In [1]:

```
import pandas as pd
```

In [2]:

```
p=pd.read_excel("MIDMARKS-MINOR1-EXAM.xlsx")
```

In [3]:

```
p
```

Out[3]:

	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

Importing pandas and reading Excel file data

In [4]:

```
p.head(91)
```

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
...	...	...	...	...	...	...	...	...
86	87	BETA	17	18	19	20	20	18
87	88	BETA	13	17	14	19	15	17
88	89	BETA	2	17	0	3	15	2
89	90	BETA	10	6	15	10	15	10
90	91	BETA	17	19	20	17	20	18

91 rows × 8 columns

In [5]:

```
p
```

Out[5]:

	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 [6]:

```
p.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 [7]:

```
p['DV'] = pd.to_numeric(p['DV'], errors='coerce').astype('Int64')
p['M-II'] = pd.to_numeric(p['M-II'], errors='coerce').astype('Int64')
p['PP'] = pd.to_numeric(p['PP'], errors='coerce').astype('Int64')
p['BEEE'] = pd.to_numeric(p['BEEE'], errors='coerce').astype('Int64')
p['FL'] = pd.to_numeric(p['FL'], errors='coerce').astype('Int64')
p['FIMS'] = pd.to_numeric(p['FIMS'], errors='coerce').astype('Int64')
```

In [8]:

```
p.info()
p["Total"]=p["DV"]+p["M-II"]+p["PP"]+p["BEEE"]+p["FL"]+p["FIMS"]
p
```

```
<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       472 non-null    Int64 
 3   M-II     465 non-null    Int64 
 4   PP       470 non-null    Int64 
 5   BEEE     464 non-null    Int64 
 6   FL       470 non-null    Int64 
 7   FIMS     466 non-null    Int64 
dtypes: Int64(6), int64(1), object(1)
memory usage: 32.9+ KB
```

Out[8]:

	S.NO	SECTION	DV	M-II	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	NaN	18	2	12	3	17	15	67
476	477	NaN	20	6	16	11	20	14	87
477	478	NaN	20	<NA>	18	13	20	18	<NA>
478	479	NaN	20	20	5	19	18	14	96
479	480	NaN	20	16	18	19	20	19	112

480 rows × 9 columns

In [9]:

```
p.rename(columns={'Total':'TOTAL'},inplace=True)
p.rename(columns={'M-II':'M2'},inplace=True)
p
```

Out[9]:

	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	NaN	18	2	12	3	17	15	67
476	477	NaN	20	6	16	11	20	14	87
477	478	NaN	20	<NA>	18	13	20	18	<NA>
478	479	NaN	20	20	5	19	18	14	96
479	480	NaN	20	16	18	19	20	19	112

480 rows × 9 columns

Renaming column 'M-II' to 'M2' in dataframe

In [10]:
```
p.fillna(0)
```

Out[10]:

	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	0	18	2	12	3	17	15	67
476	477	0	20	6	16	11	20	14	87
477	478	0	20	0	18	13	20	18	0
478	479	0	20	20	5	19	18	14	96
479	480	0	20	16	18	19	20	19	112

480 rows × 9 columns

In [11]:
```
p.head(10)
```

Out[11]:

	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
5	6	ALPHA	17	16	18	10	15	9	85
6	7	ALPHA	15	10	20	20	15	14	94
7	8	ALPHA	17	17	19	20	19	13	105
8	9	ALPHA	10	18	<NA>	20	19	15	<NA>
9	10	ALPHA	18	19	20	20	20	15	112

In [12]:
```
p = p.fillna(-1)
```

In [13]:
```
p.head(10)
```

Out[13]:

	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
5	6	ALPHA	17	16	18	10	15	9	85
6	7	ALPHA	15	10	20	20	15	14	94
7	8	ALPHA	17	17	19	20	19	13	105
8	9	ALPHA	10	18	-1	20	19	15	-1
9	10	ALPHA	18	19	20	20	20	15	112

In [16]:

```
p.loc[600:630]
```

Out[16]:

S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	TOTAL

In [17]:

```
p.loc[560:570]
```

Out[17]:

S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	TOTAL

In [18]:

