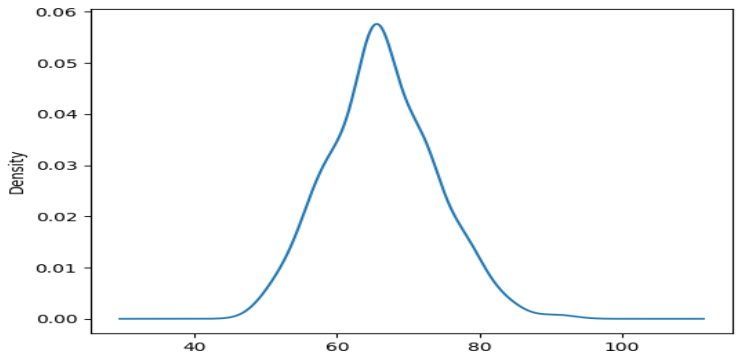



```
In [28]: df['hsc_p'].isnull().sum()
Out[28]: 0
```

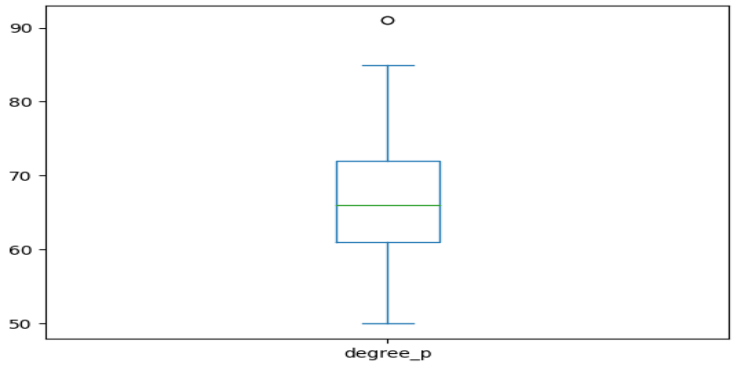
```
In [31]: df['degree_p'].plot(kind='hist',bins=20)
Out[31]: <Axes: ylabel='Frequency'>
```



```
In [32]: df['degree_p'].plot(kind='kde',figsize=(6,5))
Out[32]: <Axes: ylabel='Density'>
```




```
In [33]: df['degree_p'].plot(kind='box')
Out[33]: <Axes: >
```

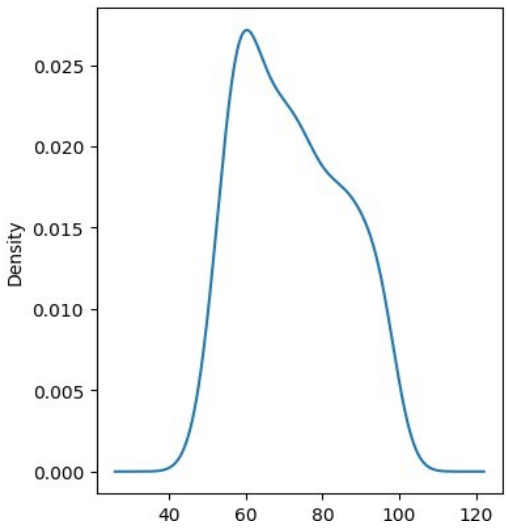



```
In [34]: df[df['degree_p']>85]
```

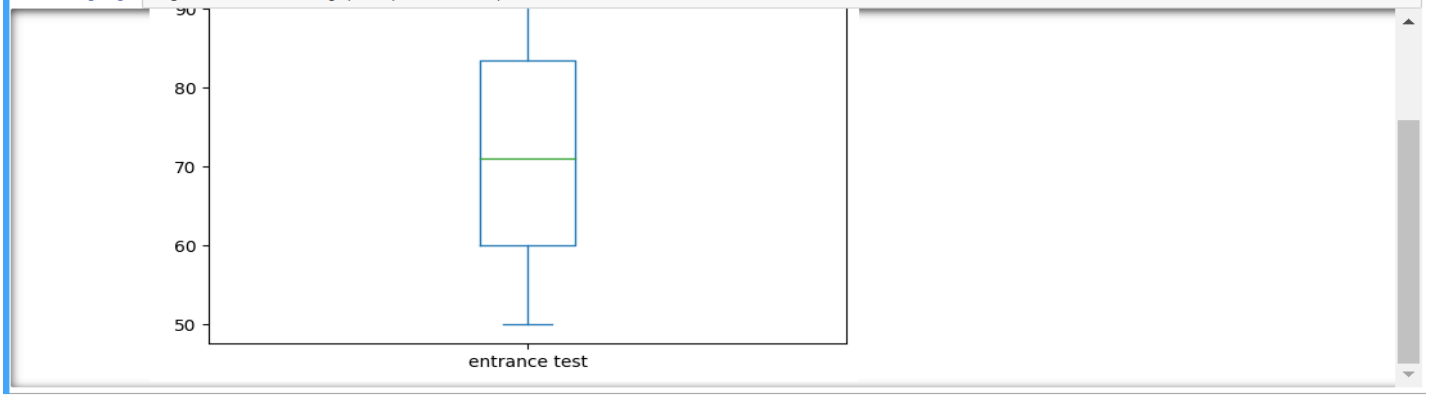
Out[34]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	work_exprience	entrance test	specialisation	mba_p	status	salary
197	198	F	83.96	Others	53.0	Others	Science	91.0	Sci&Tech	No	59.32	Mkt&HR	69.71	Placed	260000.0

```
In [37]: df['entrance test'].plot(kind='kde',figsize=(4,5))
Out[37]: <Axes: ylabel='Density'>
```



In [38]:
```
df['entrance test'].plot(kind='box')
```



In [39]:
```
df['entrance test'].skew()
```

Out[39]:
```
0.28230815846982243
```

In [40]:
```
df['mba_p'].describe()
```

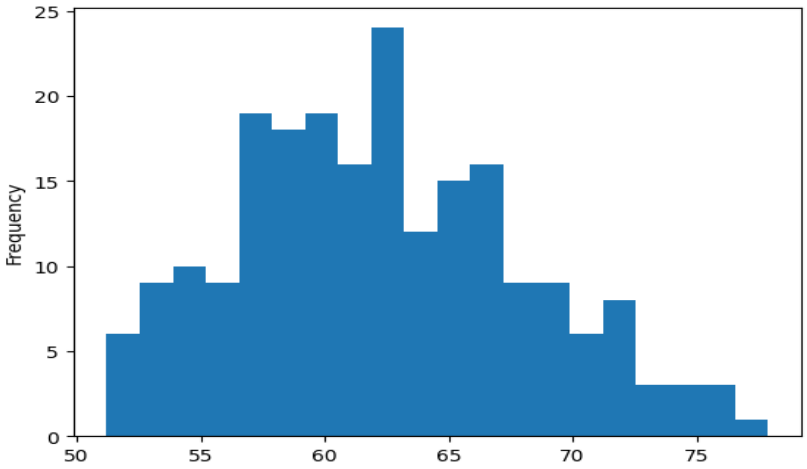
Out[40]:
```
count    215.000000
mean      62.278186
std        5.833385
min       51.210000
25%       57.945000
50%       62.000000
75%       66.255000
max       77.890000
Name: mba_p, dtype: float64
```

In [4

Out[4

In [41]:
```
df['mba_p'].plot(kind='hist',bins=20)
```

Out[41]:
```
<Axes: ylabel='Frequency'>
```



In [4

Out[4

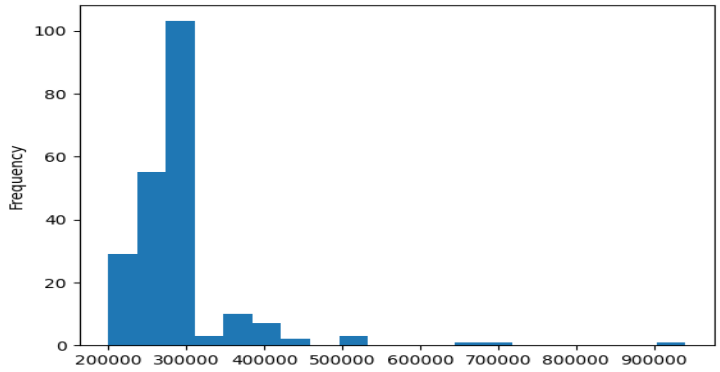
In [42]:
```
df['mba_p'].plot(kind='kde',figsize=(4,5))
```

```
In [45]: df['salary'].describe()
```

```
Out[45]: count       215.000000
         mean     288655.405405
         std       77457.900102
         min      200000.000000
         25%      250000.000000
         50%      288655.405405
         75%      288655.405405
         max      940000.000000
         Name: salary, dtype: float64
```

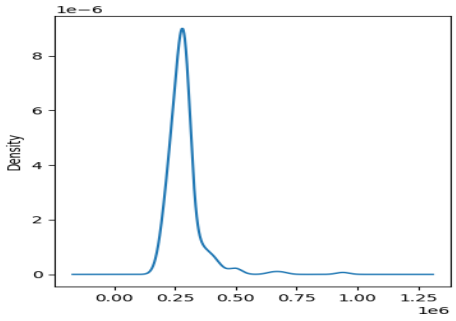
```
In [46]: df['salary'].plot(kind='hist',bins=20)
```

```
Out[46]: <Axes: ylabel='Frequency'>
```



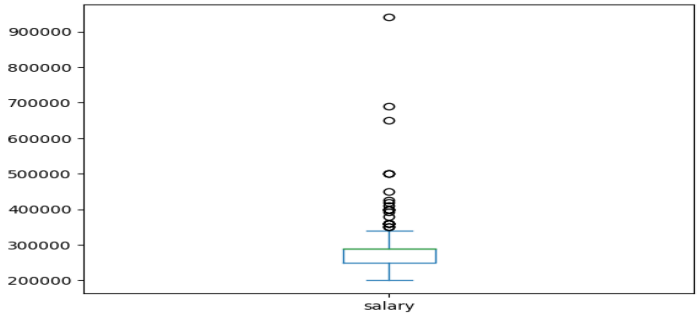

```
In [47]: df['salary'].plot(kind='kde',figsize=(4,5))
```

```
Out[47]: <Axes: ylabel='Density'>
```



