In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
df = pd.read_excel("MIDMARKS-MINOR1-EXAM.xlsx")
```

In [2]:
```
df
```

Out[2]:

	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 [3]:
```
df.info
```

Out[3]:
```
<bound method DataFrame.info of      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 x 8 columns]>
```

In [4]:
```
df['SECTION'] = df['SECTION'].fillna('ZETA')
df
```

Out[4]:

	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	ZETA	18	2	12	3	17	15
476	477	ZETA	20	6	16	11	20	14
477	478	ZETA	20	NaN	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 [ ]:
```
```

In [5]:
```
df['SECTION'].value_counts()
```

Out[5]:
```
ALPHA      60
BETA       60
DELTA      60
ZETA       60
EPSILON    60
GAMMA      60
OMEGA      60
SIGMA      60
Name: SECTION, dtype: int64
```

In [ ]:
```
```

In [6]:
```
df['DV'].value_counts()
```

Out[6]:
```
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 [7]:
```
df['DV'] = df['DV'].replace('MP',0)
df
```

Out[7]:

	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	ZETA	18	2	12	3	17	15
476	477	ZETA	20	6	16	11	20	14
477	478	ZETA	20	NaN	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 [8]:
```
df['DV'] = df['DV'].replace('A',0)
df
```

Out[8]:

	S.NO	SECTION	DV	M-II	PP	BEEE	FL	FIMS
0	1	ALPHA	12.0	0	17	9	19	15
1	2	ALPHA	19.0	12	16	16	18	3
2	3	ALPHA	18.0	14	18	18	18	16
3	4	ALPHA	15.0	9	19	17	19	15
4	5	ALPHA	18.0	17	19	19	20	18
...	...	...	...	...	...	...	...	...
475	476	ZETA	18.0	2	12	3	17	15
476	477	ZETA	20.0	6	16	11	20	14
477	478	ZETA	20.0	NaN	18	13	20	18
478	479	ZETA	20.0	20	5	19	18	14
479	480	ZETA	20.0	16	18	19	20	19

480 rows × 8 columns

In [9]:
```
df['DV'].value_counts()
```

Out[9]:
```
17.0    53
20.0    53
18.0    48
16.0    48
15.0    45
19.0    38
11.0    31
12.0    27
13.0    25
14.0    24
10.0    22
9.0     14
8.0     10
6.0      9
5.0      8
0.0      7
7.0      6
2.0      4
4.0      3
1.0      3
3.0      1
Name: DV, dtype: int64
```

In [10]:
```
df = df.replace('MP',0)
df = df.replace('A',0)
df = df.replace('AB',0)
df = df.replace('o',0)
df = df.replace('II',11)
df = df.replace('I',1)
```

In [11]:
```
df['PP'].value_counts()
```

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

In [12]:
```
df['BEEE'].value_counts()
```

Out[12]:
```
20.0    76
17.0    46
19.0    42
11.0    31
18.0    31
15.0    28
16.0    23
14.0    21
12.0    21
10.0    20
9.0     19
0.0     15
7.0     15
6.0     15
3.0     14
13.0    14
8.0     13
4.0     12
5.0     10
2.0      9
1.0      3
Name: BEEE, dtype: int64
```

In [13]:
```
df['M-II'].value_counts()
```

Out[13]:
```
20.0    44
0.0     34
3.0     34
17.0    32
8.0     29
11.0    25
12.0    24
15.0    24
18.0    23
5.0     23
4.0     22
10.0    20
13.0    18
6.0     18
1.0     18
9.0     17
14.0    17
16.0    16
7.0     14
2.0     13
19.0    12
Name: M-II, dtype: int64
```

In [14]:
```
df['FIMS'].value_counts()
```

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

In [15]:
```
df['FL'].value_counts()
```

Out[15]:
```
20.0    121
15.0     85
18.0     59
10.0     55
13.0     50
19.0     34
16.0     15
14.0     11
11.0     10
17.0      9
0.0       9
12.0      8
8.0       6
9.0       3
6.0       2
7.0       2
Name: FL, dtype: int64
```

In [16]:
```
df.isnull().sum()
```

Out[16]:
```
S.NO       0
SECTION    0
DV         1
M-II       3
PP         0
BEEE       2
FL         1
FIMS       0
dtype: int64
```

In [17]:
```
df[df['DV'].isnull()]
```

Out[17]:

	S.NO	SECTION	DV	M-II	PP	BEEE	FL	FIMS
389	390	OMEGA	NaN	17.0	17	19.0	20.0	17

In [ ]:
```

```

In [18]:
```
df['S.NO'] = range(1, len(df) + 1)
df
```

Out[18]:

	S.NO	SECTION	DV	M-II	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 [19]:
```
df['FL'] = df['FL'].fillna(0)
df=df.dropna()
```

In [20]:
```
df
```

Out[20]:

	S.NO	SECTION	DV	M-II	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
...	...	...	...	...	...	...	...	...
474	475	ZETA	11.0	4.0	2	2.0	8.0	10
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
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

474 rows × 8 columns

In [21]:
```
df.info
```

Out[21]:
```
<bound method DataFrame.info of      S.NO SECTION    DV  M-II  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
..    ...     ...   ...   ...  ..   ...   ...   ...
474   475    ZETA  11.0   4.0   2   2.0   8.0    10
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
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

[474 rows x 8 columns]>
```

In [22]:
```
df.isnull().sum()
```

Out[22]:
```
S.NO       0
SECTION    0
DV         0
M-II       0
PP         0
BEEE       0
FL         0
FIMS       0
dtype: int64
```

In [ ]:
```

```

Mid Marks Data

In [23]:
```
df.rename(columns={'M-II':'M2'},inplace=True)
```

```
C:\Users\khsbh\AppData\Local\Temp\ipykernel_26360\82161249.py:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.rename(columns={'M-II':'M2'},inplace=True)
```

Renaming M-II as M2

In [24]:
```
df
```

Out[24]:

	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
...	...	...	...	...	...	...	...	...
474	475	ZETA	11.0	4.0	2	2.0	8.0	10
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
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

474 rows × 8 columns

In [25]:
```
df['DV'] = pd.to_numeric(df['DV'], errors='coerce')
df['M2'] = pd.to_numeric(df['M2'], errors='coerce')
df['PP'] = pd.to_numeric(df['PP'], errors='coerce')
df['BEEE'] = pd.to_numeric(df['BEEE'], errors='coerce')
df['FL'] = pd.to_numeric(df['FL'], errors='coerce')
df['FIMS'] = pd.to_numeric(df['FIMS'], errors='coerce')
df.fillna(0, inplace=True)
```

```
C:\Users\khsbh\AppData\Local\Temp\ipykernel_26360\1473698917.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 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df['DV'] = pd.to_numeric(df['DV'], errors='coerce')
C:\Users\khsbh\AppData\Local\Temp\ipykernel_26360\1473698917.py:2: 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 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df['M2'] = pd.to_numeric(df['M2'], errors='coerce')
C:\Users\khsbh\AppData\Local\Temp\ipykernel_26360\1473698917.py:3: 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 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df['PP'] = pd.to_numeric(df['PP'], errors='coerce')
C:\Users\khsbh\AppData\Local\Temp\ipykernel_26360\1473698917.py:4: 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 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df['BEEE'] = pd.to_numeric(df['BEEE'], errors='coerce')
C:\Users\khsbh\AppData\Local\Temp\ipykernel_26360\1473698917.py:5: 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 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df['FL'] = pd.to_numeric(df['FL'], errors='coerce')
C:\Users\khsbh\AppData\Local\Temp\ipykernel_26360\1473698917.py:6: 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 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df['FIMS'] = pd.to_numeric(df['FIMS'], errors='coerce')
C:\Users\khsbh\AppData\Local\Temp\ipykernel_26360\1473698917.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df.fillna(0, inplace=True)
```

Converting into numeric

In [26]:
```
df['Total'] = df['DV'] + df['M2'] + df['PP'] + df['BEEE'] + df['FL'] + df['FIMS']

df
```

```
C:\Users\khsbh\AppData\Local\Temp\ipykernel_26360\4026183173.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 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df['Total'] = df['DV'] + df['M2'] + df['PP'] + df['BEEE'] + df['FL'] + df['FIMS']
```

Out[26]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total
0	1	ALPHA	12.0	0.0	17	9.0	19.0	15	72.0
1	2	ALPHA	19.0	12.0	16	16.0	18.0	3	84.0
2	3	ALPHA	18.0	14.0	18	18.0	18.0	16	102.0
3	4	ALPHA	15.0	9.0	19	17.0	19.0	15	94.0
4	5	ALPHA	18.0	17.0	19	19.0	20.0	18	111.0
...	...	...	...	...	...	...	...	...	...
474	475	ZETA	11.0	4.0	2	2.0	8.0	10	37.0
475	476	ZETA	18.0	2.0	12	3.0	17.0	15	67.0
476	477	ZETA	20.0	6.0	16	11.0	20.0	14	87.0
478	479	ZETA	20.0	20.0	5	19.0	18.0	14	96.0
479	480	ZETA	20.0	16.0	18	19.0	20.0	19	112.0

474 rows × 9 columns

Calculating total

In [27]:
```
df["Percentage"] = (df['Total']/120)*100
```

```
C:\Users\khsbh\AppData\Local\Temp\ipykernel_26360\3999239091.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 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df["Percentage"] = (df['Total']/120)*100
```

In [28]:
```
df
```

Out[28]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage
0	1	ALPHA	12.0	0.0	17	9.0	19.0	15	72.0	60.000000
1	2	ALPHA	19.0	12.0	16	16.0	18.0	3	84.0	70.000000
2	3	ALPHA	18.0	14.0	18	18.0	18.0	16	102.0	85.000000
3	4	ALPHA	15.0	9.0	19	17.0	19.0	15	94.0	78.333333
4	5	ALPHA	18.0	17.0	19	19.0	20.0	18	111.0	92.500000
...	...	...	...	...	...	...	...	...	...	...
474	475	ZETA	11.0	4.0	2	2.0	8.0	10	37.0	30.833333
475	476	ZETA	18.0	2.0	12	3.0	17.0	15	67.0	55.833333
476	477	ZETA	20.0	6.0	16	11.0	20.0	14	87.0	72.500000
478	479	ZETA	20.0	20.0	5	19.0	18.0	14	96.0	80.000000
479	480	ZETA	20.0	16.0	18	19.0	20.0	19	112.0	93.333333

474 rows × 10 columns

In [29]:
```
df['Percentage'] = df['Percentage'].round().astype(int)
df
```

```
C:\Users\khsbh\AppData\Local\Temp\ipykernel_26360\1999675392.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 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df['Percentage'] = df['Percentage'].round().astype(int)
```

Out[29]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage
0	1	ALPHA	12.0	0.0	17	9.0	19.0	15	72.0	60
1	2	ALPHA	19.0	12.0	16	16.0	18.0	3	84.0	70
2	3	ALPHA	18.0	14.0	18	18.0	18.0	16	102.0	85
3	4	ALPHA	15.0	9.0	19	17.0	19.0	15	94.0	78
4	5	ALPHA	18.0	17.0	19	19.0	20.0	18	111.0	92
...	...	...	...	...	...	...	...	...	...	...
474	475	ZETA	11.0	4.0	2	2.0	8.0	10	37.0	31
475	476	ZETA	18.0	2.0	12	3.0	17.0	15	67.0	56
476	477	ZETA	20.0	6.0	16	11.0	20.0	14	87.0	72
478	479	ZETA	20.0	20.0	5	19.0	18.0	14	96.0	80
479	480	ZETA	20.0	16.0	18	19.0	20.0	19	112.0	93

474 rows × 10 columns

In [30]:
```
def assign_grade(percentage):
    if percentage >= 90:
        return 'A'
    elif percentage >= 80:
        return 'B+'
    elif percentage >= 70:
        return 'B'
    elif percentage >= 60:
        return 'C+'
    elif percentage >=50:
        return 'C'
    elif percentage >=40:
        return 'D'
    else:
        return 'F'
df['Grade'] = df['Percentage'].apply(assign_grade)
df
```

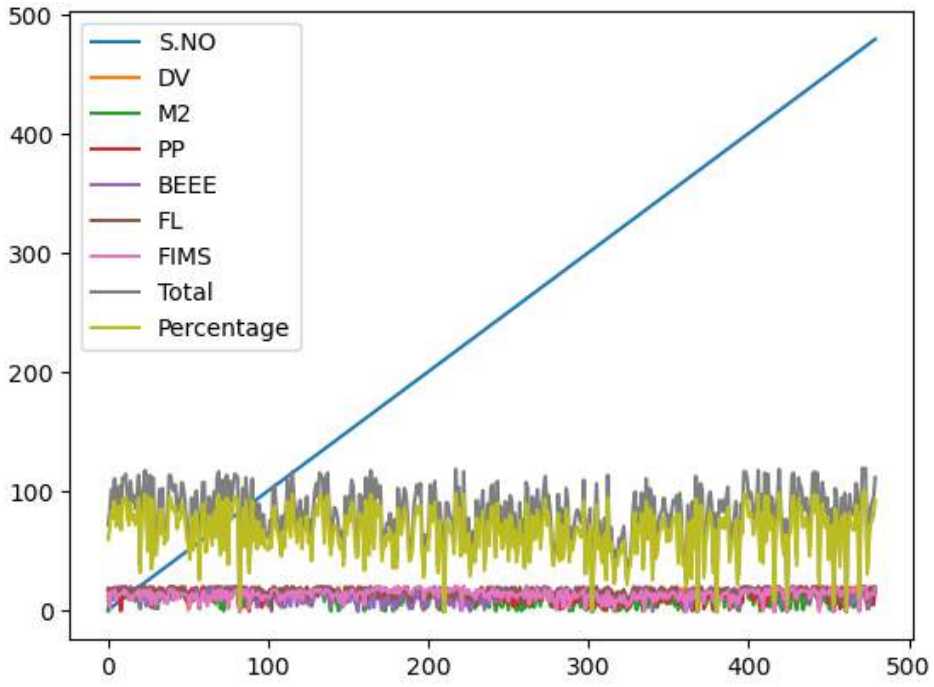
```
C:\Users\khsbh\AppData\Local\Temp\ipykernel_26360\2453317957.py:16: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df['Grade'] = df['Percentage'].apply(assign_grade)
```

Out[30]:

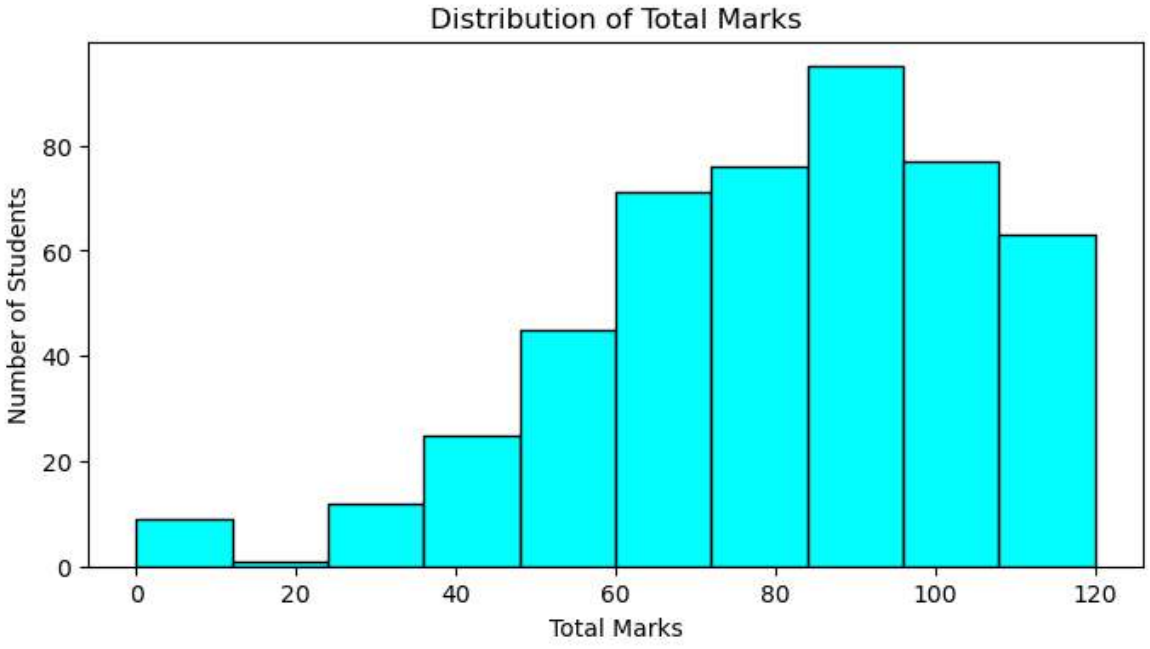
	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade
0	1	ALPHA	12.0	0.0	17	9.0	19.0	15	72.0	60	C+
1	2	ALPHA	19.0	12.0	16	16.0	18.0	3	84.0	70	B
2	3	ALPHA	18.0	14.0	18	18.0	18.0	16	102.0	85	B+
3	4	ALPHA	15.0	9.0	19	17.0	19.0	15	94.0	78	B
4	5	ALPHA	18.0	17.0	19	19.0	20.0	18	111.0	92	A
...	...	...	...	...	...	...	...	...	...	...	...
474	475	ZETA	11.0	4.0	2	2.0	8.0	10	37.0	31	F
475	476	ZETA	18.0	2.0	12	3.0	17.0	15	67.0	56	C
476	477	ZETA	20.0	6.0	16	11.0	20.0	14	87.0	72	B
478	479	ZETA	20.0	20.0	5	19.0	18.0	14	96.0	80	B+
479	480	ZETA	20.0	16.0	18	19.0	20.0	19	112.0	93	A

474 rows × 11 columns

In [31]:
```
df.plot()
plt.show()
```



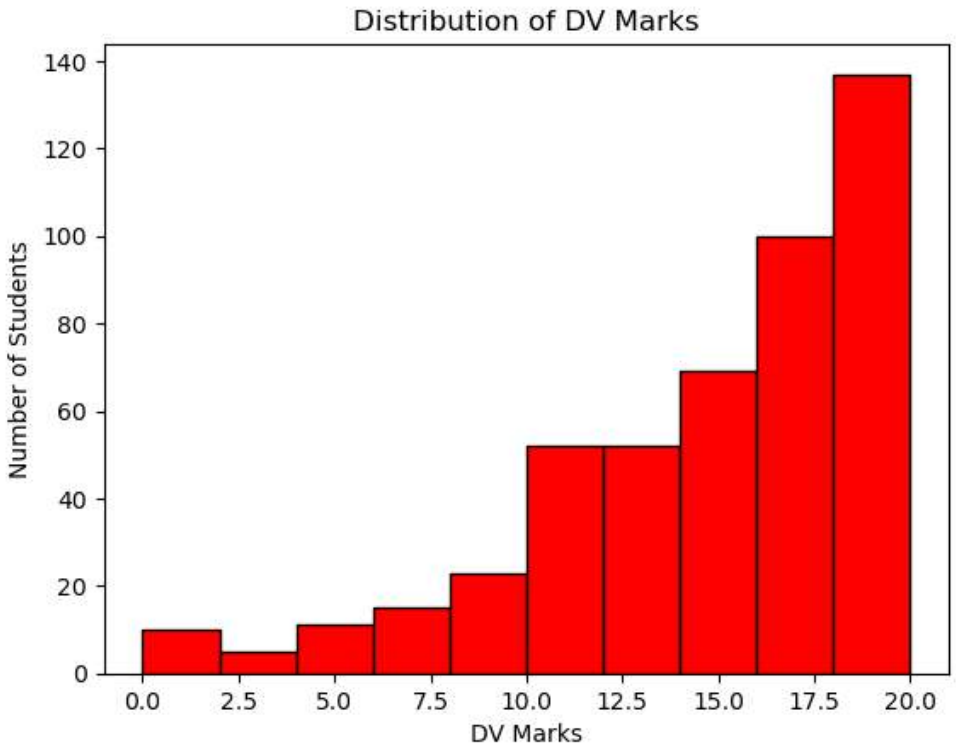
In [32]:
```
plt.figure(figsize=[8, 4])
plt.hist(df['Total'], color='cyan', bins=10, edgecolor='black')
plt.title("Distribution of Total Marks")
plt.xlabel("Total Marks")
plt.ylabel("Number of Students")
plt.show()
```