```
p.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 480 entries, 0 to 479
Data columns (total 9 columns):
 #   Column   Non-Null Count  Dtype 
---  ------   --------------  ----- 
 0   S.NO     480 non-null    int64 
 1   SECTION  480 non-null    object
 2   DV       480 non-null    Int64 
 3   M2       480 non-null    Int64 
 4   PP       480 non-null    Int64 
 5   BEEE     480 non-null    Int64 
 6   FL       480 non-null    Int64 
 7   FIMS     480 non-null    Int64 
 8   TOTAL    480 non-null    Int64 
dtypes: Int64(7), int64(1), object(1)
memory usage: 37.2+ KB
```

In [19]:

```
p
```

Out[19]:

	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	-1	18	2	12	3	17	15	67
476	477	-1	20	6	16	11	20	14	87
477	478	-1	20	-1	18	13	20	18	-1
478	479	-1	20	20	5	19	18	14	96
479	480	-1	20	16	18	19	20	19	112

480 rows × 9 columns

In [20]:

```
p["percentage"]=(p["TOTAL"]/120)*100
```

In [21]:

```
p
```

Out[21]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	TOTAL	percentage
0	1	ALPHA	12	0	17	9	19	15	72	60.0
1	2	ALPHA	19	12	16	16	18	3	84	70.0
2	3	ALPHA	18	14	18	18	18	16	102	85.0
3	4	ALPHA	15	9	19	17	19	15	94	78.333333
4	5	ALPHA	18	17	19	19	20	18	111	92.5
...	...	...	...	...	...	...	...	...	...	...
475	476	-1	18	2	12	3	17	15	67	55.833333
476	477	-1	20	6	16	11	20	14	87	72.5
477	478	-1	20	-1	18	13	20	18	-1	-0.833333
478	479	-1	20	20	5	19	18	14	96	80.0
479	480	-1	20	16	18	19	20	19	112	93.333333

480 rows × 10 columns

Calculating percentage based on 'Total' column values

In [22]:
```
p["grade"]=((p["TOTAL"]/120)*10).round()
```

In [23]:
```
p
```

Out[23]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	TOTAL	percentage	grade
0	1	ALPHA	12	0	17	9	19	15	72	60.0	6.0
1	2	ALPHA	19	12	16	16	18	3	84	70.0	7.0
2	3	ALPHA	18	14	18	18	18	16	102	85.0	8.0
3	4	ALPHA	15	9	19	17	19	15	94	78.333333	8.0
4	5	ALPHA	18	17	19	19	20	18	111	92.5	9.0
...	...	...	...	...	...	...	...	...	...	...	...
475	476	-1	18	2	12	3	17	15	67	55.833333	6.0
476	477	-1	20	6	16	11	20	14	87	72.5	7.0
477	478	-1	20	-1	18	13	20	18	-1	-0.833333	-0.0
478	479	-1	20	20	5	19	18	14	96	80.0	8.0
479	480	-1	20	16	18	19	20	19	112	93.333333	9.0

480 rows × 11 columns

In [24]:
```
def assign_grade(percentage):
    if percentage >= 90:
        return 'A'
    elif percentage >= 80:
        return 'B'
    elif percentage >= 70:
        return 'C'
    elif percentage >= 60:
        return 'D'
    else:
        return 'F'

