```
In [104]: import pandas as pd
          df=pd.read_excel("MIDMARKS-MINOR1-EXAM.xlsx")
          df
```

```
Out[104]:
          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
     ...   ...      ...  ...  ...  ...  ...  ...   ...
     475   476      NaN  18     2  12     3  17    15
     476   477      NaN  20     6  16    11  20    14
     477   478      NaN  20   NaN  18    13  20    18
     478   479      NaN  20    20   5    19  18    14
     479   480      NaN  20    16  18    19  20    19

    480 rows × 8 columns
```

In [105]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 480 entries, 0 to 479
Data columns (total 8 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   S.NO     480 non-null    int64 
 1   SECTION  439 non-null    object
 2   DV       479 non-null    object
 3   M-II     477 non-null    object
 4   PP       480 non-null    object
 5   BEEE     478 non-null    object
 6   FL       479 non-null    object
 7   FIMS     480 non-null    object
dtypes: int64(1), object(7)
memory usage: 30.1+ KB
```

In [106]:

```
df.describe()
```

Out[106]:

	S.NO
count	480.000000
mean	240.500000
std	138.708327
min	1.000000
25%	120.750000
50%	240.500000
75%	360.250000
max	480.000000

In [107]:

```
df.dtypes
```

Out[107]:

```
S.NO          int64
SECTION      object
DV           object
M-II         object
PP           object
BEEE         object
FL           object
FIMS         object
dtype: object
```

In [108]:

```
df.isnull().sum()
```

Out[108]:

```
S.NO           0
SECTION       41
DV             1
M-II           3
PP             0
BEEE           2
FL             1
FIMS           0
dtype: int64
```

```
In [109]: df['PP'].value_counts()
```

```
Out[109]: 20    70
          18    35
          19    35
          17    31
          12    29
          16    28
          14    28
          11    28
          15    27
          9     24
          10    19
          6     18
          13    15
          5     15
          3     13
          2     13
          8     12
          4     10
          7     10
          1      7
          A      6
          0      3
          AB     3
          MP     1
          Name: PP, dtype: int64
```

In [110]:

```
df['DV'].value_counts()
```

Out[110]:

```
17    53
20    53
18    48
16    48
15    45
19    38
11    31
12    27
13    25
14    24
10    22
9     14
8     10
6      9
5      8
7      6
A      6
2      4
1      3
4      3
3      1
MP     1
Name: DV, dtype: int64
```

In [111]:
```
df['FIMS'].value_counts()
```

Out[111]:
```
18    62
15    57
16    50
17    41
14    40
13    36
19    35
9     28
11    22
12    20
10    19
20    12
8     11
AB     8
A      6
3      6
6      5
4      5
7      5
5      5
2      3
1      3
0      1
Name: FIMS, dtype: int64
```

In [112]:

```
df['M-II'].value_counts()
```

Out[112]:

```
20    44
3     34
17    32
8     29
0     24
12    24
11    24
15    24
5     23
18    23
4     22
10    20
6     18
13    18
9     17
14    17
1     17
16    16
7     14
2     13
19    12
AB     5
o      3
A      2
II     1
I      1
Name: M-II, dtype: int64
```

In [113]:

```
df['BEEE'].value_counts()
```

Out[113]:

```
20    76
17    46
19    42
18    31
11    31
15    28
16    23
12    21
14    21
10    20
9     19
6     15
7     15
13    14
3     14
A     13
8     13
4     12
5     10
2      9
1      3
o      1
0      1
Name: BEEE, dtype: int64
```

In [114]:

```
df['FL'].value_counts()
```

Out[114]:

```
20    121
15     85
18     59
10     55
13     50
19     34
16     15
14     11
11     10
17      9
A       9
12      8
8       6
9       3
6       2
7       2
Name: FL, dtype: int64
```

In [115]:

```
df.rename(columns={"M-II":"M2"},inplace=True)
df
```

Out[115]:

	S.NO	SECTION	DV	M2	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
...	...	...	...	...	...	...	...	...
475	476	NaN	18	2	12	3	17	15
476	477	NaN	20	6	16	11	20	14
477	478	NaN	20	NaN	18	13	20	18
478	479	NaN	20	20	5	19	18	14
479	480	NaN	20	16	18	19	20	19

480 rows × 8 columns

In [116]:

```
df["SECTION"].value_counts()
```

Out[116]:

```
ALPHA      60
BETA       60
DELTA      60
EPSILON    60
GAMMA      60
OMEGA      60
SIGMA      60
ZETA       19
Name: SECTION, dtype: int64
```

```
In [117]: df["SECTION"]=df["SECTION"].fillna("ZETA")
```

```
In [118]: df["SECTION"].value_counts()
```

```
Out[118]: ALPHA      60
          BETA       60
          DELTA      60
          ZETA       60
          EPSILON    60
          GAMMA      60
          OMEGA      60
          SIGMA      60
          Name: SECTION, dtype: int64
```

```
In [119]: df[df['DV'].isnull()]
```

Out[119]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS
389	390	OMEGA	NaN	17	17	19	20	17

```
In [120]: df[df['M2'].isnull()]
```

Out[120]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS
227	228	EPSILON	11	NaN	10	12	10	16
323	324	SIGMA	9	NaN	2	3	11	1
477	478	ZETA	20	NaN	18	13	20	18

```
In [121]: df[df['BEEE'].isnull()]
```

Out[121]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS
192	193	EPSILON	16	18	15	NaN	18	18
439	440	EPSILON	20	16	20	NaN	20	18

In [122]:

```
df[df['FL'].isnull()]
```

Out[122]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS
102	103	BETA	10	12	17	9	NaN	16

In [123]:

```
df.replace('A',0,inplace=True)
df.replace('AB',0,inplace=True)
df.replace('MP',0,inplace=True)
df.replace('I',1,inplace=True)
df.replace('II',11,inplace=True)
df.replace('o',0,inplace=True)
df
```

Out[123]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS
0	1	ALPHA	12.0	0.0	17	9.0	19.0	15
1	2	ALPHA	19.0	12.0	16	16.0	18.0	3
2	3	ALPHA	18.0	14.0	18	18.0	18.0	16
3	4	ALPHA	15.0	9.0	19	17.0	19.0	15
4	5	ALPHA	18.0	17.0	19	19.0	20.0	18
...	...	...	...	...	...	...	...	...
475	476	ZETA	18.0	2.0	12	3.0	17.0	15
476	477	ZETA	20.0	6.0	16	11.0	20.0	14
477	478	ZETA	20.0	NaN	18	13.0	20.0	18
478	479	ZETA	20.0	20.0	5	19.0	18.0	14
479	480	ZETA	20.0	16.0	18	19.0	20.0	19

480 rows × 8 columns

In [124]:

```
subjects = ["DV", "M2", "PP", "BEEE", "FL", "FIMS"]
df[subjects] = df[subjects].apply(lambda row: row.fillna(row.mean()), axis=1)
print("Missing marks filled correctly with row-wise mean.")
```

```
Missing marks filled correctly with row-wise mean.
```

In [125]:

```
df.iloc[102]
```

Out[125]:

```
S.NO         103
SECTION     BETA
DV          10.0
M2          12.0
PP          17.0
BEEE         9.0
FL          12.8
FIMS        16.0
Name: 102, dtype: object
```

```
In [126]: df[subjects] = df[subjects].astype(int)
          df
```

```
Out[126]:
       S.NO  SECTION  DV  M2  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
  ...   ...      ...  ..  ..  ..   ...  ..   ...
  475   476     ZETA  18   2  12     3  17    15
  476   477     ZETA  20   6  16    11  20    14
  477   478     ZETA  20  17  18    13  20    18
  478   479     ZETA  20  20   5    19  18    14
  479   480     ZETA  20  16  18    19  20    19

480 rows × 8 columns
```

```
In [127]: df.dtypes
```

```
Out[127]: S.NO        int64
          SECTION    object
          DV          int32
          M2          int32
          PP          int32
          BEEE        int32
          FL          int32
          FIMS        int32
          dtype: object
```

In [152]:

```
df["Total"]=df['DV']+df['M2']+df['PP']+df['BEEE']+df['FL']+df['FIMS']
df
```

Out[152]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total
0	1	ALPHA	12	0	17	9	19	15	72
1	2	ALPHA	19	12	16	16	18	3	84
2	3	ALPHA	18	14	18	18	18	16	102
3	4	ALPHA	15	9	19	17	19	15	94
4	5	ALPHA	18	17	19	19	20	18	111
...	...	...	...	...	...	...	...	...	...
475	476	ZETA	18	2	12	3	17	15	67
476	477	ZETA	20	6	16	11	20	14	87
477	478	ZETA	20	17	18	13	20	18	106
478	479	ZETA	20	20	5	19	18	14	96
479	480	ZETA	20	16	18	19	20	19	112

480 rows × 9 columns

In [154]:

```
df["Total"].value_counts()
```

Out[154]:

```
81     13
86     13
96     13
95     12
103    12
       ..
9       1
7       1
6       1
14      1
50      1
Name: Total, Length: 96, dtype: int64
```

In [156]:

```
def grade(total):
    if total>=110:
        return "O"
    elif total>=100:
        return "A+"
    elif total>=90:
        return "A"
    elif total>=80:
        return "B+"
    elif total>=70:
        return "B"
    elif total>=60:
        return "C+"
    elif total>=50:
        return "C"
    else:
        return "F"
df["Grade"]=df["Total"].apply(grade)
df
```

```
Out[156]:
       S.NO  SECTION  DV  M2  PP  BEEE  FL  FIMS  Total  Grade
    0     1    ALPHA  12   0  17     9  19    15     72      B
    1     2    ALPHA  19  12  16    16  18     3     84     B+
    2     3    ALPHA  18  14  18    18  18    16    102     A+
    3     4    ALPHA  15   9  19    17  19    15     94      A
    4     5    ALPHA  18  17  19    19  20    18    111      O
  ...   ...      ...  ...  ...  ...   ...  ...   ...    ...    ...
  475   476     ZETA  18   2  12     3  17    15     67     C+
  476   477     ZETA  20   6  16    11  20    14     87     B+
  477   478     ZETA  20  17  18    13  20    18    106     A+
  478   479     ZETA  20  20   5    19  18    14     96      A
  479   480     ZETA  20  16  18    19  20    19    112      O

480 rows × 10 columns
```

In [166]:

```
subjects = ["DV", "M2", "PP", "BEEE", "FL", "FIMS"]
df["Backlogs"] = (df[subjects] < 10).sum(axis=1)
df
```

Out[166]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Grade	Backlogs
0	1	ALPHA	12	0	17	9	19	15	72	B	2
1	2	ALPHA	19	12	16	16	18	3	84	B+	1
2	3	ALPHA	18	14	18	18	18	16	102	A+	0
3	4	ALPHA	15	9	19	17	19	15	94	A	1
4	5	ALPHA	18	17	19	19	20	18	111	O	0
...	...	...	...	...	...	...	...	...	...	...	...
475	476	ZETA	18	2	12	3	17	15	67	C+	2
476	477	ZETA	20	6	16	11	20	14	87	B+	1
477	478	ZETA	20	17	18	13	20	18	106	A+	0
478	479	ZETA	20	20	5	19	18	14	96	A	1
479	480	ZETA	20	16	18	19	20	19	112	O	0

480 rows × 11 columns

In [168]:

```
def programming(pp):
    if pp>18:
        return "Very Good"
    elif pp>15:
        return "Good"
    elif pp>10:
        return "Average"
    else:
        return "Poor"

df["Programming_Skills"]=df["PP"].apply(programming)
df
```

Out[168]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Grade	Backlogs	Programming_Skills
0	1	ALPHA	12	0	17	9	19	15	72	B	2	Good
1	2	ALPHA	19	12	16	16	18	3	84	B+	1	Good
2	3	ALPHA	18	14	18	18	18	16	102	A+	0	Good
3	4	ALPHA	15	9	19	17	19	15	94	A	1	Very Good
4	5	ALPHA	18	17	19	19	20	18	111	O	0	Very Good
...	...	...	...	...	...	...	...	...	...	...	...	...
475	476	ZETA	18	2	12	3	17	15	67	C+	2	Average
476	477	ZETA	20	6	16	11	20	14	87	B+	1	Good
477	478	ZETA	20	17	18	13	20	18	106	A+	0	Good
478	479	ZETA	20	20	5	19	18	14	96	A	1	Poor
479	480	ZETA	20	16	18	19	20	19	112	O	0	Good

480 rows × 12 columns

In [170]:

```
def analytical(dv):
    if dv>18:
        return "Very Good"
    elif dv>15:
        return "Good"
    elif dv>10:
        return "Average"
    else:
        return "Poor"

df["Analytical_Skills"]=df["DV"].apply(programming)
df
```

Out[170]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Grade	Backlogs	Programming_Skills	Analytical_Skills
0	1	ALPHA	12	0	17	9	19	15	72	B	2	Good	Average
1	2	ALPHA	19	12	16	16	18	3	84	B+	1	Good	Very Good
2	3	ALPHA	18	14	18	18	18	16	102	A+	0	Good	Good
3	4	ALPHA	15	9	19	17	19	15	94	A	1	Very Good	Average
4	5	ALPHA	18	17	19	19	20	18	111	O	0	Very Good	Good
...	...	...	...	...	...	...	...	...	...	...	...	...	...
475	476	ZETA	18	2	12	3	17	15	67	C+	2	Average	Good
476	477	ZETA	20	6	16	11	20	14	87	B+	1	Good	Very Good
477	478	ZETA	20	17	18	13	20	18	106	A+	0	Good	Very Good
478	479	ZETA	20	20	5	19	18	14	96	A	1	Poor	Very Good
479	480	ZETA	20	16	18	19	20	19	112	O	0	Good	Very Good

480 rows × 13 columns

In [190]:

```
df.isnull().sum()
```

Out[190]:

```
S.NO                  0
SECTION               0
DV                    0
M2                    0
PP                    0
BEEE                  0
FL                    0
FIMS                  0
Total                 0
Grade                 0
Backlogs              0
Programming_Skills    0
Analytical_Skills     0
dtype: int64
```

In [174]:

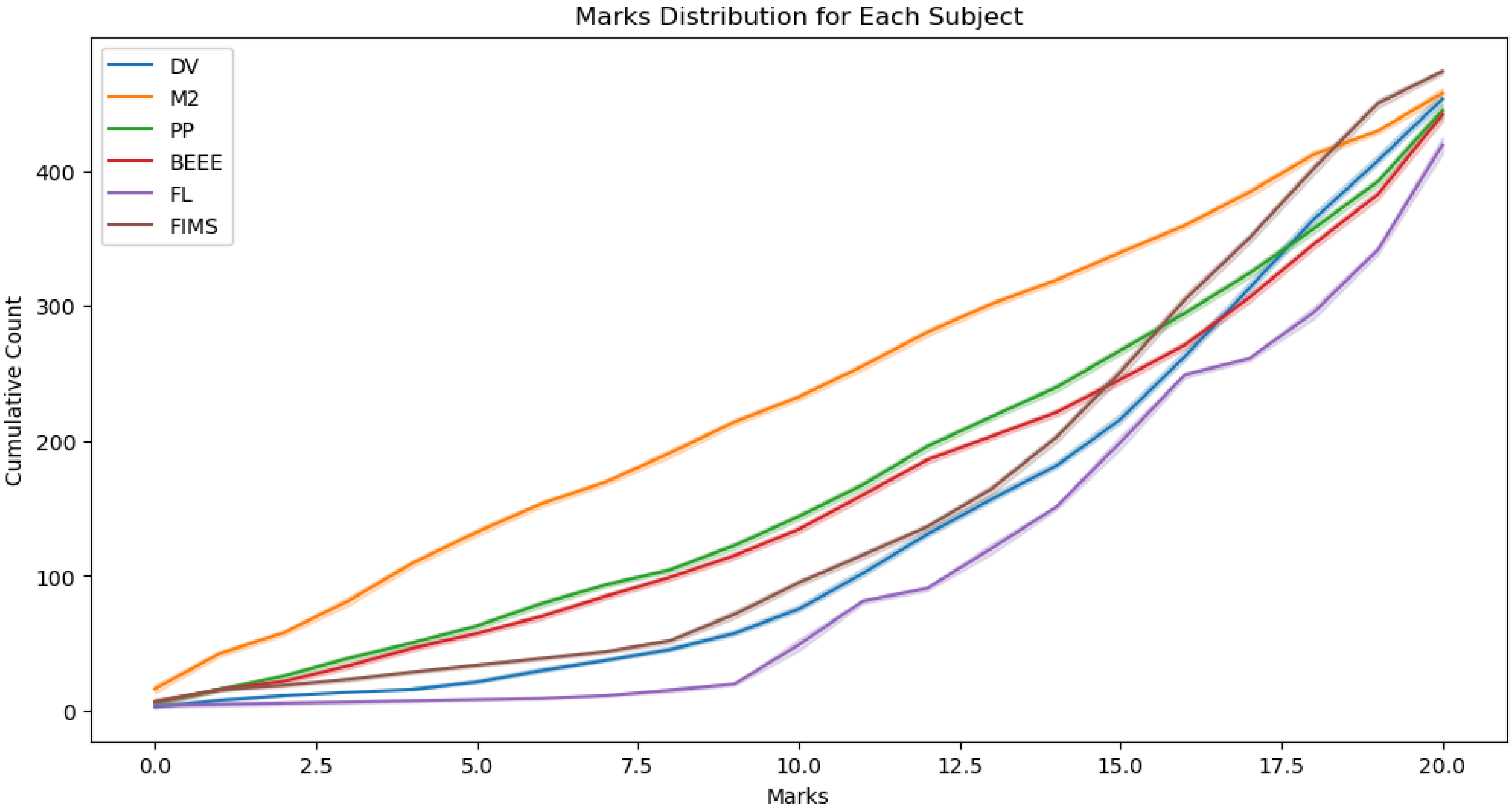
```
import matplotlib.pyplot as plt
import seaborn as sns
```

In [178]:

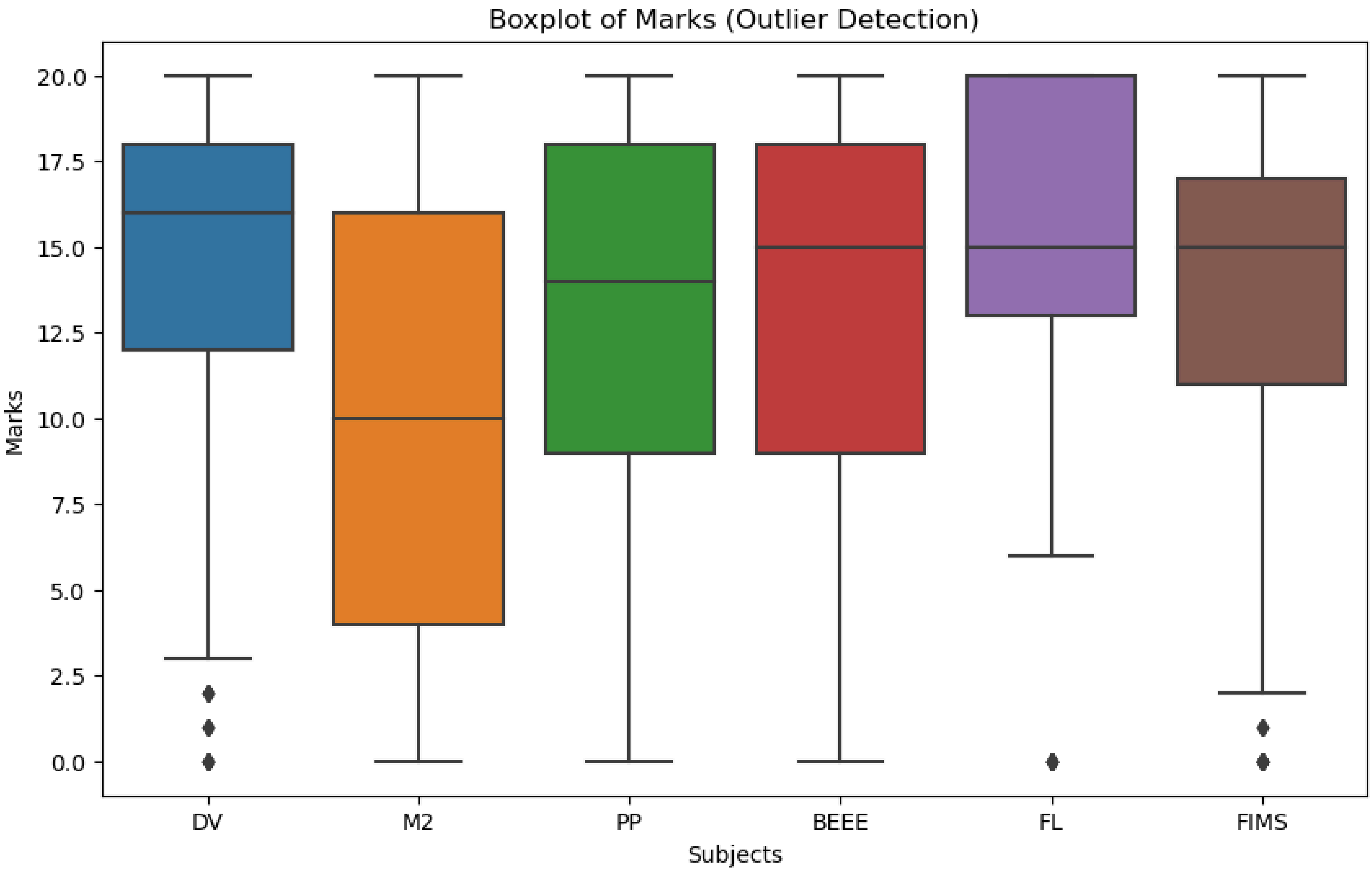
```
plt.figure(figsize=(12, 6))

for subject in subjects:
    sns.lineplot(x=sorted(df[subject]), y=range(len(df)), label=subject)

