k.likhith

2211cs010309

Question_Markes

The Excel file contains a single sheet named "SEM2 MID 1 - ALPHA". I'll now inspect the contents of this sheet to understand its structure.

Dataset Description: MIDMARKS.xlsx

The dataset contains 718 entries with 8 columns, representing students' mid-term exam scores in various subjects. However, some columns have missing values.

Columns Overview:

S.NO – Serial number of the student (some missing values).

SECTION – The section of the student (mostly "ALPHA").

DV – Marks in the subject DV.

M-II – Marks in Mathematics-II.

PP – Marks in PP.

BEEE – Marks in BEEE.

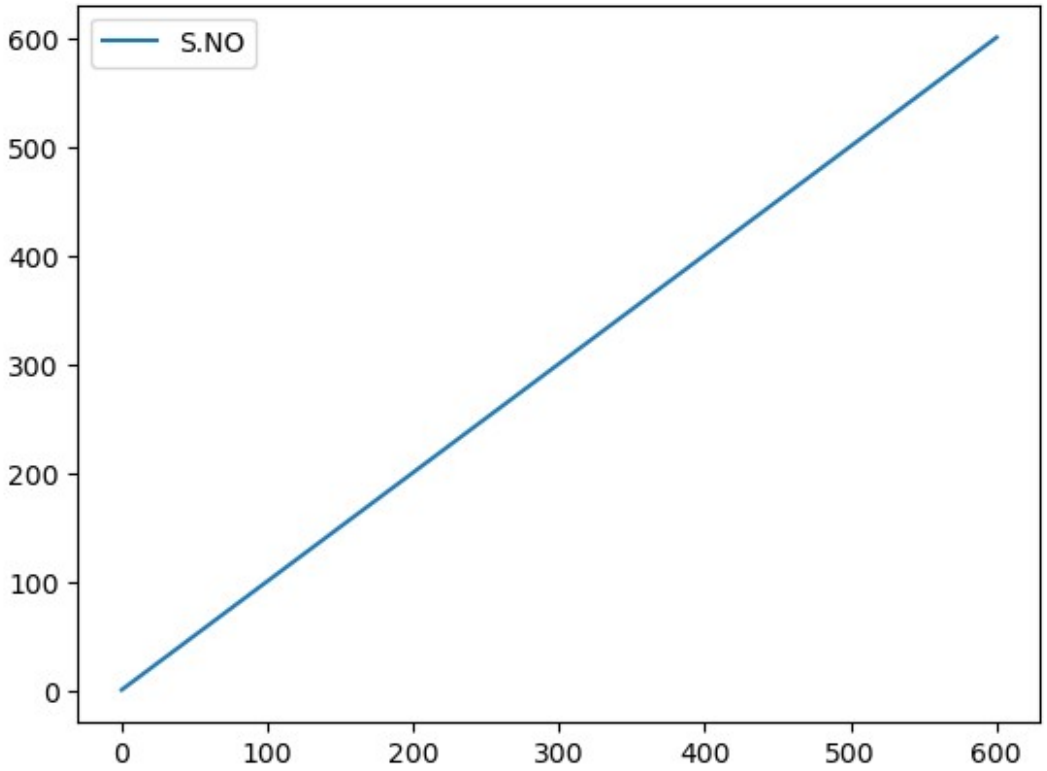
FL – Marks in FL.

FIMS – Marks in FIMS.

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_excel("MIDMARKS.xlsx")
df
```

	S.NO	SECTION	DV	M-II	PP	BEEE	FL	FIMS
0	1.0	ALPHA	12	0	17	9	19	15
1	2.0	ALPHA	19	12	16	16	18	3
2	3.0	ALPHA	18	14	18	18	18	16
3	4.0	ALPHA	15	9	19	17	19	15
4	5.0	ALPHA	18	17	19	19	20	18
..	...	...	..	...	..	...	..	...

```
713   NaN    ZETA  19    8   8   19  17   18
714   NaN    ZETA  12    1   7   10  20    8
715   NaN    ZETA  17    6  14   14  17   18
716   NaN    ZETA  12    1   6    7  15   12
717   NaN    ZETA  19   14  17   16  20   19

[718 rows x 8 columns]

df.plot()
plt.show()
```




```
df[df.DV==0]

Empty DataFrame
Columns: [S.NO, SECTION, DV, M-II, PP, BEEE, FL, FIMS]
Index: []

df[df.PP==0]

      S.NO SECTION  DV M-II PP BEEE  FL FIMS
88    89.0   ALPHA   2   17  0    3  15    2
394  395.0   GAMMA  20    8  0    0  13   13
487  488.0   OMEGA   1    5  0    A   A   AB
611    NaN     NaN   2    0  0    3  10    9
673    NaN    ZETA   2    A  0    0   2    1
```

```
df.DV.value_counts()
```

```
DV
20    103
17     79
16     74
18     69
15     63
19     60
11     43
12     41
14     41
13     30
10     26
9      20
6      12
5      11
8      11
A      10
7       8
2       6
4       4
1       3
3       1
MP      1
Name: count, dtype: int64
```

```
df['FL'] = df['FL'].fillna(0)
df=df.dropna()
```

```
df['S.NO'] = range(1, len(df) + 1)
df
```

```
C:\Users\likhi\AppData\Local\Temp\ipykernel_4912\1412173193.py:1:
SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation:
https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#
returning-a-view-versus-a-copy
  df['S.NO'] = range(1, len(df) + 1)
```

	S.NO	SECTION	DV	M-II	PP	BEEE	FL	FIMS
0	1	ALPHA	12	0	17	9	19	15
1	2	ALPHA	19	12	16	16	18	3
2	3	ALPHA	18	14	18	18	18	16
3	4	ALPHA	15	9	19	17	19	15
4	5	ALPHA	18	17	19	19	20	18
..	...	...	..	...	..	...	..	...

```
596   596   SIGMA  20   20  20   20  20   20
597   597   SIGMA  20   20  20   19  19   18
598   598   SIGMA  20   20  17   17  19   18
599   599   SIGMA  14   12  11    9  18   17
600   600   SIGMA  20   19  20   18  18   19

[600 rows x 8 columns]
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 600 entries, 0 to 600
Data columns (total 8 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   S.NO     600 non-null    int64
 1   SECTION  600 non-null    object
 2   DV       600 non-null    object
 3   M-II     600 non-null    object
 4   PP       600 non-null    object
 5   BEEE     600 non-null    object
 6   FL       600 non-null    object
 7   FIMS     600 non-null    object
dtypes: int64(1), object(7)
memory usage: 42.2+ KB
```

```
df_n = df.dropna(how='all')
print(df_n)
```

```
     S.NO SECTION  DV M-II  PP BEEE  FL FIMS
0       1   ALPHA  12    0  17    9  19   15
1       2   ALPHA  19   12  16   16  18    3
2       3   ALPHA  18   14  18   18  18   16
3       4   ALPHA  15    9  19   17  19   15
4       5   ALPHA  18   17  19   19  20   18
..    ...     ...  ..  ...  ..  ...  ..  ...
596   596   SIGMA  20   20  20   20  20   20
597   597   SIGMA  20   20  20   19  19   18
598   598   SIGMA  20   20  17   17  19   18
599   599   SIGMA  14   12  11    9  18   17
600   600   SIGMA  20   19  20   18  18   19

[600 rows x 8 columns]
```

```
import pandas as pd
file_path = 'MIDMARKS.xlsx'
df = pd.read_excel(file_path)
columns_to_sum = ["DV", "M-II", "PP", "BEEE", "FL", "FIMS"]
df[columns_to_sum] = df[columns_to_sum].apply(pd.to_numeric,
errors='coerce')
df["Total Value"] = df[columns_to_sum].sum(axis=1)
```

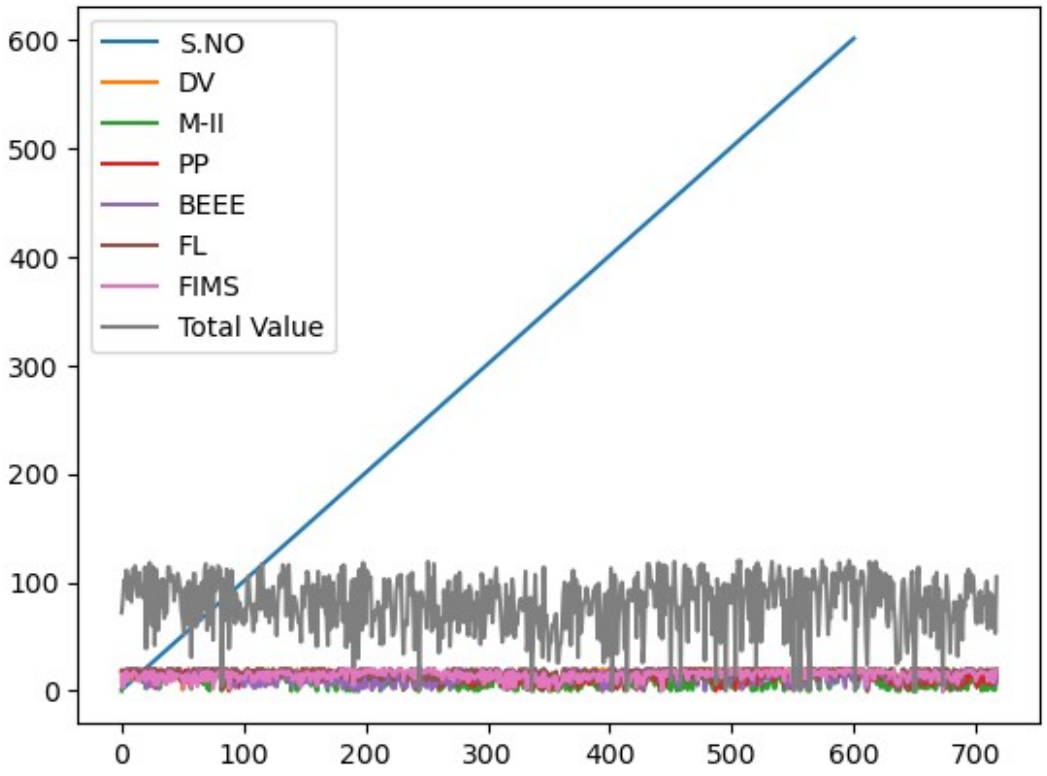
```
output_file_path = 'MIDMARKS_WITH_TOTAL.xlsx'
df.to_excel(output_file_path, index=False)
print(f"File with 'Total Value' column saved to {output_file_path}")
df
```

```
File with 'Total Value' column saved to MIDMARKS_WITH_TOTAL.xlsx
```

	S.NO	SECTION	DV	M-II	PP	BEEE	FL	FIMS	Total Value
0	1.0	ALPHA	12.0	0.0	17.0	9.0	19.0	15.0	72.0
1	2.0	ALPHA	19.0	12.0	16.0	16.0	18.0	3.0	84.0
2	3.0	ALPHA	18.0	14.0	18.0	18.0	18.0	16.0	102.0
3	4.0	ALPHA	15.0	9.0	19.0	17.0	19.0	15.0	94.0
4	5.0	ALPHA	18.0	17.0	19.0	19.0	20.0	18.0	111.0
..	...	...	...	...	...	...	...	...	...
713	NaN	ZETA	19.0	8.0	8.0	19.0	17.0	18.0	89.0
714	NaN	ZETA	12.0	1.0	7.0	10.0	20.0	8.0	58.0
715	NaN	ZETA	17.0	6.0	14.0	14.0	17.0	18.0	86.0
716	NaN	ZETA	12.0	1.0	6.0	7.0	15.0	12.0	53.0
717	NaN	ZETA	19.0	14.0	17.0	16.0	20.0	19.0	105.0

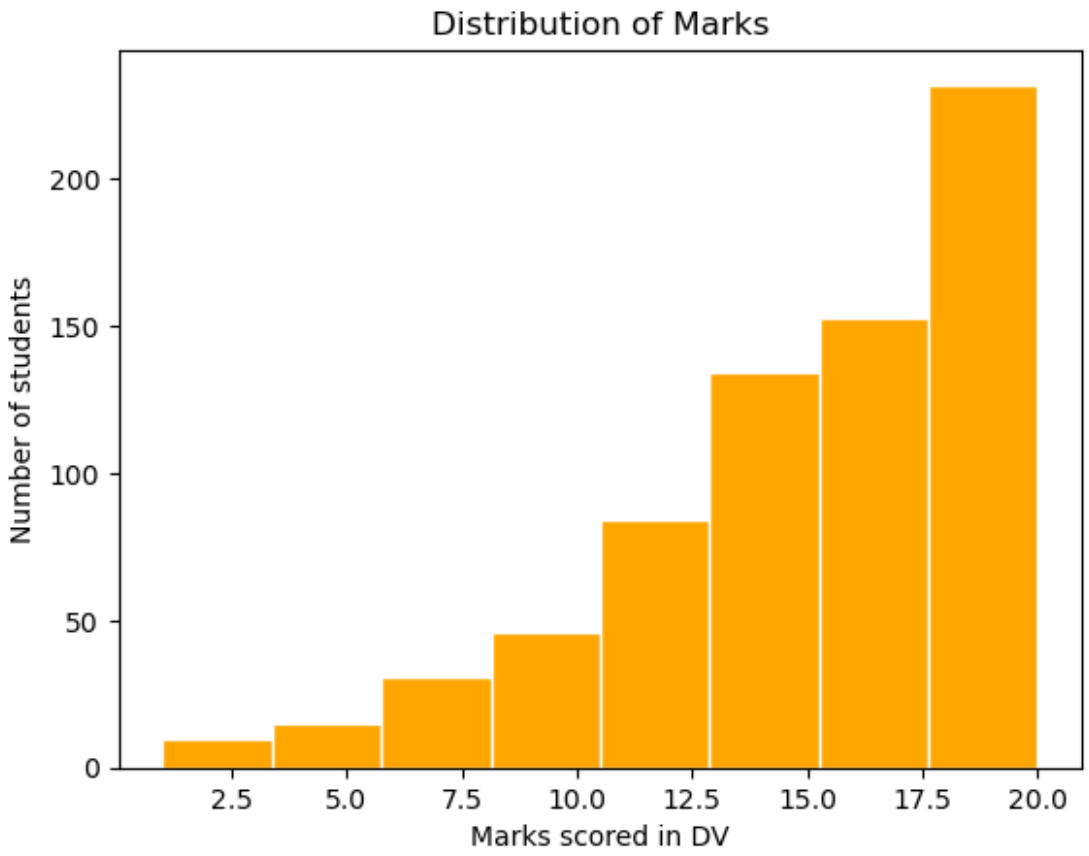
```
[718 rows x 9 columns]
```