p['Grade'] = p['percentage'].apply(assign_grade)
p
```

Out[24]:

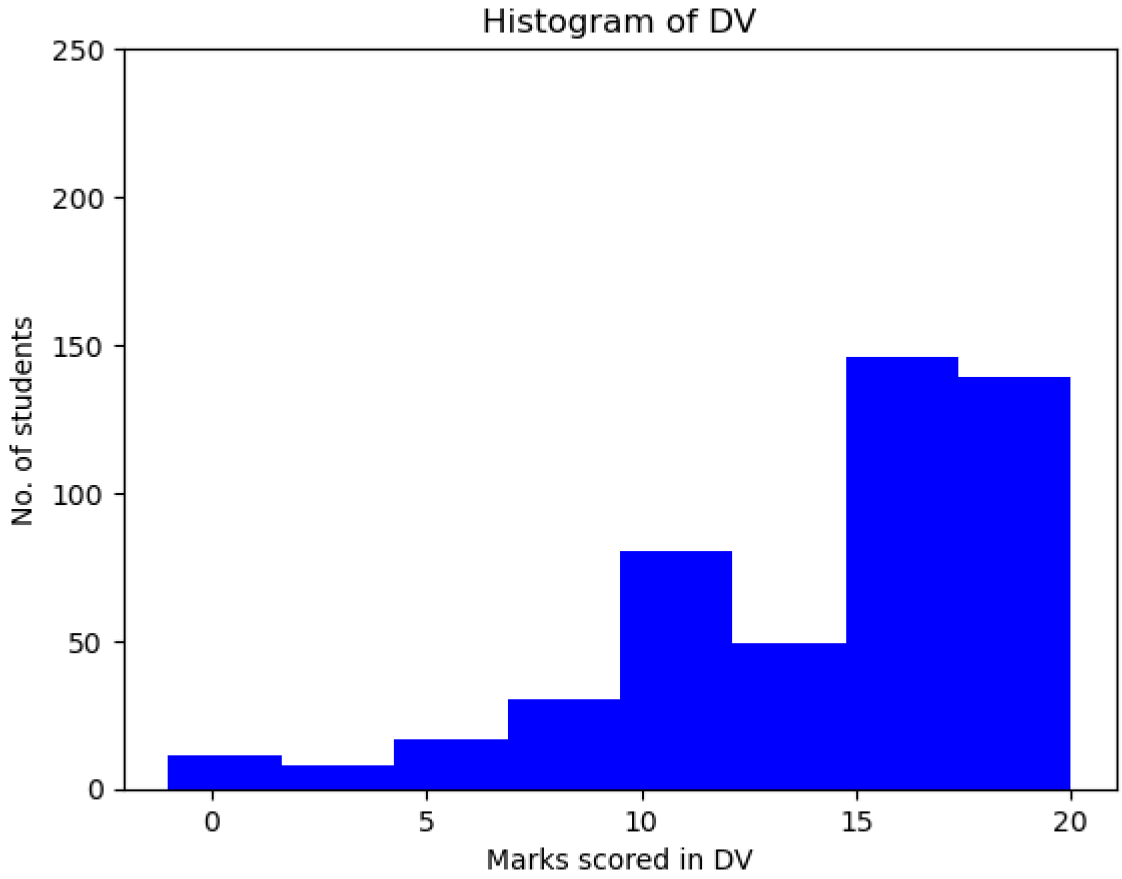
	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	TOTAL	percentage	grade	Grade
0	1	ALPHA	12	0	17	9	19	15	72	60.0	6.0	D
1	2	ALPHA	19	12	16	16	18	3	84	70.0	7.0	C
2	3	ALPHA	18	14	18	18	18	16	102	85.0	8.0	B
3	4	ALPHA	15	9	19	17	19	15	94	78.333333	8.0	C
4	5	ALPHA	18	17	19	19	20	18	111	92.5	9.0	A