Distribution of Total marks in Histogram

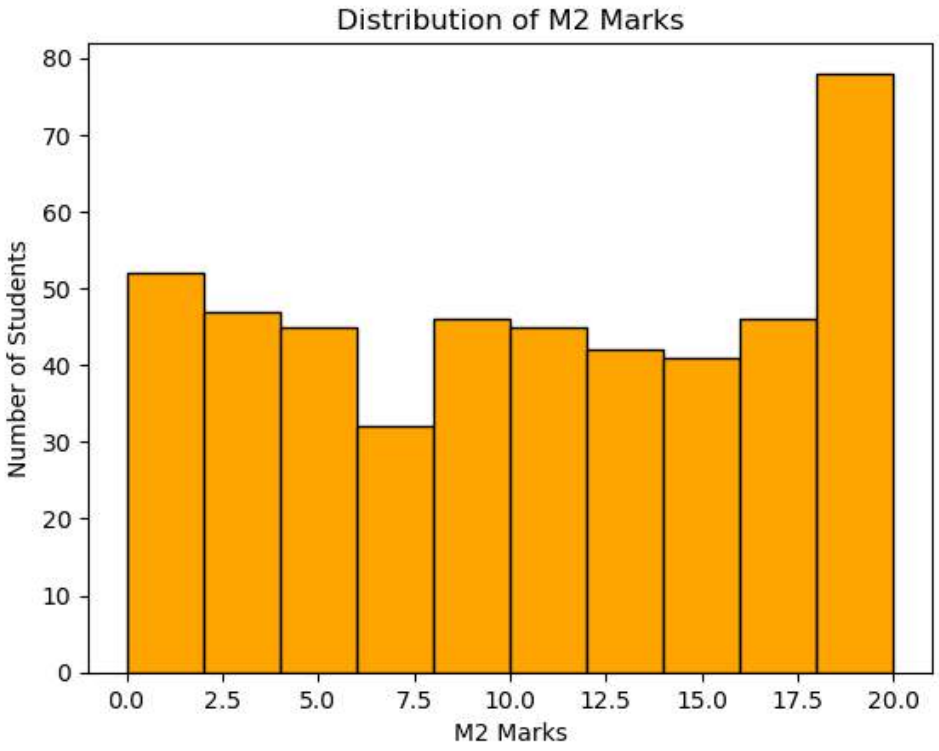
In [33]:
```
df['DV'] = pd.to_numeric(df['DV'], errors='coerce')
df = df.dropna(subset=['DV'])
plt.hist(df['DV'], bins=10, color='red', edgecolor='black')
plt.title("Distribution of DV Marks")
plt.xlabel("DV Marks")
plt.ylabel("Number of Students")
plt.show()
```

```
C:\Users\khsbh\AppData\Local\Temp\ipykernel_26360\1854147094.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 (https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)
  df['DV'] = pd.to_numeric(df['DV'], errors='coerce')
```



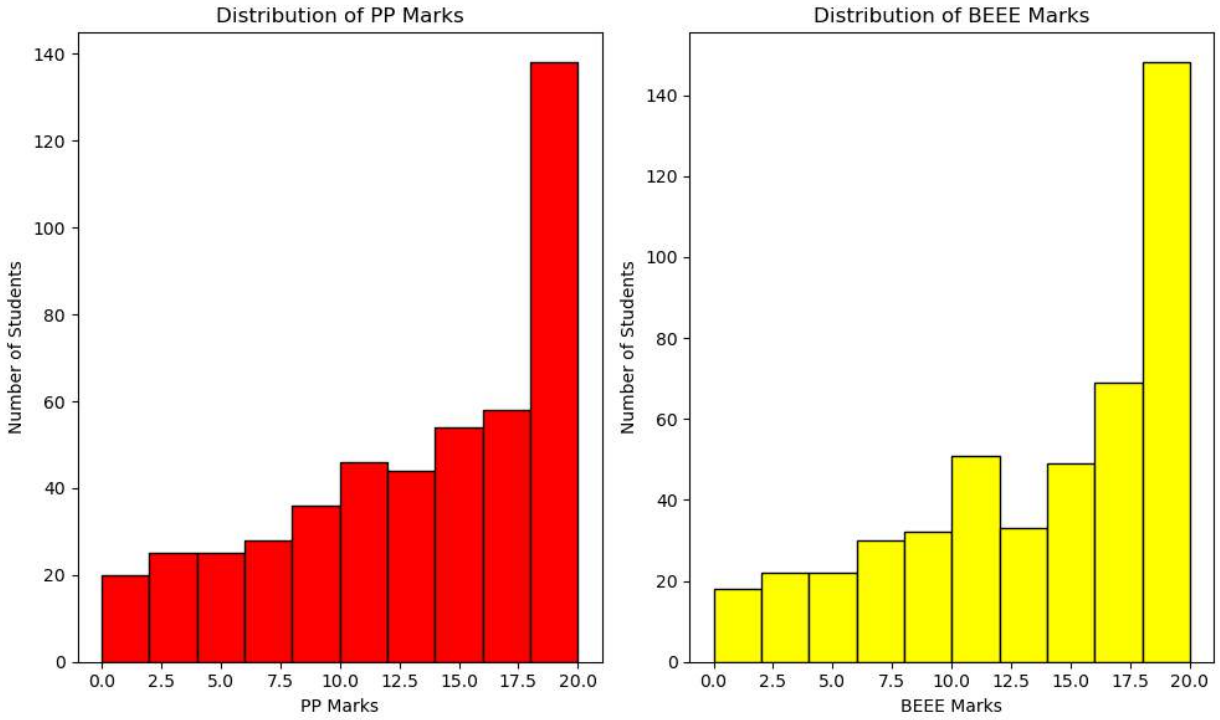
Distribution of DV Marks

In [34]:
```
df['M2'] = pd.to_numeric(df['M2'], errors='coerce')
df = df.dropna(subset=['M2'])
plt.hist(df['M2'], bins=10, color='orange', edgecolor='black')
plt.title("Distribution of M2 Marks")
plt.xlabel("M2 Marks")
plt.ylabel("Number of Students")
plt.show()
```



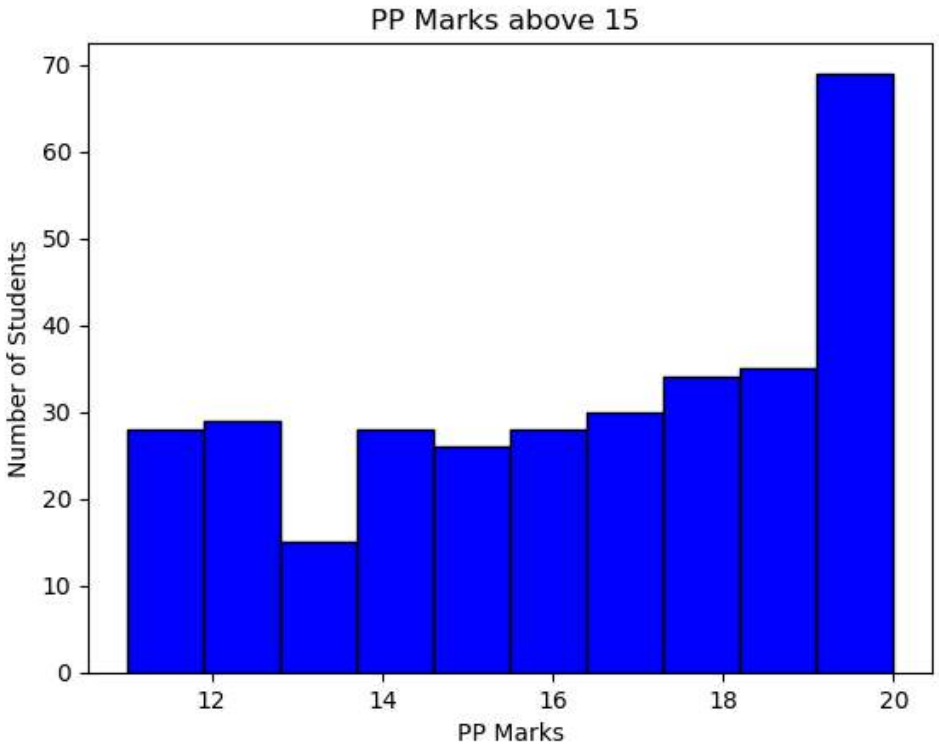
Distribution of M2 Marks

In [35]:
```
df['PP'] = pd.to_numeric(df['PP'], errors='coerce')
df['BEEE'] = pd.to_numeric(df['BEEE'], errors='coerce')
df_clean = df.dropna(subset=['PP', 'BEEE'])

plt.figure(figsize=[10, 6])

plt.subplot(1, 2, 1)
plt.hist(df_clean['PP'], bins=10, color='red', edgecolor='black')
plt.title("Distribution of PP Marks")
plt.xlabel("PP Marks")
plt.ylabel("Number of Students")

plt.subplot(1, 2, 2)
plt.hist(df_clean['BEEE'], bins=10, color='yellow', edgecolor='black')
plt.title("Distribution of BEEE Marks")
plt.xlabel("BEEE Marks")
plt.ylabel("Number of Students")

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



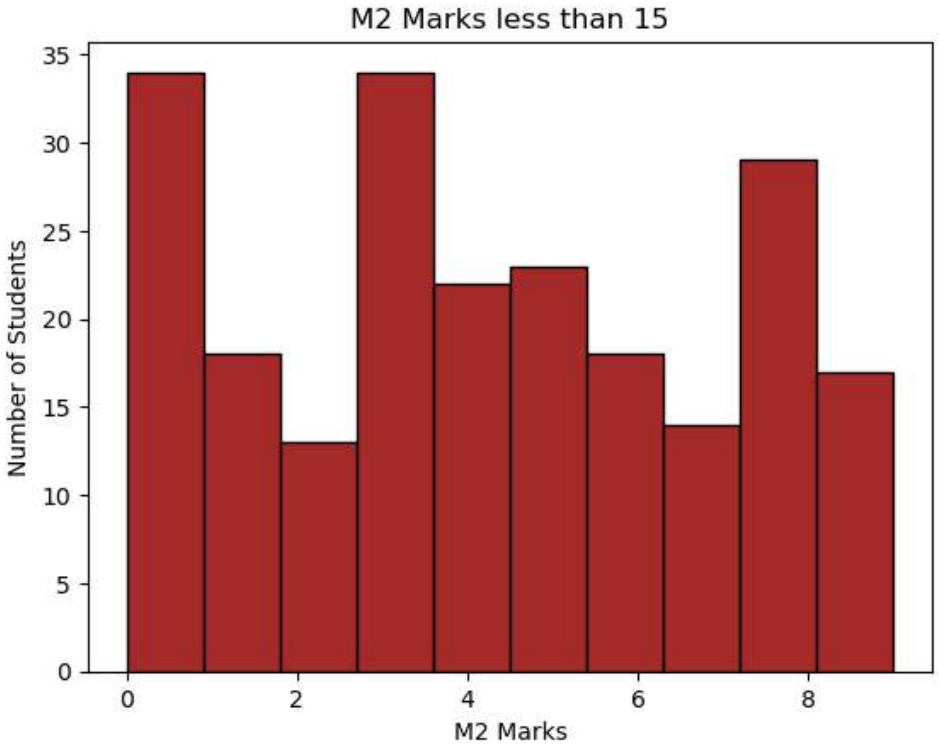
Comparision of PP marks and BEEE marks

In [36]:
```
filtered_df = df[df['PP'] > 10]
plt.hist(filtered_df['PP'], bins=10, color='blue', edgecolor='black')
plt.title("PP Marks above 15 ")
plt.xlabel("PP Marks")
plt.ylabel("Number of Students")
plt.show()
```



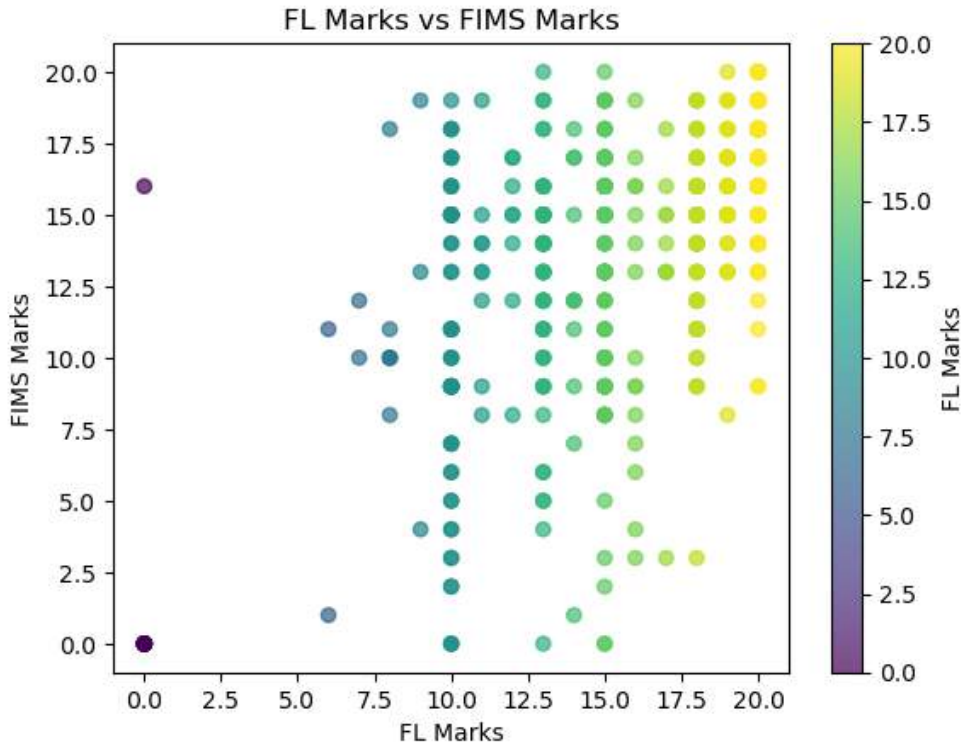
PP Marks who got more than 10

In [37]:
```
filtered_df = df[df['M2'] < 10]

plt.hist(filtered_df['M2'], bins=10, color='brown', edgecolor='black')
plt.title("M2 Marks less than 15")
plt.xlabel("M2 Marks")
plt.ylabel("Number of Students")
plt.show()
```



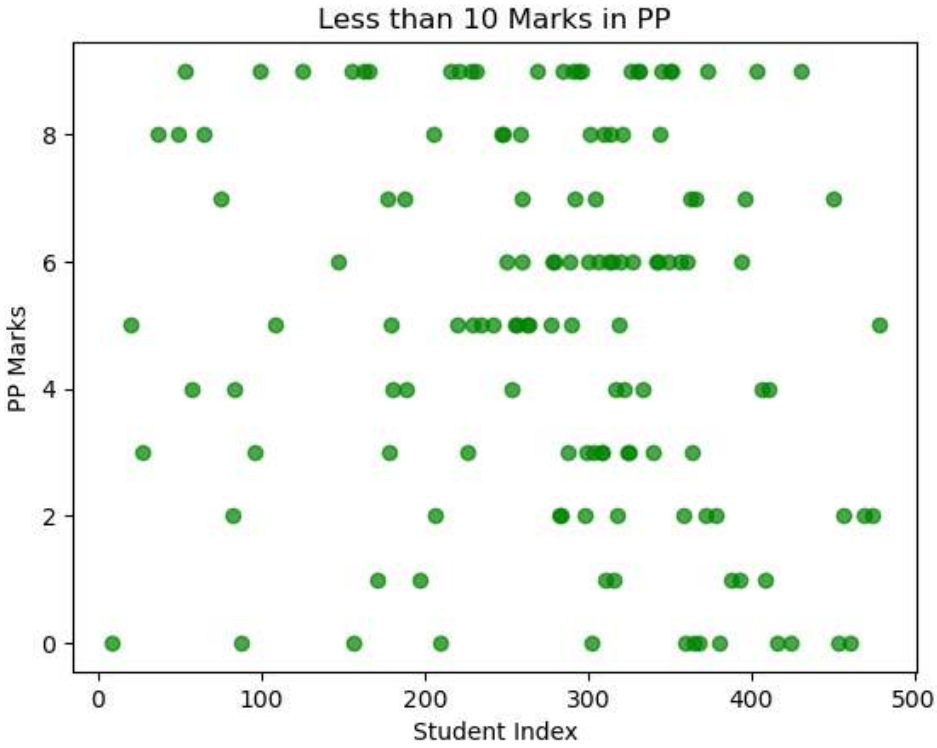
M2 Marks Who got less than 10

In [38]:
```
plt.scatter(df['FL'], df['FIMS'], c=df['FL'], cmap='viridis', alpha=0.7)
plt.title("FL Marks vs FIMS Marks")
plt.xlabel("FL Marks")
plt.ylabel("FIMS Marks")
plt.colorbar(label='FL Marks')
plt.show()
```



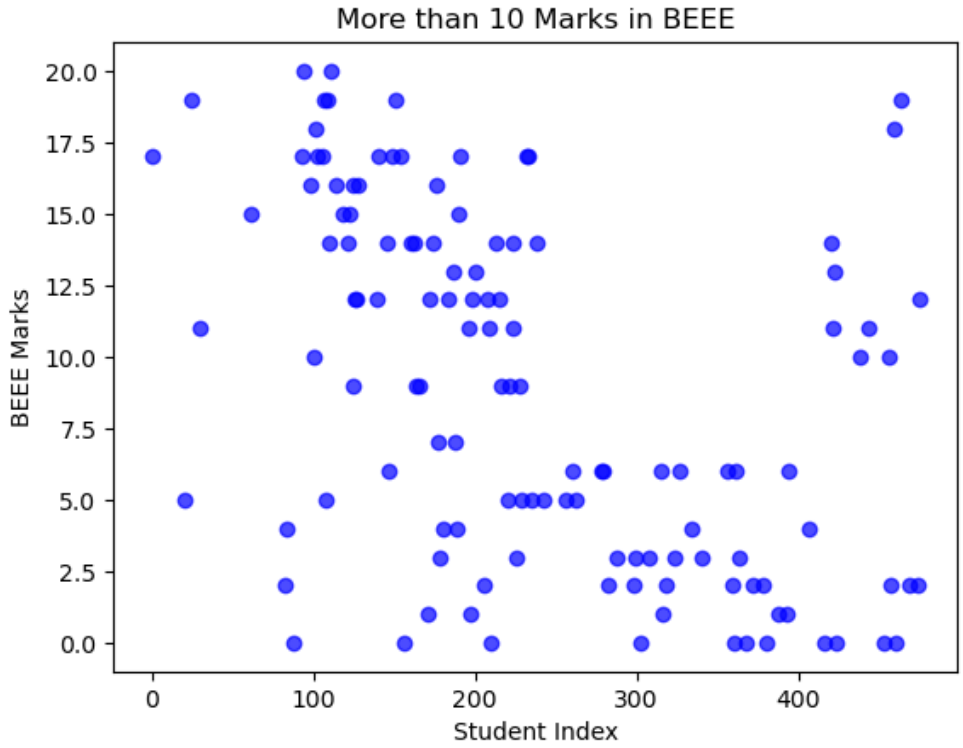
Scatter plot of FL VS FIMS Marks

In [39]:
```
filtered_df = df[df['PP'] < 10]