```
import matplotlib.pyplot as plt
df.plot()
plt.show()
```



```
import pandas as pd
import matplotlib.pyplot as plt
print(df['DV'])
df['DV'] = pd.to_numeric(df['DV'], errors='coerce')
df = df.dropna(subset=['DV'])
plt.hist(df['DV'], color='orange', edgecolor='white', bins=8)
plt.xlabel("Marks scored in DV")
plt.ylabel("Number of students")
plt.title("Distribution of Marks")
plt.show()
```

```
0      12.0
1      19.0
2      18.0
3      15.0
4      18.0
       ...
713    19.0
714    12.0
715    17.0
716    12.0
717    19.0
Name: DV, Length: 718, dtype: float64
```



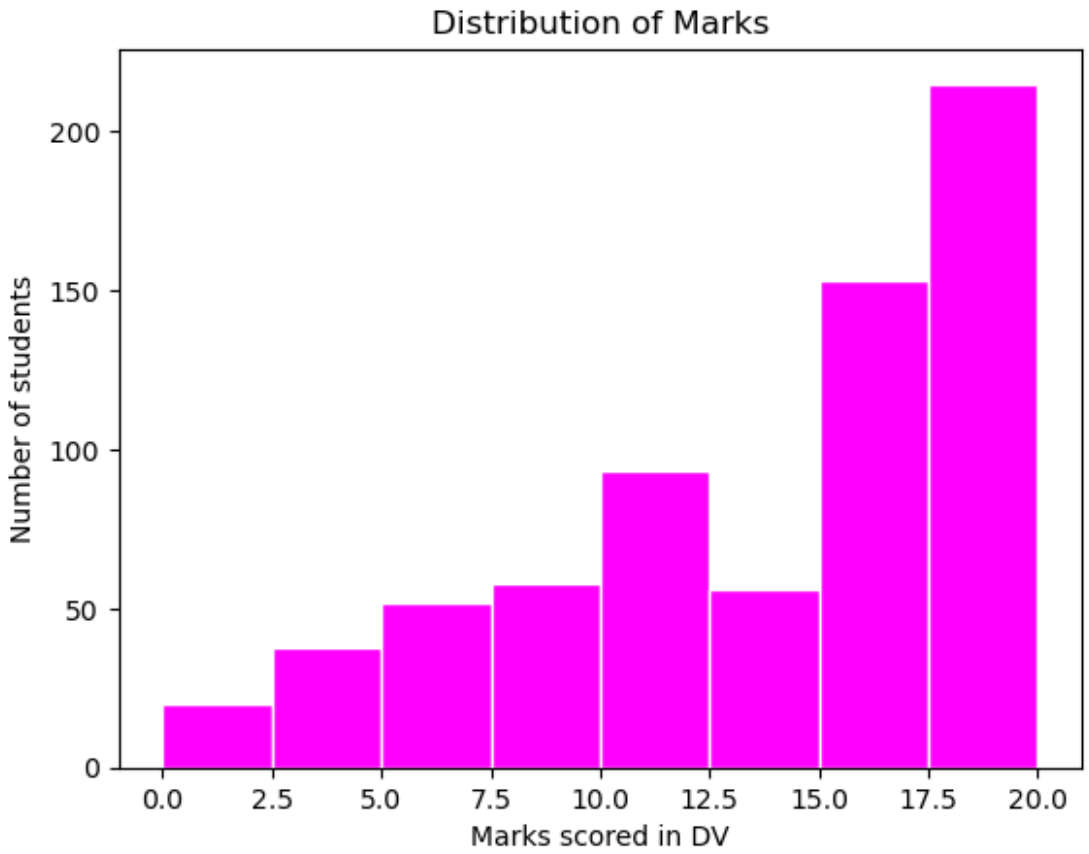
```python
import pandas as pd


df = pd.read_excel("MIDMARKS.xlsx")
df_new = df.dropna(how='all')
print(df_new)
df.fillna(0, inplace=True)


subject_columns = ['DV', 'M-II', 'PP', 'BEEE', 'FL', 'FIMS']


df[subject_columns] = df[subject_columns].apply(pd.to_numeric,
errors='coerce')


print("Any NaN values in subject columns:",
df[subject_columns].isnull().any())

df = df.dropna(subset=subject_columns)



passing_marks = 10

fail_criteria = (df[subject_columns] < passing_marks).any(axis=1)


failed_students_count = fail_criteria.sum()


print(f"Number of students failed: {failed_students_count}")

failed_students = df[fail_criteria]
print(failed_students)
```

```
     S.NO SECTION  DV M-II  PP BEEE  FL FIMS
0     1.0   ALPHA  12    0  17    9  19   15
1     2.0   ALPHA  19   12  16   16  18    3
2     3.0   ALPHA  18   14  18   18  18   16
3     4.0   ALPHA  15    9  19   17  19   15
4     5.0   ALPHA  18   17  19   19  20   18
..    ...     ...  ..  ...  ..  ...  ..  ...
713   NaN    ZETA  19    8   8   19  17   18
714   NaN    ZETA  12    1   7   10  20    8
715   NaN    ZETA  17    6  14   14  17   18
716   NaN    ZETA  12    1   6    7  15   12
717   NaN    ZETA  19   14  17   16  20   19

[717 rows x 8 columns]
Any NaN values in subject columns: DV      True
```

```
M-II    True
PP      True
BEEE    True
FL      True
FIMS    True
dtype: bool
Number of students failed: 412
     S.NO SECTION    DV  M-II    PP  BEEE    FL  FIMS
0     1.0   ALPHA  12.0   0.0  17.0   9.0  19.0  15.0
1     2.0   ALPHA  19.0  12.0  16.0  16.0  18.0   3.0
3     4.0   ALPHA  15.0   9.0  19.0  17.0  19.0  15.0
5     6.0   ALPHA  17.0  16.0  18.0  10.0  15.0   9.0
13   14.0   ALPHA  17.0  17.0  18.0  11.0  15.0   9.0
..    ...     ...   ...   ...   ...   ...   ...   ...
712   0.0    ZETA  15.0  10.0   7.0  18.0  18.0  16.0
713   0.0    ZETA  19.0   8.0   8.0  19.0  17.0  18.0
714   0.0    ZETA  12.0   1.0   7.0  10.0  20.0   8.0
715   0.0    ZETA  17.0   6.0  14.0  14.0  17.0  18.0
716   0.0    ZETA  12.0   1.0   6.0   7.0  15.0  12.0

[412 rows x 8 columns]
```

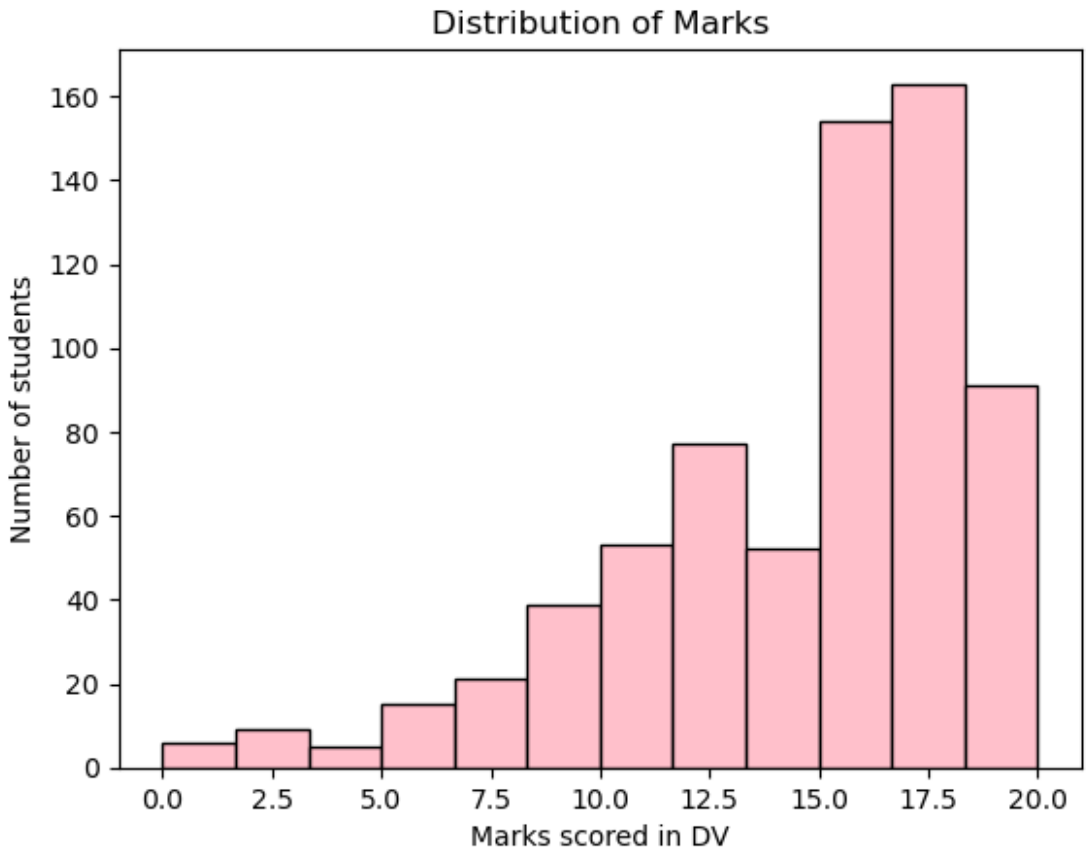
```
print(df['BEEE'])
df['BEEE'] = pd.to_numeric(df['BEEE'], errors='coerce')
df = df.dropna(subset=['BEEE'])
plt.hist(df['BEEE'], color='MAGENTA', edgecolor='white', bins=8)
plt.xlabel("Marks scored in DV")
plt.ylabel("Number of students")
plt.title("Distribution of Marks")
plt.show()
```

```
0       9.0
1      16.0
2      18.0
3      17.0
4      19.0
       ...
713    19.0
714    10.0
715    14.0
716     7.0
717    16.0
Name: BEEE, Length: 685, dtype: float64
```

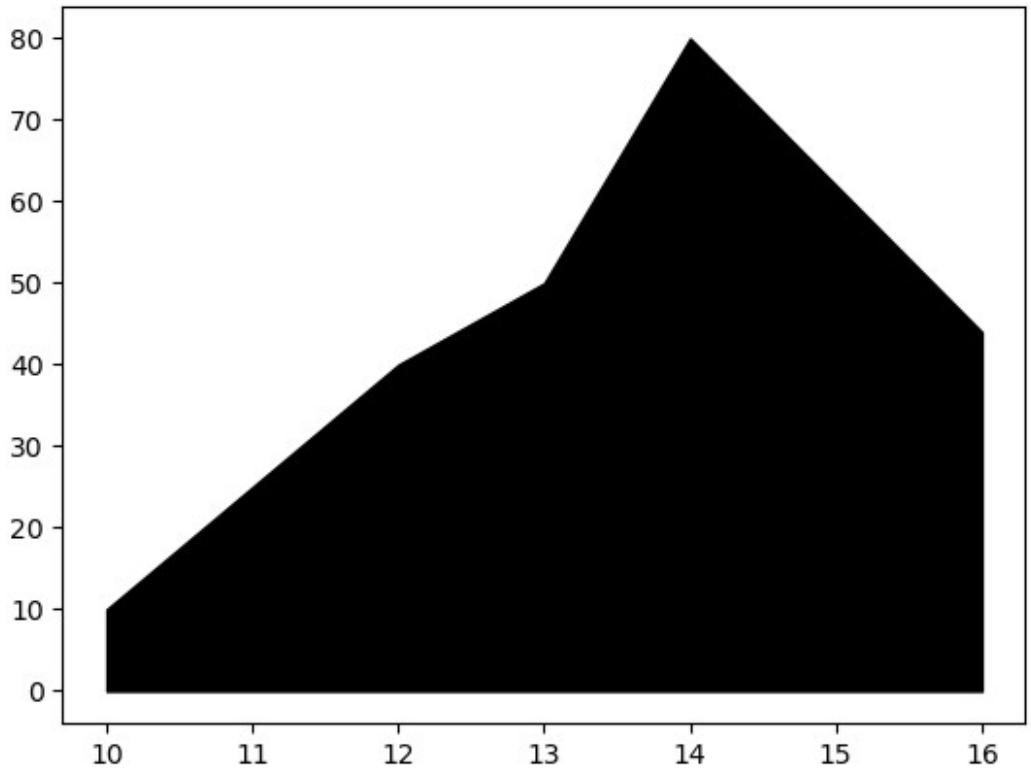