...	...	...	...	...	...	...	...	...	...	...	...	...
475	476	-1	18	2	12	3	17	15	67	55.833333	6.0	F
476	477	-1	20	6	16	11	20	14	87	72.5	7.0	C
477	478	-1	20	-1	18	13	20	18	-1	-0.833333	-0.0	F
478	479	-1	20	20	5	19	18	14	96	80.0	8.0	B
479	480	-1	20	16	18	19	20	19	112	93.333333	9.0	A

480 rows × 12 columns

Assigning grades based on percentage values in dataframe

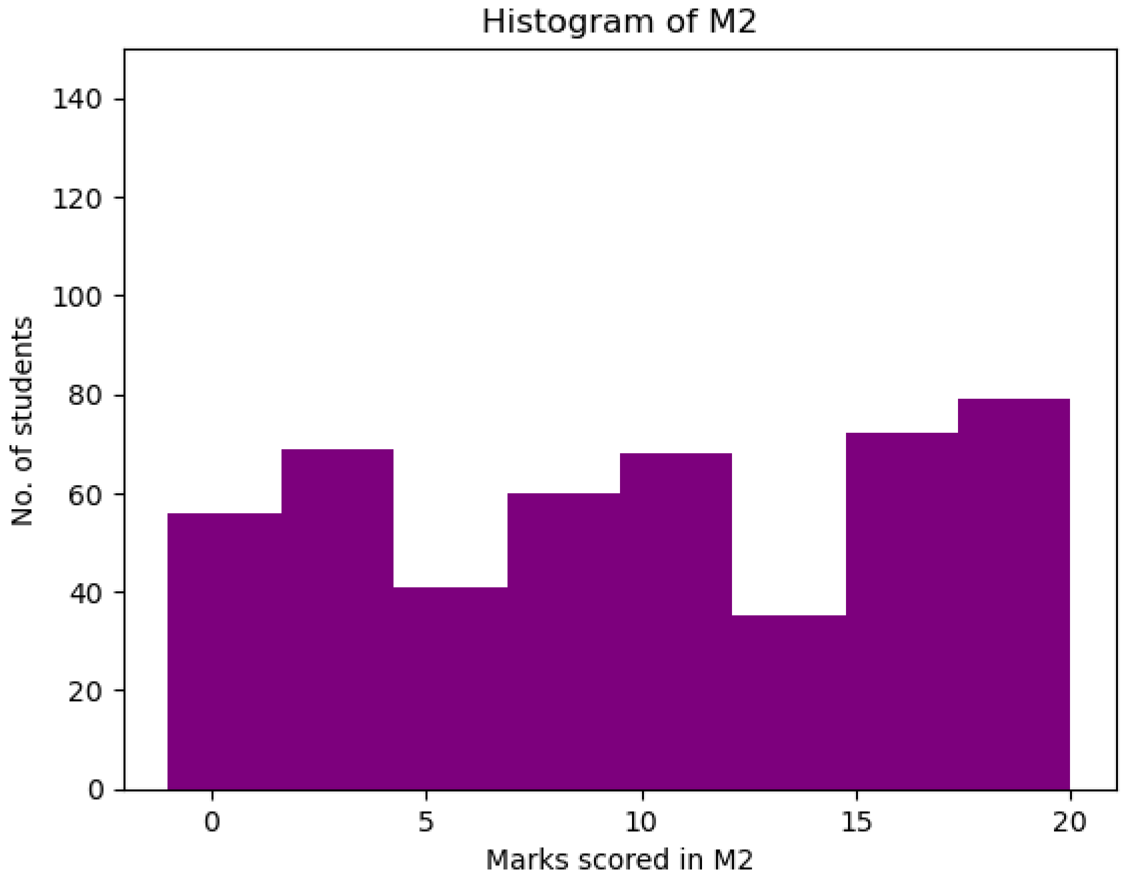
In [25]:

```python
import matplotlib.pyplot as plt
plt.hist(p['DV'], color='blue', bins=8)
plt.ylim(0, 250)
plt.xlabel("Marks scored in DV")
plt.ylabel("No. of students")
plt.title("Histogram of DV")
plt.show()
#print(p['DV'].value_counts())
```



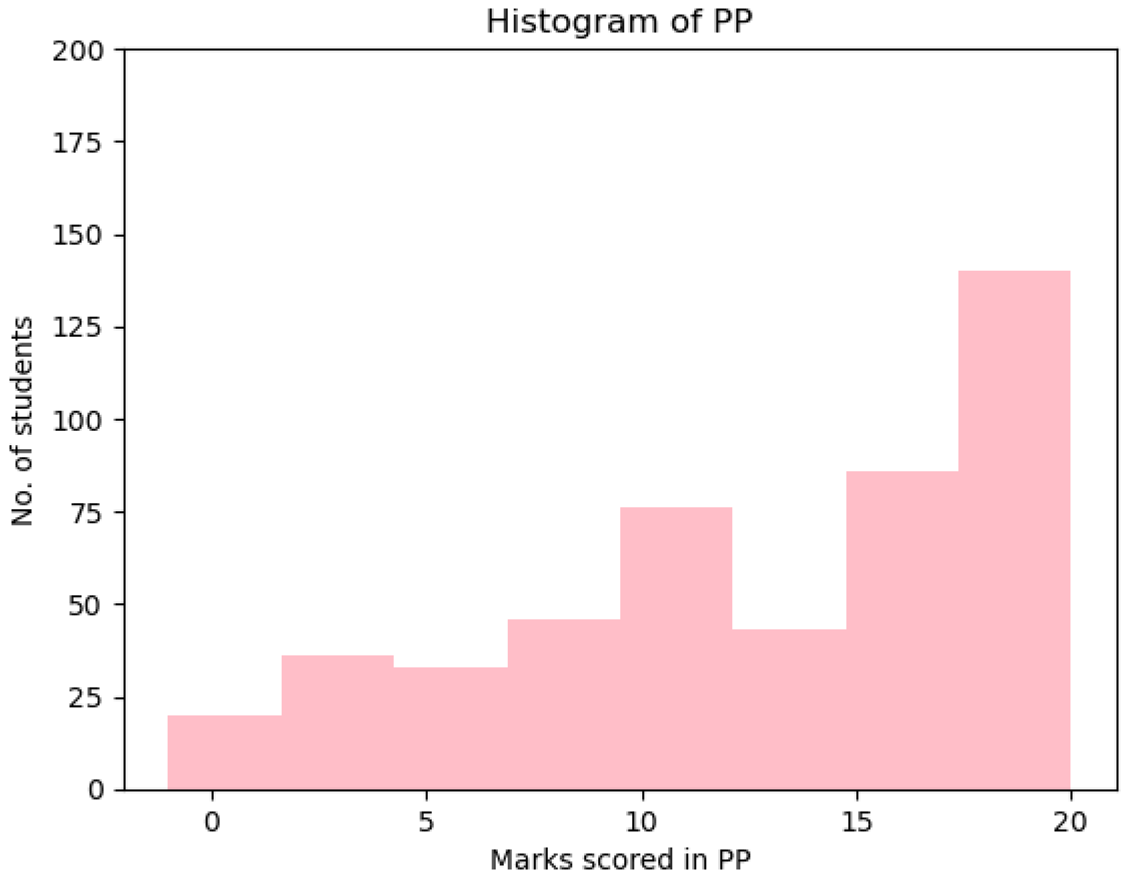
Creating histogram to visualize 'DV' subject marks distribution

In [26]:
```
plt.hist(p['M2'], color='purple', bins=8)
plt.ylim(0, 150)
plt.xlabel("Marks scored in M2")
plt.ylabel("No. of students")
plt.title("Histogram of M2")
plt.show()
#print(p['M2'].value_counts())
```



Creating histogram to visualize 'M2' subject marks distribution

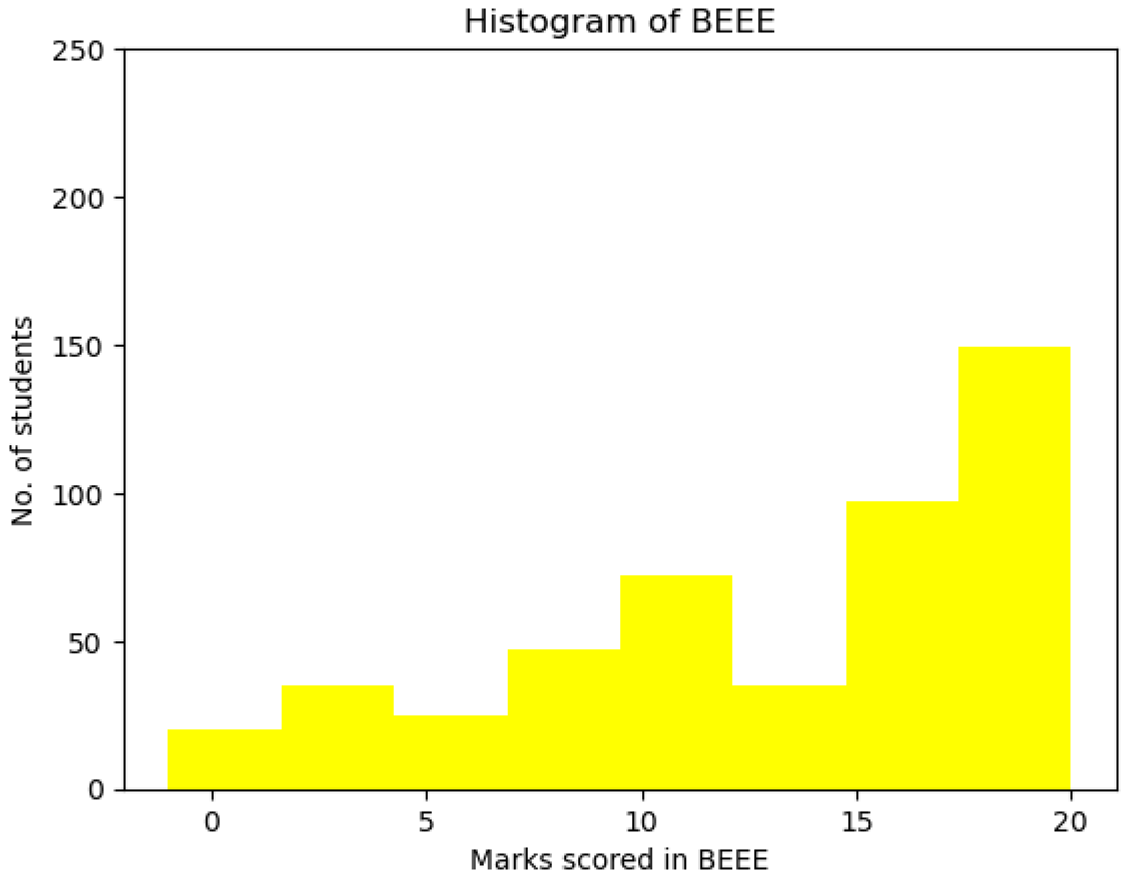
```
In [27]: plt.hist(p['PP'], color='pink', bins=8)
         plt.ylim(0, 200)
         plt.xlabel("Marks scored in PP")
         plt.ylabel("No. of students")
         plt.title("Histogram of PP")
         plt.show()
```