plt.scatter(filtered_df.index, filtered_df['PP'], alpha=0.7, color='green')
plt.title("Less than 10 Marks in PP")
plt.xlabel("Student Index")
plt.ylabel("PP Marks")
plt.show()
```



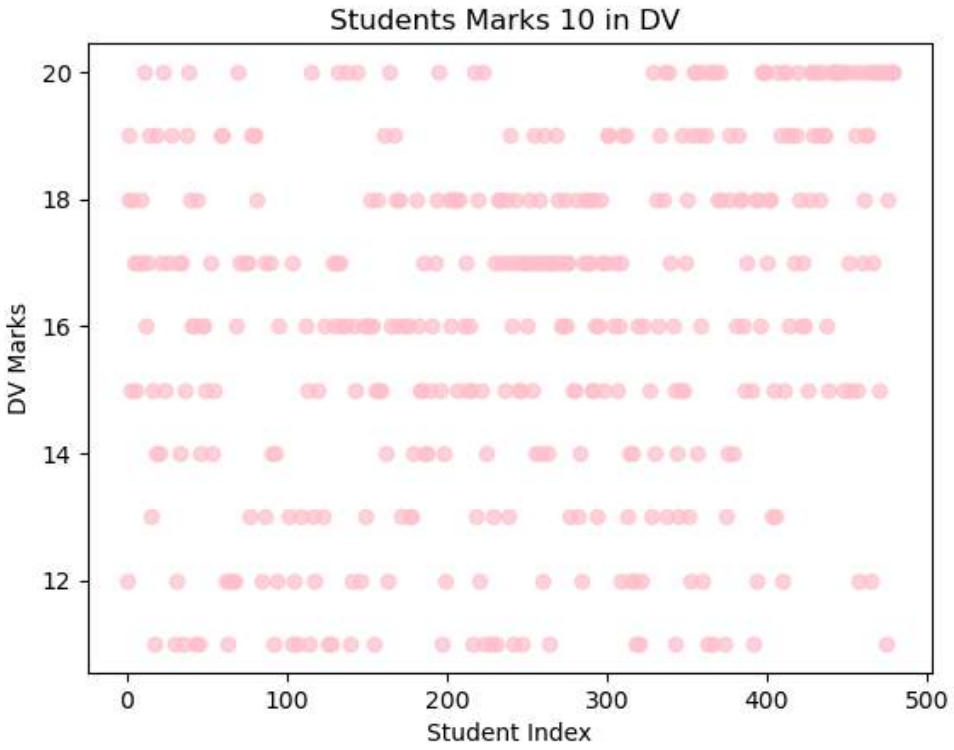
Scoring of PP Marks Less than 10

In [40]:
```
filtered_df = df[df['BEEE'] < 10]