```
print(df['FIMS'])
df['FIMS'] = pd.to_numeric(df['FIMS'], errors='coerce')
df = df.dropna(subset=['FIMS'])
plt.hist(df['FIMS'], color='PINK', edgecolor='BLACK', bins=12)
plt.xlabel("Marks scored in DV")
plt.ylabel("Number of students")
plt.title("Distribution of Marks")
plt.show()
```

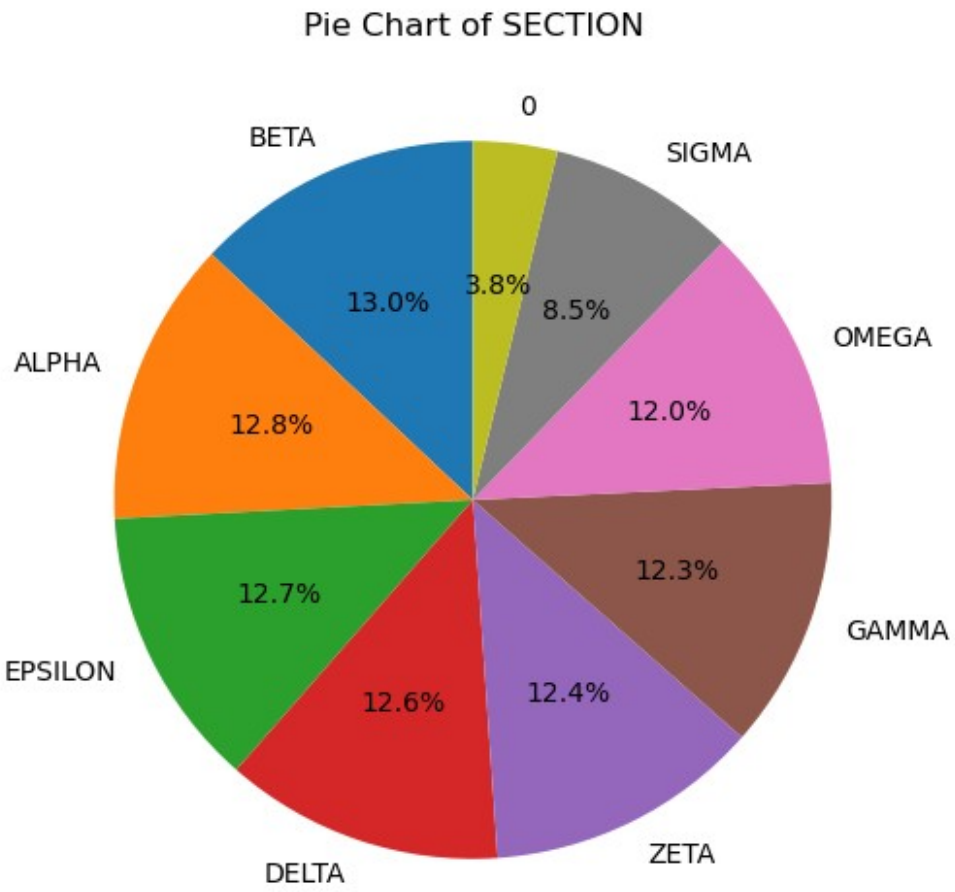
```
0      15.0
1       3.0
2      16.0
3      15.0
4      18.0
       ...
713    18.0
714     8.0
715    18.0
716    12.0
717    19.0
Name: FIMS, Length: 685, dtype: float64
```



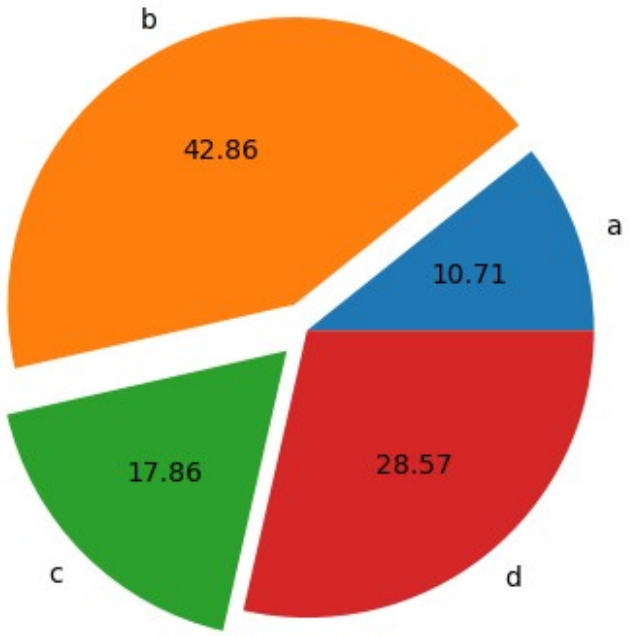

```
x=[10,12,13,14,16]
y=[10,40,50,80,44]

plt.fill_between(x, y,color='BLACK')
plt.show()
```




```
import matplotlib.pyplot as plt
section_counts = df['SECTION'].value_counts()
section_counts.plot(kind='pie', figsize=(6, 6), autopct='%1.1f%%',
startangle=90)
plt.title("Pie Chart of SECTION")
plt.ylabel("")
plt.show()
```

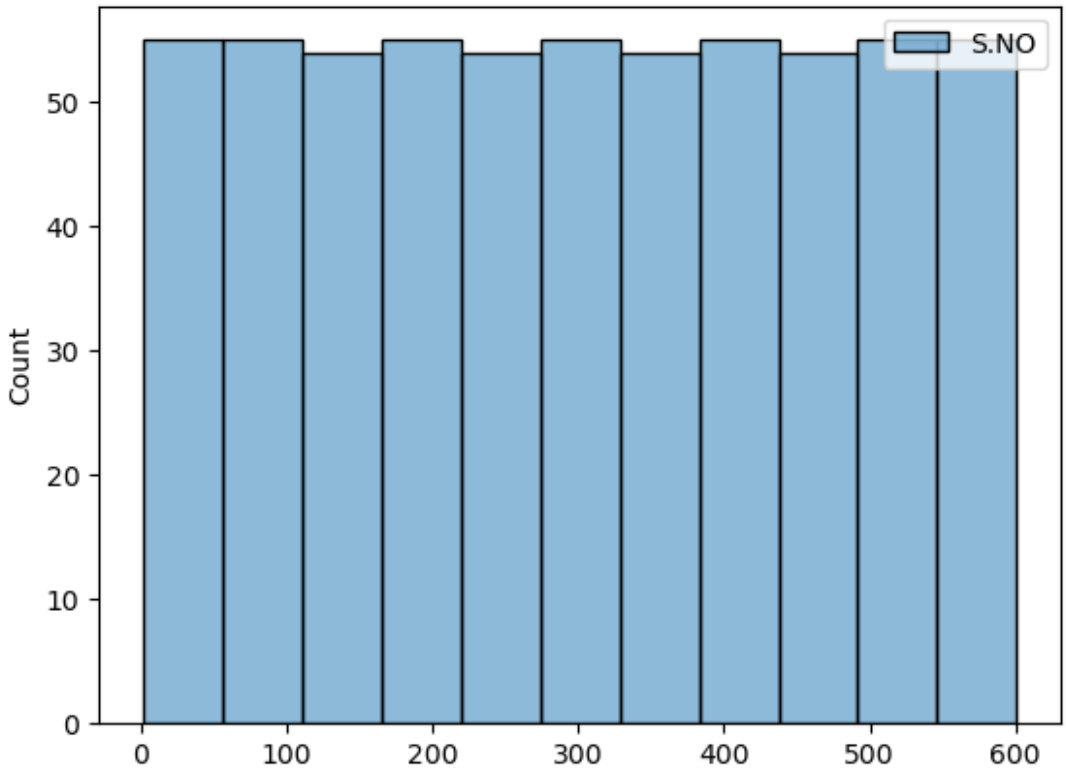



```
import matplotlib.pyplot as plt
values = [3, 12, 5, 8]
mylabels = ['a', 'b', 'c', 'd']
plt.pie(values, labels=mylabels, autopct='%.2f',
explode=[0,0.1,0.1,0])
plt.show()
```

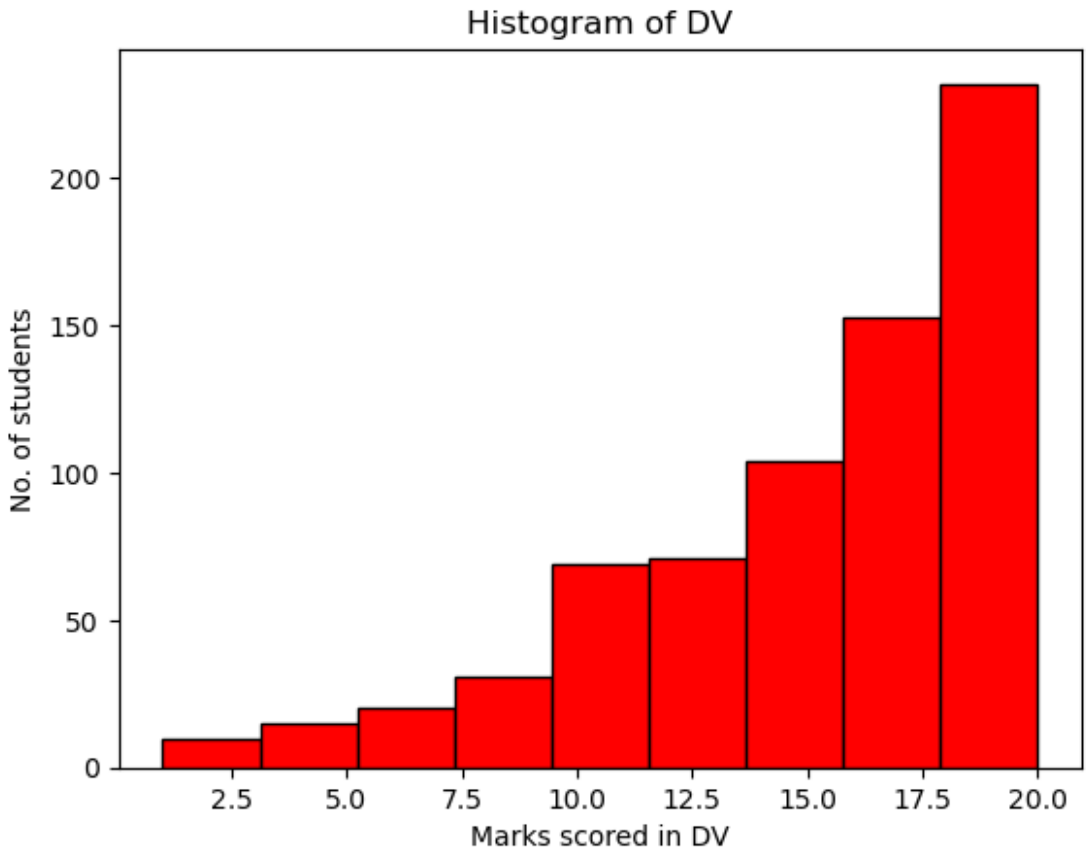



```
import pandas as pd
import seaborn as sns
df=pd.read_excel('MIDMARKS.xlsx')
sns.histplot(df)
```

```
<Axes: ylabel='Count'>
```




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



df = pd.read_excel("MIDMARKS.XLSX")


df['DV'] = pd.to_numeric(df['DV'], errors='coerce')


df = df.dropna(subset=['DV'])


plt.hist(df['DV'], color='red', edgecolor='black', bins=9)
plt.xlabel("Marks scored in DV")
plt.ylabel("No. of students")
plt.title("Histogram of DV")
plt.show()


passing_marks = 10
failures = df[df['DV'] < passing_marks]
num_failures = len(failures)

print(f"Number of students who failed in DV: {num_failures}")
```




```
Number of students who failed in DV: 76
```

```
import pandas as pd
data = pd.read_excel("MIDMARKS.XLSX")
subjects = ["DV"]

for subject in subjects:
    data[subject] = pd.to_numeric(data[subject],
errors='coerce').fillna(0)

def is_backlog(score):
    if score < 10:
        return 1
    else:
        return 0

for subject in subjects:
    data[f"{subject}_Backlog"] = data[subject].apply(is_backlog)

data["Backlog Count"] = data[[f"{subject}_Backlog" for subject in
subjects]].sum(axis=1)
data
```

```
     S.NO SECTION    DV M-II  PP BEEE  FL FIMS  DV_Backlog  Backlog
Count
0     1.0   ALPHA  12.0    0  17    9  19   15           0
0
1     2.0   ALPHA  19.0   12  16   16  18    3           0
0
2     3.0   ALPHA  18.0   14  18   18  18   16           0
0
3     4.0   ALPHA  15.0    9  19   17  19   15           0
0
4     5.0   ALPHA  18.0   17  19   19  20   18           0
0
..    ...     ...   ...  ...  ..  ...  ..  ...         ...
...
713   NaN    ZETA  19.0    8   8   19  17   18           0
0
714   NaN    ZETA  12.0    1   7   10  20    8           0
0
715   NaN    ZETA  17.0    6  14   14  17   18           0
0
716   NaN    ZETA  12.0    1   6    7  15   12           0
0
717   NaN    ZETA  19.0   14  17   16  20   19           0
0

[718 rows x 10 columns]
```

```
import pandas as pd
data = pd.read_excel("MIDMARKS.XLSX")
subjects = ["DV"]

for subject in subjects:
    data[subject] = pd.to_numeric(data[subject],
errors='coerce').fillna(0)

def is_backlog(score):
    if score < 10:
        return 1
    else:
        return 0

for subject in subjects:
    data[f"{subject}_Backlog"] = data[subject].apply(is_backlog)

data["Backlog Count"] = data[[f"{subject}_Backlog" for subject in
subjects]].sum(axis=1)
data
```

```
     S.NO SECTION    DV M-II  PP BEEE  FL FIMS  DV_Backlog  Backlog
Count
```

```
0     1.0   ALPHA  12.0    0  17    9  19   15           0
0
1     2.0   ALPHA  19.0   12  16   16  18    3           0
0
2     3.0   ALPHA  18.0   14  18   18  18   16           0
0
3     4.0   ALPHA  15.0    9  19   17  19   15           0
0
4     5.0   ALPHA  18.0   17  19   19  20   18           0
0
..    ...     ...   ...  ...  ..  ...  ..  ...         ...
...
713   NaN    ZETA  19.0    8   8   19  17   18           0
0
714   NaN    ZETA  12.0    1   7   10  20    8           0
0
715   NaN    ZETA  17.0    6  14   14  17   18           0
0
716   NaN    ZETA  12.0    1   6    7  15   12           0
0
717   NaN    ZETA  19.0   14  17   16  20   19           0
0

[718 rows x 10 columns]
```

```
import pandas as pd
data = pd.read_excel("MIDMARKS.XLSX")
subjects = ["PP"]

for subject in subjects:
    data[subject] = pd.to_numeric(data[subject],
errors='coerce').fillna(0)

def is_backlog(score):
    if score < 10:
        return 1
    else:
        return 0

for subject in subjects:
    data[f"{subject}_Backlog"] = data[subject].apply(is_backlog)

data["Backlog Count"] = data[[f"{subject}_Backlog" for subject in
subjects]].sum(axis=1)
data
```

```
     S.NO SECTION  DV M-II    PP BEEE  FL FIMS  PP_Backlog  Backlog
Count
0     1.0   ALPHA  12    0  17.0    9  19   15           0
0
```

```
1     2.0   ALPHA  19   12  16.0   16  18    3           0
0
2     3.0   ALPHA  18   14  18.0   18  18   16           0
0
3     4.0   ALPHA  15    9  19.0   17  19   15           0
0
4     5.0   ALPHA  18   17  19.0   19  20   18           0
0
..    ...     ...  ..  ...   ...  ...  ..  ...         ...
...
713   NaN    ZETA  19    8   8.0   19  17   18           1
1
714   NaN    ZETA  12    1   7.0   10  20    8           1
1
715   NaN    ZETA  17    6  14.0   14  17   18           0
0
716   NaN    ZETA  12    1   6.0    7  15   12           1
1
717   NaN    ZETA  19   14  17.0   16  20   19           0
0

[718 rows x 10 columns]
```

```
import pandas as pd
data = pd.read_excel("MIDMARKS.XLSX")
subjects = ["BEEE"]

for subject in subjects:
    data[subject] = pd.to_numeric(data[subject],
errors='coerce').fillna(0)

def is_backlog(score):
    if score < 10:
        return 1
    else:
        return 0

for subject in subjects:
    data[f"{subject}_Backlog"] = data[subject].apply(is_backlog)

data["Backlog Count"] = data[[f"{subject}_Backlog" for subject in
subjects]].sum(axis=1)
data
```

```
     S.NO SECTION  DV M-II  PP  BEEE  FL FIMS  BEEE_Backlog  Backlog
Count
0     1.0   ALPHA  12    0  17   9.0  19   15             1
1
1     2.0   ALPHA  19   12  16  16.0  18    3             0
0
```

```
2     3.0   ALPHA  18   14  18  18.0  18   16             0
0
3     4.0   ALPHA  15    9  19  17.0  19   15             0
0
4     5.0   ALPHA  18   17  19  19.0  20   18             0
0
..    ...     ...  ..  ...  ..   ...  ..  ...           ...
...
713   NaN    ZETA  19    8   8  19.0  17   18             0
0
714   NaN    ZETA  12    1   7  10.0  20    8             0
0
715   NaN    ZETA  17    6  14  14.0  17   18             0
0
716   NaN    ZETA  12    1   6   7.0  15   12             1
1
717   NaN    ZETA  19   14  17  16.0  20   19             0
0

[718 rows x 10 columns]
```

```
import pandas as pd
data = pd.read_excel("MIDMARKS.XLSX")
subjects = ["FL"]

for subject in subjects:
    data[subject] = pd.to_numeric(data[subject],
errors='coerce').fillna(0)

def is_backlog(score):
    if score < 10:
        return 1
    else:
        return 0

for subject in subjects:
    data[f"{subject}_Backlog"] = data[subject].apply(is_backlog)

data["Backlog Count"] = data[[f"{subject}_Backlog" for subject in
subjects]].sum(axis=1)
data
```

```
     S.NO SECTION  DV M-II  PP BEEE    FL FIMS  FL_Backlog  Backlog
Count
0     1.0   ALPHA  12    0  17    9  19.0   15           0
0
1     2.0   ALPHA  19   12  16   16  18.0    3           0
0
2     3.0   ALPHA  18   14  18   18  18.0   16           0
0
```