```
In [ ]:
```

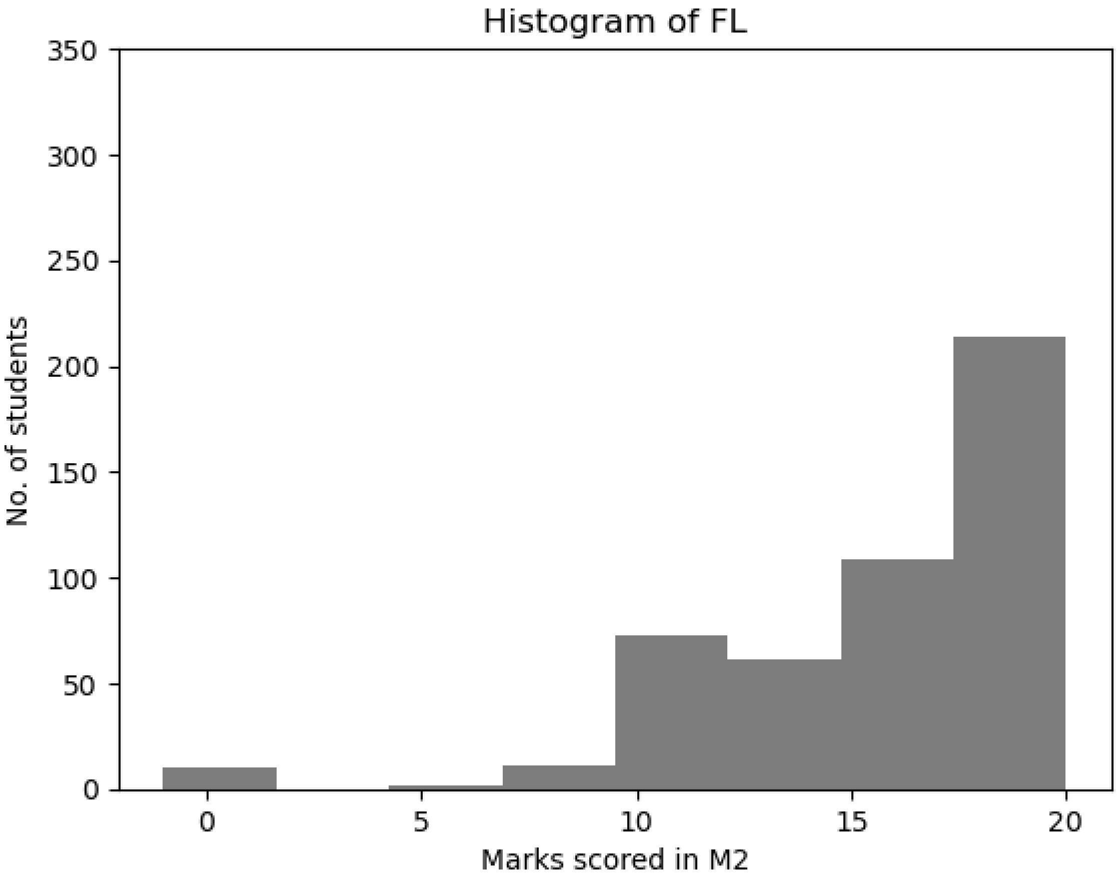
Creating histogram to visualize 'PP' subject marks distribution