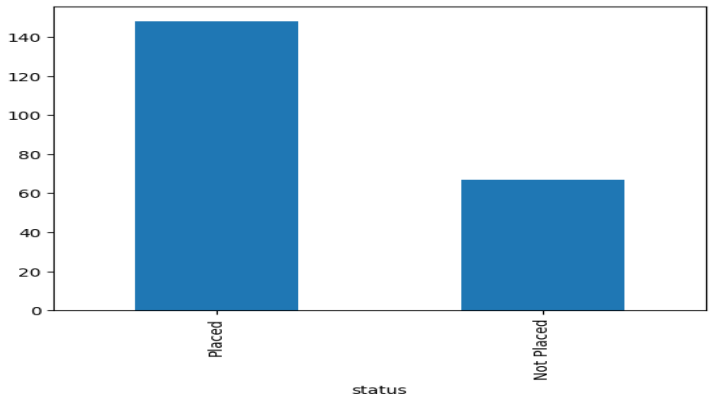

```
In [48]: df['salary'].plot(kind="box")
```

```
Out[48]: <Axes: >
```

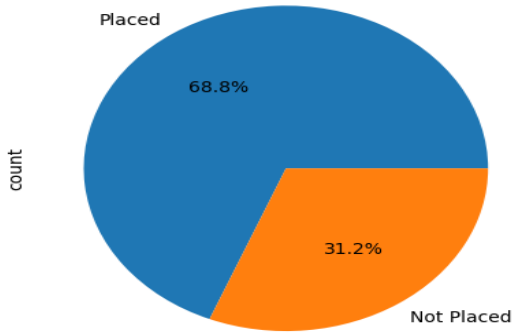



```
In [49]: df[df['salary']>35000]
```

```
In [53]: df['status'].value_counts().plot(kind='bar')
Out[53]: <Axes: xlabel='status'>
```




```
In [54]: df['status'].value_counts().plot(kind='pie',autopct='%0.1f%%')
Out[54]: <Axes: ylabel='count'>
```

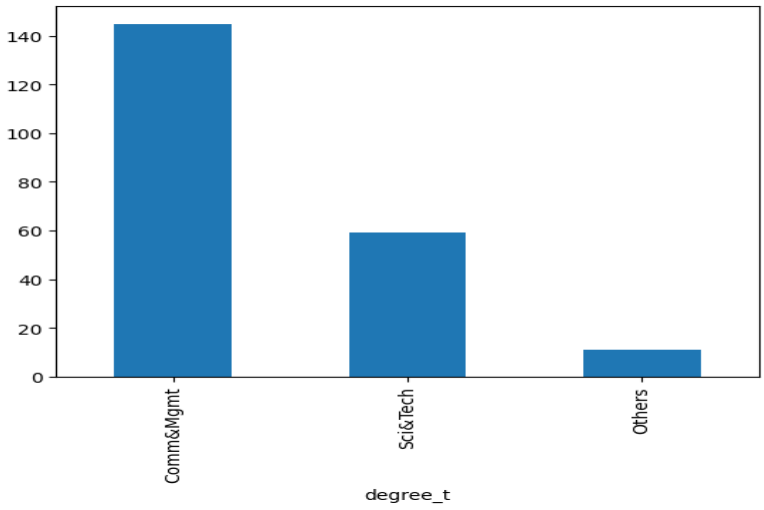



```
In [55]: df['degree_t'].count()
Out[55]: 215

In [56]: df['degree_t'].value_counts()
Out[56]: degree_t
         Comm&Mgmt    145
         Sci&Tech      59
         Others        11
         Name: count, dtype: int64
```


```
In [55]: df['degree_t'].count()
Out[55]: 215

In [56]: df['degree_t'].value_counts()
Out[56]: degree_t
         Comm&Mgmt    145
         Sci&Tech      59
         Others        11
         Name: count, dtype: int64
```

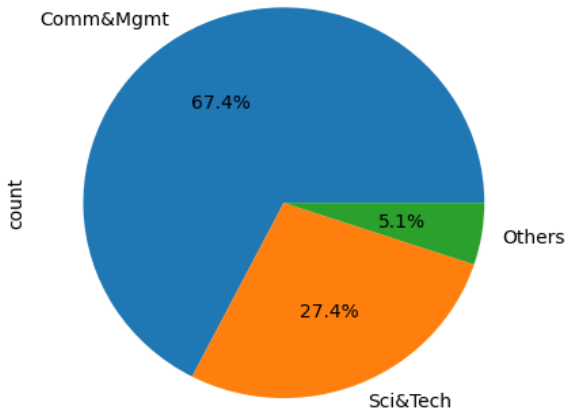

```
In [57]: df['degree_t'].value_counts().plot(kind='bar')
```

Out[57]: <Axes: xlabel='degree_t'>



In [58]:
```
df['degree_t'].value_counts().plot(kind='pie',autopct='%0.1f%%')
```

Out[58]: <Axes: ylabel='count'>



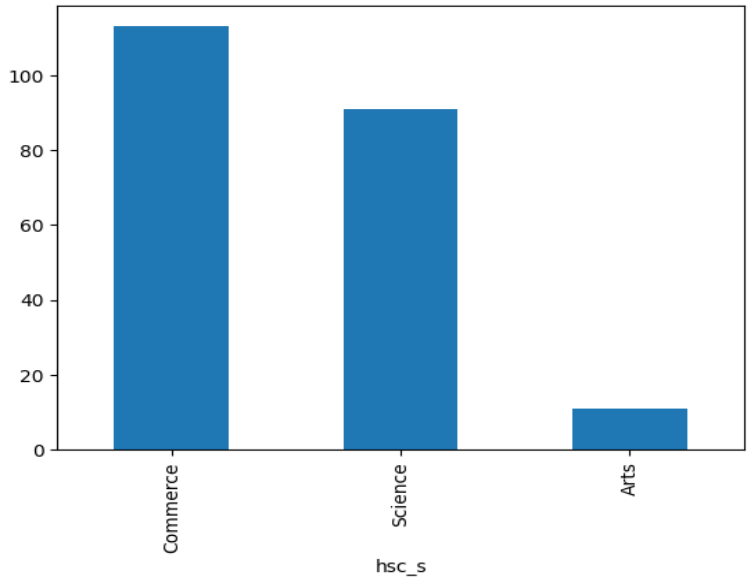
In [59]:
```
df['hsc_s'].count()
```

Out[59]: 215

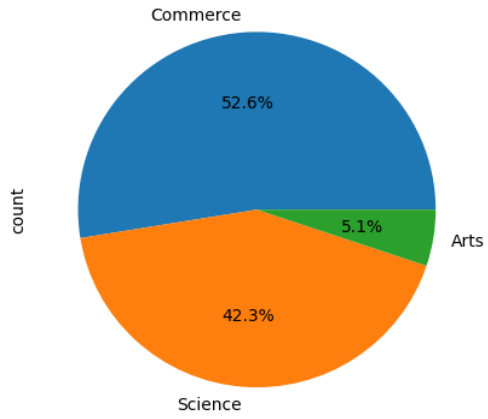
In [60]:
```
df['hsc_s'].value_counts()
```

Out[60]:
```
hsc_s
Commerce    113
Science      91
Arts         11
Name: count, dtype: int64
```

```
In [61]: df['hsc_s'].value_counts().plot(kind='bar')
Out[61]: <Axes: xlabel='hsc_s'>
```

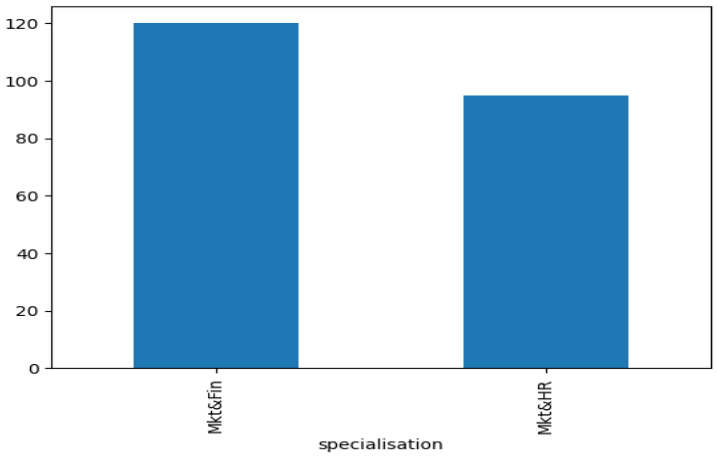



```
In [62]: df['hsc_s'].value_counts().plot(kind='pie',autopct='%0.1f%%')
Out[62]: <Axes: ylabel='count'>
```

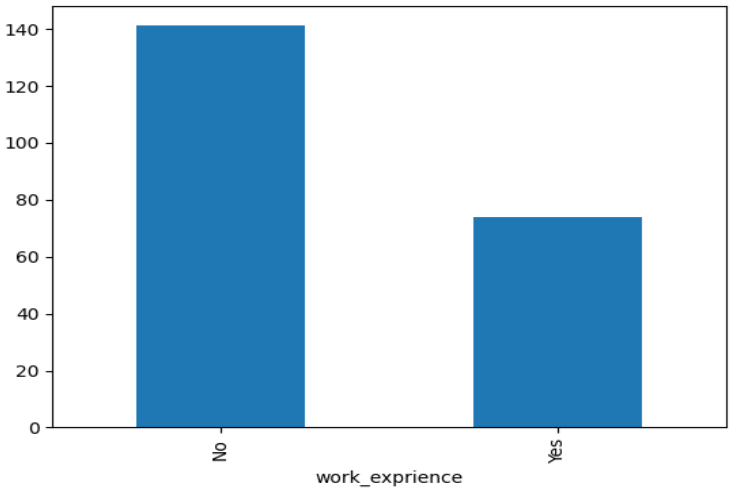



```
In [63]: df['specialisation'].count()
Out[63]: 215

In [64]: df['specialisation'].value_counts()
Out[64]: specialisation
         Mkt&Fin    120
         Mkt&HR      95
         Name: count, dtype: int64
```

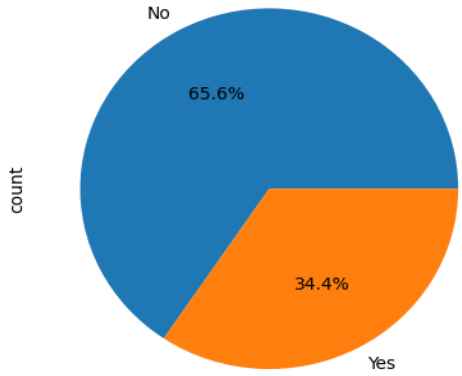
```
In [65]: df['specialisation'].value_counts().plot(kind='bar')
Out[65]: <Axes: xlabel='specialisation'>
```




```
In [67]: df['work_exprience'].count()
Out[67]: 215

In [68]: df['work_exprience'].value_counts()
Out[68]: work_exprience
         No     141
         Yes     74
         Name: count, dtype: int64

In [69]: df['work_exprience'].value_counts().plot(kind='bar')
Out[69]: <Axes: xlabel='work_exprience'>
```