plt.scatter(filtered_df.index, filtered_df['PP'], alpha=0.7, color='blue')
plt.title("More than 10 Marks in BEEE")
plt.xlabel("Student Index")
plt.ylabel("BEEE Marks")
plt.show()
```



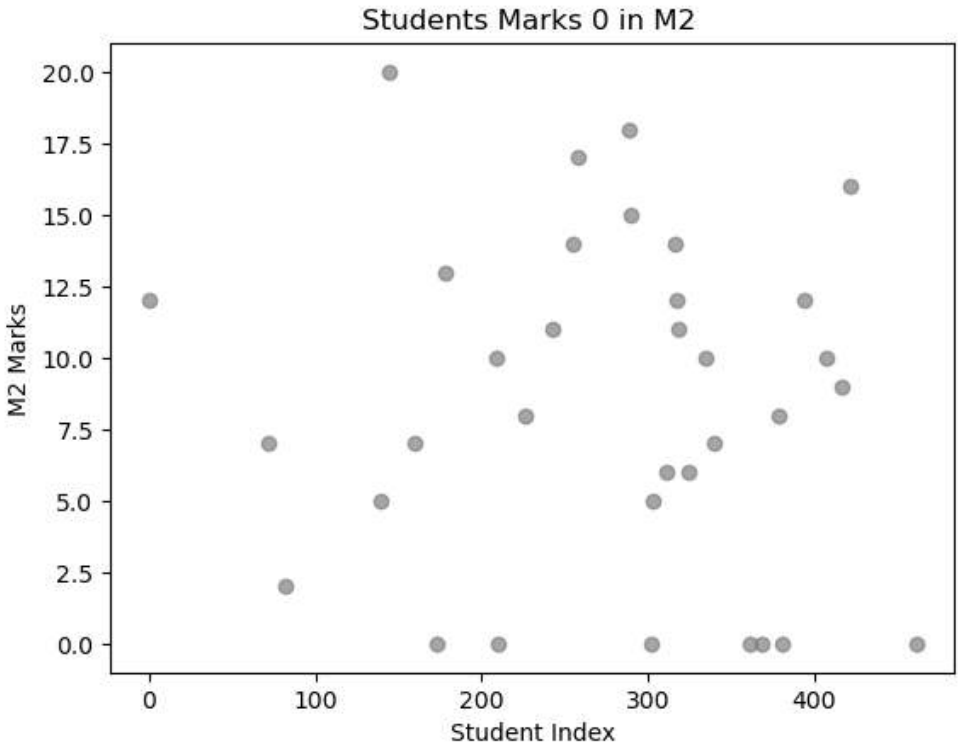
Scoring of BEEE Marks More than 15

In [41]:
```
filtered_df = df[df['DV'] > 10]
plt.scatter(filtered_df.index, filtered_df['DV'], alpha=0.7, color='pink')
plt.title("Students Marks 10 in DV ")
plt.xlabel("Student Index")
plt.ylabel("DV Marks")
plt.show()
```



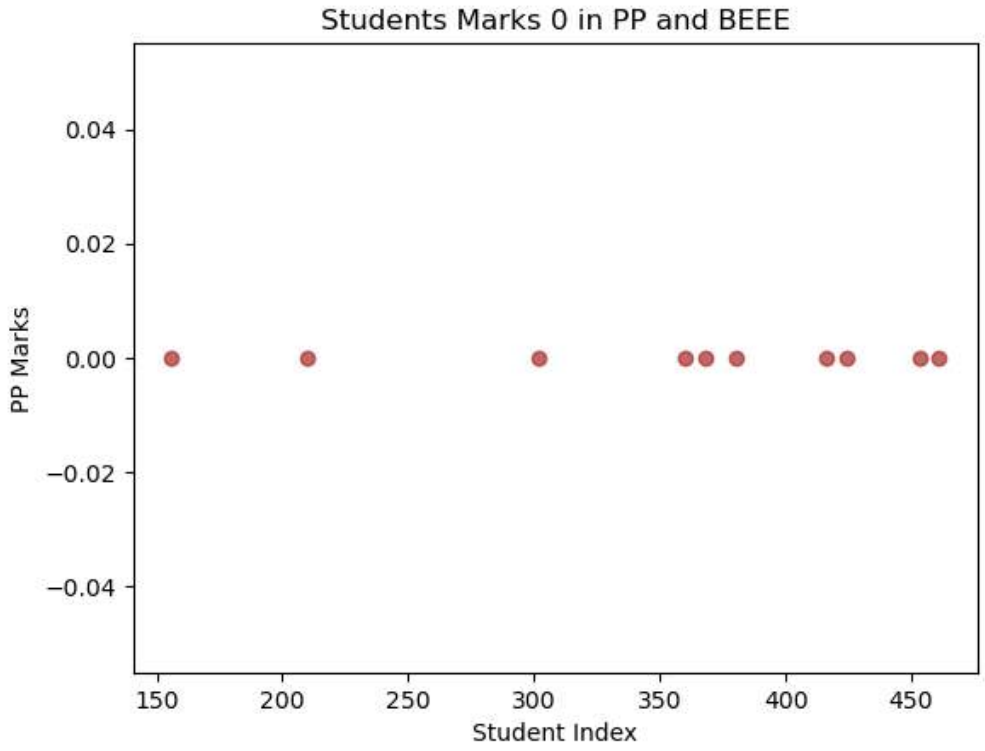
Students who scored 10 marks in DV

In [42]:
```
filtered_df = df[df['M2'] == 0]
plt.scatter(filtered_df.index, filtered_df['DV'], alpha=0.7, color='grey')
plt.title("Students Marks 0 in M2 ")
plt.xlabel("Student Index")
plt.ylabel("M2 Marks")
plt.show()
```



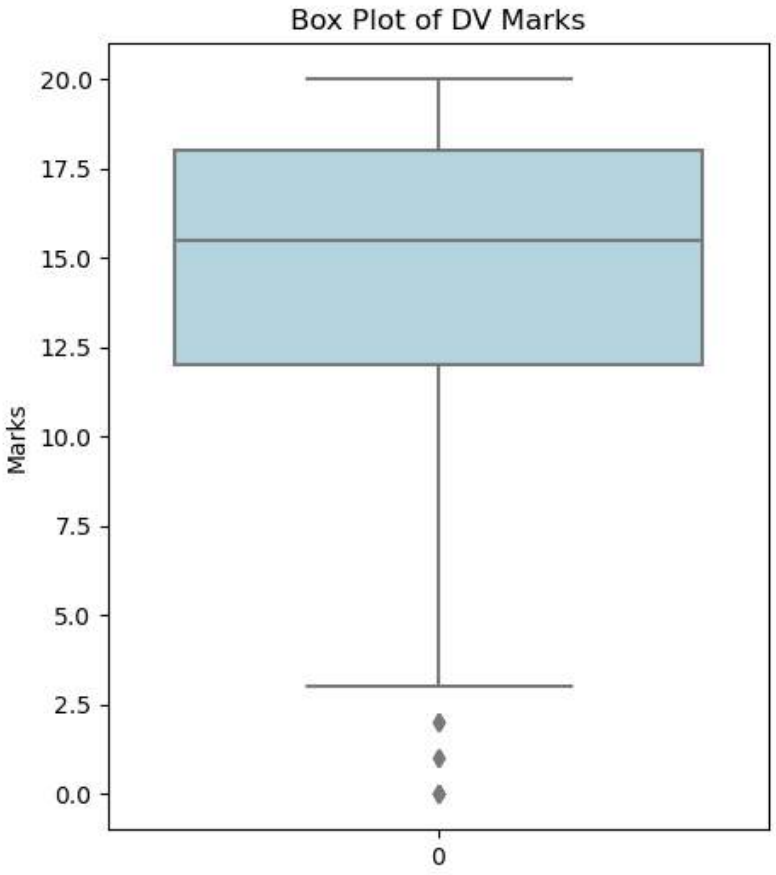
Students who scored 0 Marks in M2

In [43]:
```
filtered_df = df[(df['PP'] == 0) & (df['BEEE'] == 0)]
plt.scatter(filtered_df.index, filtered_df['PP'], alpha=0.7, color='brown')
plt.title("Students Marks 0 in PP and BEEE")
plt.xlabel("Student Index")
plt.ylabel("PP Marks")
plt.show()
```



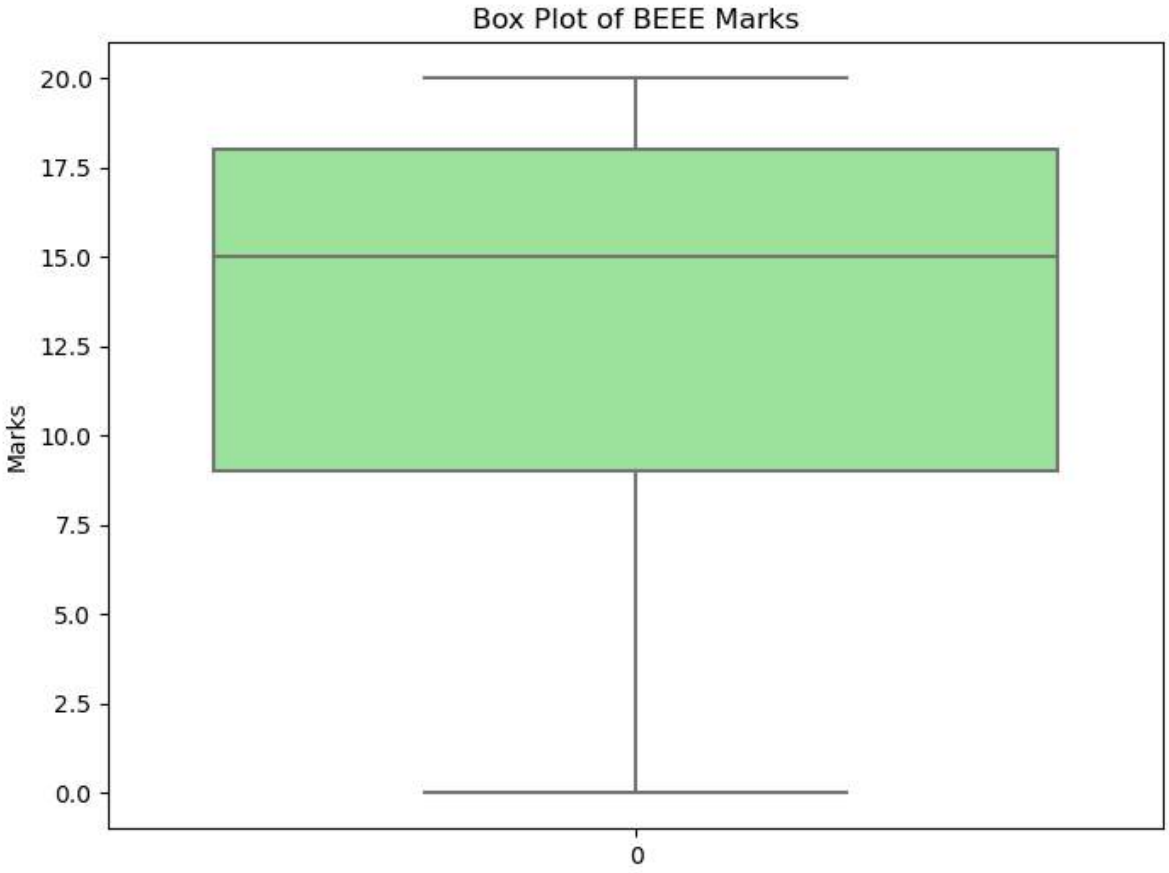
Students scored 0 Marks in PP and BEEE

In [44]:
```
plt.figure(figsize=(5, 6))
sns.boxplot(data=df['DV'], color='lightblue')
plt.title("Box Plot of DV Marks")
plt.ylabel("Marks")
plt.show()
```



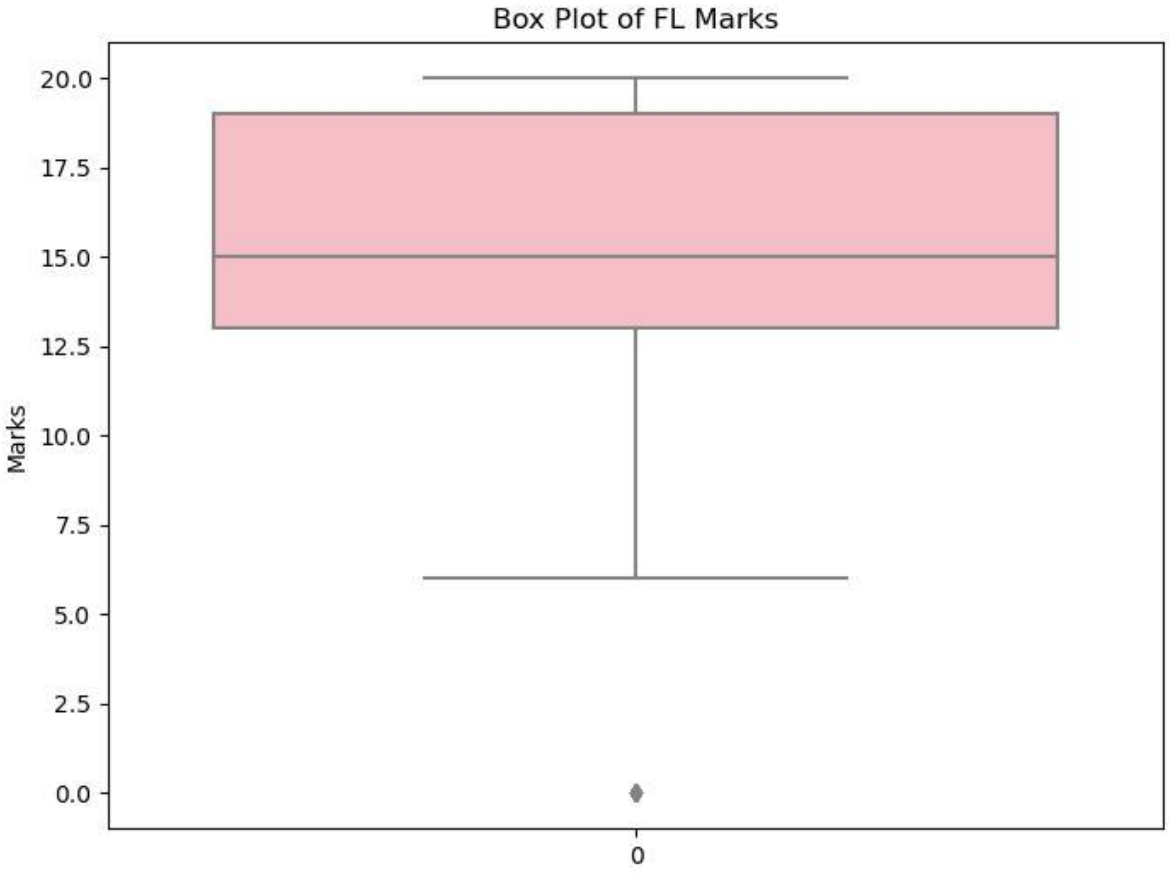
Box plot for the DV Marks

In [45]:
```
plt.figure(figsize=(8, 6))
sns.boxplot(data=df['BEEE'], color='lightgreen')
plt.title("Box Plot of BEEE Marks")
plt.ylabel("Marks")
plt.show()
```



Box plot of BEEE Marks

In [46]:
```
plt.figure(figsize=(8, 6))
sns.boxplot(data=df['FL'], color='lightpink')
plt.title("Box Plot of FL Marks")
plt.ylabel("Marks")
plt.show()
```



Box plot for the FL Marks

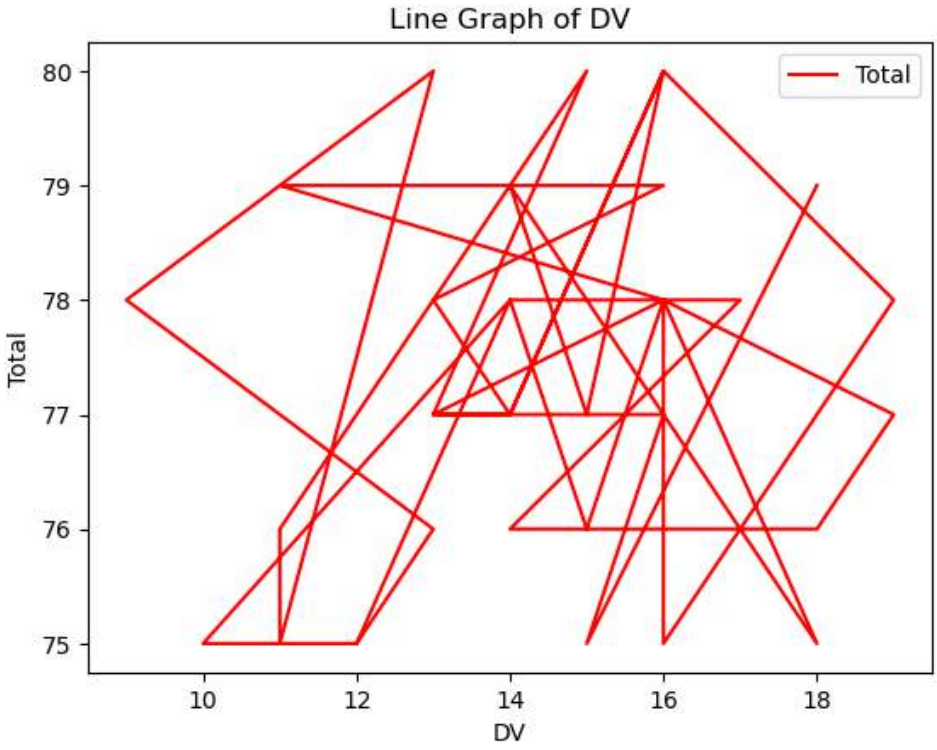
In [47]:
```
a=df.loc[(df['Total'] >= 75) & (df['Total'] <= 80)]
a=a.reset_index()
a
```

Out[47]:

	index	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade
0	31	32	ALPHA	12.0	2.0	17	11.0	18.0	15	75.0	62	C+
1	33	34	ALPHA	14.0	10.0	17	12.0	13.0	12	78.0	65	C+
2	56	57	ALPHA	10.0	17.0	12	17.0	10.0	9	75.0	62	C+
3	67	68	BETA	12.0	6.0	13	20.0	15.0	9	75.0	62	C+
4	101	102	BETA	13.0	12.0	18	4.0	18.0	11	76.0	63	C+
5	106	107	BETA	9.0	13.0	17	6.0	18.0	15	78.0	65	C+
6	109	110	BETA	13.0	12.0	19	4.0	18.0	14	80.0	67	C+
7	114	115	BETA	11.0	14.0	16	4.0	18.0	12	75.0	62	C+
8	126	127	DELTA	11.0	14.0	12	7.0	15.0	17	76.0	63	C+
9	143	144	DELTA	15.0	5.0	16	19.0	10.0	15	80.0	67	C+
10	149	150	DELTA	13.0	16.0	17	7.0	13.0	11	77.0	64	C+
11	179	180	DELTA	14.0	13.0	5	12.0	13.0	20	77.0	64	C+
12	182	183	EPSILON	16.0	8.0	18	11.0	13.0	14	80.0	67	C+
13	183	184	EPSILON	15.0	5.0	12	9.0	18.0	18	77.0	64	C+
14	187	188	EPSILON	14.0	5.0	13	9.0	20.0	18	79.0	66	C+
15	208	209	EPSILON	18.0	6.0	12	8.0	15.0	16	75.0	62	C+
16	214	215	EPSILON	16.0	6.0	14	13.0	13.0	16	78.0	65	C+
17	215	216	EPSILON	15.0	6.0	12	9.0	15.0	19	76.0	63	C+
18	225	226	EPSILON	14.0	8.0	12	13.0	13.0	18	78.0	65	C+
19	244	245	GAMMA	17.0	8.0	11	15.0	15.0	12	78.0	65	C+
20	259	260	GAMMA	14.0	9.0	7	14.0	16.0	16	76.0	63	C+
21	296	297	GAMMA	18.0	4.0	9	10.0	20.0	15	76.0	63	C+
22	300	301	OMEGA	19.0	2.0	6	20.0	17.0	13	77.0	64	C+
23	304	305	OMEGA	16.0	3.0	7	16.0	20.0	16	78.0	65	C+
24	307	308	OMEGA	16.0	4.0	12	17.0	15.0	11	75.0	62	C+
25	310	311	OMEGA	19.0	1.0	8	19.0	18.0	13	78.0	65	C+
26	332	333	SIGMA	16.0	5.0	10	20.0	18.0	11	80.0	67	C+
27	344	345	SIGMA	14.0	9.0	8	18.0	13.0	15	77.0	64	C+
28	374	375	ZETA	13.0	8.0	11	17.0	13.0	16	78.0	65	C+
29	381	382	OMEGA	16.0	14.0	11	18.0	15.0	5	79.0	66	C+
30	391	392	OMEGA	11.0	15.0	11	19.0	15.0	8	79.0	66	C+
31	396	397	OMEGA	16.0	11.0	7	17.0	13.0	14	78.0	65	C+
32	403	404	OMEGA	13.0	5.0	9	18.0	15.0	17	77.0	64	C+
33	423	424	SIGMA	16.0	12.0	13	4.0	15.0	17	77.0	64	C+
34	425	426	SIGMA	15.0	4.0	15	11.0	11.0	19	75.0	62	C+
35	426	427	SIGMA	18.0	2.0	12	18.0	12.0	17	79.0	66	C+

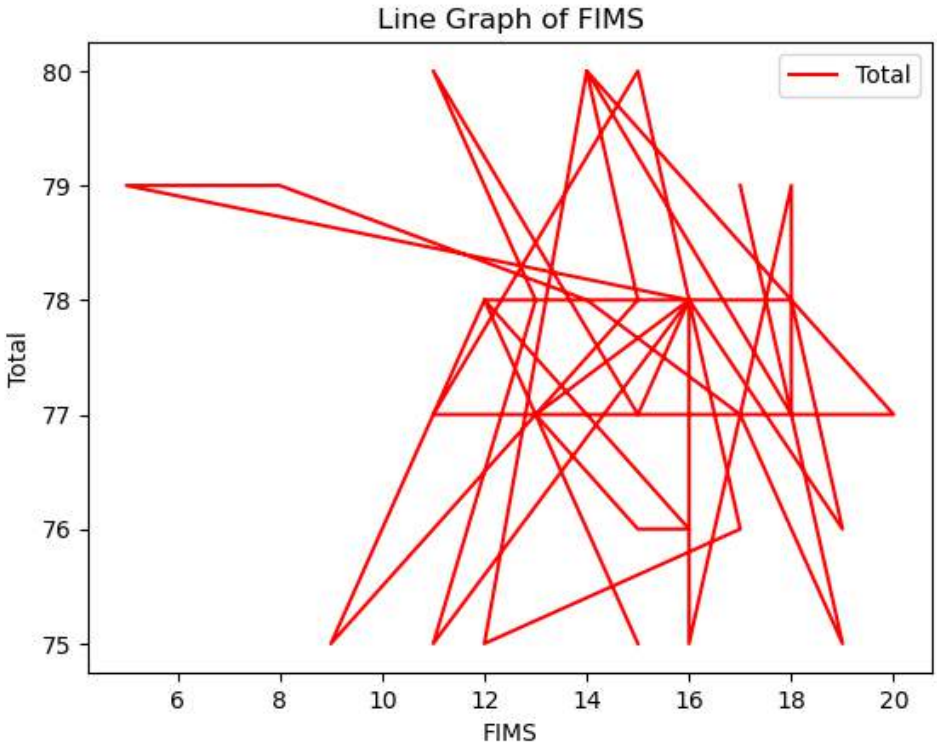
Total marks between 75 to 80

In [48]:
```
a.plot.line(x='DV',y='Total',color='red')
plt.title("Line Graph of DV")
plt.ylabel("Total")
plt.show()
```



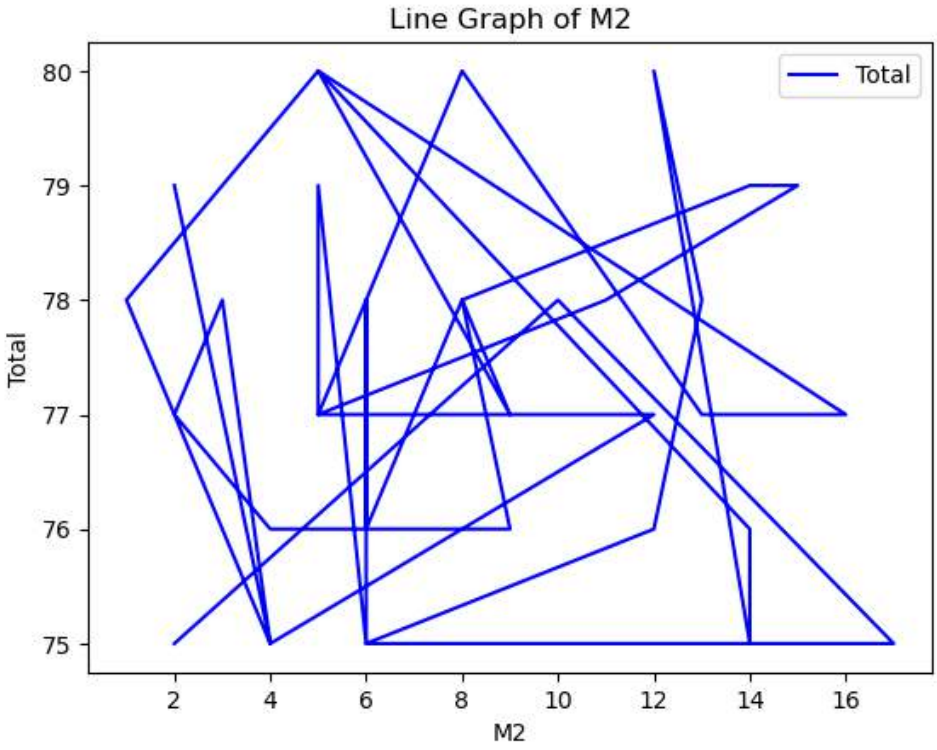
Lineplot for total and DV

In [49]:
```
a.plot.line(x='FIMS',y='Total',color='red')
plt.title("Line Graph of FIMS")
plt.ylabel("Total")
plt.show()
```



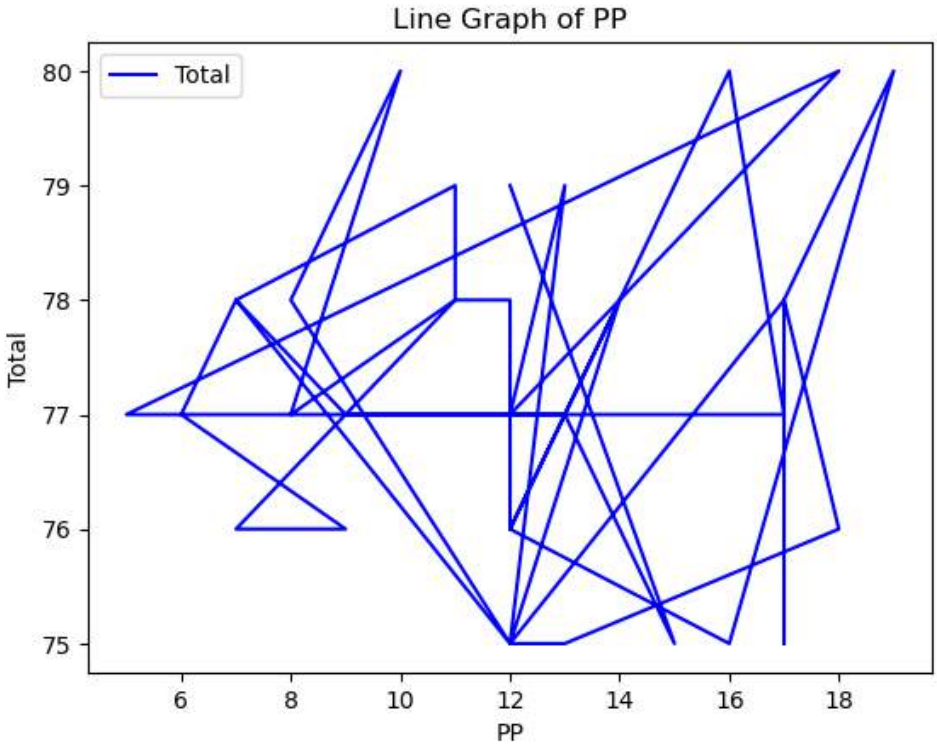
Lineplot for total and FIMS

In [50]:
```
a.plot.line(x='M2',y='Total',color='blue')
plt.title("Line Graph of M2")
plt.ylabel("Total")
plt.show()
```



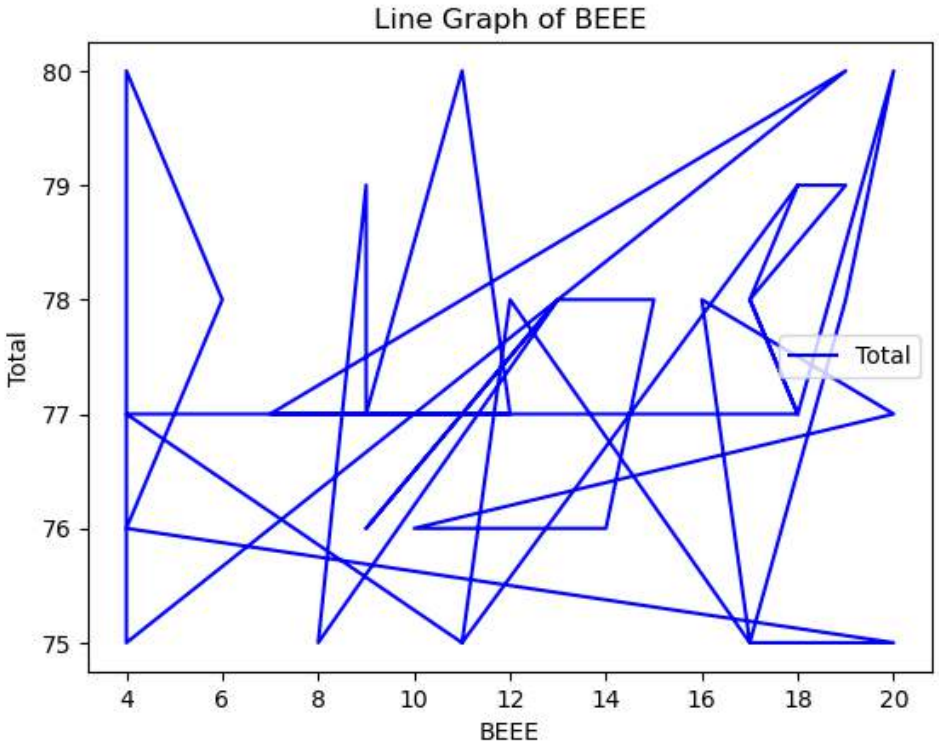
Lineplot for total and M2

In [51]:
```
a.plot.line(x='PP',y='Total',color='blue')
plt.title("Line Graph of PP")
plt.ylabel("Total")
plt.show()
```



Lineplot for total and M2

In [52]:
```
a.plot.line(x='BEEE',y='Total',color='blue')
plt.title("Line Graph of BEEE")
plt.ylabel("Total")
plt.show()
```



Lineplot for total and BEEE

In [53]:
```
b=df.loc[(df['Total'] >= 115) & (df['Total'] <= 120)]
b=b.reset_index()
b
```

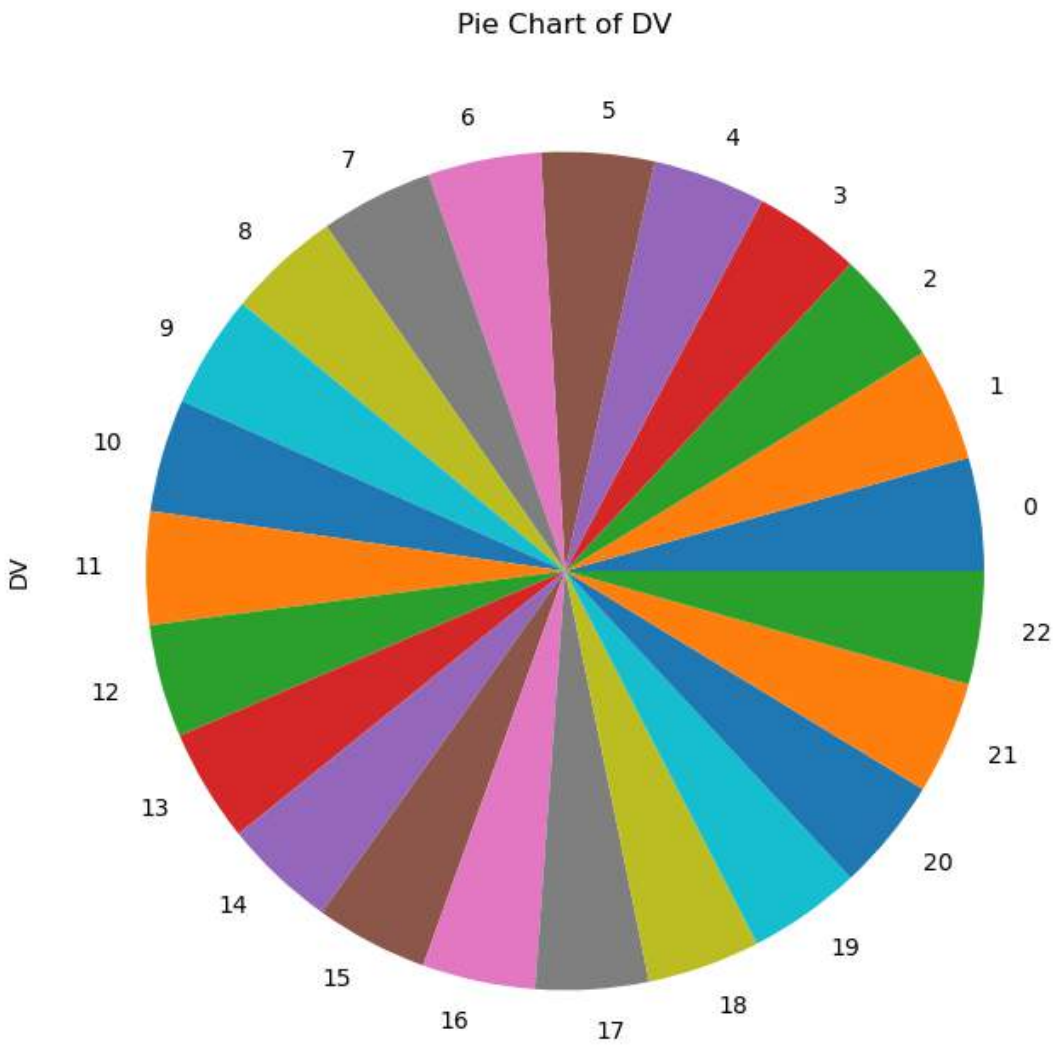
Out[53]:

	index	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade
0	11	12	ALPHA	20.0	20.0	20	20.0	19.0	16	115.0	96	A
1	23	24	ALPHA	20.0	20.0	20	20.0	20.0	18	118.0	98	A
2	69	70	BETA	20.0	20.0	20	19.0	20.0	18	117.0	98	A
3	79	80	BETA	19.0	20.0	20	19.0	20.0	17	115.0	96	A
4	115	116	BETA	20.0	20.0	20	20.0	20.0	17	117.0	98	A
5	132	133	DELTA	20.0	18.0	20	20.0	20.0	18	116.0	97	A
6	137	138	DELTA	20.0	20.0	20	20.0	18.0	18	116.0	97	A
7	164	165	DELTA	20.0	20.0	20	20.0	20.0	18	118.0	98	A
8	217	218	EPSILON	20.0	20.0	20	19.0	20.0	20	119.0	99	A
9	222	223	EPSILON	20.0	20.0	20	17.0	20.0	20	117.0	98	A