```
In [28]: plt.hist(p['BEEE'], color='yellow', bins=8)
         plt.ylim(0, 250)
         plt.xlabel("Marks scored in BEEE")
         plt.ylabel("No. of students")
         plt.title("Histogram of BEEE")
         plt.show()
```



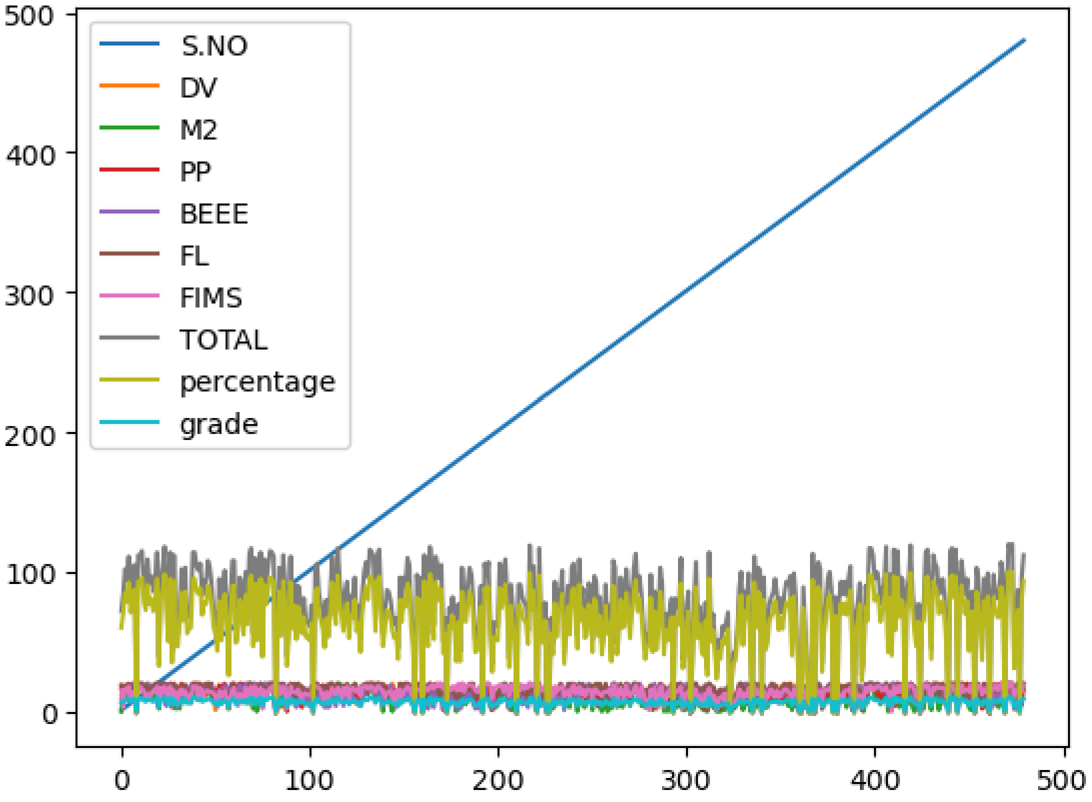
Creating histogram to visualize 'BEEE' subject marks distribution