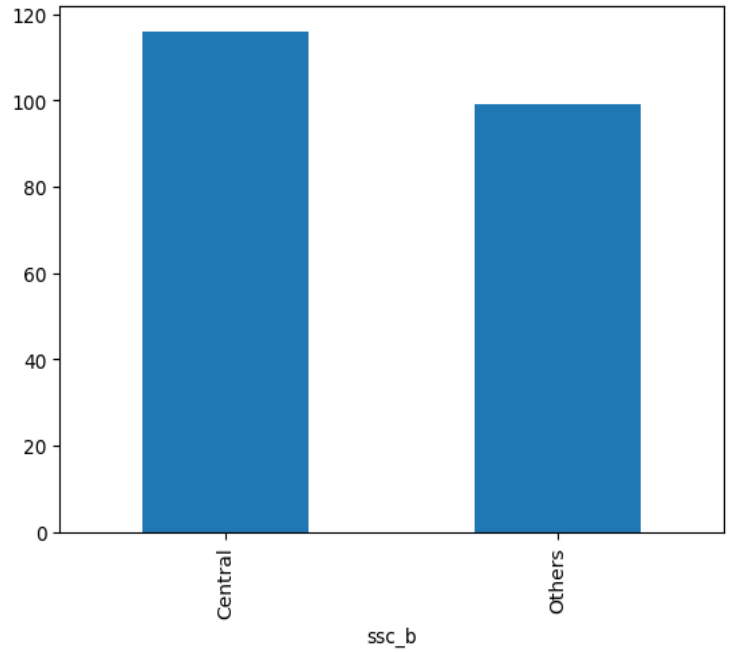


```
In [70]: df['work_exprience'].value_counts().plot(kind='pie',autopct='%0.1f%%')
Out[70]: <Axes: ylabel='count'>
```



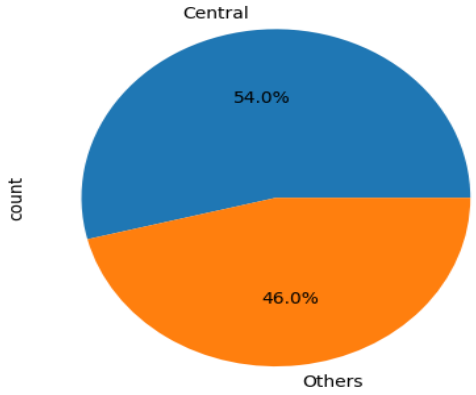

```
In [71]:    df['ssc_b'].count()
Out[71]: 215

In [72]: df['ssc_b'].value_counts()
Out[72]: ssc_b
         Central    116
         Others      99
         Name: count, dtype: int64
```

```
In [73]: df['ssc_b'].value_counts().plot(kind='bar')
Out[73]: <Axes: xlabel='ssc_b'>
```



In [74]:
```
df['ssc_b'].value_counts().plot(kind='pie',autopct='%0.1f%%')
```

Out[74]: <Axes: ylabel='count'>



In [75]:
```
df['hsc_s'].count()
```

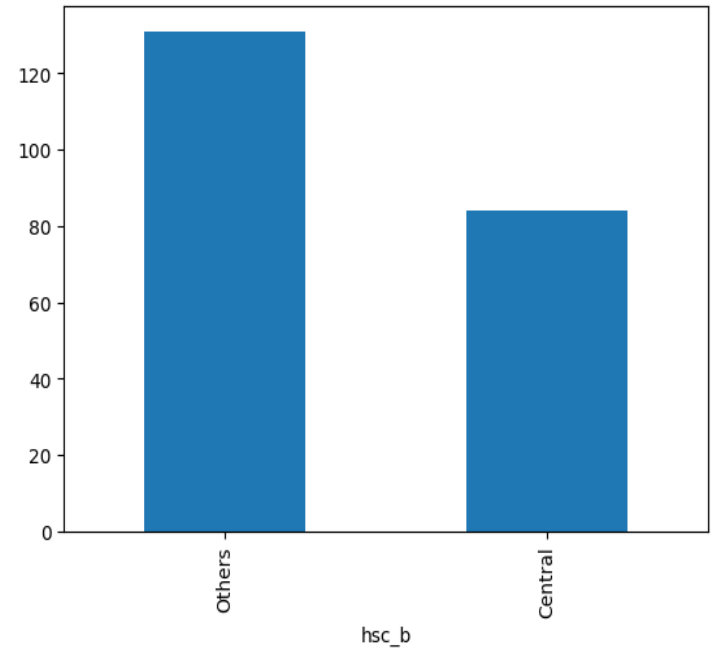
Out[75]: 215

In [76]:
```
df['hsc_b'].value_counts()
```

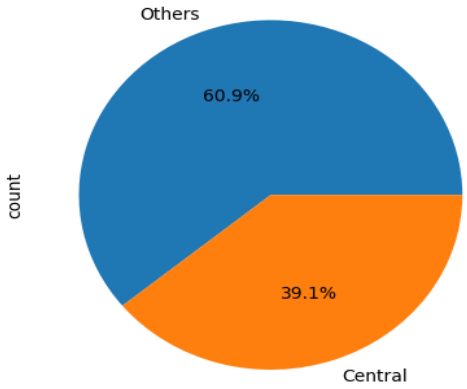
Out[76]:
```
hsc_b
Others     131
Central     84
Name: count, dtype: int64
```

In [77]:
```
df['hsc_b'].value_counts().plot(kind='bar')
```

Out[77]: <Axes: xlabel='hsc_b'>

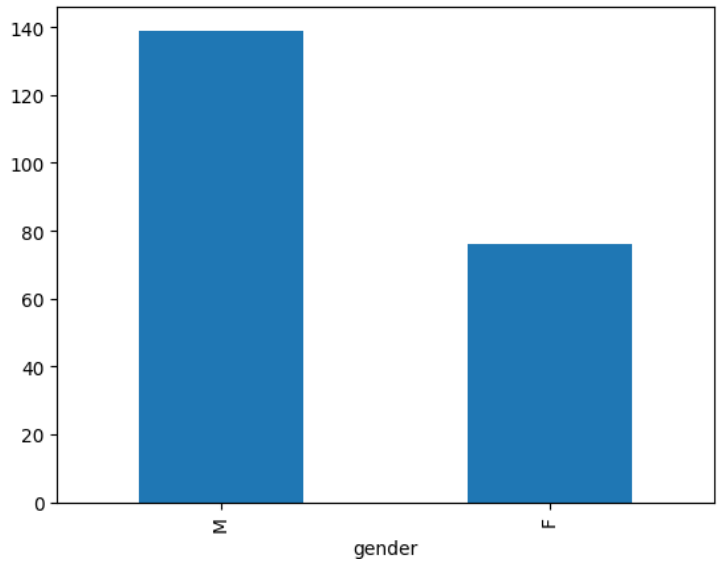


```
In [78]: df['hsc_b'].value_counts().plot(kind='pie',autopct='%0.1f%%')
Out[78]: <Axes: ylabel='count'>
```



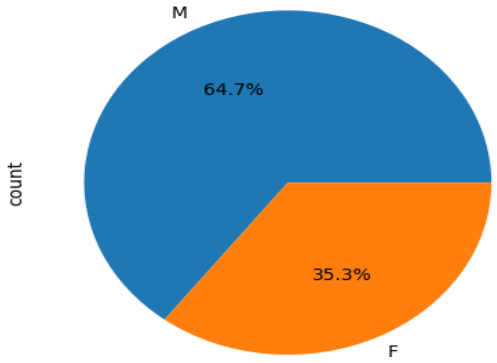

```
In [79]: df['gender'].count()
Out[79]: 215

In [80]: df['gender'].value_counts()
Out[80]: gender
         M    139
         F     76
         Name: count, dtype: int64
```

```
In [81]: df['gender'].value_counts().plot(kind='bar')
Out[81]: <Axes: xlabel='gender'>
```



In [82]:
```
df['gender'].value_counts().plot(kind='pie',autopct='%0.1f%%')
```

Out[82]: <Axes: ylabel='count'>



In [83]:
```
df.head()
```

Out[83]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	work_exprience	entrance test	specialisation	mba_p	status	salary
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.000000
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.000000
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.000000
3	4	M	56.00	Central	52.00	Central	Science	52.00	Sci&Tech	No	66.0	Mkt&HR	59.43	Not Placed	288655.405405
4	5	M	85.80	Central	73.60	Central	Commerce	73.30	Comm&Mgmt	No	96.8	Mkt&Fin	55.50	Placed	425000.000000

bivarient analysis

In [84]:
```
pd.crosstab(df['status'],df['gender'])              #Bivarient analysis of two categorical coloumns
```

Out[84]:

gender	F	M
status		
Not Placed	28	39
Placed	48	100

In [85]:
```
pd.crosstab(df['status'],df['gender'],normalize='columns')*100
```

Out[85]:

gender	F	M
status		
Not Placed	36.842105	28.057554
Placed	63.157895	71.942446

```
In [90]: pd.crosstab(df['status'],df['hsc_b'])
```

Out[90]:

status	Central	Others
Not Placed	27	40
Placed	57	91

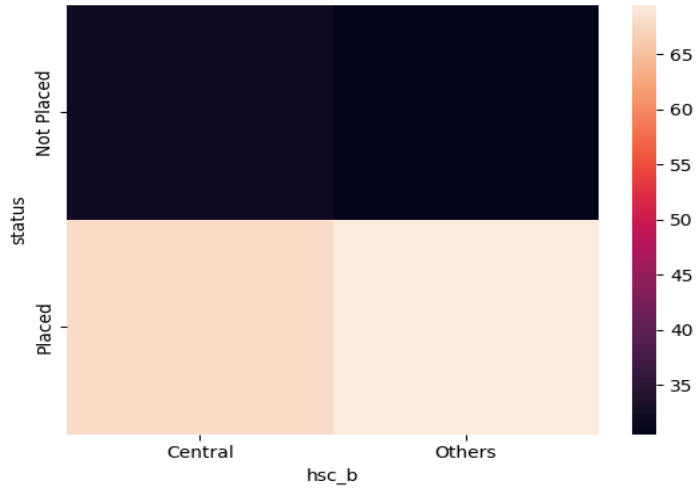
```
In [91]: pd.crosstab(df['status'],df['hsc_b'],normalize='columns')*100
```