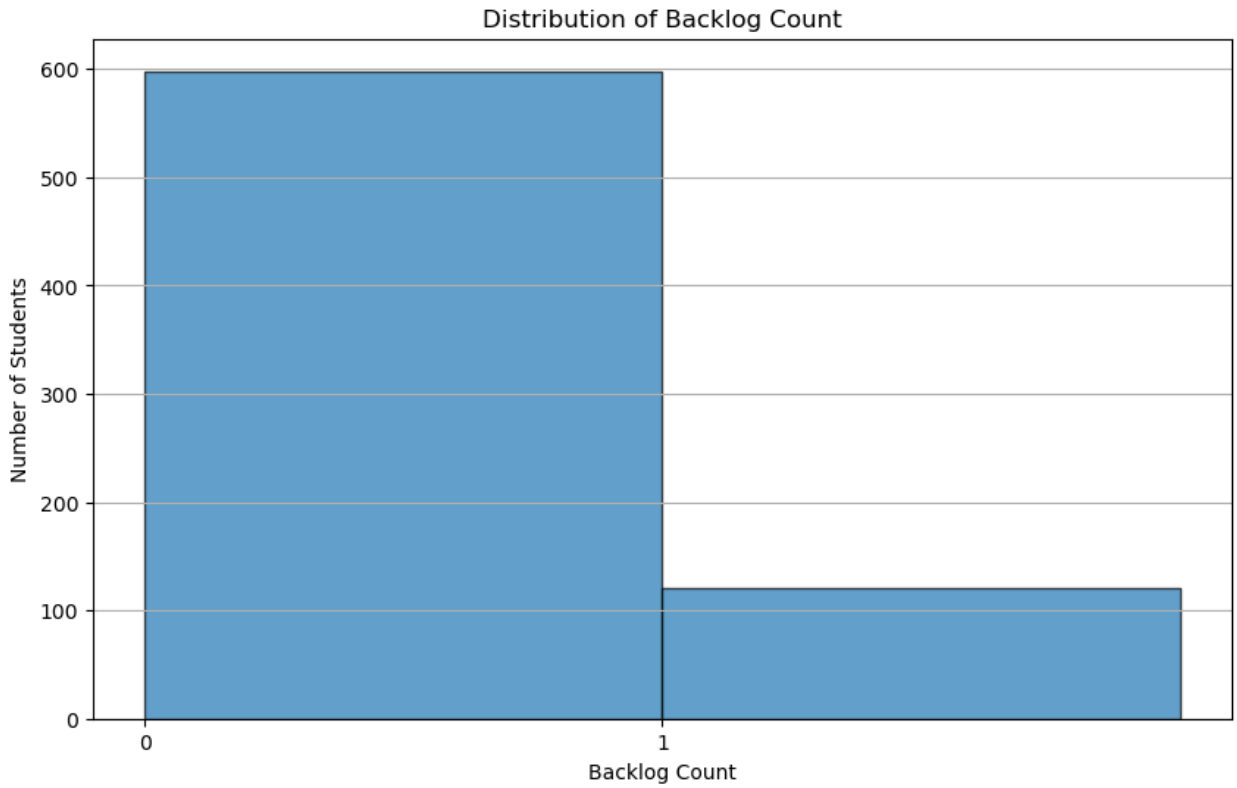
```
3     4.0   ALPHA  15    9  19   17  19.0   15           0
0
4     5.0   ALPHA  18   17  19   19  20.0   18           0
0
..    ...     ...  ..  ...  ..  ...   ...  ...         ...
...
713   NaN    ZETA  19    8   8   19  17.0   18           0
0
714   NaN    ZETA  12    1   7   10  20.0    8           0
0
715   NaN    ZETA  17    6  14   14  17.0   18           0
0
716   NaN    ZETA  12    1   6    7  15.0   12           0
0
717   NaN    ZETA  19   14  17   16  20.0   19           0
0

[718 rows x 10 columns]
```

```
import pandas as pd
data = pd.read_excel("MIDMARKS.XLSX")
subjects = ["FIMS"]

for subject in subjects:
    data[subject] = pd.to_numeric(data[subject],
errors='coerce').fillna(0)

def is_backlog(score):
    if score < 10:
        return 1
    else:
        return 0

for subject in subjects:
    data[f"{subject}_Backlog"] = data[subject].apply(is_backlog)

data["Backlog Count"] = data[[f"{subject}_Backlog" for subject in
subjects]].sum(axis=1)
data
```

```
     S.NO SECTION  DV M-II  PP BEEE  FL  FIMS  FIMS_Backlog  Backlog
Count
0     1.0   ALPHA  12    0  17    9  19  15.0             0
0
1     2.0   ALPHA  19   12  16   16  18   3.0             1
1
2     3.0   ALPHA  18   14  18   18  18  16.0             0
0
3     4.0   ALPHA  15    9  19   17  19  15.0             0
0
```

```
4     5.0   ALPHA  18   17  19   19  20  18.0             0
0
..    ...     ...  ..  ...  ..  ...  ..   ...           ...
...
713   NaN    ZETA  19    8   8   19  17  18.0             0
0
714   NaN    ZETA  12    1   7   10  20   8.0             1
1
715   NaN    ZETA  17    6  14   14  17  18.0             0
0
716   NaN    ZETA  12    1   6    7  15  12.0             0
0
717   NaN    ZETA  19   14  17   16  20  19.0             0
0

[718 rows x 10 columns]
```

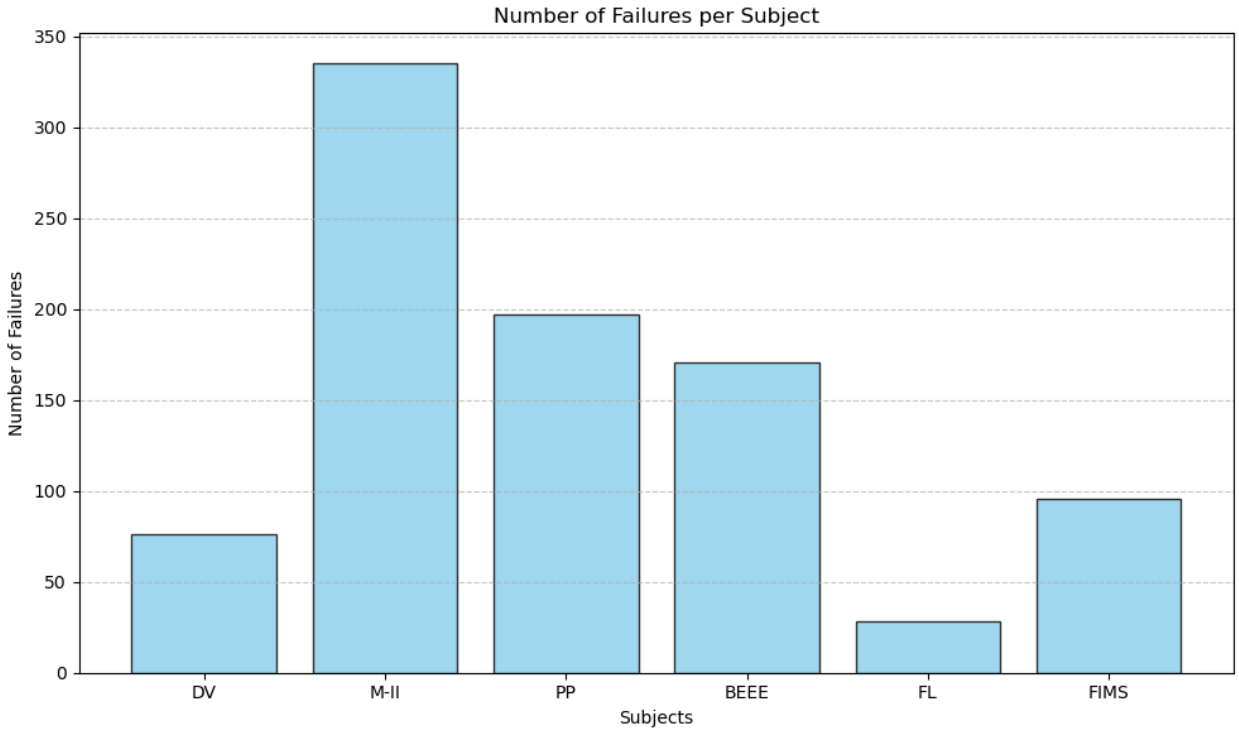
```python
import matplotlib.pyplot as plt
subjects = ["DA"]
plt.figure(figsize=(10, 6))
plt.hist(data["Backlog Count"], bins=range(data["Backlog Count"].max()
+ 2), edgecolor='black', alpha=0.7)
plt.title("Distribution of Backlog Count")
plt.xlabel("Backlog Count")
plt.ylabel("Number of Students")
plt.xticks(range(data["Backlog Count"].max() + 1))
plt.grid(axis='y')
plt.show()
```




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

subjects = ["DV", "M-II", "PP", "BEEE", "FL", "FIMS"]
passing_marks = 10

df[subjects] = df[subjects].apply(pd.to_numeric, errors='coerce')

for subject in subjects:
    df[f"{subject}_Backlog"] = (df[subject] <
passing_marks).astype(int)

failure_counts = {subject: df[f"{subject}_Backlog"].sum() for subject
in subjects}

most_failed_subject = max(failure_counts, key=failure_counts.get)

print("Failures per subject:")
for subject, count in failure_counts.items():
    print(f"{subject}: {count} students failed")

print(f"\nThe subject most students failed in is:
{most_failed_subject}")
plt.figure(figsize=(10, 6))
plt.bar(failure_counts.keys(), failure_counts.values(),
color='skyblue', edgecolor='black', alpha=0.8)
```