plt.title("Marks Distribution for Each Subject")
plt.xlabel("Marks")
plt.ylabel("Cumulative Count")
plt.legend()
plt.show()
```

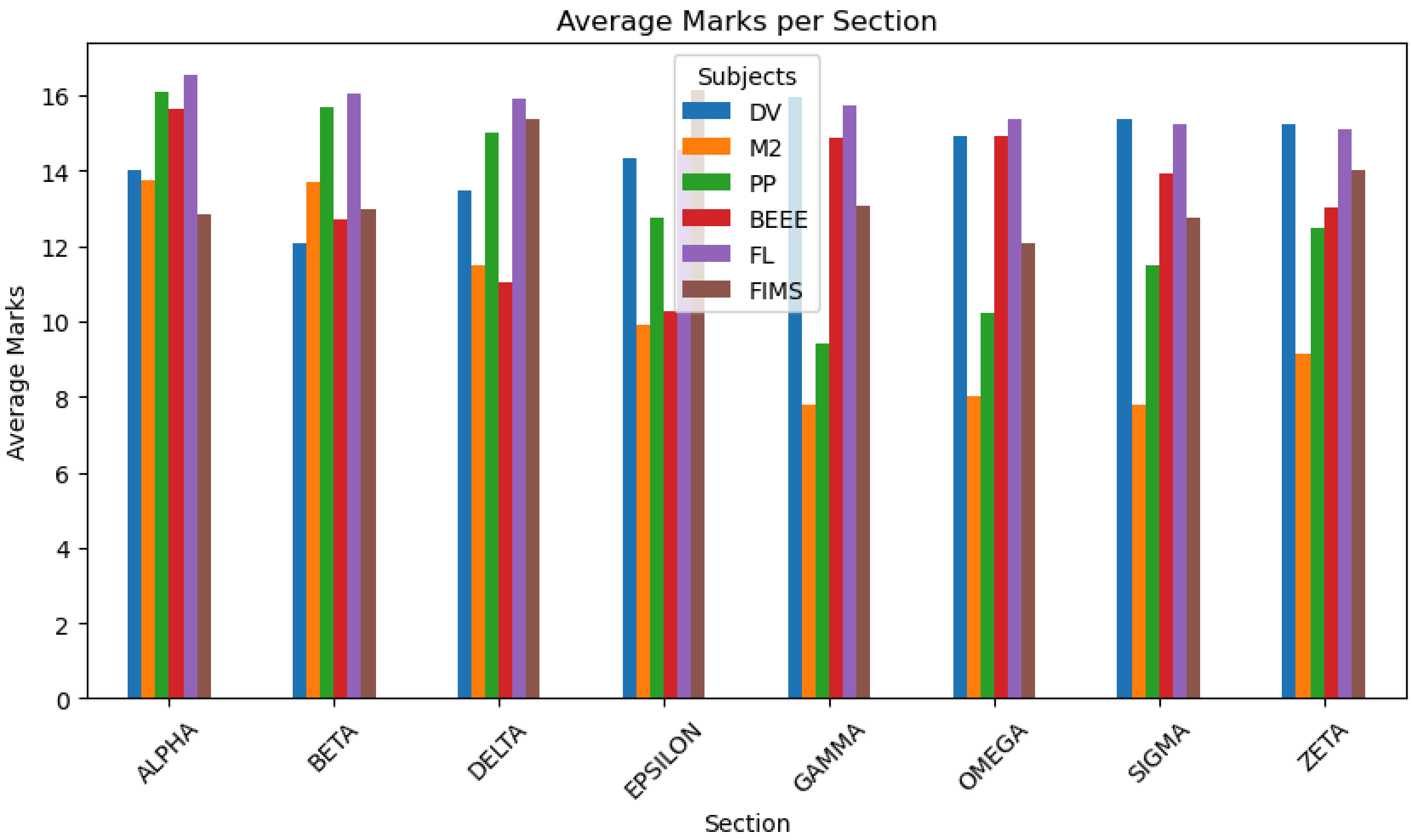


```
In [182]: plt.figure(figsize=(10, 6))
          sns.boxplot(data=df[subjects])
          plt.title("Boxplot of Marks (Outlier Detection)")
          plt.xlabel("Subjects")
          plt.ylabel("Marks")
          plt.show()
```



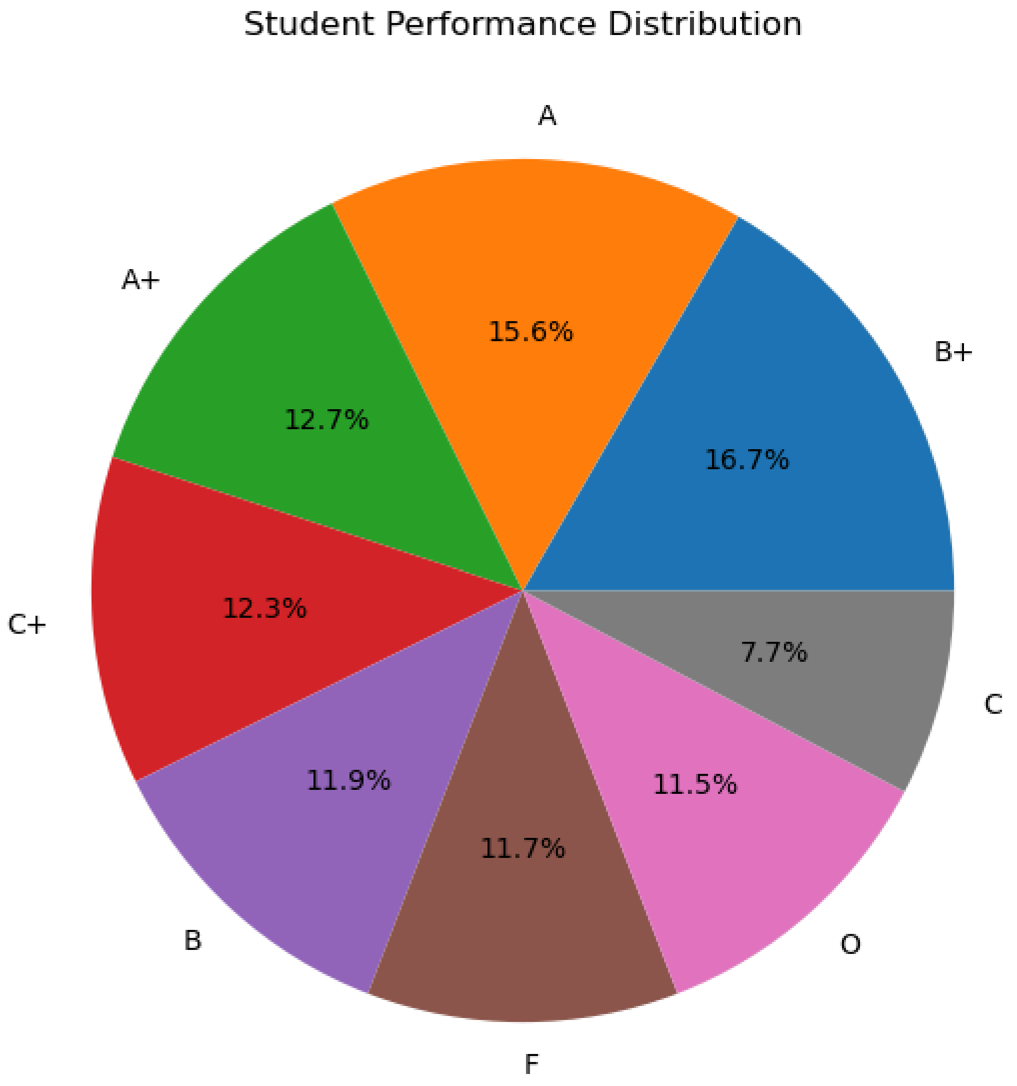
```
In [188]: plt.figure(figsize=(10, 5))
          df.groupby("SECTION")[subjects].mean().plot(kind="bar", figsize=(10, 5))
          plt.title("Average Marks per Section")
          plt.xlabel("Section")
          plt.ylabel("Average Marks")
          plt.xticks(rotation=45)
          plt.legend(title="Subjects")
          plt.show()
```