Out[91]:

status	Central	Others
Not Placed	32.142857	30.534351
Placed	67.857143	69.465649

```
In [92]: sns.heatmap(pd.crosstab(df['status'],df['hsc_b'],normalize='columns')*100)
```

Out[92]: <Axes: xlabel='hsc_b', ylabel='status'>




```
In [93]: pd.crosstab(df['status'],df['hsc_s'])
```

Out[93]:

status	Arts	Commerce	Science
Not Placed	5	34	28
Placed	6	79	63

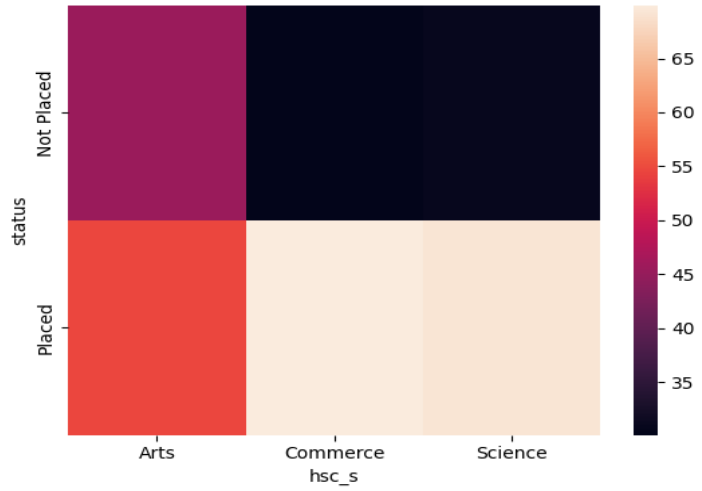
```
In [94]: pd.crosstab(df['status'],df['hsc_s'],normalize='columns')*100
```

Out[94]:

status	Arts	Commerce	Science
Not Placed	45.454545	30.088496	30.769231
Placed	54.545455	69.911504	69.230769

In [95]:
```
sns.heatmap(pd.crosstab(df['status'],df['hsc_s'],normalize='columns')*100)
```

Out[95]: <Axes: xlabel='hsc_s', ylabel='status'>



In [96]:
```
pd.crosstab(df['status'],df['degree_t'])
```

Out[96]:

degree_t	Comm&Mgmt	Others	Sci&Tech
status			
Not Placed	43	6	18
Placed	102	5	41

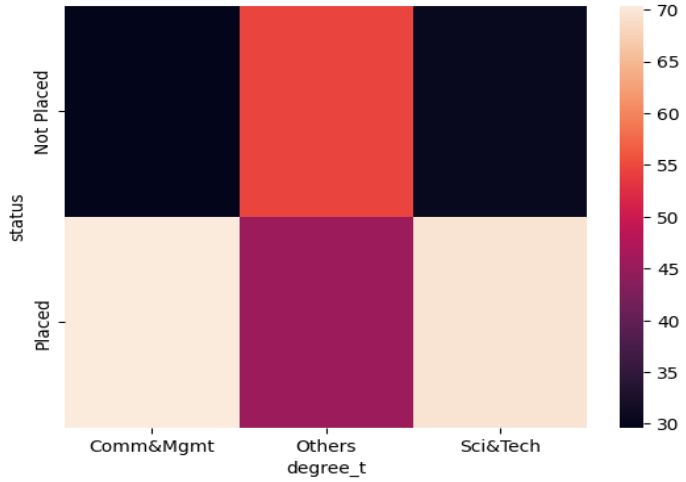
In [97]:
```
pd.crosstab(df['status'],df['degree_t'],normalize='columns')*100
```

Out[97]:

degree_t	Comm&Mgmt	Others	Sci&Tech
status			
Not Placed	29.655172	54.545455	30.508475
Placed	70.344828	45.454545	69.491525

In [98]:
```
sns.heatmap(pd.crosstab(df['status'],df['degree_t'],normalize='columns')*100)
```

Out[98]: <Axes: xlabel='degree_t', ylabel='status'>



In [99]:
```
pd.crosstab(df['status'],df['work_exprience'])
```

Out[99]:

status	work_exprience No	Yes
Not Placed	57	10
Placed	84	64

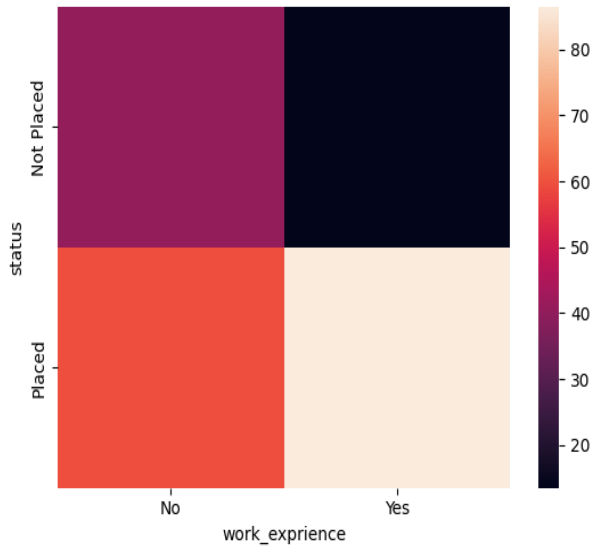
In [100]:
```
pd.crosstab(df['status'],df['work_exprience'],normalize='columns')*100
```

Out[100]:

status	work_exprience No	Yes
Not Placed	40.425532	13.513514
Placed	59.574468	86.486486

In [101]:
```
sns.heatmap(pd.crosstab(df['status'],df['work_exprience'],normalize='columns')*100)
```

Out[101]: <Axes: xlabel='work_exprience', ylabel='status'>



In [102]:
```
pd.crosstab(df['status'],df['specialisation'])
```

Out[102]:

status	specialisation Mkt&Fin	Mkt&HR
Not Placed	25	42
Placed	95	53

Placed	95	53

In [103]:
```
pd.crosstab(df['status'],df['specialisation'],normalize='columns')*100
```

Out[103]:

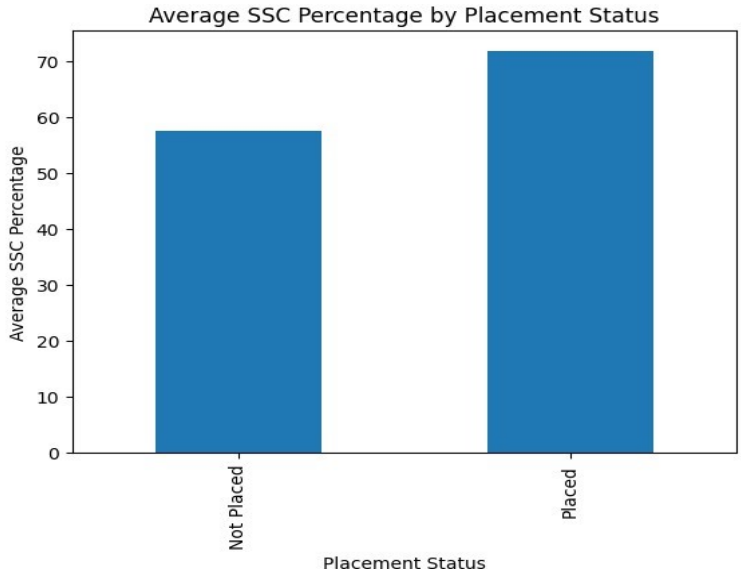
specialisation	Mkt&Fin	Mkt&HR
status		
Not Placed	20.833333	44.210526
Placed	79.166667	55.789474

In [104]:
```
sns.heatmap(pd.crosstab(df['status'],df['specialisation'],normalize='columns')*100)
```

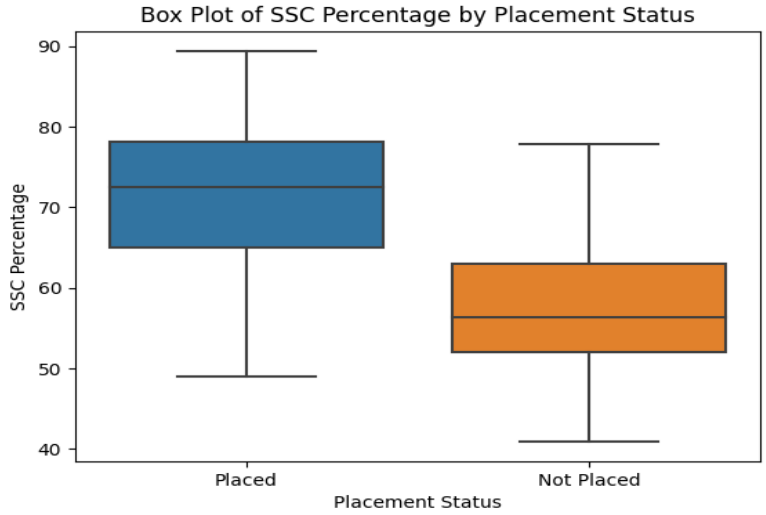
Out[104]: <Axes: xlabel='specialisation', ylabel='status'>

Bivarient Analysis between categorical and numerical data

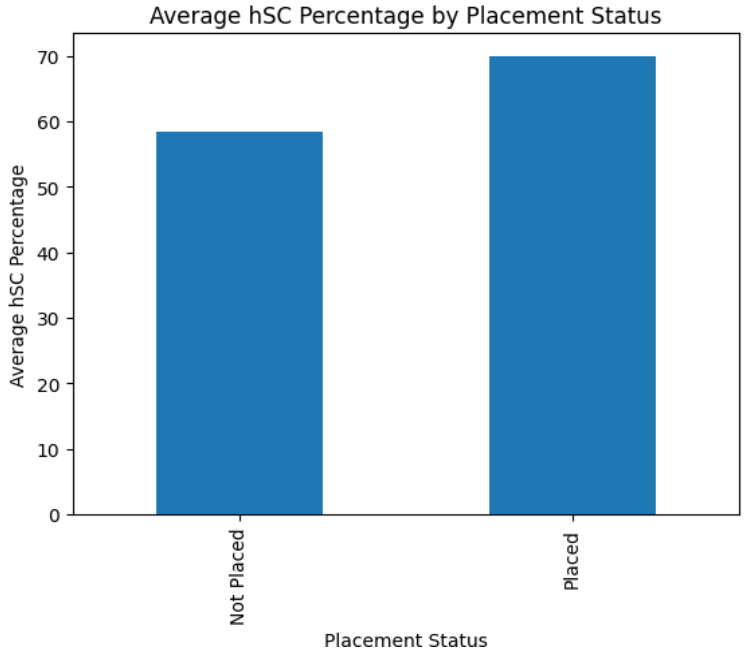
In [105]:
```
# Bar chart to compare the average SSC percentage for each placement status
df.groupby('status')['ssc_p'].mean().plot(kind='bar')
plt.title('Average SSC Percentage by Placement Status')
plt.xlabel('Placement Status')
plt.ylabel('Average SSC Percentage')
plt.show()
```