```
plt.title("Number of Failures per Subject")
plt.xlabel("Subjects")
plt.ylabel("Number of Failures")
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.tight_layout()
plt.show()
```

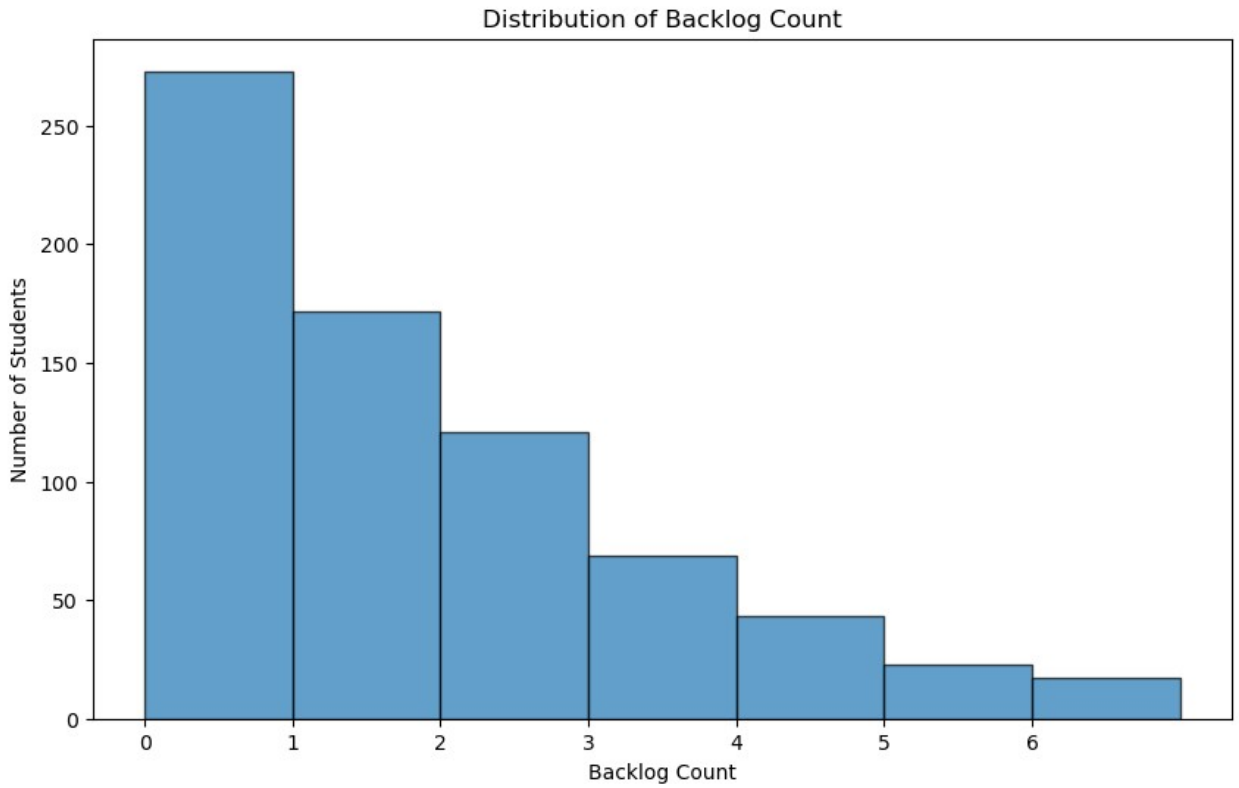
```
Failures per subject:
DV: 76 students failed
M-II: 335 students failed
PP: 197 students failed
BEEE: 171 students failed
FL: 28 students failed
FIMS: 96 students failed

The subject most students failed in is: M-II
```



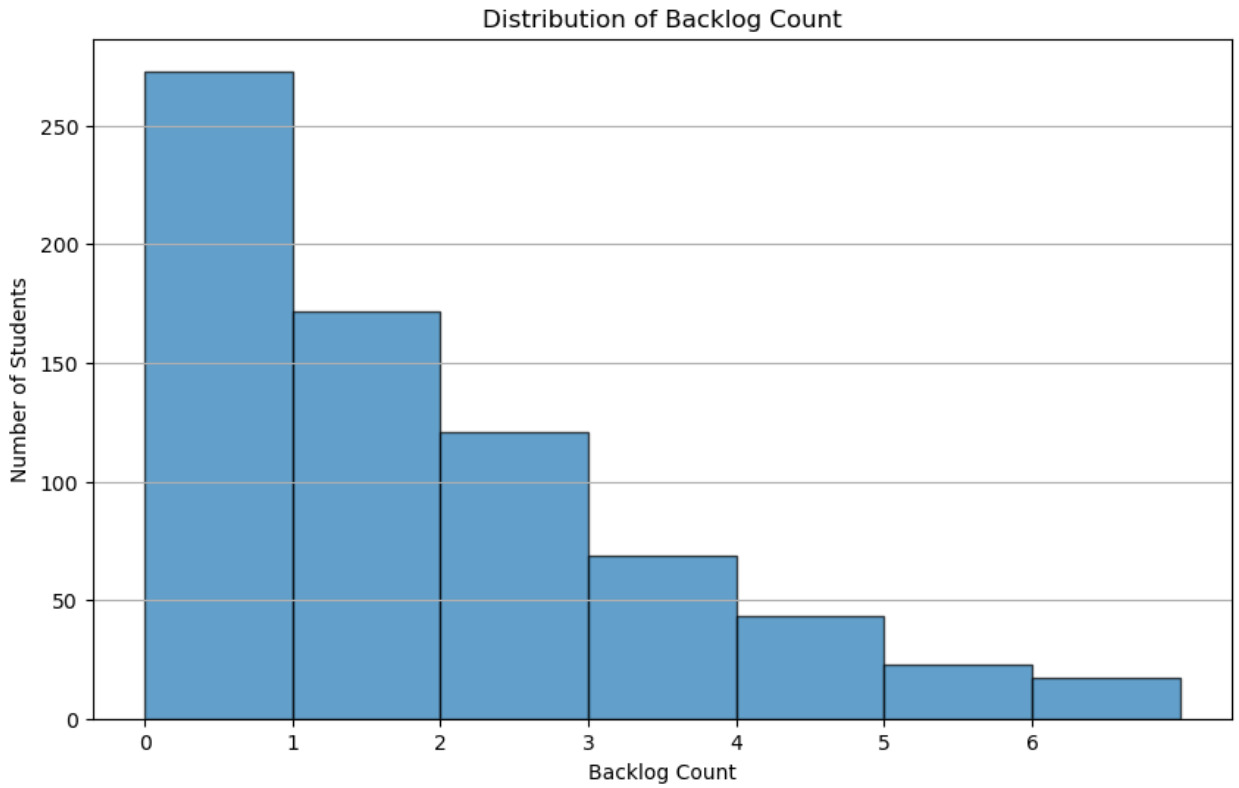

```
import pandas as pd
import matplotlib.pyplot as plt

data = pd.read_excel("MIDMARKS.XLSX")


subjects = ["DV", "M-II", "PP", "BEEE", "FL", "FIMS"]

for subject in subjects:
    data[subject] = pd.to_numeric(data[subject],
```

```
errors='coerce').fillna(0)


def is_backlog(score):
    return 1 if score < 10 else 0


for subject in subjects:
    data[f"{subject}_Backlog"] = data[subject].apply(is_backlog)

print("Failures per subject:")
for subject, count in failure_counts.items():
    print(f"{subject}: {count} students failed")

print(f"The subject most students failed is: {most_failed_subject}")



data["Backlog Count"] = data[[f"{subject}_Backlog" for subject in
subjects]].sum(axis=1)


plt.figure(figsize=(10, 6))
plt.hist(data["Backlog Count"], bins=range(data["Backlog Count"].max()
+ 2), edgecolor='black', alpha=0.7)
plt.title("Distribution of Backlog Count")
plt.xlabel("Backlog Count")
plt.ylabel("Number of Students")
plt.xticks(range(data["Backlog Count"].max() + 1))

plt.show()
```

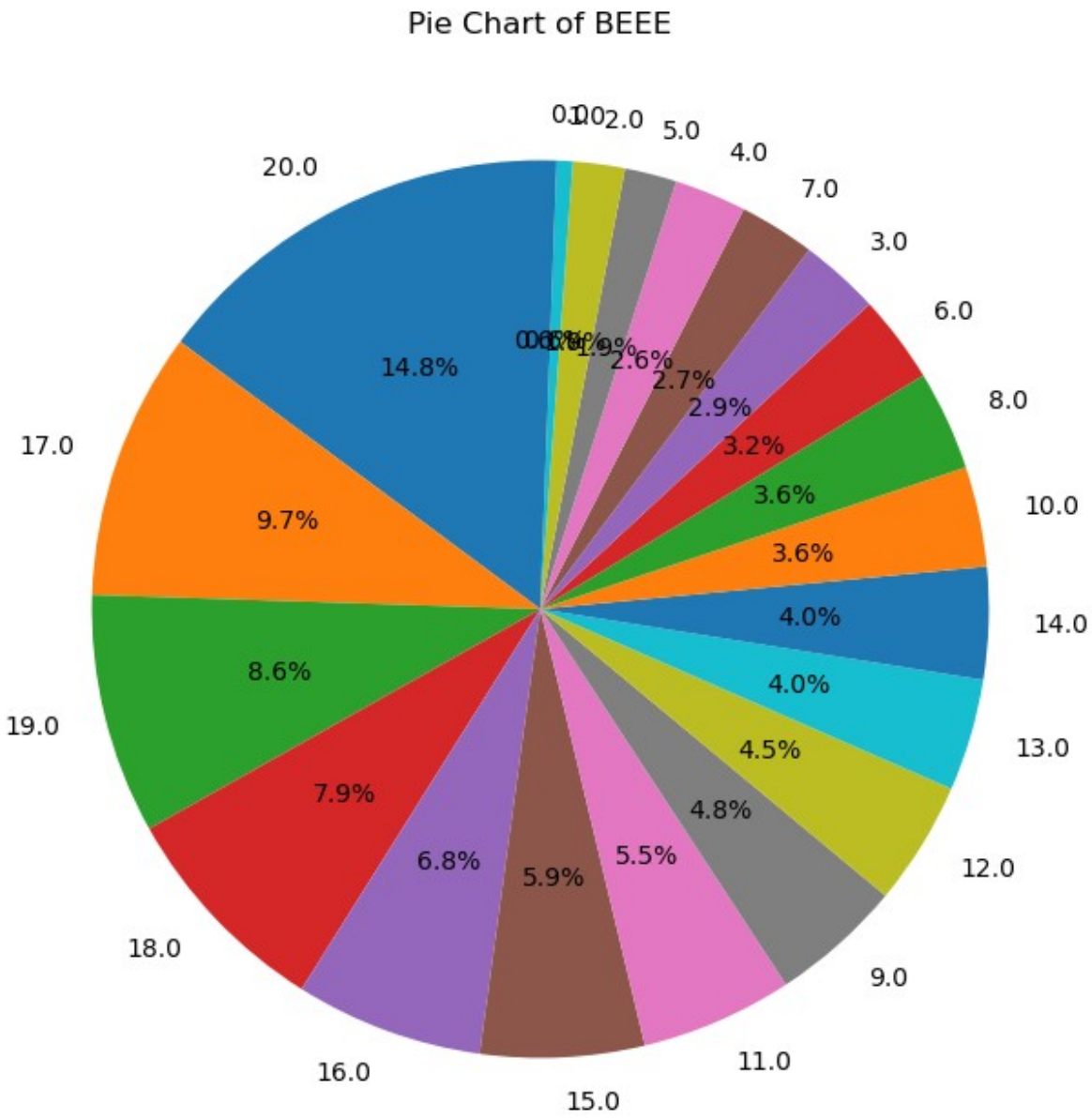
```
Failures per subject:
DV: 76 students failed
M-II: 335 students failed
PP: 197 students failed
BEEE: 171 students failed
FL: 28 students failed
FIMS: 96 students failed
The subject most students failed is: M-II
```




```
import matplotlib.pyplot as plt
subjects = ["DA"]
plt.figure(figsize=(10, 6))
plt.hist(data["Backlog Count"], bins=range(data["Backlog Count"].max()
+ 2), edgecolor='black', alpha=0.7)
plt.title("Distribution of Backlog Count")
plt.xlabel("Backlog Count")
plt.ylabel("Number of Students")
plt.xticks(range(data["Backlog Count"].max() + 1))
plt.grid(axis='y')
plt.show()
```




```
import matplotlib.pyplot as plt
bee_counts = df['BEEE'].value_counts()
bee_counts.plot(kind='pie', figsize=(8, 8), autopct='%1.1f%%',
startangle=90)
plt.title("Pie Chart of BEEE")
plt.ylabel("")
plt.show()
```




```
import pandas as pd
data = pd.read_excel("MIDMARKS.XLSX")
subjects = ["PP"]
data.dropna(inplace=True)

for subject in subjects:
    data[subject] = pd.to_numeric(data[subject],
errors='coerce').fillna(0)

def assign_grade(percentage):
    if 18 <= percentage <= 20:
        return 'Very Good'
```

```
    elif 13 <= percentage <= 14:
        return 'Average'
    elif 13 <= percentage <= 17:
        return 'Good'
    else:
        return 'poor'

data['PP_Grade'] = data['PP'].apply(assign_grade)
print(data)
```

```
      S.NO SECTION  DV M-II    PP BEEE  FL FIMS   PP_Grade
0      1.0   ALPHA  12    0  17.0    9  19   15       Good
1      2.0   ALPHA  19   12  16.0   16  18    3       Good
2      3.0   ALPHA  18   14  18.0   18  18   16  Very Good
3      4.0   ALPHA  15    9  19.0   17  19   15  Very Good
4      5.0   ALPHA  18   17  19.0   19  20   18  Very Good
..     ...     ...  ..  ...   ...  ...  ..  ...        ...
596  597.0   SIGMA  20   20  20.0   20  20   20  Very Good
597  598.0   SIGMA  20   20  20.0   19  19   18  Very Good
598  599.0   SIGMA  20   20  17.0   17  19   18       Good
599  600.0   SIGMA  14   12  11.0    9  18   17       poor
600  601.0   SIGMA  20   19  20.0   18  18   19  Very Good

[599 rows x 9 columns]
```

```
import pandas as pd
data = pd.read_excel("MIDMARKS.XLSX")
subjects = ["DV"]
data.dropna(inplace=True)

for subject in subjects:
    data[subject] = pd.to_numeric(data[subject],
errors='coerce').fillna(0)

def assign_grade(percentage):
    if 18 <= percentage <= 20:
        return 'Very Good'
    elif 13 <= percentage <= 14:
        return 'Average'
    elif 13 <= percentage <= 17:
        return 'Good'
    else:
        return 'poor'

data['DV_Grade'] = data['DV'].apply(assign_grade)
print(data)
```

```
      S.NO SECTION    DV M-II  PP BEEE  FL FIMS   DV_Grade
0      1.0   ALPHA  12.0    0  17    9  19   15       poor
1      2.0   ALPHA  19.0   12  16   16  18    3  Very Good
```