```
<Figure size 1000x500 with 0 Axes>
```



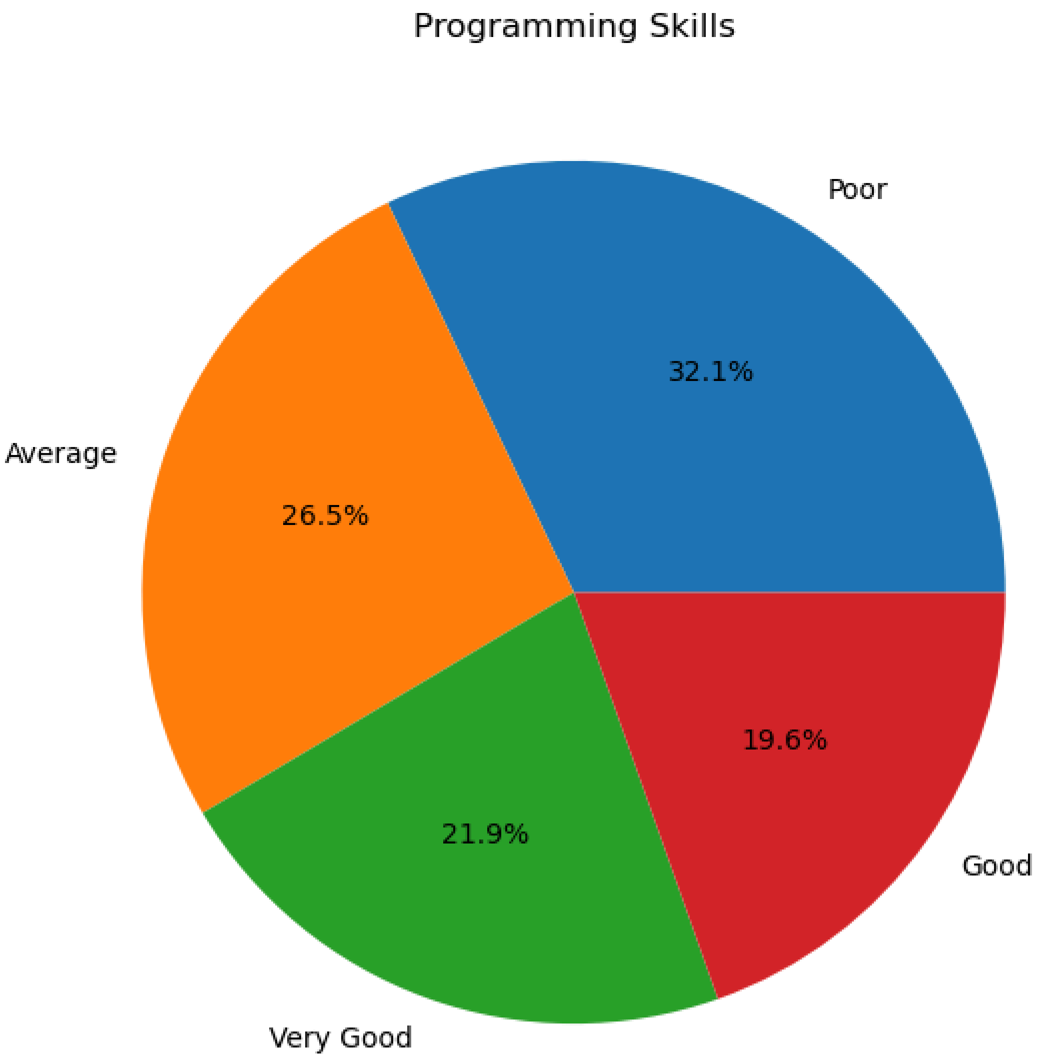
In [192]:

```
grades= df["Grade"].value_counts()
plt.figure(figsize=(7, 7))
plt.pie(grades, labels=grades.index, autopct="%1.1f%%")
plt.title("Student Performance Distribution")
plt.show()
```



In [198]:

```
programming_skills=df['Programming_Skills'].value_counts()
plt.figure(figsize=(7, 7))
plt.pie(programming_skills, labels=programming_skills.index, autopct="%1.1f%%")
plt.title("Programming Skills")
plt.show()
```



In [204]:

```
def pass_fail(x):
    if x==0:
        return "Pass"
    else:
        return "Fail"

df["Pass/Fail"]=df["Backlogs"].apply(pass_fail)
df
```

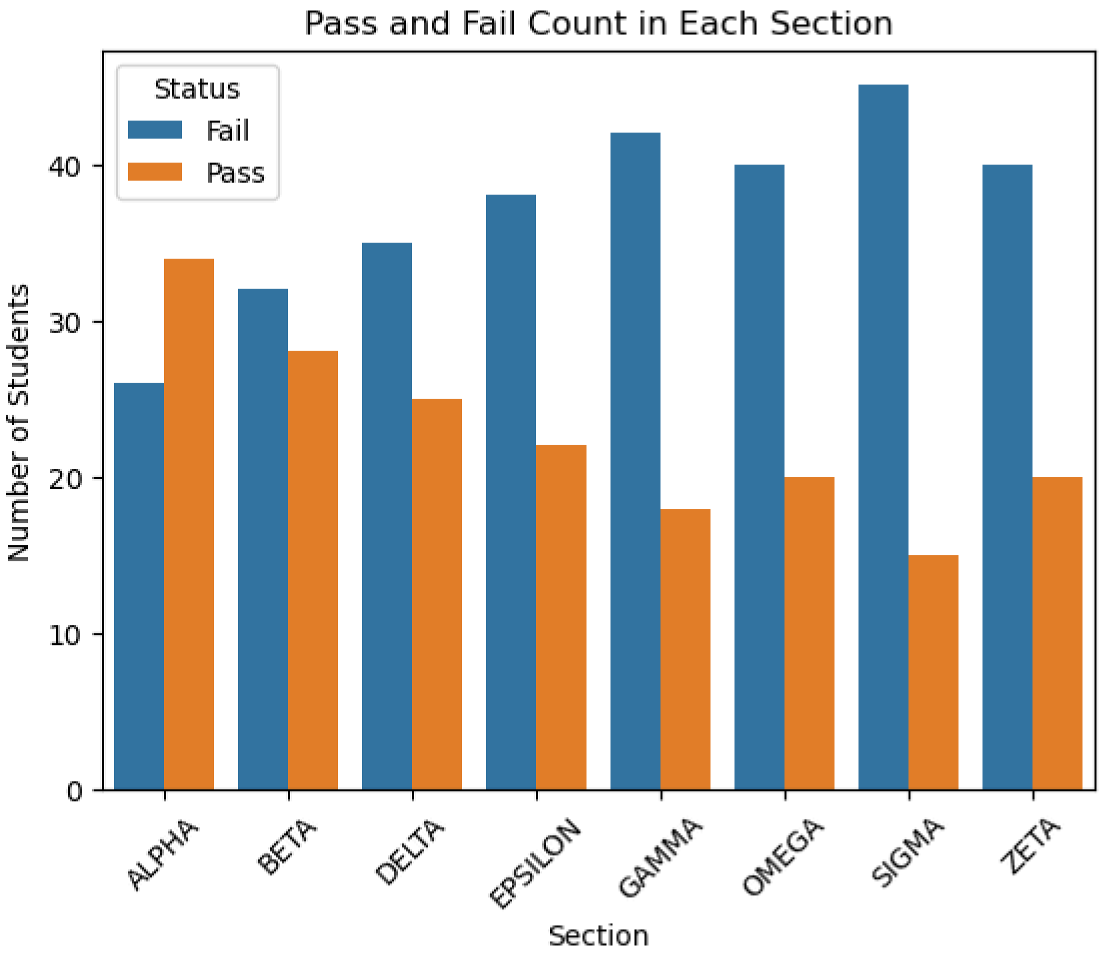
Out[204]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Grade	Backlogs	Programming_Skills	Analytical_Skills	Pass/Fail
0	1	ALPHA	12	0	17	9	19	15	72	B	2	Good	Average	Fail
1	2	ALPHA	19	12	16	16	18	3	84	B+	1	Good	Very Good	Fail
2	3	ALPHA	18	14	18	18	18	16	102	A+	0	Good	Good	Pass
3	4	ALPHA	15	9	19	17	19	15	94	A	1	Very Good	Average	Fail
4	5	ALPHA	18	17	19	19	20	18	111	O	0	Very Good	Good	Pass
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
475	476	ZETA	18	2	12	3	17	15	67	C+	2	Average	Good	Fail
476	477	ZETA	20	6	16	11	20	14	87	B+	1	Good	Very Good	Fail
477	478	ZETA	20	17	18	13	20	18	106	A+	0	Good	Very Good	Pass
478	479	ZETA	20	20	5	19	18	14	96	A	1	Poor	Very Good	Fail
479	480	ZETA	20	16	18	19	20	19	112	O	0	Good	Very Good	Pass

480 rows × 14 columns

In [214]:

```
pass_fail_counts=df.groupby(['SECTION','Pass/Fail']).size().reset_index(name="Count")
sns.barplot(x="SECTION", y="Count", hue="Pass/Fail", data=pass_fail_counts)
plt.title("Pass and Fail Count in Each Section")
plt.xlabel("Section")
plt.ylabel("Number of Students")
plt.xticks(rotation=45)
plt.legend(title="Status")
plt.show()
```



In [218]:

```
aggregated_data=df.groupby(['SECTION']).mean()
aggregated_data
```

```
C:\Users\subha\AppData\Local\Temp\ipykernel_8836\113419422.py:1: FutureWarning: The default value of numeric_only in
DataFrameGroupBy.mean is deprecated. In a future version, numeric_only will default to False. Either specify numeric_
only or select only columns which should be valid for the function.
  aggregated_data=df.groupby(['SECTION']).mean()
```

Out[218]:

	S.NO	DV	M2	PP	BEEE	FL	FIMS	Total	Backlogs
SECTION									
ALPHA	30.500000	14.033333	13.733333	16.066667	15.616667	16.550000	12.850000	88.850000	0.716667
BETA	90.500000	12.083333	13.683333	15.666667	12.716667	16.033333	12.983333	83.166667	1.133333
DELTA	150.500000	13.483333	11.466667	15.016667	11.050000	15.916667	15.350000	82.283333	1.216667
EPSILON	214.816667	14.333333	9.900000	12.750000	10.283333	14.566667	16.116667	77.950000	1.366667
GAMMA	270.500000	15.933333	7.800000	9.400000	14.866667	15.716667	13.050000	76.766667	1.533333
OMEGA	369.833333	14.900000	8.000000	10.216667	14.900000	15.350000	12.066667	75.433333	1.700000
SIGMA	370.166667	15.383333	7.783333	11.483333	13.916667	15.233333	12.750000	76.550000	1.616667
ZETA	427.183333	15.216667	9.133333	12.483333	13.000000	15.116667	14.000000	78.950000	1.650000

In [220]:

```
std_data=df.groupby(['SECTION']).std()
std_data
```

```
C:\Users\subha\AppData\Local\Temp\ipykernel_8836\475094170.py:1: FutureWarning: The default value of numeric_only in
DataFrameGroupBy.std is deprecated. In a future version, numeric_only will default to False. Either specify numeric_o
nly or select only columns which should be valid for the function.
  std_data=df.groupby(['SECTION']).std()
```

Out[220]:

	S.NO	DV	M2	PP	BEEE	FL	FIMS	Total	Backlogs
SECTION									
ALPHA	17.464249	4.654018	5.161351	5.085262	4.476271	3.402018	4.037221	20.844725	1.090664
BETA	17.464249	4.465657	5.484931	5.183634	6.031251	3.817740	4.343285	22.804376	1.395716
DELTA	17.464249	4.268496	6.091023	4.942008	5.664115	3.585714	3.545467	21.055288	1.316025
EPSILON	34.117688	4.082483	5.876411	5.130913	5.689990	4.393241	4.100813	23.889878	1.389631
GAMMA	17.464249	2.208356	5.885345	3.945390	4.537851	3.884309	4.350823	19.912833	1.346265
OMEGA	43.493295	4.803600	6.711993	6.311388	5.745153	4.884514	5.550589	29.319032	1.768845
SIGMA	43.493295	4.166418	5.740406	5.435264	5.630702	4.110205	4.714295	24.448788	1.574066
ZETA	54.155298	6.268273	6.662485	6.867984	6.762308	5.285387	5.474130	31.942971	1.857874

In [224]:

```
group1=df[df['SECTION']=='ALPHA']['DV']
print(group1)
```

```
0     12
1     19
2     18
3     15
4     18
5     17
6     15
7     17
8     10
9     18
10    17
11    20
12    16
13    17
14    19
15    13
16    15
17    11
18    14
19    19
20     4
21    14
22    17
23    20
24    15
25     6
26    17
27     5
28    19
29     8
30    11
31    12
32    17
33    14
34    17
35     8
36    11
37    15
38    19
39    20
40    18
```

```
41    16
42    16
43    11
44    18
45    11
46    14
47    16
48    16
49    15
50     1
51     6
52    17
53     8
54    14
55    15
56    10
57     2
58    10
59    19
Name: DV, dtype: int32
```

In [226]:

```
group2=df[df['SECTION']=='BETA']['DV']
print(group2)
```

```
60     19
61      8
62     12
63     11
64     12
65      9
66     12
67     12
68     16
69     20
70      4
71     17
72      7
73     10
74     17
75      5
76     17
77     13
78     19
79     19
80     19
81     18
82      2
83     10
84     12
85      3
86     17
87     13
88      2
89     10
90     17
91     14
92     11
93     14
94     12
95     16
96      8
97      8
98      6
99      9
100    10
```

```
101    13
102    10
103    11
104    17
105    12
106     9
107    11
108    10
109    13
110     8
111    10
112    16
113    15
114    11
115    20
116    13
117    12
118     9
119    15
Name: DV, dtype: int32
```

In [236]:

```
from scipy.stats import ttest_1samp
t_statistic,p_value=ttest_1samp(group1,14.41)
print(t_statistic,p_value)
if p_value<0.5:
    print("Reject H0")
else:
    print("Accept H0")
```

```
-0.6269093116996493 0.5331371479713868
Reject H0
```

In [238]:

```
from scipy.stats import ttest_ind
t_statistic,p_value=ttest_ind(group1, group2, equal_var=False)
print(t_statistic,p_value)
if p_value<0.5:
    print("Reject H0")
else:
    print("Accept H0")
```

```
2.34181859243181 0.020869348905772172
Accept H0
```

In [244]:

```
from scipy.stats import ttest_rel
t_statistic,p_value=ttest_rel(group1, group2)
print(t_statistic,p_value)
if p_value<0.5:
    print("Reject H0")
else:
    print("Accept H0")
```

```
2.3172456109384103 0.023979527821469917
Reject H0
```

In [242]:

```
import pandas as pd
from scipy.stats import chi2_contingency
contingency_table = pd.crosstab(df["Pass/Fail"], df["SECTION"])
stat, p, dof, expected = chi2_contingency(contingency_table)
alpha = 0.05
print(f"p-value is {p}")
if p <= alpha:
    print("Pass/Fail is dependent on Section (Reject H0)")
else:
    print("Pass/Fail is independent of Section (H0 holds true)")
```

```
p-value is 0.010968332427338603
Pass/Fail is dependent on Section (Reject H0)
```