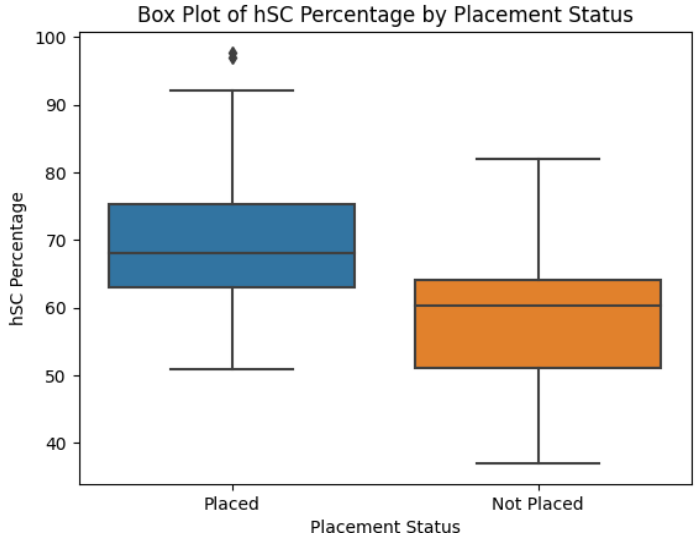
In [106]:
```
# Box plot to compare SSC percentage for each placement status
sns.boxplot(x='status', y='ssc_p', data=df)
plt.title('Box Plot of SSC Percentage by Placement Status')
plt.xlabel('Placement Status')
plt.ylabel('SSC Percentage')
plt.show()
```



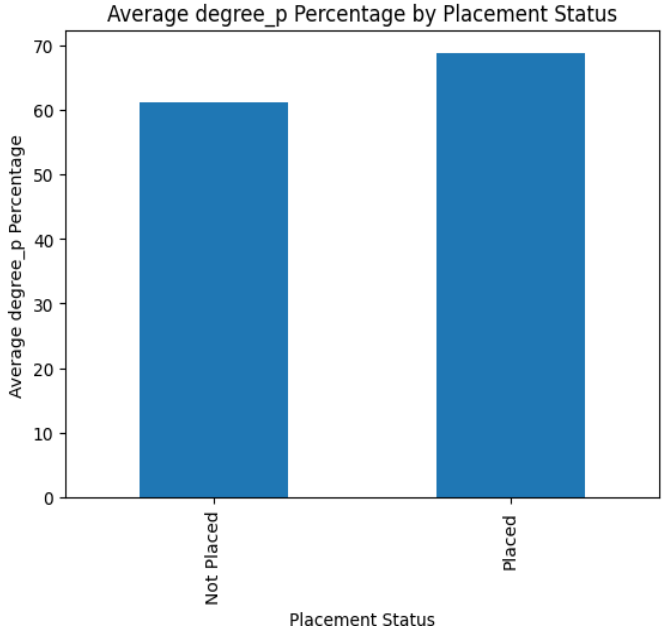
In [107]:
```
# Bar chart to compare the average hSC percentage for each placement status
df.groupby('status')['hsc_p'].mean().plot(kind='bar')
plt.title('Average hSC Percentage by Placement Status')
plt.xlabel('Placement Status')
plt.ylabel('Average hSC Percentage')
plt.show()
```



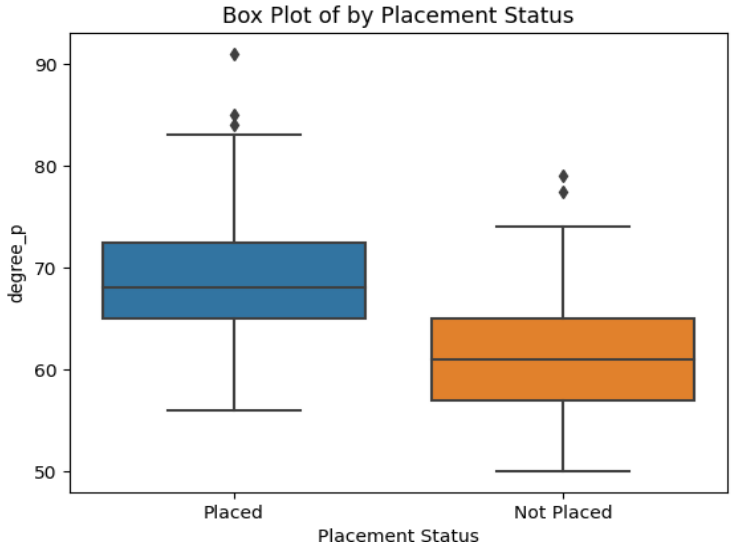
In [108]:
```
# Box plot to compare SSC percentage for each placement status
sns.boxplot(x='status', y='hsc_p', data=df)
plt.title('Box Plot of hSC Percentage by Placement Status')
plt.xlabel('Placement Status')
plt.ylabel('hSC Percentage')
plt.show()
```



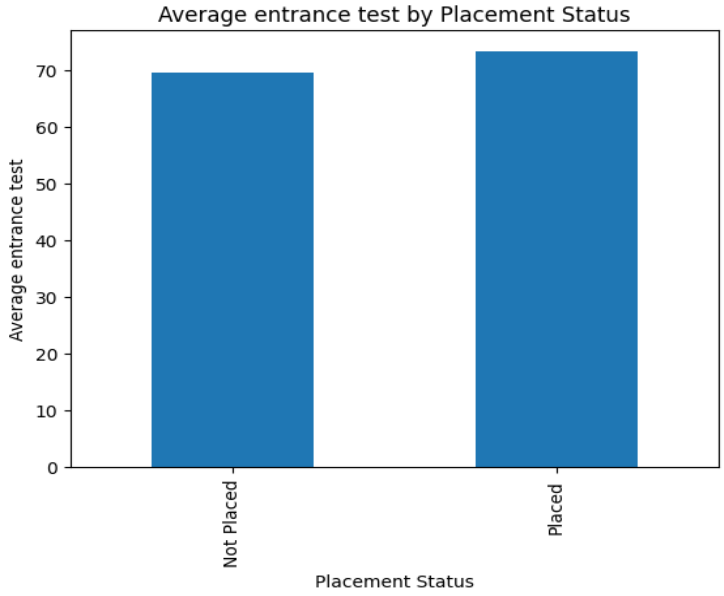
In [109]:
```
# Bar chart to compare the average degree_p percentage for each placement status
df.groupby('status')['degree_p'].mean().plot(kind='bar')
plt.title('Average degree_p Percentage by Placement Status')
plt.xlabel('Placement Status')
plt.ylabel('Average degree_p Percentage')
plt.show()
```



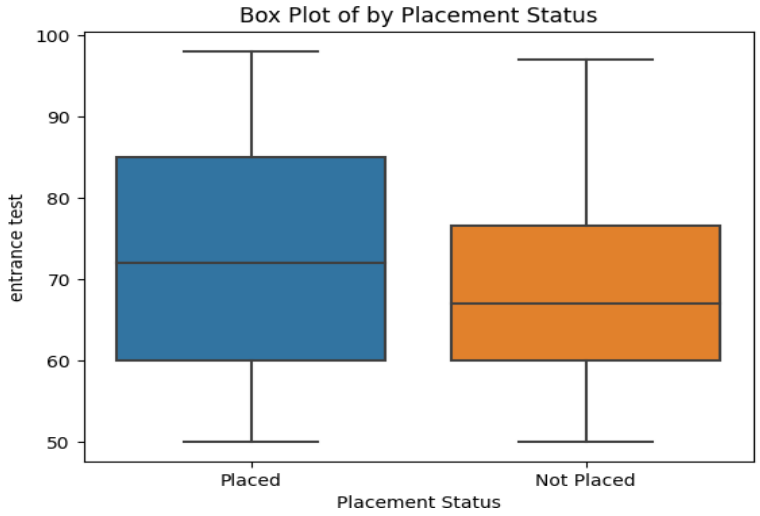
In [110]:
```
# Box plot to compare degree_p for each placement status
sns.boxplot(x='status', y='degree_p', data=df)
plt.title('Box Plot of by Placement Status')
plt.xlabel('Placement Status')
plt.ylabel('degree_p')
plt.show()
```



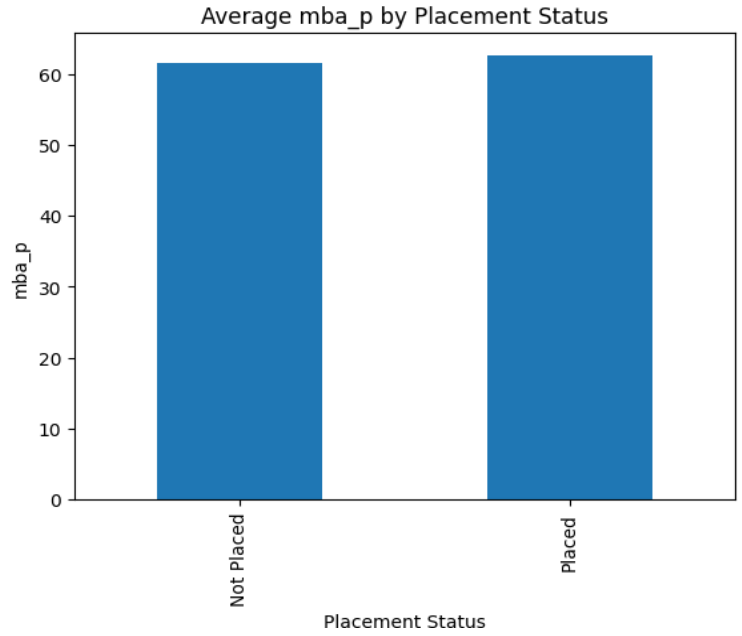
In [111]:
```
# Bar chart to compare the averageentrance entrance test for each placement status
df.groupby('status')['entrance test'].mean().plot(kind='bar')
plt.title('Average entrance test by Placement Status')
plt.xlabel('Placement Status')
plt.ylabel('Average entrance test')
plt.show()
```