10	397	398	OMEGA	20.0	20.0	19	20.0	20.0	18	117.0	98	A
11	398	399	OMEGA	20.0	20.0	19	20.0	20.0	18	117.0	98	A
12	406	407	OMEGA	20.0	20.0	20	20.0	20.0	18	118.0	98	A
13	410	411	OMEGA	20.0	20.0	20	20.0	18.0	18	116.0	97	A
14	412	413	OMEGA	20.0	20.0	17	20.0	20.0	18	115.0	96	A
15	419	420	SIGMA	20.0	20.0	20	20.0	20.0	19	119.0	99	A
16	428	429	SIGMA	20.0	17.0	20	20.0	20.0	19	116.0	97	A
17	440	441	ZETA	20.0	18.0	20	19.0	20.0	18	115.0	96	A
18	441	442	ZETA	20.0	19.0	20	19.0	20.0	19	117.0	98	A
19	443	444	ZETA	20.0	20.0	20	20.0	20.0	16	116.0	97	A
20	471	472	ZETA	20.0	20.0	20	20.0	20.0	20	120.0	100	A
21	472	473	ZETA	20.0	18.0	20	20.0	20.0	19	117.0	98	A
22	473	474	ZETA	20.0	20.0	20	20.0	20.0	20	120.0	100	A

students between 115 and 120

In [54]:
```
b['DV'].plot(kind='pie',subplots=True,figsize=(8,8))
plt.title("Pie Chart of DV")
```

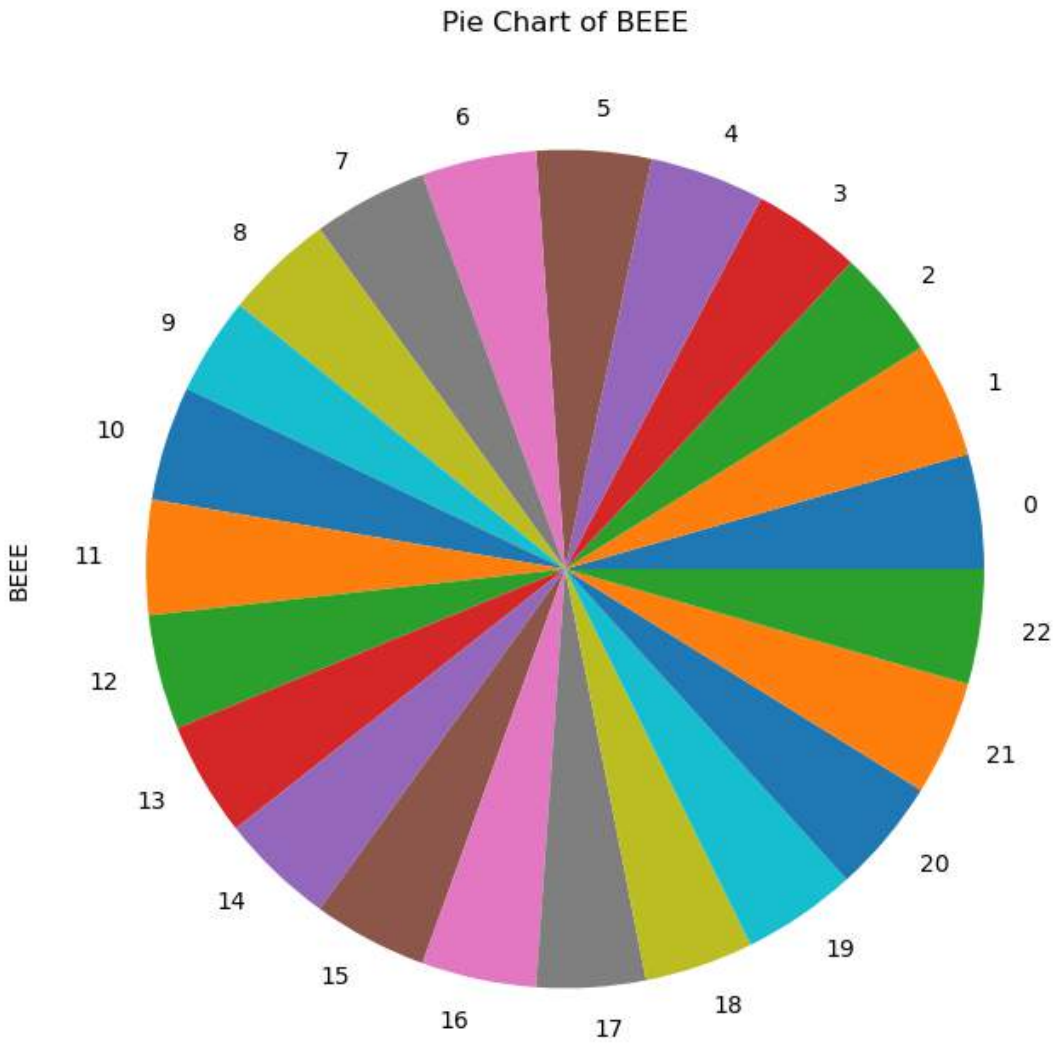
Out[54]:
```
Text(0.5, 1.0, 'Pie Chart of DV')
```



PIE Chart of DV for 32 students

In [55]:
```
b['BEEE'].plot(kind='pie',subplots=True,figsize=(8,8))
plt.title("Pie Chart of BEEE")
```

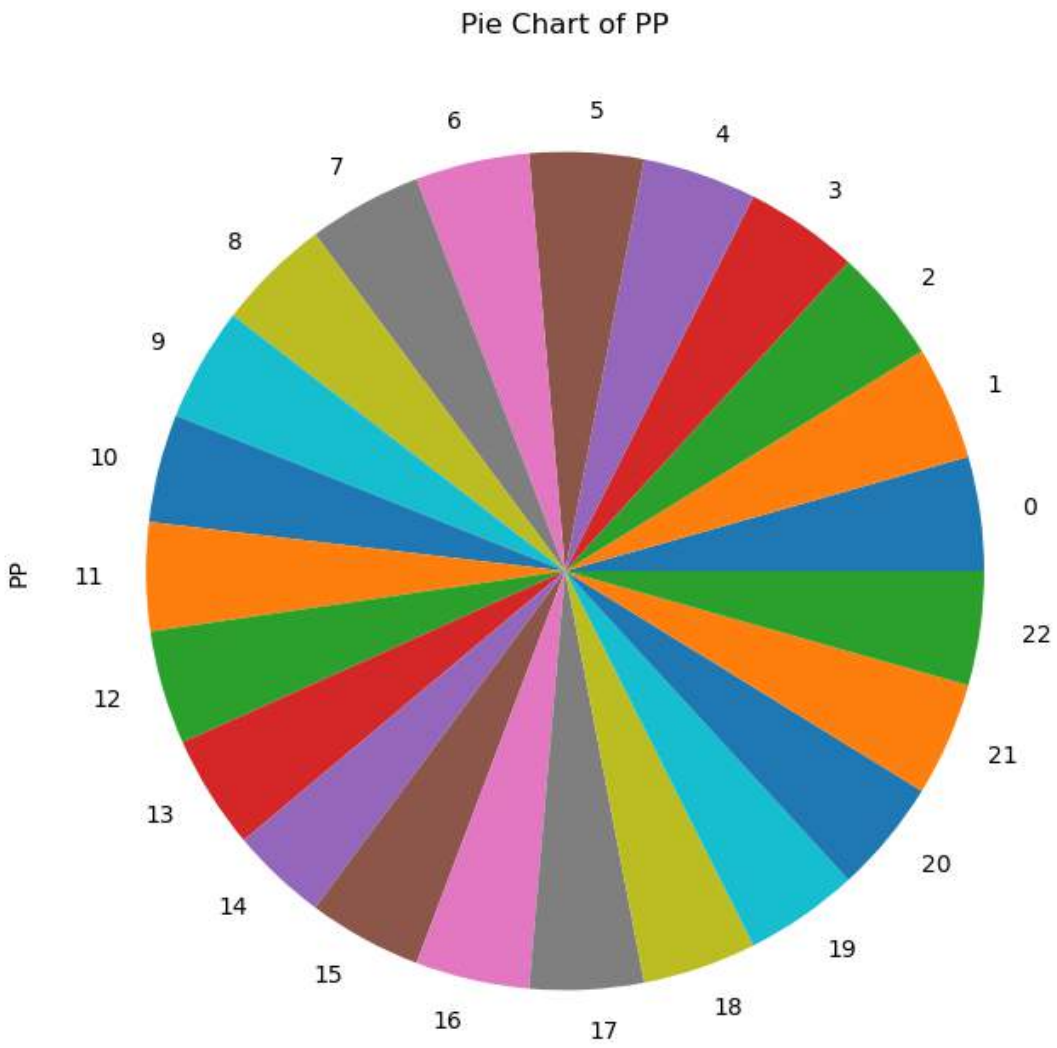
Out[55]: Text(0.5, 1.0, 'Pie Chart of BEEE')



PIE Chart of BEEE for 32 students

In [56]:
```
b['PP'].plot(kind='pie',subplots=True,figsize=(8,8))
plt.title("Pie Chart of PP")
```

Out[56]:
```
Text(0.5, 1.0, 'Pie Chart of PP')
```



PIE Chart of PP for 32 students

In [57]:
```
df.sort_values('Total').tail(10)
```

Out[57]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade
441	442	ZETA	20.0	19.0	20	19.0	20.0	19	117.0	98	A
397	398	OMEGA	20.0	20.0	19	20.0	20.0	18	117.0	98	A
472	473	ZETA	20.0	18.0	20	20.0	20.0	19	117.0	98	A
23	24	ALPHA	20.0	20.0	20	20.0	20.0	18	118.0	98	A
164	165	DELTA	20.0	20.0	20	20.0	20.0	18	118.0	98	A
406	407	OMEGA	20.0	20.0	20	20.0	20.0	18	118.0	98	A
419	420	SIGMA	20.0	20.0	20	20.0	20.0	19	119.0	99	A
217	218	EPSILON	20.0	20.0	20	19.0	20.0	20	119.0	99	A
473	474	ZETA	20.0	20.0	20	20.0	20.0	20	120.0	100	A
471	472	ZETA	20.0	20.0	20	20.0	20.0	20	120.0	100	A

In [58]:
```
df.sort_values('DV').tail(20)
```

Out[58]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade
39	40	ALPHA	20.0	17.0	20	19.0	20.0	18	114.0	95	A
365	366	ZETA	20.0	18.0	0	19.0	18.0	17	92.0	77	B
367	368	ZETA	20.0	20.0	19	20.0	18.0	16	113.0	94	A
440	441	ZETA	20.0	18.0	20	19.0	20.0	18	115.0	96	A
434	435	SIGMA	20.0	1.0	17	17.0	12.0	17	84.0	70	B
431	432	SIGMA	20.0	19.0	20	20.0	16.0	17	112.0	93	A
428	429	SIGMA	20.0	17.0	20	20.0	20.0	19	116.0	97	A
427	428	SIGMA	20.0	16.0	20	20.0	19.0	19	114.0	95	A
11	12	ALPHA	20.0	20.0	20	20.0	19.0	16	115.0	96	A
419	420	SIGMA	20.0	20.0	20	20.0	20.0	19	119.0	99	A
412	413	OMEGA	20.0	20.0	17	20.0	20.0	18	115.0	96	A
410	411	OMEGA	20.0	20.0	20	20.0	18.0	18	116.0	97	A
406	407	OMEGA	20.0	20.0	20	20.0	20.0	18	118.0	98	A
399	400	OMEGA	20.0	20.0	19	20.0	18.0	15	112.0	93	A
398	399	OMEGA	20.0	20.0	19	20.0	20.0	18	117.0	98	A
397	398	OMEGA	20.0	20.0	19	20.0	20.0	18	117.0	98	A
23	24	ALPHA	20.0	20.0	20	20.0	20.0	18	118.0	98	A
370	371	ZETA	20.0	17.0	16	20.0	20.0	14	107.0	89	B+
442	443	ZETA	20.0	11.0	18	14.0	20.0	15	98.0	82	B+
479	480	ZETA	20.0	16.0	18	19.0	20.0	19	112.0	93	A

In [59]:
```
df.sort_values('DV').head(50)
```

Out[59]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade
361	362	ZETA	0.0	0.0	6	7.0	13.0	0	26.0	22	F
173	174	DELTA	0.0	0.0	16	10.0	20.0	19	65.0	54	C
461	462	ZETA	0.0	0.0	0	0.0	0.0	0	0.0	0	F
368	369	ZETA	0.0	0.0	0	0.0	0.0	0	0.0	0	F
380	381	OMEGA	0.0	0.0	0	0.0	0.0	0	0.0	0	F
302	303	OMEGA	0.0	0.0	0	0.0	0.0	0	0.0	0	F
210	211	EPSILON	0.0	0.0	0	0.0	0.0	0	0.0	0	F
469	470	ZETA	1.0	1.0	2	0.0	10.0	0	14.0	12	F
50	51	ALPHA	1.0	16.0	15	13.0	10.0	11	66.0	55	C
453	454	ZETA	1.0	5.0	0	0.0	0.0	0	6.0	5	F
57	58	ALPHA	2.0	2.0	4	10.0	10.0	3	31.0	26	F
82	83	BETA	2.0	0.0	2	0.0	0.0	0	4.0	3	F
88	89	BETA	2.0	17.0	0	3.0	15.0	2	39.0	32	F
393	394	OMEGA	2.0	5.0	1	2.0	10.0	6	26.0	22	F
85	86	BETA	3.0	4.0	14	13.0	18.0	13	65.0	54	C
189	190	EPSILON	4.0	15.0	4	5.0	10.0	14	52.0	43	D
70	71	BETA	4.0	2.0	16	10.0	15.0	9	56.0	47	D
20	21	ALPHA	4.0	2.0	5	3.0	16.0	9	39.0	32	F
75	76	BETA	5.0	8.0	7	15.0	10.0	2	47.0	39	F
326	327	SIGMA	5.0	3.0	9	10.0	10.0	7	44.0	37	F
27	28	ALPHA	5.0	4.0	3	12.0	13.0	5	42.0	35	F
139	140	DELTA	5.0	0.0	12	4.0	20.0	15	56.0	47	D
303	304	OMEGA	5.0	0.0	3	11.0	7.0	10	36.0	30	F
364	365	ZETA	5.0	3.0	3	2.0	10.0	9	32.0	27	F
125	126	DELTA	5.0	16.0	9	7.0	18.0	14	69.0	57	C
444	445	ZETA	5.0	2.0	11	0.0	10.0	0	28.0	23	F
430	431	SIGMA	6.0	1.0	9	11.0	8.0	10	45.0	38	F
311	312	OMEGA	6.0	0.0	1	11.0	9.0	4	31.0	26	F
25	26	ALPHA	6.0	10.0	10	11.0	13.0	10	60.0	50	C
206	207	EPSILON	6.0	6.0	2	3.0	10.0	11	38.0	32	F
424	425	SIGMA	6.0	1.0	0	0.0	0.0	0	7.0	6	F
324	325	SIGMA	6.0	0.0	3	3.0	10.0	4	26.0	22	F
450	451	ZETA	6.0	3.0	7	11.0	11.0	14	52.0	43	D
98	99	BETA	6.0	7.0	16	9.0	13.0	13	64.0	53	C
51	52	ALPHA	6.0	12.0	10	11.0	10.0	3	52.0	43	D
72	73	BETA	7.0	0.0	15	10.0	18.0	11	61.0	51	C
188	189	EPSILON	7.0	10.0	7	2.0	10.0	15	51.0	42	D
340	341	SIGMA	7.0	0.0	3	0.0	13.0	8	31.0	26	F
127	128	DELTA	7.0	4.0	12	4.0	13.0	11	51.0	42	D
160	161	DELTA	7.0	0.0	14	5.0	10.0	5	41.0	34	F
372	373	ZETA	7.0	2.0	2	6.0	10.0	7	34.0	28	F
378	379	ZETA	8.0	0.0	2	6.0	15.0	8	39.0	32	F

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade
121	122	DELTA	8.0	11.0	14	6.0	15.0	9	63.0	52	C
29	30	ALPHA	8.0	2.0	11	10.0	13.0	12	56.0	47	D
53	54	ALPHA	8.0	13.0	9	11.0	10.0	10	61.0	51	C
226	227	EPSILON	8.0	0.0	3	0.0	10.0	14	35.0	29	F
61	62	BETA	8.0	8.0	15	9.0	10.0	9	59.0	49	D
110	111	BETA	8.0	11.0	14	3.0	18.0	10	64.0	53	C
97	98	BETA	8.0	16.0	14	17.0	13.0	13	81.0	68	C+
96	97	BETA	8.0	17.0	3	12.0	18.0	9	67.0	56	C

In [60]:
```
h = df[