```
2      3.0   ALPHA  18.0   14  18   18  18   16  Very Good
3      4.0   ALPHA  15.0    9  19   17  19   15       Good
4      5.0   ALPHA  18.0   17  19   19  20   18  Very Good
..     ...     ...   ...  ...  ..  ...  ..  ...        ...
596  597.0   SIGMA  20.0   20  20   20  20   20  Very Good
597  598.0   SIGMA  20.0   20  20   19  19   18  Very Good
598  599.0   SIGMA  20.0   20  17   17  19   18  Very Good
599  600.0   SIGMA  14.0   12  11    9  18   17    Average
600  601.0   SIGMA  20.0   19  20   18  18   19  Very Good

[599 rows x 9 columns]
```

```
import pandas as pd

data = {
    'Programming_skills': ['Very Good', 'Good', 'Very Good',
'Excellent', 'Good', 'Very Good']
}

df = pd.DataFrame(data)

very_good_count = (df['Programming_skills'] == 'Very Good').sum()

print(f"Number of 'Very Good': {very_good_count}")
```

```
Number of 'Very Good': 3
```

```
import pandas as pd

data = pd.read_excel("MIDMARKS.XLSX")

subjects = ["DV"]

data.dropna(inplace=True)

for subject in subjects:
    data[subject] = pd.to_numeric(data[subject],
errors='coerce').fillna(0)

def assign_grade(percentage):
    if 18 <= percentage <= 20:
        return 'Very Good'
    elif 13 <= percentage <= 14:
        return 'Average'
    elif 13 <= percentage <= 17:
        return 'Good'
    else:
        return 'Poor'

data['Programming_skills'] = data['DV'].apply(assign_grade)
```

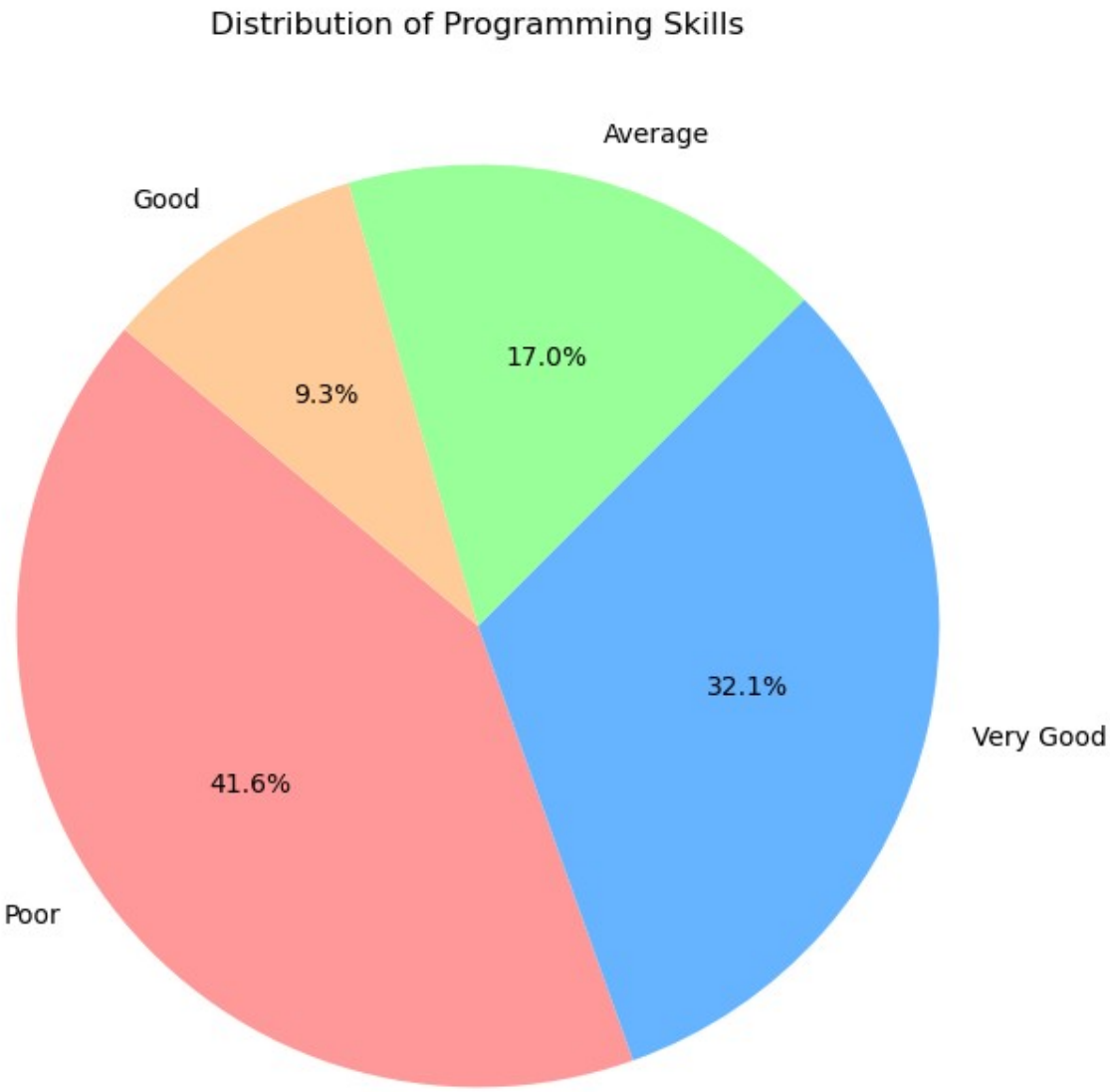
```
very_good_count = (data['Programming_skills'] == 'Very Good').sum()

print(data)
print(f"\nNumber of 'Very Good' students: {very_good_count}")
```

```
      S.NO SECTION    DV M-II  PP BEEE  FL FIMS Programming_skills
0      1.0   ALPHA  12.0    0  17    9  19   15                Poor
1      2.0   ALPHA  19.0   12  16   16  18    3           Very Good
2      3.0   ALPHA  18.0   14  18   18  18   16           Very Good
3      4.0   ALPHA  15.0    9  19   17  19   15                Good
4      5.0   ALPHA  18.0   17  19   19  20   18           Very Good
..     ...     ...   ...  ...  ..  ...  ..  ...                 ...
596  597.0   SIGMA  20.0   20  20   20  20   20           Very Good
597  598.0   SIGMA  20.0   20  20   19  19   18           Very Good
598  599.0   SIGMA  20.0   20  17   17  19   18           Very Good
599  600.0   SIGMA  14.0   12  11    9  18   17             Average
600  601.0   SIGMA  20.0   19  20   18  18   19           Very Good

[599 rows x 9 columns]

Number of 'Very Good' students: 190
```

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

df = pd.read_excel("MIDMARKS.XLSX")

subjects = ["PP"]
df.dropna(inplace=True)

for subject in subjects:
    df[subject] = pd.to_numeric(df[subject],
errors='coerce').fillna(0)


def assign_grade(percentage):
    if 18 <= percentage <= 20:
        return 'Very Good'
    elif 15 <= percentage <= 17:
        return 'Average'
    elif 13 <= percentage <= 14:
        return 'Good'
    else:
        return 'Poor'

df['Programming_skills'] = df['PP'].apply(assign_grade)

most_common_grade = df['Programming_skills'].value_counts().idxmax()

print(df)
print("Most frequent skill level:", most_common_grade)
```

```
skill_counts = df['Programming_skills'].value_counts()
plt.figure(figsize=(8, 8))
plt.pie(skill_counts, labels=skill_counts.index, autopct='%1.1f%%',
startangle=140, colors=['#ff9999','#66b3ff','#99ff99','#ffcc99'])
plt.title('Distribution of Programming Skills')
plt.show()
```

```
      S.NO SECTION  DV M-II    PP BEEE  FL FIMS Programming_skills
0      1.0   ALPHA  12    0  17.0    9  19   15            Average
1      2.0   ALPHA  19   12  16.0   16  18    3            Average
2      3.0   ALPHA  18   14  18.0   18  18   16          Very Good
3      4.0   ALPHA  15    9  19.0   17  19   15          Very Good
4      5.0   ALPHA  18   17  19.0   19  20   18          Very Good
..     ...     ...  ..  ...   ...  ...  ..  ...                ...
596  597.0   SIGMA  20   20  20.0   20  20   20          Very Good
597  598.0   SIGMA  20   20  20.0   19  19   18          Very Good
598  599.0   SIGMA  20   20  17.0   17  19   18            Average
599  600.0   SIGMA  14   12  11.0    9  18   17               Poor
600  601.0   SIGMA  20   19  20.0   18  18   19          Very Good

[599 rows x 9 columns]
Most frequent skill level: Poor
```



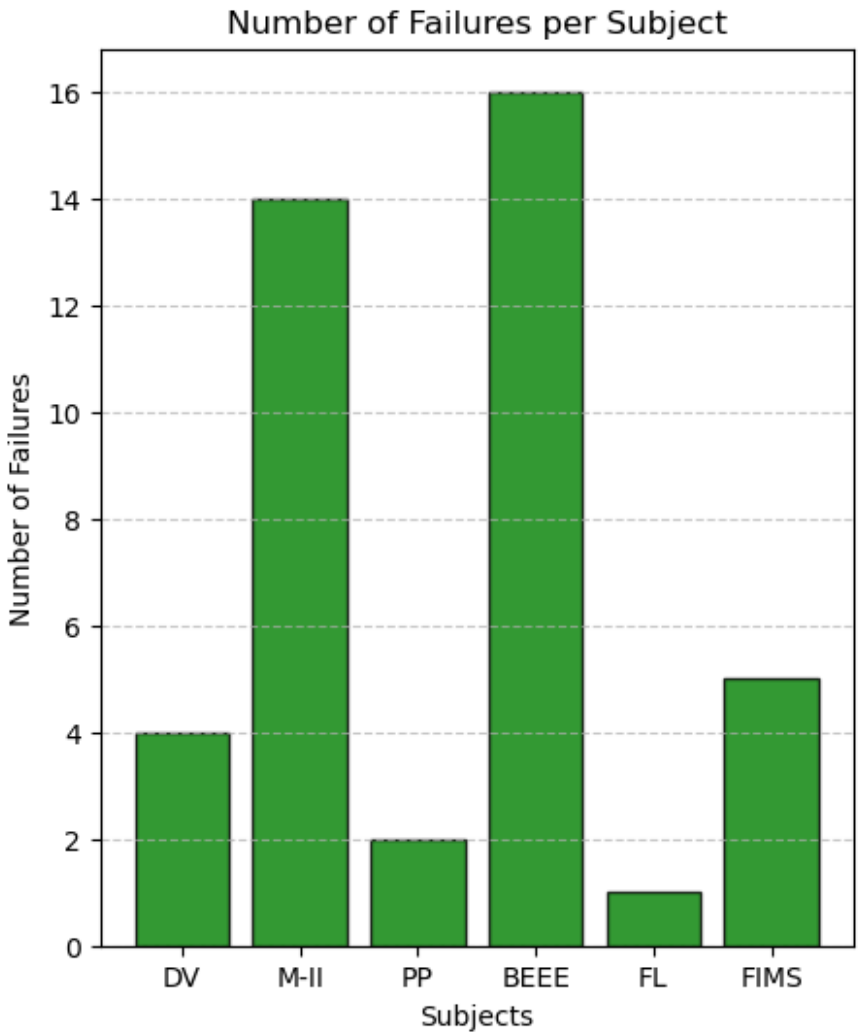

```
subjects = ['DV', 'M-II', 'PP', 'BEEE', 'FL', 'FIMS']
subset = df[df[subjects].eq(20).any(axis=1)]
print("Subset of students who scored 20 in any subject:")
print(subset)
for subject in subjects:
    count_20 = (df[subject] == 20).sum()
    print(f"Students who scored 20 in {subject}: {count_20}")
```

```
Subset of students who scored 20 in any subject:
      S.NO SECTION  DV M-II    PP BEEE  FL FIMS Programming_skills
4      5.0   ALPHA  18   17  19.0   19  20   18          Very Good
6      7.0   ALPHA  15   10  20.0   20  15   14          Very Good
7      8.0   ALPHA  17   17  19.0   20  19   13          Very Good
```

```
8      9.0   ALPHA  10   18   0.0   20  19   15                 Poor
9     10.0   ALPHA  18   19  20.0   20  20   15            Very Good
..     ...     ...  ..  ...   ...  ...  ..  ...                  ...
595  596.0   SIGMA  17   14  16.0   18  20   18              Average
596  597.0   SIGMA  20   20  20.0   20  20   20            Very Good
597  598.0   SIGMA  20   20  20.0   19  19   18            Very Good
598  599.0   SIGMA  20   20  17.0   17  19   18              Average
600  601.0   SIGMA  20   19  20.0   18  18   19            Very Good

[253 rows x 9 columns]
Students who scored 20 in DV: 88
Students who scored 20 in M-II: 56
Students who scored 20 in PP: 104
Students who scored 20 in BEEE: 89
Students who scored 20 in FL: 159
Students who scored 20 in FIMS: 27

subset = df[df.iloc[:, 1:].eq(20).any(axis=1)]
print(subset)

      S.NO SECTION  DV M-II    PP BEEE  FL FIMS Programming_skills
4      5.0   ALPHA  18   17  19.0   19  20   18            Very Good
6      7.0   ALPHA  15   10  20.0   20  15   14            Very Good
7      8.0   ALPHA  17   17  19.0   20  19   13            Very Good
8      9.0   ALPHA  10   18   0.0   20  19   15                 Poor
9     10.0   ALPHA  18   19  20.0   20  20   15            Very Good
..     ...     ...  ..  ...   ...  ...  ..  ...                  ...
595  596.0   SIGMA  17   14  16.0   18  20   18              Average
596  597.0   SIGMA  20   20  20.0   20  20   20            Very Good
597  598.0   SIGMA  20   20  20.0   19  19   18            Very Good
598  599.0   SIGMA  20   20  17.0   17  19   18              Average
600  601.0   SIGMA  20   19  20.0   18  18   19            Very Good

[253 rows x 9 columns]

import matplotlib.pyplot as plt
import pandas as pd

data = pd.DataFrame({
    "DV_Backlog": [0, 1, 2, 1],
    "M-II_Backlog": [3, 2, 4, 5],
    "PP_Backlog": [1, 0, 0, 1],
    "BEEE_Backlog": [4, 4, 4, 4],
    "FL_Backlog": [0, 0, 1, 0],
    "FIMS_Backlog": [1, 1, 1, 2]
})

subjects = ["DV", "M-II", "PP", "BEEE", "FL", "FIMS"]

failure_counts = {subject: data[f"{subject}_Backlog"].sum() for
```