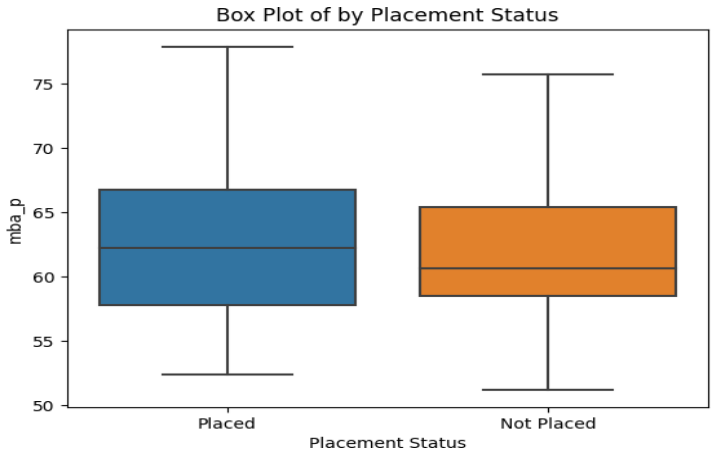
In [112]:
```
# Box plot to compare degree_p for each placement status
sns.boxplot(x='status', y='entrance test', data=df)
plt.title('Box Plot of by Placement Status')
plt.xlabel('Placement Status')
plt.ylabel('entrance test')
plt.show()
```



In [113]:
```
# Bar chart to compare the averageentrance mba_p for each placement status
df.groupby('status')['mba_p'].mean().plot(kind='bar')
plt.title('Average mba_p by Placement Status')
plt.xlabel('Placement Status')
plt.ylabel('mba_p')
plt.show()
```



In [114]:
```
# Box plot to compare mba_p for each placement status
sns.boxplot(x='status', y='mba_p', data=df)
plt.title('Box Plot of by Placement Status')
plt.xlabel('Placement Status')
plt.ylabel('mba_p')
plt.show()
```



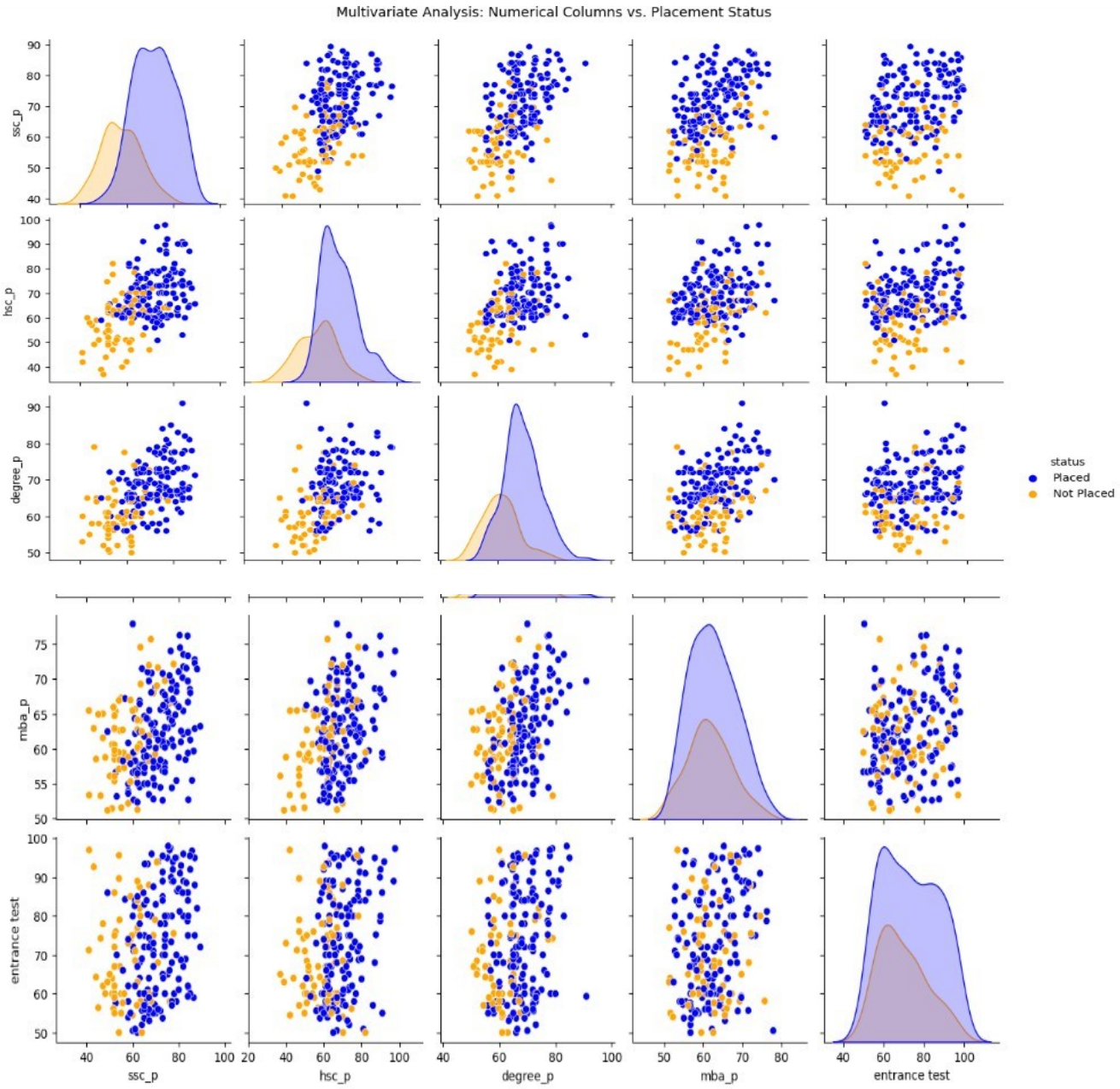
In [115]:
```
df.head(3)
```

Out[115]:

	sl_no	gender	ssc_p	ssc_b	hsc_p	hsc_b	hsc_s	degree_p	degree_t	work_exprience	entrance test	specialisation	mba_p	status	salary
0	1	M	67.00	Others	91.00	Others	Commerce	58.00	Sci&Tech	No	55.0	Mkt&HR	58.80	Placed	270000.0
1	2	M	79.33	Central	78.33	Others	Science	77.48	Sci&Tech	Yes	86.5	Mkt&Fin	66.28	Placed	200000.0
2	3	M	65.00	Central	68.00	Central	Arts	64.00	Comm&Mgmt	No	75.0	Mkt&Fin	57.80	Placed	250000.0

Multivarient Analysis

In [116]:
```
# Select numerical columns (excluding 'salary') for multivariate analysis       #there Are use in paire plot graph
numerical_columns = ['ssc_p', 'hsc_p', 'degree_p', 'mba_p', 'entrance test']

# Pairplot to visualize relationships between numerical columns based on 'status'
sns.pairplot(df, hue='status', vars=numerical_columns, diag_kind='kde', palette={'Placed': 'blue', 'Not Placed': 'orange'})
plt.suptitle('Multivariate Analysis: Numerical Columns vs. Placement Status', y=1.02)
plt.show()
```



In [117]:
```
numeric_columns = df.select_dtypes(include=[np.number])
correlation_matrix = numeric_columns.corr()
```

In [118]:
```
numeric_columns.head(3)
```

Out[118]:

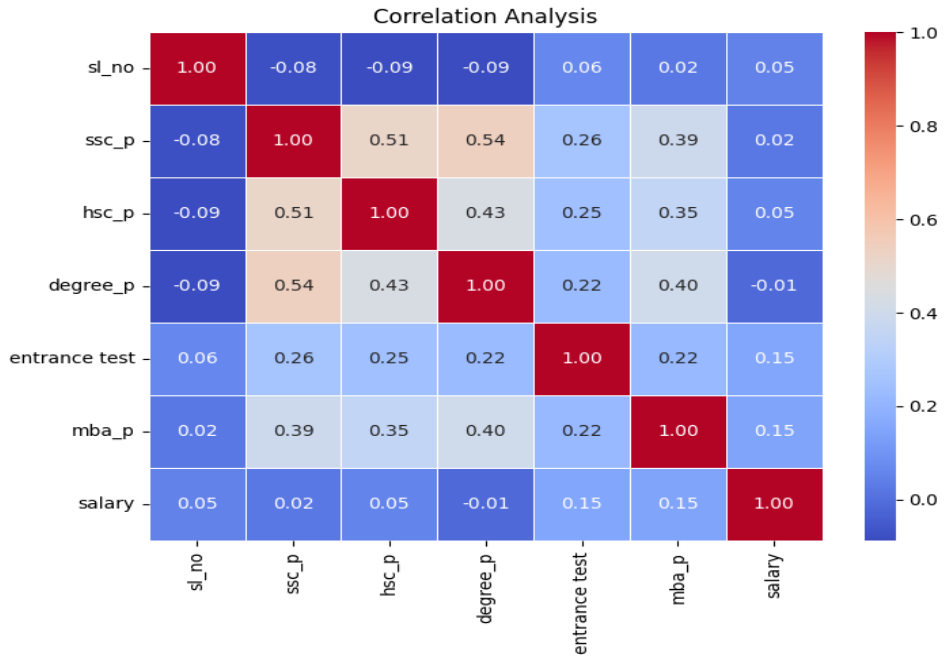
	sl_no	ssc_p	hsc_p	degree_p	entrance test	mba_p	salary
0	1	67.00	91.00	58.00	55.0	58.80	270000.0
1	2	79.33	78.33	77.48	86.5	66.28	200000.0
2	3	65.00	68.00	64.00	75.0	57.80	250000.0

In [119]:
```
correlation_matrix
```

Out[119]:

	sl_no	ssc_p	hsc_p	degree_p	entrance test	mba_p	salary
sl_no	1.000000	-0.078155	-0.085711	-0.088281	0.063636	0.022327	0.051550
ssc_p	-0.078155	1.000000	0.511472	0.538404	0.261993	0.388478	0.023571
hsc_p	-0.085711	0.511472	1.000000	0.434206	0.245113	0.354823	0.054506
degree_p	-0.088281	0.538404	0.434206	1.000000	0.224470	0.402364	-0.014148
entrance test	0.063636	0.261993	0.245113	0.224470	1.000000	0.218055	0.152829
mba_p	0.022327	0.388478	0.354823	0.402364	0.218055	1.000000	0.146324
salary	0.051550	0.023571	0.054506	-0.014148	0.152829	0.146324	1.000000