    (df['DV'] < 10.0) |
    (df['PP'] < 10.0) |
    (df['M2'] < 10.0) |
    (df['BEEE'] < 10.0) |
    (df['FL'] < 10.0) |
    (df['FIMS'] < 10.0)
]
h['SECTION'].value_counts()
```

Out[60]:
```
SIGMA      44
GAMMA      42
ZETA       40
OMEGA      40
EPSILON    38
DELTA      35
BETA       32
ALPHA      26
Name: SECTION, dtype: int64
```

In [61]:
```
df['backlogs'] = (df[['DV', 'M2', 'PP', 'BEEE', 'FL', 'FIMS']] < 10).sum(axis=1)
df
```

Out[61]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs
0	1	ALPHA	12.0	0.0	17	9.0	19.0	15	72.0	60	C+	2
1	2	ALPHA	19.0	12.0	16	16.0	18.0	3	84.0	70	B	1
2	3	ALPHA	18.0	14.0	18	18.0	18.0	16	102.0	85	B+	0
3	4	ALPHA	15.0	9.0	19	17.0	19.0	15	94.0	78	B	1
4	5	ALPHA	18.0	17.0	19	19.0	20.0	18	111.0	92	A	0
...	...	...	...	...	...	...	...	...	...	...	...	...
474	475	ZETA	11.0	4.0	2	2.0	8.0	10	37.0	31	F	4
475	476	ZETA	18.0	2.0	12	3.0	17.0	15	67.0	56	C	2
476	477	ZETA	20.0	6.0	16	11.0	20.0	14	87.0	72	B	1
478	479	ZETA	20.0	20.0	5	19.0	18.0	14	96.0	80	B+	1
479	480	ZETA	20.0	16.0	18	19.0	20.0	19	112.0	93	A	0

In [62]:
```
j=df.sort_values('backlogs')
j
```

Out[62]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs
479	480	ZETA	20.0	16.0	18	19.0	20.0	19	112.0	93	A	0
164	165	DELTA	20.0	20.0	20	20.0	20.0	18	118.0	98	A	0
165	166	DELTA	16.0	14.0	18	18.0	15.0	19	100.0	83	B+	0
167	168	DELTA	19.0	18.0	20	16.0	19.0	19	111.0	92	A	0
168	169	DELTA	18.0	17.0	18	11.0	20.0	20	104.0	87	B+	0
...	...	...	...	...	...	...	...	...	...	...	...	...
82	83	BETA	2.0	0.0	2	0.0	0.0	0	4.0	3	F	6
416	417	OMEGA	9.0	0.0	0	0.0	0.0	0	9.0	8	F	6
453	454	ZETA	1.0	5.0	0	0.0	0.0	0	6.0	5	F	6
210	211	EPSILON	0.0	0.0	0	0.0	0.0	0	0.0	0	F	6
302	303	OMEGA	0.0	0.0	0	0.0	0.0	0	0.0	0	F	6

474 rows × 12 columns

In [63]:
```
j.value_counts('backlogs')
```

Out[63]:
```
backlogs
0    177
1    122
2     77
3     46
4     31
5     12
6      9
dtype: int64
```

In [64]:
```
df["BC"]=None
```

In [65]:
```
df
```

Out[65]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs	BC
0	1	ALPHA	12.0	0.0	17	9.0	19.0	15	72.0	60	C+	2	None
1	2	ALPHA	19.0	12.0	16	16.0	18.0	3	84.0	70	B	1	None
2	3	ALPHA	18.0	14.0	18	18.0	18.0	16	102.0	85	B+	0	None
3	4	ALPHA	15.0	9.0	19	17.0	19.0	15	94.0	78	B	1	None
4	5	ALPHA	18.0	17.0	19	19.0	20.0	18	111.0	92	A	0	None
...	...	...	...	...	...	...	...	...	...	...	...	...	...
474	475	ZETA	11.0	4.0	2	2.0	8.0	10	37.0	31	F	4	None
475	476	ZETA	18.0	2.0	12	3.0	17.0	15	67.0	56	C	2	None
476	477	ZETA	20.0	6.0	16	11.0	20.0	14	87.0	72	B	1	None
478	479	ZETA	20.0	20.0	5	19.0	18.0	14	96.0	80	B+	1	None
479	480	ZETA	20.0	16.0	18	19.0	20.0	19	112.0	93	A	0	None

474 rows × 13 columns

In [66]:
```
j.value_counts('backlogs')
```

Out[66]:
```
backlogs
0    177
1    122
2     77
3     46
4     31
5     12
6      9
dtype: int64
```

Backlogs couunt

In [67]:
```
a = df[df['BC']==6]
a.value_counts('SECTION')
```

Out[67]:
```
Series([], dtype: int64)
```

Students belongs to section who got 6 backlogs

backlogs count for each subject in each section

In [68]:
```
a = df[df['DV']<10]
a.value_counts('SECTION').sum()
a = df[df['DV']<10]
a.value_counts('SECTION')
```

Out[68]:
```
SECTION
BETA       15
DELTA      11
ZETA       11
ALPHA       9
EPSILON     6
OMEGA       6
SIGMA       6
dtype: int64
```

In [69]:
```
a = df[df['M2']<10]
a.value_counts('SECTION').sum()
a = df[df['M2']<10]
a.value_counts('SECTION')
```

Out[69]:
```
SECTION
GAMMA      39
SIGMA      38
ZETA       37
OMEGA      35
EPSILON    29
DELTA      21
BETA       13
ALPHA      10
dtype: int64
```

In [70]:
```
a = df[df['PP']<10]
a.value_counts('SECTION').sum()
a = df[df['PP']<10]
a.value_counts('SECTION')
```

Out[70]:
```
SECTION
GAMMA      28
OMEGA      27
SIGMA      22
ZETA       18
EPSILON    14
DELTA      10
BETA        8
ALPHA       7
dtype: int64
```

In [71]:
```
a = df[df['BEEE']<10]
a.value_counts('SECTION').sum()
a = df[df['BEEE']<10]
a.value_counts('SECTION')
```

Out[71]:
```
SECTION
EPSILON    28
DELTA      25
BETA       18
ZETA       17
OMEGA      11
SIGMA      11
GAMMA      10
ALPHA       4
dtype: int64
```

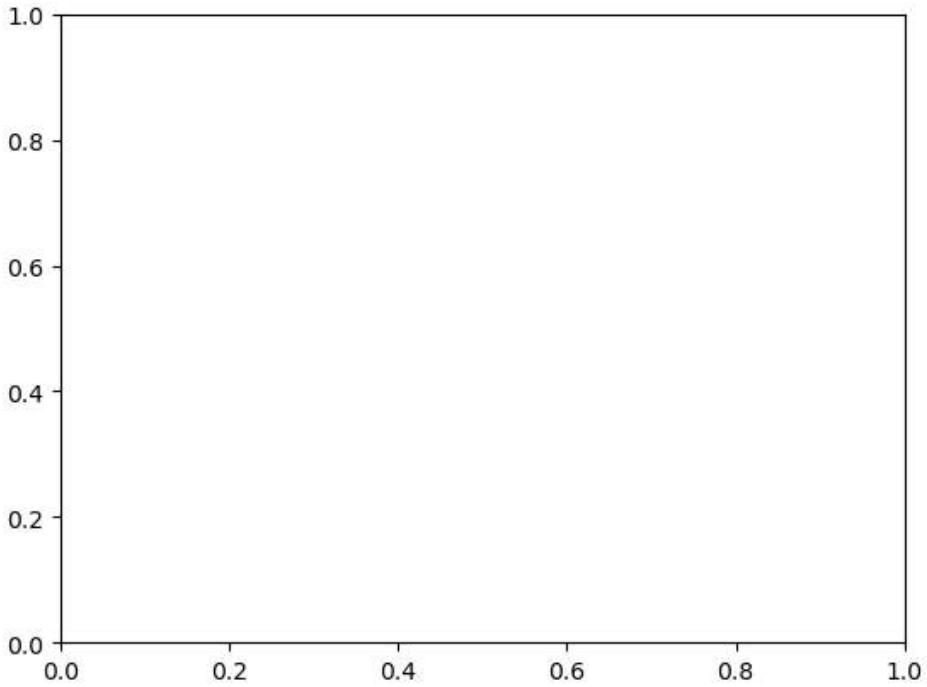
In [72]:
```
a = df[df['FL']<10]
a.value_counts('SECTION').sum()
a = df[df['FL']<10]
a.value_counts('SECTION')
```

Out[72]:
```
SECTION
ZETA       7
OMEGA      6
GAMMA      4
SIGMA      3
BETA       2
EPSILON    1
dtype: int64
```

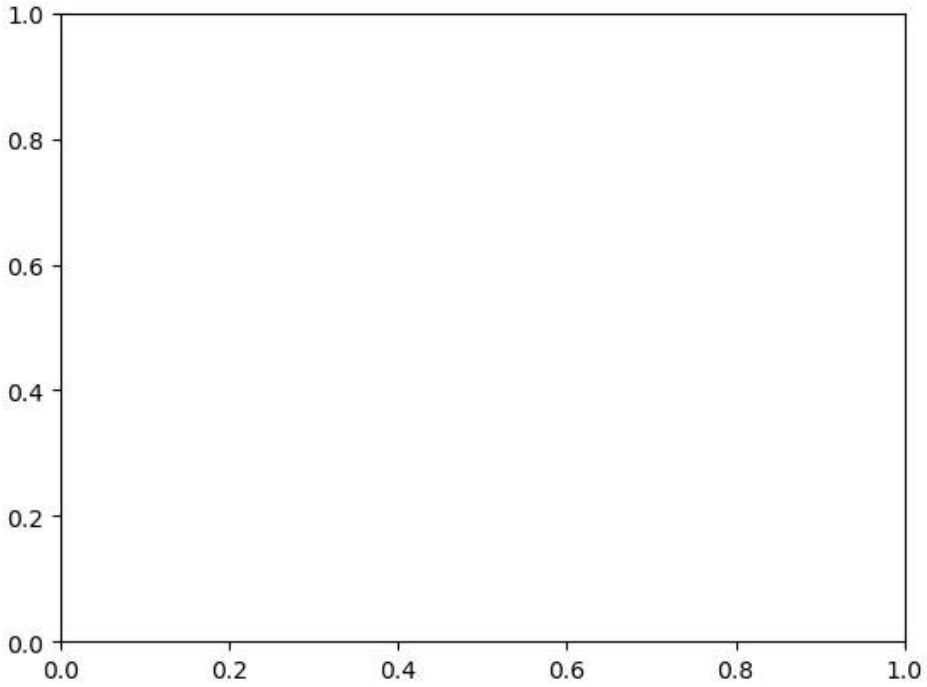
In [73]:
```
a = df[df['FIMS']<10]
a.value_counts('SECTION').sum()
a = df[df['FIMS']<10]
a.value_counts('SECTION')
```

Out[73]:
```
SECTION
OMEGA      17
ALPHA      13
BETA       13
SIGMA      12
GAMMA      11
ZETA        9
DELTA       6
EPSILON     4
dtype: int64
```

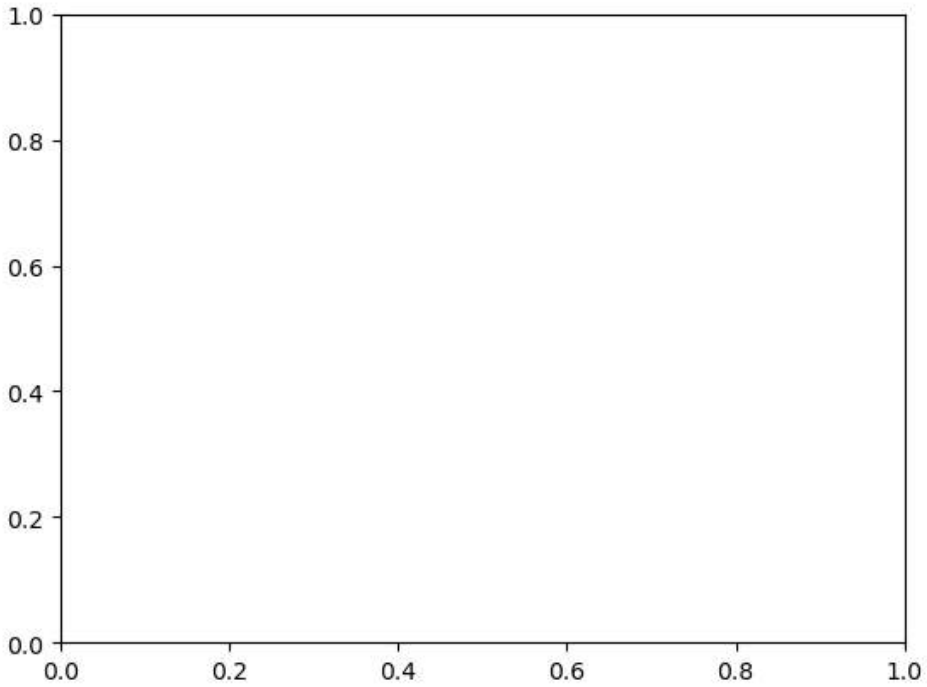
In [74]:
```
sns.scatterplot(x='BC', y='DV', data=df,color = 'red')
plt.show()
```



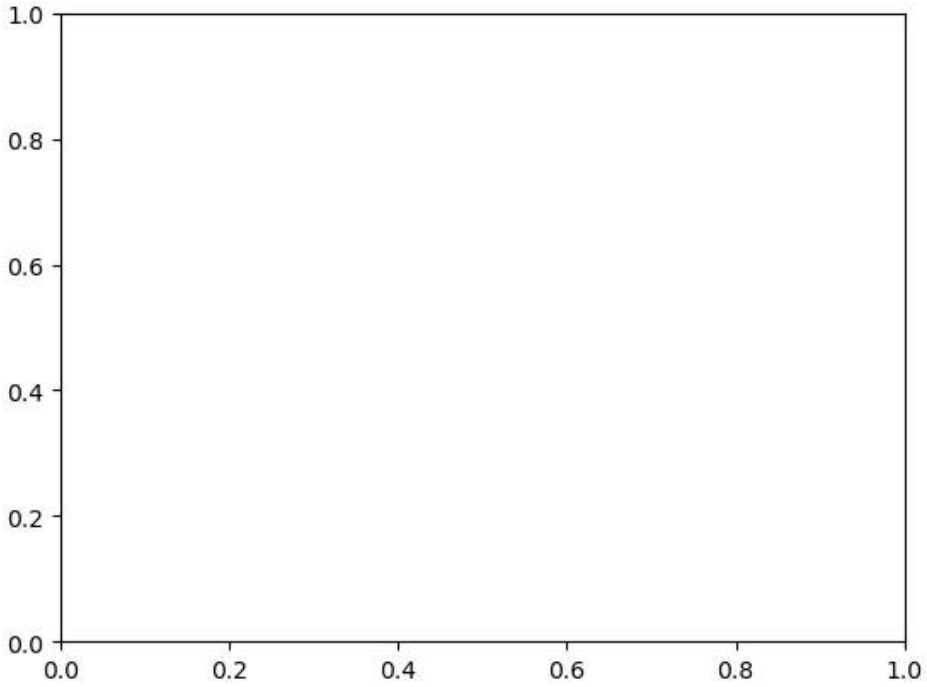
In [75]:
```
sns.scatterplot(x='BC', y='PP', data=df , color = 'green')
plt.show()
```



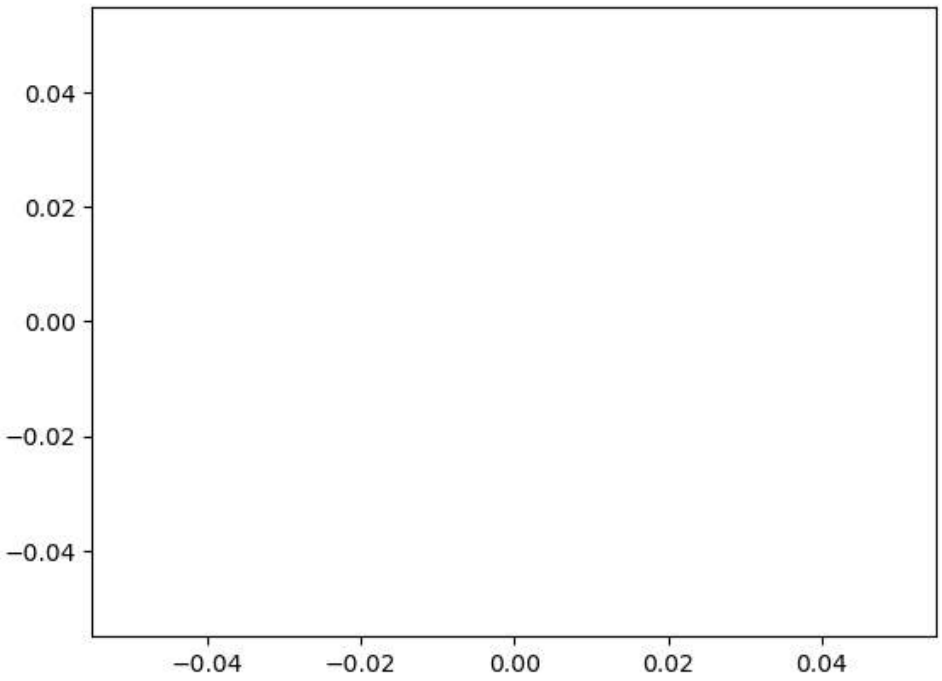
In [76]:
```
sns.scatterplot(x='BC', y='BEEE', data=df, color = 'yellow')
plt.show()
```



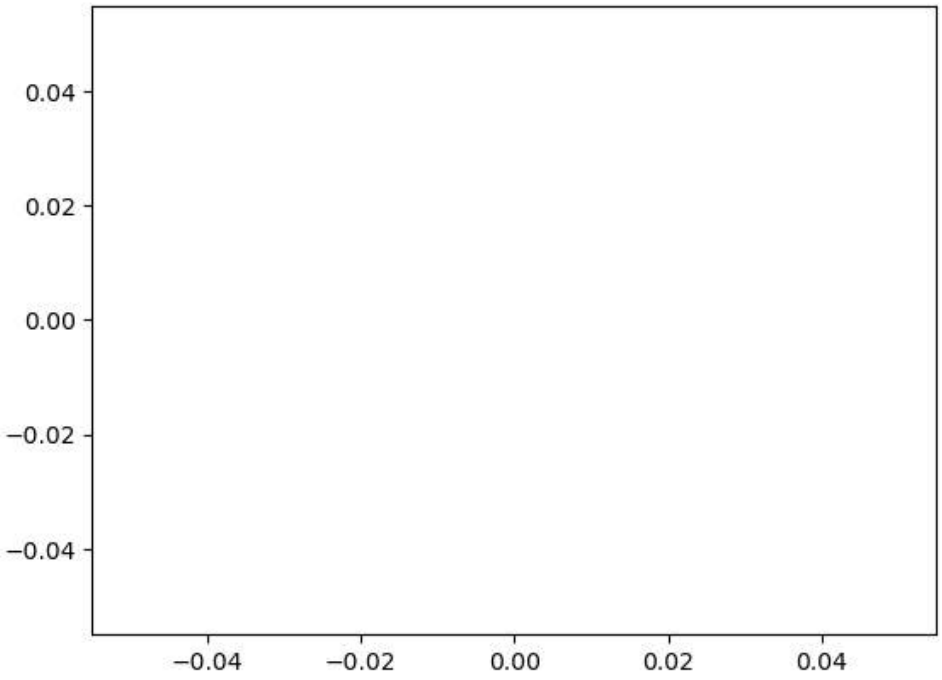
In [77]:
```
sns.scatterplot(x='BC', y='M2', data=df,color = 'blue')
plt.show()
```



In [78]:
```
sns.scatterplot(x='BC', y='FL', data=df)
plt.show()
```



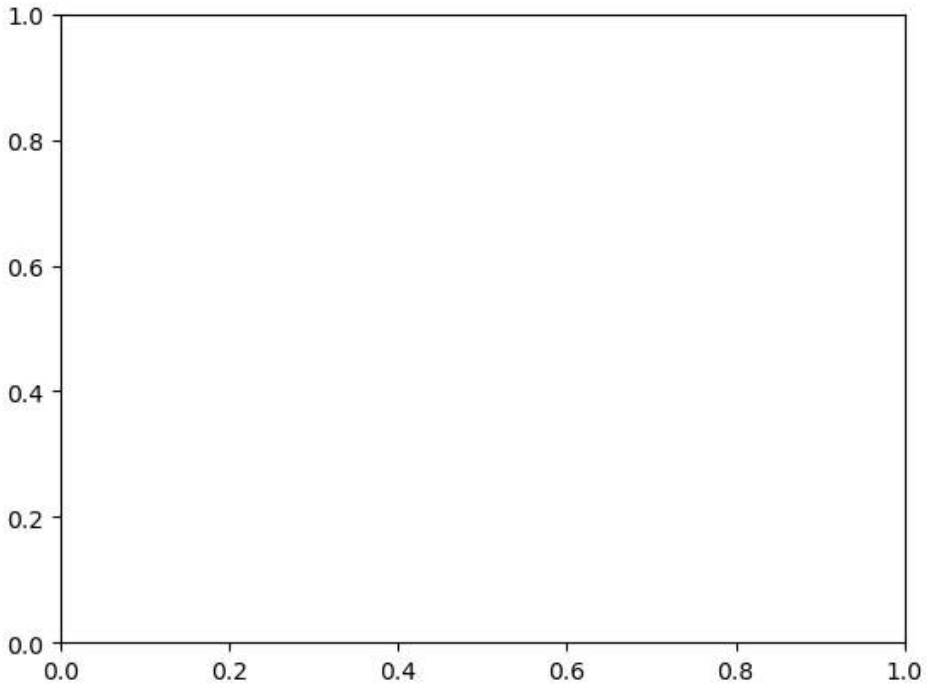
In [79]:
```
sns.scatterplot(x='BC', y='FIMS', data=df)
plt.show()
```



In [80]:
```
sns.lineplot(x='BC', y='DV', data=df, color = 'green')
plt.show()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3802, in Index.get_loc(self, ke
y, method, tolerance)
   3801 try:
-> 3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:

File ~\anaconda3\lib\site-packages\pandas\_libs\index.pyx:138, in pandas._libs.index.IndexEngi
ne.get_loc()

File ~\anaconda3\lib\site-packages\pandas\_libs\index.pyx:165, in pandas._libs.index.IndexEngi
ne.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:5745, in pandas._libs.hashtable.PyObjectHashTabl
e.get_item()

File pandas\_libs\hashtable_class_helper.pxi:5753, in pandas._libs.hashtable.PyObjectHashTabl
e.get_item()

KeyError: 'x'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[80], line 1
----> 1 sns.lineplot(x='BC', y='DV', data=df, color = 'green')
      2 plt.show()

File ~\anaconda3\lib\site-packages\seaborn\relational.py:645, in lineplot(data, x, y, hue, siz
e, style, units, palette, hue_order, hue_norm, sizes, size_order, size_norm, dashes, markers,
style_order, estimator, errorbar, n_boot, seed, orient, sort, err_style, err_kws, legend, ci,
ax, **kwargs)
    642 color = kwargs.pop("color", kwargs.pop("c", None))
    643 kwargs["color"] = _default_color(ax.plot, hue, color, kwargs)
--> 645 p.plot(ax, kwargs)
    646 return ax

File ~\anaconda3\lib\site-packages\seaborn\relational.py:432, in _LinePlotter.plot(self, ax, k
ws)
    427     sort_cols = [var for var in sort_vars if var in self.variables]
    428     sub_data = sub_data.sort_values(sort_cols)
    430 if (
    431     self.estimator is not None
--> 432     and sub_data[orient].value_counts().max() > 1
    433 ):
    434     if "units" in self.variables:
    435         # TODO eventually relax this constraint
    436         err = "estimator must be None when specifying units"

File ~\anaconda3\lib\site-packages\pandas\core\frame.py:3807, in DataFrame.__getitem__(self, k
ey)
   3805 if self.columns.nlevels > 1:
   3806     return self._getitem_multilevel(key)
-> 3807 indexer = self.columns.get_loc(key)
   3808 if is_integer(indexer):
   3809     indexer = [indexer]

File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3804, in Index.get_loc(self, ke
y, method, tolerance)
   3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:
-> 3804     raise KeyError(key) from err
   3805 except TypeError:
   3806     # If we have a listlike key, _check_indexing_error will raise
   3807     #  InvalidIndexError. Otherwise we fall through and re-raise
   3808     #  the TypeError.
```

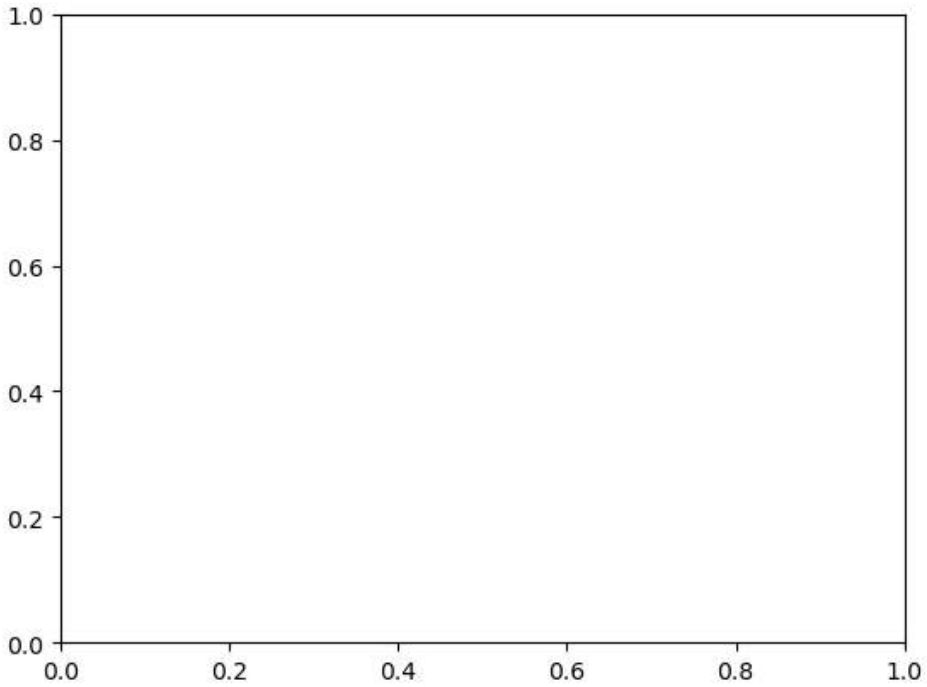
```
   3809     self._check_indexing_error(key)

KeyError: 'x'
```



In [81]:
```
sns.lineplot(x='BC', y='PP', data=df, color = 'red')
plt.show()
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3802, in Index.get_loc(self, ke
y, method, tolerance)
   3801 try:
-> 3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:

File ~\anaconda3\lib\site-packages\pandas\_libs\index.pyx:138, in pandas._libs.index.IndexEngi
ne.get_loc()

File ~\anaconda3\lib\site-packages\pandas\_libs\index.pyx:165, in pandas._libs.index.IndexEngi
ne.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:5745, in pandas._libs.hashtable.PyObjectHashTabl
e.get_item()

File pandas\_libs\hashtable_class_helper.pxi:5753, in pandas._libs.hashtable.PyObjectHashTabl
e.get_item()

KeyError: 'x'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
Cell In[81], line 1
----> 1 sns.lineplot(x='BC', y='PP', data=df, color = 'red')
      2 plt.show()

File ~\anaconda3\lib\site-packages\seaborn\relational.py:645, in lineplot(data, x, y, hue, siz
e, style, units, palette, hue_order, hue_norm, sizes, size_order, size_norm, dashes, markers,
style_order, estimator, errorbar, n_boot, seed, orient, sort, err_style, err_kws, legend, ci,
ax, **kwargs)
    642 color = kwargs.pop("color", kwargs.pop("c", None))
    643 kwargs["color"] = _default_color(ax.plot, hue, color, kwargs)
--> 645 p.plot(ax, kwargs)
    646 return ax

File ~\anaconda3\lib\site-packages\seaborn\relational.py:432, in _LinePlotter.plot(self, ax, k
ws)
    427     sort_cols = [var for var in sort_vars if var in self.variables]
    428     sub_data = sub_data.sort_values(sort_cols)
    430 if (
    431     self.estimator is not None
--> 432     and sub_data[orient].value_counts().max() > 1
    433 ):
    434     if "units" in self.variables:
    435         # TODO eventually relax this constraint
    436         err = "estimator must be None when specifying units"

File ~\anaconda3\lib\site-packages\pandas\core\frame.py:3807, in DataFrame.__getitem__(self, k
ey)
   3805 if self.columns.nlevels > 1:
   3806     return self._getitem_multilevel(key)
-> 3807 indexer = self.columns.get_loc(key)
   3808 if is_integer(indexer):
   3809     indexer = [indexer]

File ~\anaconda3\lib\site-packages\pandas\core\indexes\base.py:3804, in Index.get_loc(self, ke
y, method, tolerance)
   3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:
-> 3804     raise KeyError(key) from err
   3805 except TypeError:
   3806     # If we have a listlike key, _check_indexing_error will raise
   3807     #  InvalidIndexError. Otherwise we fall through and re-raise
   3808     #  the TypeError.
```

```
    3809     self._check_indexing_error(key)

KeyError: 'x'
```



In [ ]:
```
sns.lineplot(x='BC', y='BEEE', data=df, color = 'orange')
plt.show()
```

In [ ]:
```
sns.lineplot(x='BC', y='M2', data=df, color = 'violet')
plt.show()
```

In [ ]:
```
sns.lineplot(x='BC', y='FL', data=df, color = 'purple')
plt.show()
```

In [ ]:
```
sns.lineplot(x='BC', y='FIMS', data=df)
plt.show()
```

In [ ]:
```
sns.barplot(x='BC', y='DV', data=df)
plt.show()
```

In [ ]:
```
sns.barplot(x='BC', y='BEEE', data=df, color = 'red')
plt.show()
```

In [ ]:
```
sns.barplot(x='BC', y='PP', data=df, color = 'green')
plt.show()
```

In [ ]:
```
sns.barplot(x='BC', y='M2', data=df, color = 'yellow')
plt.show()
```

In [ ]:
```
sns.histplot(data=df['BC'], bins=30,color = 'red')
plt.show()
```

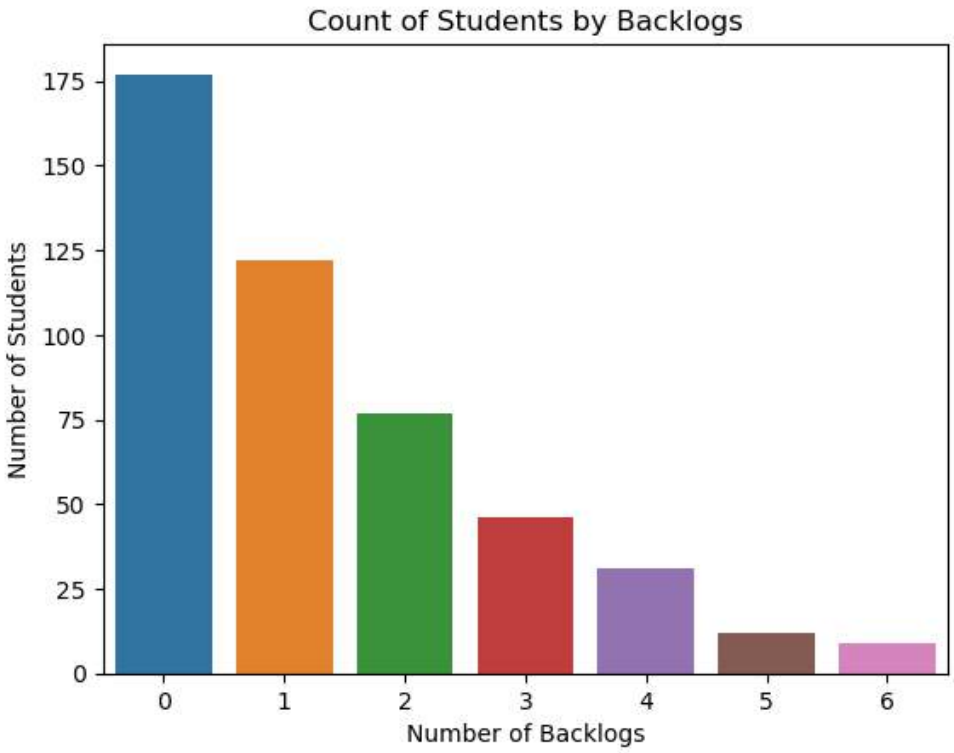
In [ ]:
```
sns.histplot(data=df['DV'], bins=30, color = 'green')
plt.show()
```

In [ ]:
```
sns.histplot(data=df['BEEE'], bins=30, color = 'blue')
plt.show()
```

In [ ]:
```
sns.histplot(data=df['FL'], bins=30, color = 'yellow')
plt.show()
```

In [ ]:
```

```

In [82]:
```
sns.countplot(x='backlogs', data=df)
plt.title('Count of Students by Backlogs')
plt.xlabel('Number of Backlogs')
plt.ylabel('Number of Students')
plt.show()
```



Bar chart for backlogs and students and some of the students has 6 backlogs also

In [83]:
```
h = df[
    (df['DV'] == 20.0) |
    (df['PP'] == 20.0) |
    (df['M2'] == 20.0) |
    (df['BEEE'] == 20.0) |
    (df['FL'] == 20.0) |
    (df['FIMS'] == 20.0)
]
h
```

Out[83]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs	BC
4	5	ALPHA	18.0	17.0	19	19.0	20.0	18	111.0	92	A	0	None
6	7	ALPHA	15.0	10.0	20	20.0	15.0	14	94.0	78	B	0	None
7	8	ALPHA	17.0	17.0	19	20.0	19.0	13	105.0	88	B+	0	None
8	9	ALPHA	10.0	18.0	0	20.0	19.0	15	82.0	68	C+	1	None
9	10	ALPHA	18.0	19.0	20	20.0	20.0	15	112.0	93	A	0	None
...	...	...	...	...	...	...	...	...	...	...	...	...	...
472	473	ZETA	20.0	18.0	20	20.0	20.0	19	117.0	98	A	0	None
473	474	ZETA	20.0	20.0	20	20.0	20.0	20	120.0	100	A	0	None
476	477	ZETA	20.0	6.0	16	11.0	20.0	14	87.0	72	B	1	None
478	479	ZETA	20.0	20.0	5	19.0	18.0	14	96.0	80	B+	1	None
479	480	ZETA	20.0	16.0	18	19.0	20.0	19	112.0	93	A	0	None

182 rows × 13 columns

In [84]:
```
h.value_counts("BC")
```

Out[84]:
```
Series([], dtype: int64)
```

In [85]:
```
h[h["BC"] == 4]
```

Out[85]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs	BC

In [86]:
```
h[h["BC"] == 3]
```

Out[86]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs	BC

In [87]:
```
h.value_counts("DV")
```

Out[87]:
```
DV
20.0    51
18.0    28
17.0    25
19.0    24
16.0    19
12.0     7
15.0     7
13.0     6
14.0     6
11.0     4
10.0     3
0.0      1
5.0      1
dtype: int64
```

In [88]:
```
h.value_counts("M2")
```

Out[88]:
```
M2
20.0    44
17.0    19
18.0    15
15.0    15
19.0    10
11.0    10
8.0      8
12.0     7
9.0      6
10.0     6
6.0      5
14.0     5
5.0      5
16.0     5
4.0      5
3.0      4
1.0      3
7.0      3
13.0     3
0.0      3
2.0      1
dtype: int64
```

In [89]:
```
h.value_counts("BEEE")
```

Out[89]:
```
BEEE
20.0    76
19.0    24
17.0    17
18.0    14
16.0    10
14.0    10
15.0     5
13.0     4
12.0     4
10.0     4
11.0     3
9.0      3
4.0      2
0.0      2
8.0      1
7.0      1
6.0      1
5.0      1
dtype: int64
```

In [90]:
```
h.value_counts("FIMS")
```

Out[90]:
```
FIMS
18    45
16    26
19    21
17    20
15    17
14    16
20    12
13    11
9      5
11     3
0      2
12     2
8      1
10     1
dtype: int64
```

In [91]:
```
h.value_counts("PP")
```

Out[91]:
```
PP
20    69
19    20
18    15
16    13
14    11
17     8
15     7
12     7
11     6
13     6
10     4
9      4
0      4
8      2
7      2
5      2
1      1
6      1
dtype: int64
```

In [92]:
```
h.value_counts("FL")
```

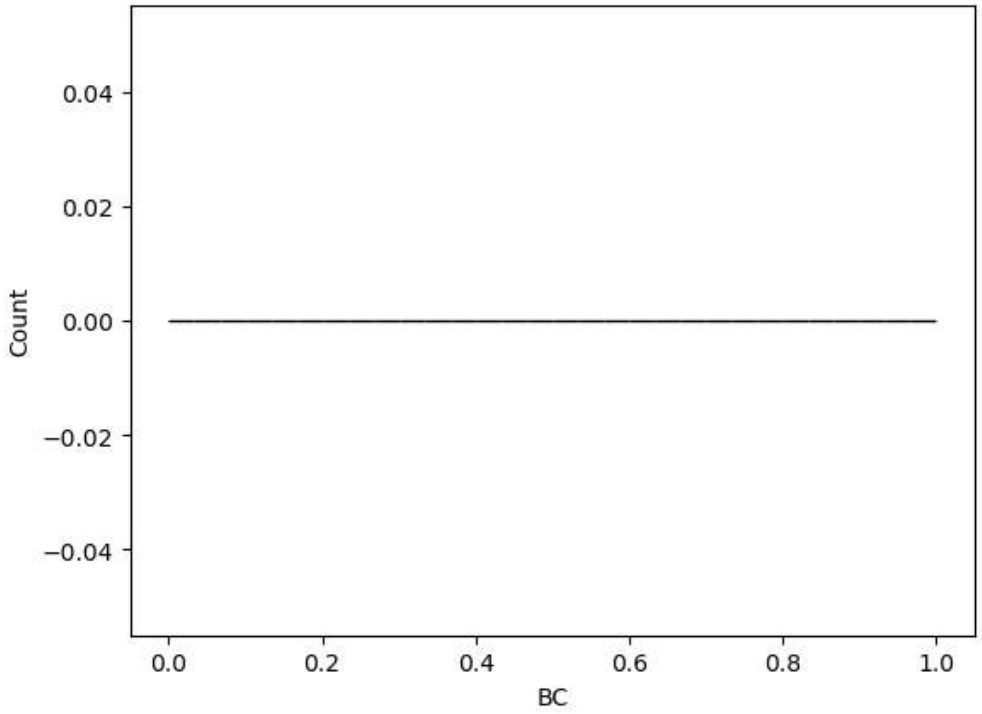
Out[92]:
```
FL
20.0    118
18.0     17
15.0     14
19.0     13
13.0      8
10.0      5
17.0      3
16.0      2
12.0      1
14.0      1
dtype: int64
```

In [93]:
```
h.describe()
```

Out[93]:

	S.NO	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage
count	182.000000	182.000000	182.000000	182.000000	182.000000	182.000000	182.000000	182.000000	182.000000
mean	233.824176	17.247253	14.076923	16.395604	17.175824	18.604396	16.192308	99.692308	83.082418
std	147.962000	3.035584	5.771392	4.648719	4.023669	2.495871	3.081104	14.027355	11.690239
min	5.000000	0.000000	0.000000	0.000000	0.000000	10.000000	0.000000	48.000000	40.000000
25%	97.250000	16.000000	10.000000	14.000000	16.000000	18.000000	15.000000	91.250000	76.250000
50%	221.500000	18.000000	16.000000	18.000000	19.000000	20.000000	17.000000	103.000000	86.000000
75%	367.500000	20.000000	19.000000	20.000000	20.000000	20.000000	18.000000	111.000000	92.000000
max	480.000000	20.000000	20.000000	20.000000	20.000000	20.000000	20.000000	120.000000	100.000000

In [94]:
```
sns.histplot(data=df['BC'], bins=30, color = 'red')
plt.show()
```



In [95]:
```
z=df.loc[(df['BC'] >= 3) & (df['BC'] <= 6)]
z=z.reset_index()
y = z.value_counts('BC')
y
```

Out[95]:
```
Series([], dtype: int64)
```

In [96]:
```
def number_format(pct, all_values):
    absolute = int(round(pct / 100. * sum(all_values)))
    return f"{absolute}"