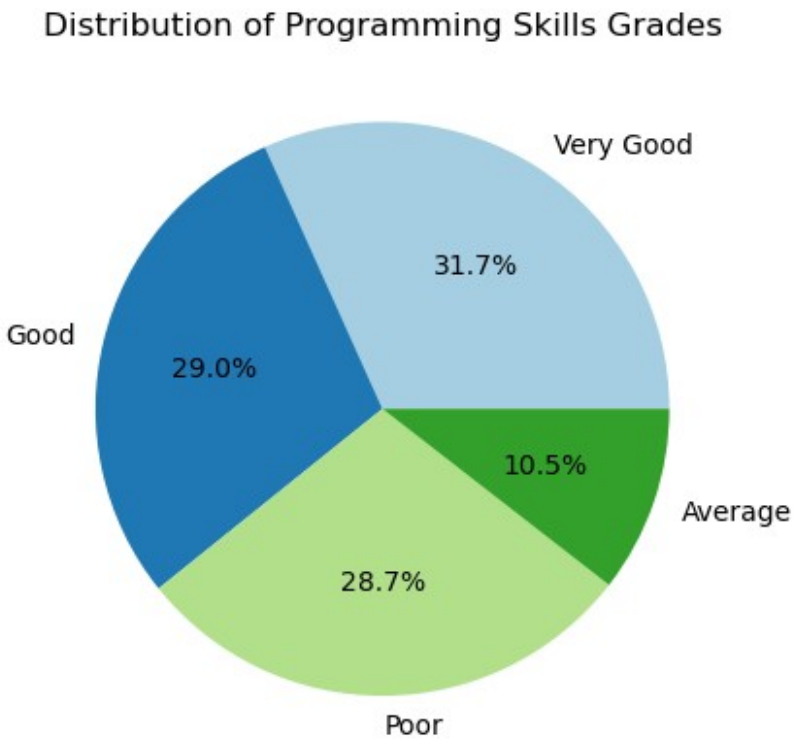
```
subject in subjects}

most_failed_subject = max(failure_counts, key=failure_counts.get)

print("Failures per subject:")
for subject, count in failure_counts.items():
    print(f"{subject}: {count} students failed")

print(f"\nThe subject most students failed is: {most_failed_subject}")

plt.figure(figsize=(5, 6))
plt.bar(failure_counts.keys(), failure_counts.values(), color='green',
edgecolor='black', alpha=0.8)
plt.title("Number of Failures per Subject")
plt.xlabel("Subjects")
plt.ylabel("Number of Failures")
plt.grid(axis='y', linestyle='--', alpha=0.7)
plt.show()
```

```
Failures per subject:
DV: 4 students failed
M-II: 14 students failed
PP: 2 students failed
BEEE: 16 students failed
FL: 1 students failed
FIMS: 5 students failed

The subject most students failed is: BEEE
```




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

data = pd.read_excel("MIDMARKS.XLSX")

subjects = ["DV"]

data.dropna(inplace=True)

for subject in subjects:
    data[subject] = pd.to_numeric(data[subject],
errors='coerce').fillna(0)

def assign_grade(percentage):
    if 18 <= percentage <= 20:
        return 'Very Good'
    elif 13 <= percentage <= 14:
        return 'Average'
```

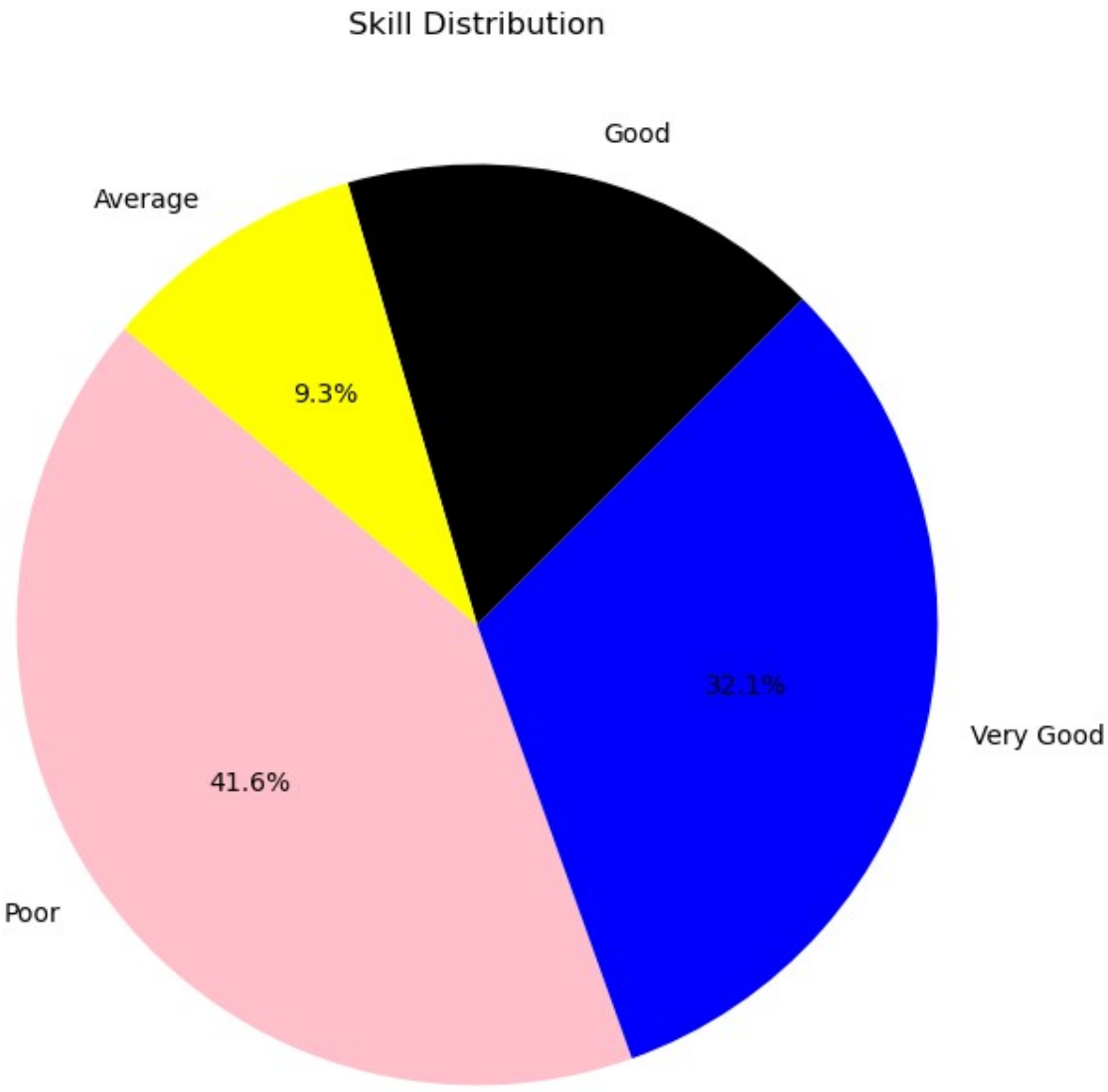
```
    elif 13 <= percentage <= 17:
        return 'Good'
    else:
        return 'Poor'

data['Programming_skills'] = data['DV'].apply(assign_grade)

grade_counts = data['Programming_skills'].value_counts()

plt.pie(grade_counts, labels=grade_counts.index, autopct='%1.1f%%',
colors=plt.cm.Paired.colors)
plt.title("Distribution of Programming Skills Grades")
plt.show()

print(data)
```




```
      S.NO SECTION    DV M-II  PP BEEE  FL FIMS Programming_skills
0      1.0   ALPHA  12.0    0  17    9  19   15               Poor
1      2.0   ALPHA  19.0   12  16   16  18    3          Very Good
2      3.0   ALPHA  18.0   14  18   18  18   16          Very Good
3      4.0   ALPHA  15.0    9  19   17  19   15               Good
4      5.0   ALPHA  18.0   17  19   19  20   18          Very Good
..     ...     ...   ...  ...  ..  ...  ..  ...                ...
596  597.0   SIGMA  20.0   20  20   20  20   20          Very Good
597  598.0   SIGMA  20.0   20  20   19  19   18          Very Good
```

```
598  599.0   SIGMA  20.0   20  17   17  19   18          Very Good
599  600.0   SIGMA  14.0   12  11    9  18   17            Average
600  601.0   SIGMA  20.0   19  20   18  18   19          Very Good

[599 rows x 9 columns]
```

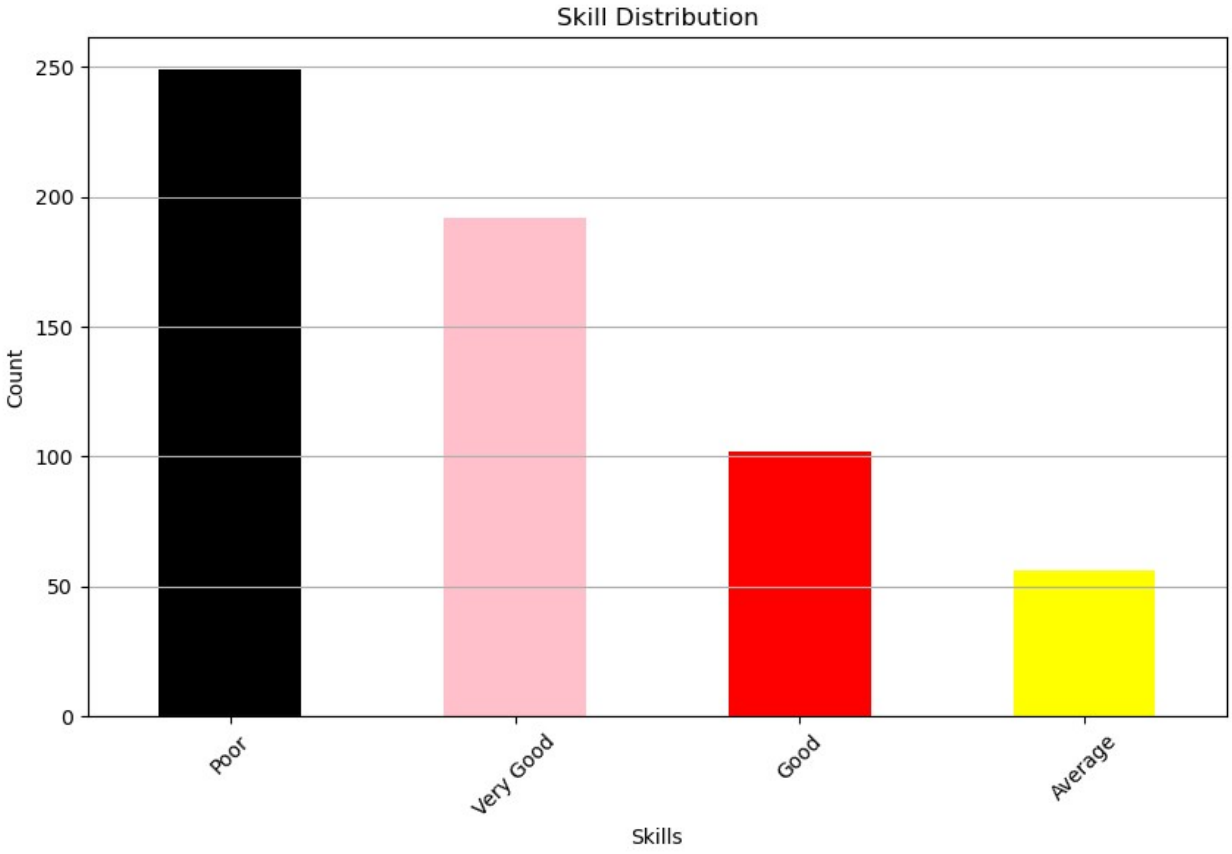
```
df['skills'] = df['PP'].apply(assign_grade)
skill_counts = df['skills'].value_counts()
plt.figure(figsize=(8, 8))
plt.pie(skill_counts, labels=skill_counts.index, autopct='%1.1f%%',
startangle=140, colors=['pink','blue','black','yellow'])
plt.title('Skill Distribution')
plt.show()
df.skills.value_counts()
```




```
skills
Poor         249
Very Good    192
Good         102
Average       56
Name: count, dtype: int64
```

```
import matplotlib.pyplot as plt

skill_counts = df['skills'].value_counts()

plt.figure(figsize=(10, 6))
skill_counts.plot(kind='bar', color=['black', 'pink', 'red',
```

```
'yellow'])
plt.title('Skill Distribution')
plt.xlabel('Skills')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()

print(df.skills.value_counts())
```




```
skills
Poor         249
Very Good    192
Good         102
Average       56
Name: count, dtype: int64
```

```
import pandas as pd
import numpy as np

file_path = 'MIDMARKS.xlsx'
df = pd.read_excel(file_path)
```

```
df.columns = df.columns.str.strip().str.lower()

columns_to_sum = ["dv", "m-ii", "pp", "beee", "fl", "fims"]

df.fillna(0, inplace=True)

df[columns_to_sum] = df[columns_to_sum].apply(pd.to_numeric,
errors='coerce')

df["total_marks"] = df[columns_to_sum].sum(axis=1)

df["prediction_probability"] = df.groupby("section")
["total_marks"].transform(
    lambda x: np.round((x - x.min()) / (x.max() - x.min()), 2) if
x.max() != x.min() else 1
)

df["prediction_probability"] = df["prediction_probability"].clip(0, 1)

output_file_path = 'MIDMARKS_WITH_SECTION_PROBABILITY.xlsx'
df.to_excel(output_file_path, index=False)

print("Updated DataFrame with Section-Wise Prediction Probability:")
print(df[["section", "total_marks",
"prediction_probability"]].head(10))
```