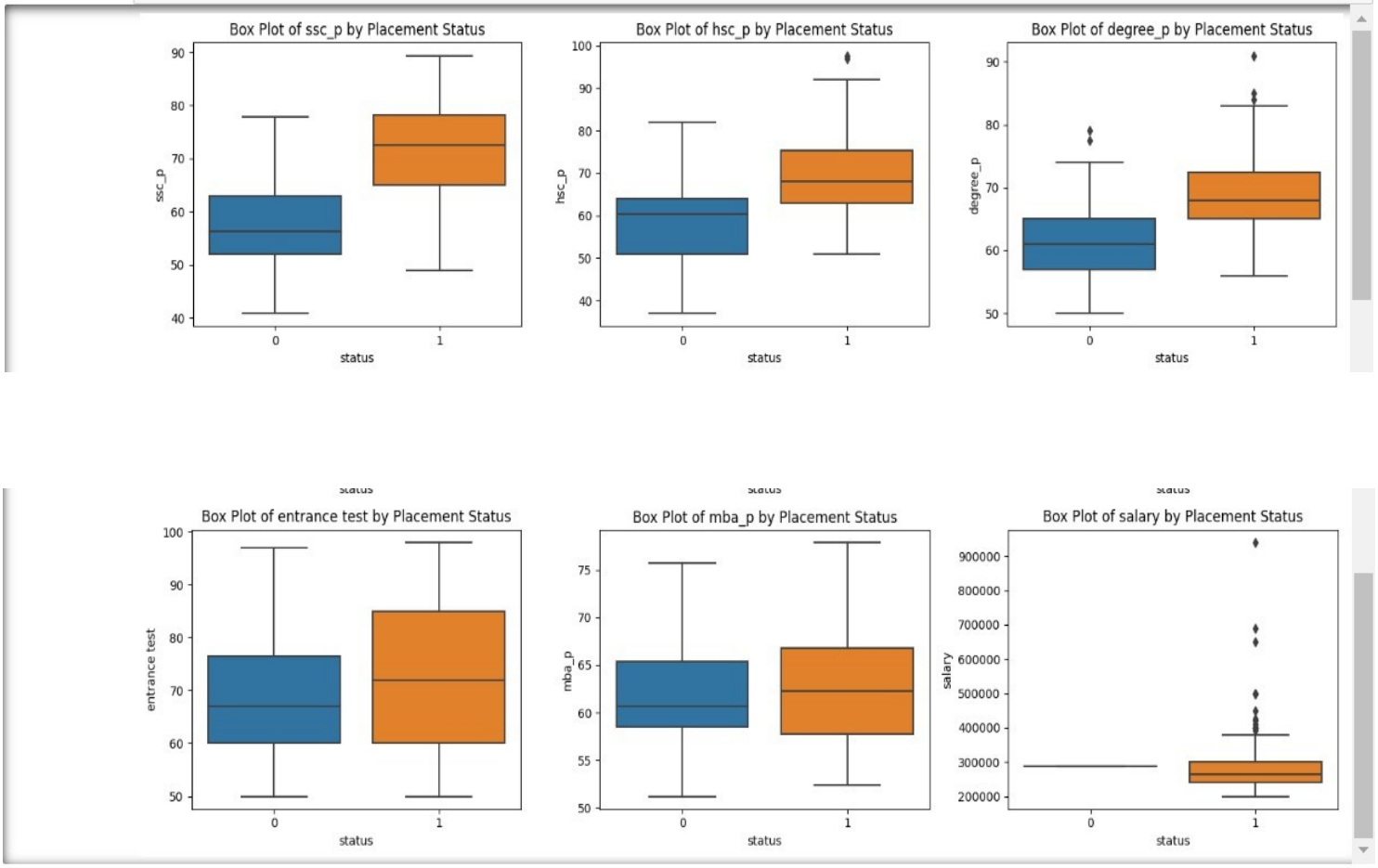
In [120]:
```
plt.figure(figsize=(8, 6))
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f", linewidths=.5)
plt.title("Correlation Analysis")
plt.show()
```



Secondary education percentage (ssc_p), higher secondary education percentage (hsc_p), and undergraduate degree percentage (degree_p) show a clear positive correlation with placement status. Entrance test scores and MBA percentages seem to have a weaker correlation.

Further Data Exploration:

In [123]:
```
# Box plots for numerical columns
numerical_columns = ['ssc_p', 'hsc_p', 'degree_p', 'entrance test', 'mba_p', 'salary']

plt.figure(figsize=(15, 8))
for i, column in enumerate(numerical_columns, 1):
    plt.subplot(2, 3, i)
    sns.boxplot(x='status', y=column, data=df)
    plt.title(f'Box Plot of {column} by Placement Status')

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

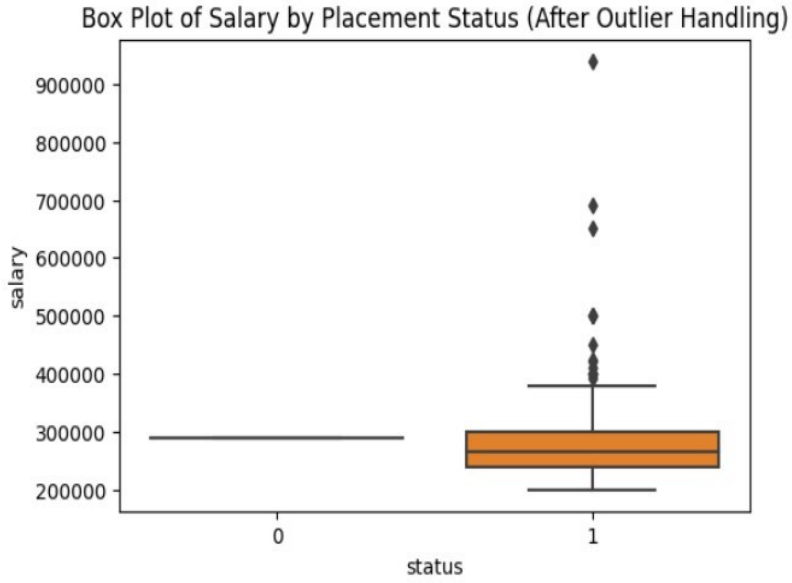


Secondary education percentage (ssc_p), higher secondary education percentage (hsc_p), and undergraduate degree percentage (degree_p) show a clear positive correlation with placement status.

Entrance test scores and MBA percentages seem to have a weaker correlation.

Salary, not surprisingly, strongly correlates with placement status.

In [124]:
```
# Re-run the box plot for 'salary' after handling outliers
plt.figure(figsize=(6, 4))
sns.boxplot(x='status', y='salary', data=df)
plt.title('Box Plot of Salary by Placement Status (After Outlier Handling)')
plt.show()
```



Hypothesis testing

In [125]:
```
from scipy.stats import chi2_contingency

# Create a contingency table
contingency_table = pd.crosstab(df['status'], df['gender'])

# Perform Chi-square test
chi2, p, _, _ = chi2_contingency(contingency_table)

# Print the results
print(f"Chi-square value: {chi2}")
print(f"P-value: {p}")

# Interpret the results
if p < 0.05:
    print("There is a significant difference in placement based on gender.")
else:
    print("There is no significant difference in placement based on gender.")
```

```
Chi-square value: 1.3817539668505106
P-value: 0.23980260881037568
There is no significant difference in placement based on gender.
```

Machine Learning Model Evaluation :

In [128]:
```
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [129]:
```
# Import necessary libraries
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline

# Assuming 'df' is your DataFrame with the processed data

# Separate features and target variable
X = df.drop('status', axis=1)  # Features
y = df['status']  # Target variable

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

# Define numerical and categorical features
numerical_features = X.select_dtypes(include=['float64', 'int64']).columns
categorical_features = X.select_dtypes(include=['object']).columns

# Create a column transformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', 'passthrough', numerical_features),
        ('cat', OneHotEncoder(), categorical_features)
    ])

# Create a pipeline with the preprocessor and the RandomForestClassifier
pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                           ('classifier', RandomForestClassifier(random_state=42))])

# Train the model on the training set
pipeline.fit(X_train, y_train)
```

```
                           ('classifier', RandomForestClassifier(random_state=42))])

# Train the model on the training set
pipeline.fit(X_train, y_train)

# Make predictions on the testing set
y_pred = pipeline.predict(X_test)

# Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

# Print the results
print(f"Accuracy: {accuracy:.2f}")
print("\nConfusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(classification_rep)
```