plt.figure(figsize=(8, 8))
y.plot.pie(
    autopct=lambda pct: number_format(pct, y),
    startangle=90,
    cmap="viridis"
)
plt.title("BC count")
plt.ylabel("")
plt.show()
```

In [97]:
```
df[df["SECTION"] == "ALPHA"].count()
```

Out[97]:
```
S.NO          60
SECTION       60
DV            60
M2            60
PP            60
BEEE          60
FL            60
FIMS          60
Total         60
Percentage    60
Grade         60
backlogs      60
BC             0
dtype: int64
```

In [98]:
```
df['DV'] = pd.to_numeric(df['DV'], errors='coerce')
df['M2'] = pd.to_numeric(df['M2'], errors='coerce')
df['PP'] = pd.to_numeric(df['PP'], errors='coerce')
df['BEEE'] = pd.to_numeric(df['BEEE'], errors='coerce')
df['FL'] = pd.to_numeric(df['FL'], errors='coerce')
df['FIMS'] = pd.to_numeric(df['FIMS'], errors='coerce')
df.fillna(0, inplace=True)
```

In [99]:
```
df.isnull().sum()
```

Out[99]:
```
S.NO          0
SECTION       0
DV            0
M2            0
PP            0
BEEE          0
FL            0
FIMS          0
Total         0
Percentage    0
Grade         0
backlogs      0
BC            0
dtype: int64
```

In [100]:
```
df
```

Out[100]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs	BC
0	1	ALPHA	12.0	0.0	17	9.0	19.0	15	72.0	60	C+	2	0
1	2	ALPHA	19.0	12.0	16	16.0	18.0	3	84.0	70	B	1	0
2	3	ALPHA	18.0	14.0	18	18.0	18.0	16	102.0	85	B+	0	0
3	4	ALPHA	15.0	9.0	19	17.0	19.0	15	94.0	78	B	1	0
4	5	ALPHA	18.0	17.0	19	19.0	20.0	18	111.0	92	A	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
474	475	ZETA	11.0	4.0	2	2.0	8.0	10	37.0	31	F	4	0
475	476	ZETA	18.0	2.0	12	3.0	17.0	15	67.0	56	C	2	0
476	477	ZETA	20.0	6.0	16	11.0	20.0	14	87.0	72	B	1	0
478	479	ZETA	20.0	20.0	5	19.0	18.0	14	96.0	80	B+	1	0
479	480	ZETA	20.0	16.0	18	19.0	20.0	19	112.0	93	A	0	0

474 rows × 13 columns

In [101]:
```
df['SECTION'] = df['SECTION'].fillna('SIGMA')
df['SECTION'] = df['SECTION'].replace('', 'SIGMA')
df
```

Out[101]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs	BC
0	1	ALPHA	12.0	0.0	17	9.0	19.0	15	72.0	60	C+	2	0
1	2	ALPHA	19.0	12.0	16	16.0	18.0	3	84.0	70	B	1	0
2	3	ALPHA	18.0	14.0	18	18.0	18.0	16	102.0	85	B+	0	0
3	4	ALPHA	15.0	9.0	19	17.0	19.0	15	94.0	78	B	1	0
4	5	ALPHA	18.0	17.0	19	19.0	20.0	18	111.0	92	A	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
474	475	ZETA	11.0	4.0	2	2.0	8.0	10	37.0	31	F	4	0
475	476	ZETA	18.0	2.0	12	3.0	17.0	15	67.0	56	C	2	0
476	477	ZETA	20.0	6.0	16	11.0	20.0	14	87.0	72	B	1	0
478	479	ZETA	20.0	20.0	5	19.0	18.0	14	96.0	80	B+	1	0
479	480	ZETA	20.0	16.0	18	19.0	20.0	19	112.0	93	A	0	0

474 rows × 13 columns

In [102]:
```
df['S.NO'] = range(1, len(df) + 1)
df
```

Out[102]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs	BC
0	1	ALPHA	12.0	0.0	17	9.0	19.0	15	72.0	60	C+	2	0
1	2	ALPHA	19.0	12.0	16	16.0	18.0	3	84.0	70	B	1	0
2	3	ALPHA	18.0	14.0	18	18.0	18.0	16	102.0	85	B+	0	0
3	4	ALPHA	15.0	9.0	19	17.0	19.0	15	94.0	78	B	1	0
4	5	ALPHA	18.0	17.0	19	19.0	20.0	18	111.0	92	A	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
474	470	ZETA	11.0	4.0	2	2.0	8.0	10	37.0	31	F	4	0
475	471	ZETA	18.0	2.0	12	3.0	17.0	15	67.0	56	C	2	0
476	472	ZETA	20.0	6.0	16	11.0	20.0	14	87.0	72	B	1	0
478	473	ZETA	20.0	20.0	5	19.0	18.0	14	96.0	80	B+	1	0
479	474	ZETA	20.0	16.0	18	19.0	20.0	19	112.0	93	A	0	0

474 rows × 13 columns

In [103]:
```
df['FL'] = df['FL'].fillna(0)
df=df.dropna()
df
```

Out[103]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs	BC
0	1	ALPHA	12.0	0.0	17	9.0	19.0	15	72.0	60	C+	2	0
1	2	ALPHA	19.0	12.0	16	16.0	18.0	3	84.0	70	B	1	0
2	3	ALPHA	18.0	14.0	18	18.0	18.0	16	102.0	85	B+	0	0
3	4	ALPHA	15.0	9.0	19	17.0	19.0	15	94.0	78	B	1	0
4	5	ALPHA	18.0	17.0	19	19.0	20.0	18	111.0	92	A	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
474	470	ZETA	11.0	4.0	2	2.0	8.0	10	37.0	31	F	4	0
475	471	ZETA	18.0	2.0	12	3.0	17.0	15	67.0	56	C	2	0
476	472	ZETA	20.0	6.0	16	11.0	20.0	14	87.0	72	B	1	0
478	473	ZETA	20.0	20.0	5	19.0	18.0	14	96.0	80	B+	1	0
479	474	ZETA	20.0	16.0	18	19.0	20.0	19	112.0	93	A	0	0

474 rows × 13 columns

In [104]:
```
df.shape
```

Out[104]: (474, 13)

In [105]:
```
df['S.NO'] = range(1, len(df) + 1)
df
```

Out[105]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs	BC
0	1	ALPHA	12.0	0.0	17	9.0	19.0	15	72.0	60	C+	2	0
1	2	ALPHA	19.0	12.0	16	16.0	18.0	3	84.0	70	B	1	0
2	3	ALPHA	18.0	14.0	18	18.0	18.0	16	102.0	85	B+	0	0
3	4	ALPHA	15.0	9.0	19	17.0	19.0	15	94.0	78	B	1	0
4	5	ALPHA	18.0	17.0	19	19.0	20.0	18	111.0	92	A	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...
474	470	ZETA	11.0	4.0	2	2.0	8.0	10	37.0	31	F	4	0
475	471	ZETA	18.0	2.0	12	3.0	17.0	15	67.0	56	C	2	0
476	472	ZETA	20.0	6.0	16	11.0	20.0	14	87.0	72	B	1	0
478	473	ZETA	20.0	20.0	5	19.0	18.0	14	96.0	80	B+	1	0
479	474	ZETA	20.0	16.0	18	19.0	20.0	19	112.0	93	A	0	0

474 rows × 13 columns

In [106]:
```
df.shape
```

Out[106]:
```
(474, 13)
```

In [107]:
```
df.isnull().sum()
```

Out[107]:
```
S.NO          0
SECTION       0
DV            0
M2            0
PP            0
BEEE          0
FL            0
FIMS          0
Total         0
Percentage    0
Grade         0
backlogs      0
BC            0
dtype: int64
```

In [108]:
```
df.info
```

Out[108]:
```
<bound method DataFrame.info of      S.NO SECTION    DV    M2  PP  BEEE    FL  FIMS  Total  Pe
rcentage Grade  \
0       1   ALPHA  12.0   0.0  17   9.0  19.0    15   72.0          60     C+
1       2   ALPHA  19.0  12.0  16  16.0  18.0     3   84.0          70      B
2       3   ALPHA  18.0  14.0  18  18.0  18.0    16  102.0          85     B+
3       4   ALPHA  15.0   9.0  19  17.0  19.0    15   94.0          78      B
4       5   ALPHA  18.0  17.0  19  19.0  20.0    18  111.0          92      A
..    ...     ...   ...   ...  ..   ...   ...   ...    ...         ...    ...
474   470    ZETA  11.0   4.0   2   2.0   8.0    10   37.0          31      F
475   471    ZETA  18.0   2.0  12   3.0  17.0    15   67.0          56      C
476   472    ZETA  20.0   6.0  16  11.0  20.0    14   87.0          72      B
478   473    ZETA  20.0  20.0   5  19.0  18.0    14   96.0          80     B+
479   474    ZETA  20.0  16.0  18  19.0  20.0    19  112.0          93      A

     backlogs  BC
0           2   0
1           1   0
2           0   0
3           1   0
4           0   0
..        ...  ..
474         4   0
475         2   0
476         1   0
478         1   0
479         0   0

[474 rows x 13 columns]>
```

In [109]:
```
df.describe()
```

Out[109]:

	S.NO	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage
count	474.000000	474.000000	474.000000	474.000000	474.000000	474.000000	474.000000	474.000000	474.000000
mean	237.500000	14.405063	10.139241	12.875527	13.289030	15.523207	13.632911	79.864979	66.535865
std	136.976275	4.602653	6.380270	5.870998	5.853642	4.277935	4.693598	24.718857	20.609420
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	119.250000	12.000000	4.000000	9.000000	9.000000	13.000000	11.000000	64.000000	53.000000
50%	237.500000	15.500000	10.000000	14.000000	15.000000	15.000000	15.000000	83.000000	69.000000
75%	355.750000	18.000000	16.000000	18.000000	18.000000	19.000000	17.000000	98.000000	82.000000
max	474.000000	20.000000	20.000000	20.000000	20.000000	20.000000	20.000000	120.000000	100.000000

In [110]:
```
from scipy.stats import ttest_ind
from scipy.stats import ttest_rel
```

In [111]:
```
ttest_ind(df[df['SECTION'] == 'ALPHA']['DV'], df[df['SECTION']== 'BETA']['DV'])
```

Out[111]:
```
Ttest_indResult(statistic=2.3418185924318102, pvalue=0.020866453244001094)
```

In [112]:
```
ttest_rel(df[df['SECTION'] == 'ALPHA']['DV'], df[df['SECTION']== 'BETA']['DV'])
```

Out[112]:
```
TtestResult(statistic=2.3172456109384103, pvalue=0.023979527821469917, df=59)
```

In [113]:
```
from scipy.stats import chi2_contingency
```

In [114]:
```
data = (df[df['SECTION'] == 'ALPHA']['DV'], df[df['SECTION']== 'BETA']['DV'])
stat, p, dof, expected = chi2_contingency(data)
print(stat)
print(p)
print(dof)
```

```
113.91846891456558
2.3496708155645757e-05
59
```

In [115]:
```
data = (df[df['SECTION'] == 'ALPHA']['DV'], df[df['SECTION']== 'BETA']['DV'])
stat, p, dof, expected = chi2_contingency(data)
chi2_contingency(data)
```

Out[115]:
```
Chi2ContingencyResult(statistic=113.91846891456558, pvalue=2.3496708155645757e-05, dof=59, exp
ected_freq=array([[16.65730696, 14.50797703, 16.11997447, 13.97064454, 16.11997447,
        13.97064454, 14.50797703, 15.58264199, 13.97064454, 20.41863433,
        11.28398213, 19.88130185, 12.3586471 , 14.50797703, 19.34396937,
         9.67198468, 17.19463944, 12.89597958, 17.73197192, 20.41863433,
        12.3586471 , 17.19463944, 10.20931717, 16.11997447, 14.50797703,
         4.83599234, 18.2693044 ,  9.67198468, 11.28398213,  9.67198468,
        15.04530951, 13.97064454, 15.04530951, 15.04530951, 15.58264199,
        12.89597958, 10.20931717, 12.3586471 , 13.43331206, 15.58264199,
        15.04530951, 15.58264199, 13.97064454, 11.82131461, 18.80663689,
        12.3586471 , 12.3586471 , 14.50797703, 13.97064454, 15.04530951,
         4.83599234,  8.59731972, 17.73197192, 12.3586471 , 13.43331206,
        18.80663689, 12.3586471 ,  7.52265475, 10.20931717, 18.2693044 ],
       [14.34269304, 12.49202297, 13.88002553, 12.02935546, 13.88002553,
        12.02935546, 12.49202297, 13.41735801, 12.02935546, 17.58136567,
         9.71601787, 17.11869815, 10.6413529 , 12.49202297, 16.65603063,
         8.32801532, 14.80536056, 11.10402042, 15.26802808, 17.58136567,
        10.6413529 , 14.80536056,  8.79068283, 13.88002553, 12.49202297,
         4.16400766, 15.7306956 ,  8.32801532,  9.71601787,  8.32801532,
        12.95469049, 12.02935546, 12.95469049, 12.95469049, 13.41735801,
        11.10402042,  8.79068283, 10.6413529 , 11.56668794, 13.41735801,
        12.95469049, 13.41735801, 12.02935546, 10.17868539, 16.19336311,
        10.6413529 , 10.6413529 , 12.49202297, 12.02935546, 12.95469049,
         4.16400766,  7.40268028, 15.26802808, 10.6413529 , 11.56668794,
        16.19336311, 10.6413529 ,  6.47734525,  8.79068283, 15.7306956 ]]))
```

In [116]:
```
df[df['SECTION'] == 'ALPHA'].DV.mean()
```

Out[116]:
```
14.033333333333333
```

In [117]:
```
import scipy.stats as stats
```

In [118]:
```
t_statistic, p_value = stats.ttest_1samp(df[df['SECTION'] == 'ALPHA']['DV'] , df.DV.mean())
print(t_statistic, p_value)
```

```
-0.618692845981171 0.5385001708772008
```

In [119]:
```
t_statistic, p_value = stats.ttest_1samp(df[df['SECTION'] == 'BETA']['DV'] , df.DV.mean())
print(t_statistic, p_value)
```

```
-4.0271887976567315 0.0001634927314623452
```

In [120]:
```
t_statistic, p_value = stats.ttest_1samp(df[df['SECTION'] == 'GAMMA']['DV'] , df.DV.mean())
print(t_statistic, p_value)
```

```
5.360516566254314 1.4453792736728807e-06
```

In [121]:
```
t_statistic, p_value = stats.ttest_1samp(df[df['SECTION'] == 'DELTA']['DV'] , df.DV.mean())
print(t_statistic, p_value)
```

```
-1.672647664892037 0.09969192507810967
```

In [122]:
```
t_statistic, p_value = stats.ttest_1samp(df[df['SECTION'] == 'SIGMA']['DV'] , df.DV.mean())
print(t_statistic, p_value)
```

```
2.02737462602529 0.04722782875442706
```

In [123]:
```
t_statistic, p_value = stats.ttest_1samp(df[df['SECTION'] == 'ZETA']['DV'] , df.DV.mean())
print(t_statistic, p_value)
```

```
0.8920612807760117 0.37604618564503456
```

In [124]:
```
t_statistic, p_value = stats.ttest_1samp(df[df['SECTION'] == 'OMEGA']['DV'] , df.DV.mean())
print(t_statistic, p_value)
```

```
0.7039164464119071 0.48430139497653446
```

In [125]:
```
t_statistic, p_value = stats.ttest_1samp(df[df['SECTION'] == 'EPSILON']['DV'] , df.DV.mean())
print(t_statistic, p_value)
```

```
-0.26190775637361224 0.7943535438301653
```

In [126]:
```
df.describe()
```

Out[126]:

	S.NO	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage
count	474.000000	474.000000	474.000000	474.000000	474.000000	474.000000	474.000000	474.000000	474.000000
mean	237.500000	14.405063	10.139241	12.875527	13.289030	15.523207	13.632911	79.864979	66.535865
std	136.976275	4.602653	6.380270	5.870998	5.853642	4.277935	4.693598	24.718857	20.609420
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	119.250000	12.000000	4.000000	9.000000	9.000000	13.000000	11.000000	64.000000	53.000000
50%	237.500000	15.500000	10.000000	14.000000	15.000000	15.000000	15.000000	83.000000	69.000000
75%	355.750000	18.000000	16.000000	18.000000	18.000000	19.000000	17.000000	98.000000	82.000000
max	474.000000	20.000000	20.000000	20.000000	20.000000	20.000000	20.000000	120.000000	100.000000

In [ ]:
```
```