```
Updated DataFrame with Section-Wise Prediction Probability:
  section  total_marks  prediction_probability
0   ALPHA         72.0                    0.60
1   ALPHA         84.0                    0.70
2   ALPHA        102.0                    0.86
3   ALPHA         94.0                    0.79
4   ALPHA        111.0                    0.94
5   ALPHA         85.0                    0.71
6   ALPHA         94.0                    0.79
7   ALPHA        105.0                    0.89
8   ALPHA         82.0                    0.68
9   ALPHA        112.0                    0.95
```

```
 import pandas as pd
from tabulate import tabulate

file_path = "MIDMARKS.xlsx"
df = pd.read_excel(file_path, sheet_name="SEM2 MID 1 - ALPHA")
subject_columns = df.columns[2:]

df[subject_columns] = df[subject_columns].apply(pd.to_numeric,
errors='coerce')

average_marks = df.groupby("SECTION")[subject_columns].mean()
```

```
subject_percentage = (average_marks / 20) * 100

pass_marks = 10
pass_percentage = df.groupby("SECTION")[subject_columns].apply(lambda
x: (x >= pass_marks).mean() * 100)

predicted_probability = pass_percentage / 100

print("\n Average Percentage for Each Subject by Section:")
print(tabulate(subject_percentage, headers="keys",
tablefmt="fancy_grid"))

print("\n Pass Percentage for Each Subject by Section:")
print(tabulate(pass_percentage, headers="keys",
tablefmt="fancy_grid"))

print("\n Predicted Probability of Passing for Each Subject by
Section:")
print(tabulate(predicted_probability, headers="keys",
tablefmt="fancy_grid"))
```

```
 Average Percentage for Each Subject by Section:
╒═══════════╤═══════════╤═══════════╤═══════════╤═══════════╤═══════════╤═══
═══╕
│ SECTION   │      DV │    M-II │      PP │    BEEE │      FL │
FIMS │
╞═══════════╪═══════════╪═══════════╪═══════════╪═══════════╪═══════════╪═══
═══╡
│ ALPHA     │ 67.0556 │ 68.5556 │ 80.5618 │ 78.9888 │ 81.7978 │
64.2135 │
├───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───
──┤
│ BETA      │ 65      │ 60.6111 │ 80.7303 │ 54.8876 │ 80.7865 │
70.2222 │
├───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───
──┤
│ DELTA     │ 70.9091 │ 48.5795 │ 62.809  │ 48.3523 │ 73.5955 │
84.7727 │
├───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───
──┤
│ EPSILON   │ 76.954  │ 33.8506 │ 43.3333 │ 72.9885 │ 77.4713 │
61.0345 │
├───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───
──┤
│ GAMMA     │ 76.6092 │ 48.046  │ 55.7471 │ 76.4943 │ 79.9432 │
65.0581 │
├───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼───
──┤
│ OMEGA     │ 84.6591 │ 42.2727 │ 75.5747 │ 72.4405 │ 82.7647 │
```

81.8675 |
├──────────┼──────────┼──────────┼──────────┼──────────┼──────────┼──────────┤
| SIGMA | 83.4167 | 65.3333 | 76.3559 | 68.5 | 84.3333 |
82.5862 |
├──────────┼──────────┼──────────┼──────────┼──────────┼──────────┼──────────┤
| ZETA | 77.1348 | 36.7614 | 47.1023 | 73.8068 | 75.5056 |
70.5172 |
╞══════════╪══════════╪══════════╪══════════╪══════════╪══════════╪══════════╡

 Pass Percentage for Each Subject by Section:
╞══════════╪══════════╪══════════╪══════════╪══════════╪══════════╪══════════╡
| SECTION | DV | M-II | PP | BEEE | FL |
FIMS |
╞══════════╪══════════╪══════════╪══════════╪══════════╪══════════╪══════════╡
| ALPHA | 81.1111 | 80 | 86.6667 | 91.1111 | 98.8889 |
75.5556 |
├──────────┼──────────┼──────────┼──────────┼──────────┼──────────┼──────────┤
| BETA | 81.1111 | 72.2222 | 91.1111 | 54.4444 | 98.8889 |
87.7778 |
├──────────┼──────────┼──────────┼──────────┼──────────┼──────────┼──────────┤
| DELTA | 84.4444 | 50 | 68.8889 | 48.8889 | 98.8889 |
95.5556 |
├──────────┼──────────┼──────────┼──────────┼──────────┼──────────┼──────────┤
| EPSILON | 95.4545 | 27.2727 | 44.3182 | 80.6818 | 90.9091 | 75
|
├──────────┼──────────┼──────────┼──────────┼──────────┼──────────┼──────────┤
| GAMMA | 87.7778 | 48.8889 | 60 | 81.1111 | 97.7778 |
74.4444 |
├──────────┼──────────┼──────────┼──────────┼──────────┼──────────┼──────────┤
| OMEGA | 88.8889 | 35.5556 | 82.2222 | 76.6667 | 86.6667 |
92.2222 |
├──────────┼──────────┼──────────┼──────────┼──────────┼──────────┼──────────┤
| SIGMA | 90.4762 | 68.254 | 80.9524 | 73.0159 | 95.2381 |
87.3016 |
├──────────┼──────────┼──────────┼──────────┼──────────┼──────────┼──────────┤
| ZETA | 91.1111 | 30 | 44.4444 | 80 | 83.3333 |
77.7778 |

```
╚═══════════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════
═══╝

 Predicted Probability of Passing for Each Subject by Section:
╒═══════════╤═══════════╤═══════════╤═══════════╤═══════════╤═══════════╤═══════
═══════╕
│ SECTION   │        DV │      M-II │        PP │      BEEE │        FL │
FIMS │
╞═══════════╪═══════════╪═══════════╪═══════════╪═══════════╪═══════════╪═══════
═══════╡
│ ALPHA     │ 0.811111  │ 0.8       │ 0.866667  │ 0.911111  │ 0.988889  │
0.755556 │
├───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼──
──────┤
│ BETA      │ 0.811111  │ 0.722222  │ 0.911111  │ 0.544444  │ 0.988889  │
0.877778 │
├───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼──
──────┤
│ DELTA     │ 0.844444  │ 0.5       │ 0.688889  │ 0.488889  │ 0.988889  │
0.955556 │
├───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼──
──────┤
│ EPSILON   │ 0.954545  │ 0.272727  │ 0.443182  │ 0.806818  │ 0.909091  │
0.75     │
├───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼──
──────┤
│ GAMMA     │ 0.877778  │ 0.488889  │ 0.6       │ 0.811111  │ 0.977778  │
0.744444 │
├───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼──
──────┤
│ OMEGA     │ 0.888889  │ 0.355556  │ 0.822222  │ 0.766667  │ 0.866667  │
0.922222 │
├───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼──
──────┤
│ SIGMA     │ 0.904762  │ 0.68254   │ 0.809524  │ 0.730159  │ 0.952381  │
0.873016 │
├───────────┼───────────┼───────────┼───────────┼───────────┼───────────┼──
──────┤
│ ZETA      │ 0.911111  │ 0.3       │ 0.444444  │ 0.8       │ 0.833333  │
0.777778 │
╘═══════════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════════╧═══════
═══════╛
```

```
df[df["SECTION"] == "ALPHA"].mean(numeric_only=True)
```

```
S.NO    45.500000
DV      13.411111
M-II    13.711111
PP      16.112360
BEEE    15.797753
```

```
FL      16.359551
FIMS    12.842697
dtype: float64
```

```
df[df["SECTION"] == "ALPHA"].select_dtypes(include="number").mean()
df[df["SECTION"] == "ALPHA"].apply(pd.to_numeric,
errors="coerce").mean()
```

```
S.NO       45.500000
SECTION          NaN
DV         13.411111
M-II       13.711111
PP         16.112360
BEEE       15.797753
FL         16.359551
FIMS       12.842697
dtype: float64
```

```
df[df["SECTION"] == "ALPHA"].std(numeric_only=True)
```

```
S.NO    26.124701
DV       4.991891
M-II     5.595432
PP       5.095538
BEEE     4.530653
FL       3.415364
FIMS     4.314086
dtype: float64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 718 entries, 0 to 717
Data columns (total 8 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   S.NO     601 non-null    float64
 1   SECTION  691 non-null    object
 2   DV       705 non-null    float64
 3   M-II     704 non-null    float64
 4   PP       701 non-null    float64
 5   BEEE     697 non-null    float64
 6   FL       702 non-null    float64
 7   FIMS     694 non-null    float64
dtypes: float64(7), object(1)
memory usage: 45.0+ KB
```

```
import pandas as pd
from scipy.stats import ttest_ind
```

```
print(df['DV'].dtype)
print(df[['DV', ]].head())
```

```
float64
     DV
0  12.0
1  19.0
2  18.0
3  15.0
4  18.0
```

```
df[df['SECTION']=="ALPHA"]["DV"]
```

```
0     12.0
1     19.0
2     18.0
3     15.0
4     18.0
      ...
85     3.0
86    17.0
87    13.0
88     2.0
89    10.0
Name: DV, Length: 90, dtype: float64
```

```
import pandas as pd
from scipy.stats import ttest_ind

ttest_ind(df[df["SECTION"] == "ALPHA"]['DV'],df[df["SECTION"] ==
"BETA"]['DV'] )
```

```
TtestResult(statistic=0.6207084248259586, pvalue=0.5355854399866022,
df=178.0)
```

```
import pandas as pd
from scipy.stats import ttest_rel

ttest_rel(df[df["SECTION"] == "ALPHA"]['DV'],df[df["SECTION"] ==
"BETA"]['DV'] )
```

```
TtestResult(statistic=0.6677896583545824, pvalue=0.5059958752914141,
df=89)
```

```
from scipy.stats import chi2_contingency


data = [df[df["SECTION"] == "ALPHA"]['DV'],df[df["SECTION"] == "BETA"]
['DV']]
stat, p, dof, expected = chi2_contingency(data)
```

```
alpha = 0.05
print("p value is " + str(p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (H0 holds true)')
```

```
p value is 0.0011991006253874233
Dependent (reject H0)
```

```
stat
```

```
135.02226175700022
```

```
p
```

```
0.0011991006253874233
```

```
dof
```

```
89
```

```
expected
```

```
array([[14.72570467, 16.75683635, 14.72570467, 14.72570467,
15.23348759,
        16.75683635, 11.67900715, 12.69457299,  8.12452671,
13.71013883,
        13.71013883, 16.75683635, 13.20235591, 14.21792175,
18.28018511,
        12.69457299, 12.18679007, 11.17122423, 12.18679007,
16.24905343,
         6.09339504, 12.18679007, 16.75683635, 17.77240219,
13.20235591,
        13.20235591, 15.23348759,  8.63230963, 14.21792175,
11.67900715,
        10.66344131, 10.15565839, 13.71013883, 13.71013883,
16.75683635,
         6.60117796, 11.17122423, 11.17122423, 15.23348759,
18.78796803,
        17.26461927, 16.75683635, 18.28018511, 14.21792175,
17.26461927,
        10.15565839, 15.23348759, 18.28018511, 13.20235591,
10.15565839,
         6.09339504,  9.14009255, 16.75683635, 11.67900715,
17.26461927,
        12.69457299, 11.17122423,  5.58561212, 13.20235591,
16.24905343,
        17.77240219, 12.18679007, 15.23348759, 13.71013883,
11.17122423,
        10.15565839, 13.71013883, 15.23348759, 15.74127051,
```

```
17.77240219,
         5.58561212, 18.78796803, 10.15565839, 12.18679007,
16.24905343,
        10.66344131, 16.75683635, 15.23348759, 18.28018511,
14.21792175,
        19.80353387, 14.21792175,  4.57004628, 13.71013883,
11.67900715,
         8.63230963, 11.67900715, 12.69457299,  9.14009255,
12.69457299],
       [14.27429533, 16.24316365, 14.27429533, 14.27429533,
14.76651241,
        16.24316365, 11.32099285, 12.30542701,  7.87547329,
13.28986117,
        13.28986117, 16.24316365, 12.79764409, 13.78207825,
17.71981489,
        12.30542701, 11.81320993, 10.82877577, 11.81320993,
15.75094657,
         5.90660496, 11.81320993, 16.24316365, 17.22759781,
12.79764409,
        12.79764409, 14.76651241,  8.36769037, 13.78207825,
11.32099285,
        10.33655869,  9.84434161, 13.28986117, 13.28986117,
16.24316365,
         6.39882204, 10.82877577, 10.82877577, 14.76651241,
18.21203197,
        16.73538073, 16.24316365, 17.71981489, 13.78207825,
16.73538073,
         9.84434161, 14.76651241, 17.71981489, 12.79764409,
9.84434161,
         5.90660496,  8.85990745, 16.24316365, 11.32099285,
16.73538073,
        12.30542701, 10.82877577,  5.41438788, 12.79764409,
15.75094657,
        17.22759781, 11.81320993, 14.76651241, 13.28986117,
10.82877577,
         9.84434161, 13.28986117, 14.76651241, 15.25872949,
17.22759781,
         5.41438788, 18.21203197,  9.84434161, 11.81320993,
15.75094657,
        10.33655869, 16.24316365, 14.76651241, 17.71981489,
13.78207825,
        19.19646613, 13.78207825,  4.42995372, 13.28986117,
11.32099285,
         8.36769037, 11.32099285, 12.30542701,  8.85990745,
12.30542701]])

from scipy import stats

t_statistic, p_value = stats.ttest_1samp(df[df['SECTION'] == 'ALPHA']
['DV'], df['DV'].mean())
```

```
print(f"T-statistic: {t_statistic}, P-value: {p_value}")
```

```
T-statistic: -3.0546455966439896, P-value: 0.002972712305590382
```

```
import scipy.stats as stats

dv_alpha = df[df['SECTION'] == 'ALPHA']['DV']
dv_beta = df[df['SECTION'] == 'BETA']['DV']

t_statistic, p_value = stats.ttest_ind(dv_alpha, dv_beta,
equal_var=False)  # Welch's t-test

print("T-statistic:", t_statistic)
print("P-value:", p_value)
```

```
T-statistic: 0.6207084248259586
P-value: 0.5356400219163465
```