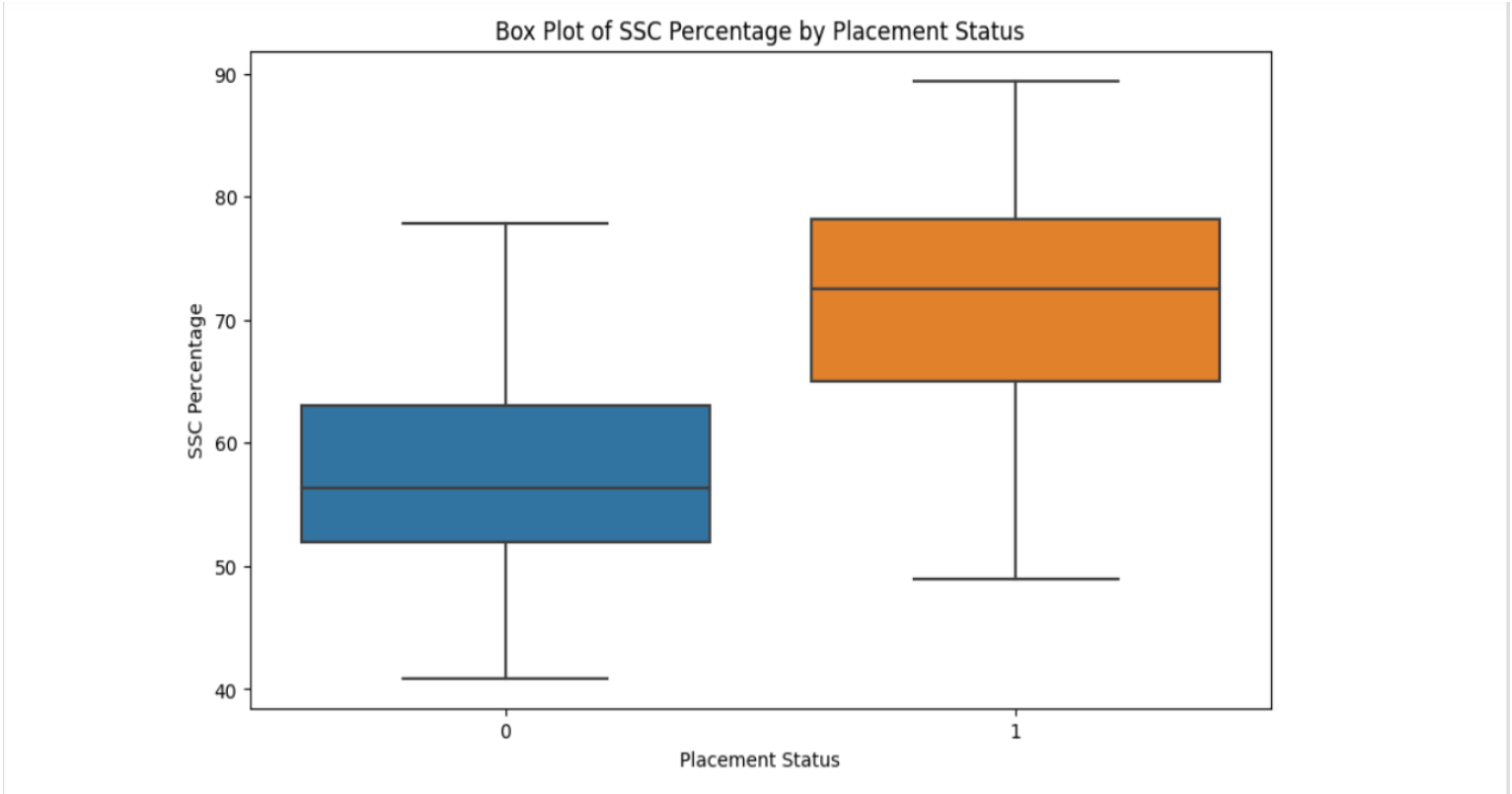
```
Accuracy: 0.93

Confusion Matrix:
[[10  2]
 [ 1 30]]

Classification Report:
              precision    recall  f1-score   support

           0       0.91      0.83      0.87        12
           1       0.94      0.97      0.95        31

    accuracy                           0.93        43
   macro avg       0.92      0.90      0.91        43
weighted avg       0.93      0.93      0.93        43
```

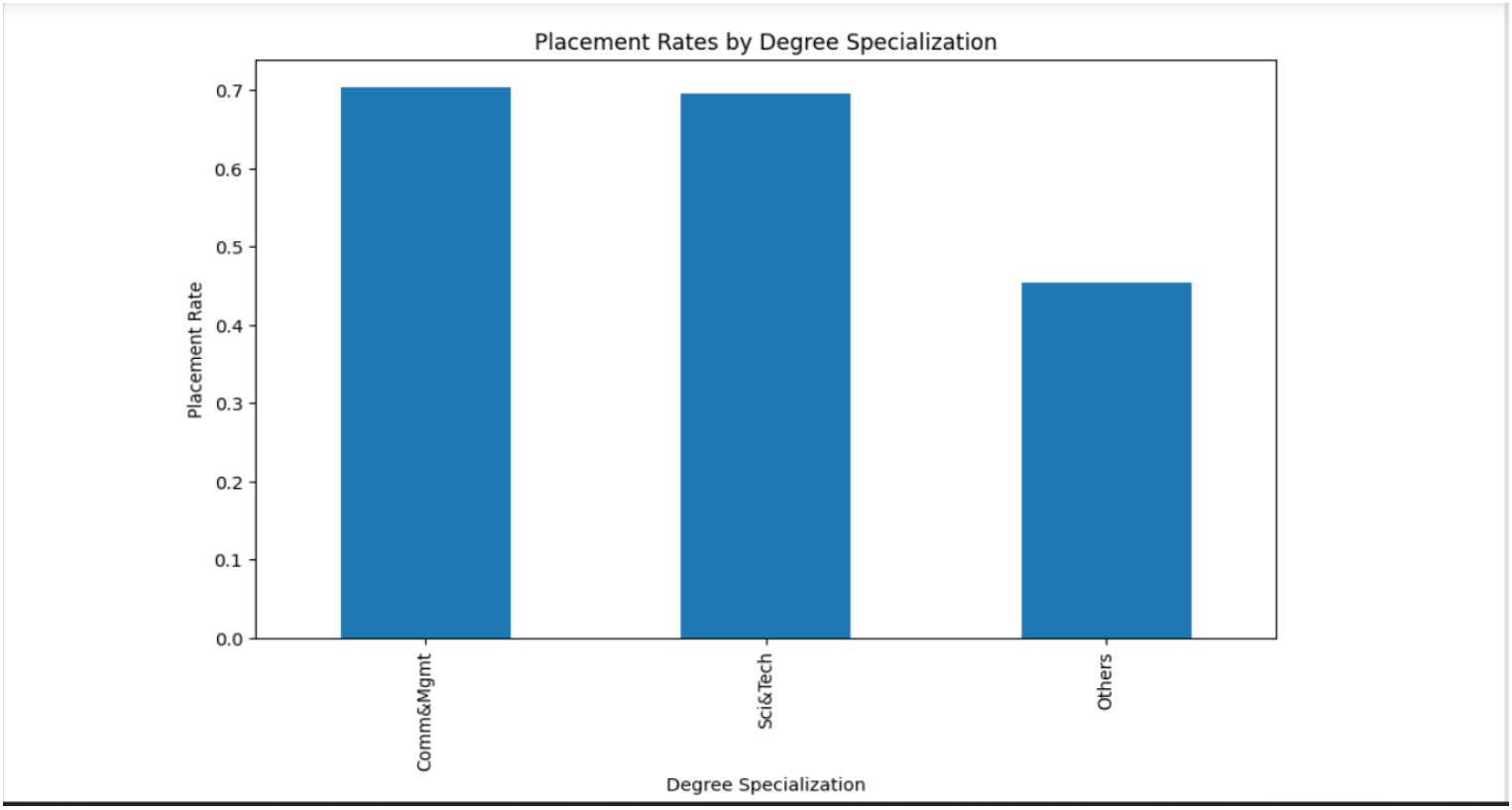
Fine-Tuning the Model:

In [130]:

```python
# Import necessary libraries
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.preprocessing import OneHotEncoder
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report

# Assuming 'df' is your DataFrame with the processed data

# Separate features and target variable
X = df.drop('status', axis=1)  # Features
y = df['status']  # Target variable

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

# Define numerical and categorical features
numerical_features = X.select_dtypes(include=['float64', 'int64']).columns
categorical_features = X.select_dtypes(include=['object']).columns

# Create a column transformer
preprocessor = ColumnTransformer(
    transformers=[
        ('num', 'passthrough', numerical_features),
        ('cat', OneHotEncoder(), categorical_features)
    ])
```

```python
# Create the pipeline with the preprocessor and the RandomForestClassifier
pipeline = Pipeline(steps=[('preprocessor', preprocessor),
                           ('classifier', RandomForestClassifier(random_state=42))])

# Define the parameter grid to search
param_grid = {
    'classifier__n_estimators': [50, 100, 200],
    'classifier__max_depth': [None, 10, 20, 30],
    'classifier__min_samples_split': [2, 5, 10],
    'classifier__min_samples_leaf': [1, 2, 4]
}

# Create the GridSearchCV object
grid_search = GridSearchCV(pipeline, param_grid, cv=5, scoring='accuracy')

# Fit the grid search to the data
grid_search.fit(X_train, y_train)

# Get the best parameters and best model
best_params = grid_search.best_params_
best_model = grid_search.best_estimator_

# Evaluate the best model on the test set
y_pred_tuned = best_model.predict(X_test)

# Print the best parameters and evaluation metrics
print("Best Parameters:", best_params)
print("\nModel Evaluation After Fine-Tuning:")
print("Accuracy:", accuracy_score(y_test, y_pred_tuned))
print("Confusion Matrix:\n", confusion_matrix(y_test, y_pred_tuned))
print("Classification Report:\n", classification_report(y_test, y_pred_tuned))
```


Box Plot of SSC Percentage by Placement Status
SSC Percentage
90
80
70
60
50
40
0
1
Placement Status


Placement Rates by Degree Specialization
Placement Rate
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Comm&Mgmt
Sci&Tech
Others
Degree Specialization

8.1 Testing Method Used:

The project may have employed various testing methods to assess the performance of the implemented system or model. Common testing methods include:

- **Unit Testing** :- This involves testing individual components or functions of the system to ensure they work as intended.
- **Integration Testing:** Verifying that different modules or components of the system work together seamlessly.
- **System Testing**: Evaluating the system as a whole to ensure it meets the specified requirements.
- **User Acceptance Testing (UAT):** Involving end-users to validate that the system meets their expectations.

8.2 Test Case with Results:

- Test cases are scenarios designed to validate the functionality and performance of the system. Here are examples of test cases and their expected results:

a.) Test Case 1: Input validation for academic percentages :-

- Input: An out-of-range academic percentage (e.g., 120%)
- Expected Result: The system should reject the invalid input and provide an error message.

b.) Test Case 2: Machine Learning Model Evaluation:-

- Input: A set of features representing a student's academic and personal information.
- Expected Result: The model should predict the placement status accurately

c.) Test Case 3: Hypothesis Testing:-

- Input: Data related to factors like gender, work experience, etc.
- Expected Result: The hypothesis tests should provide insights into significant factors influencing placement.

8.3 Results Analysis:

- After conducting the tests, the results are analyzed to draw meaningful conclusions. For example:

Analysis of Machine Learning Model Results:

Accuracy, precision, recall, and F1-score can be analyzed to evaluate the performance of the placement prediction model.

Analysis of Hypothesis Testing:

P-values and chi-square statistics are interpreted to determine the significance of factors like gender, work experience, etc., in placement outcomes.

Analysis of System Testing:

Check whether the system meets the specified requirements. Identify and address any deviations or issues discovered during testing.

8. References

Dataset Source:

- Clearly state that the dataset used in your project was obtained from Kaggle.
- Provide details about the dataset, such as the name, purpose, and the kind of information it contains.

Jupyter Notebook Integration:

- Highlight the use of Jupyter Notebook as your development environment.
- Mention specific Jupyter Notebook features or tools you used during the project.

.

Project Overview:

- Briefly outline the purpose and goals of your campus recruitment project.
- Highlight the key features or functionalities you implemented using the dataset.

Methodology:

- Explain the steps you took in the Jupyter Notebook to clean, explore, and analyze the data.
- Mention any specific techniques, algorithms, or libraries you used.