```
In [29]: plt.hist(p['FL'], color='gray', bins=8)
         plt.ylim(0, 350)
         plt.xlabel("Marks scored in M2")
         plt.ylabel("No. of students")
         plt.title("Histogram of FL")
         plt.show()
```



Creating histogram to visualize 'FL' subject marks distribution

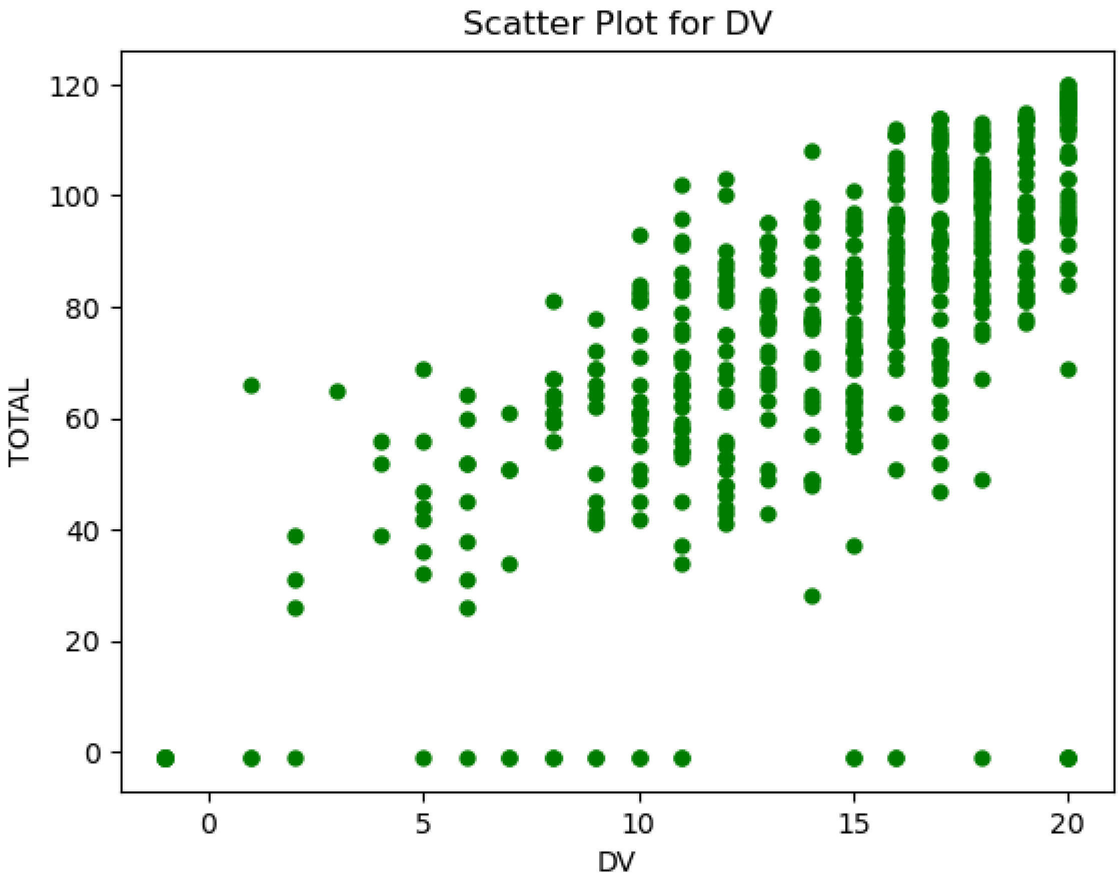
In [30]:
```
p.plot()
plt.show()
```



Plotting all columns in the dataframe for visualization

In [31]:
```
p.plot.scatter(x = 'DV', y = 'TOTAL',color='green',s=25)
plt.title("Scatter Plot for DV")
```

Out[31]:
```
Text(0.5, 1.0, 'Scatter Plot for DV')
```



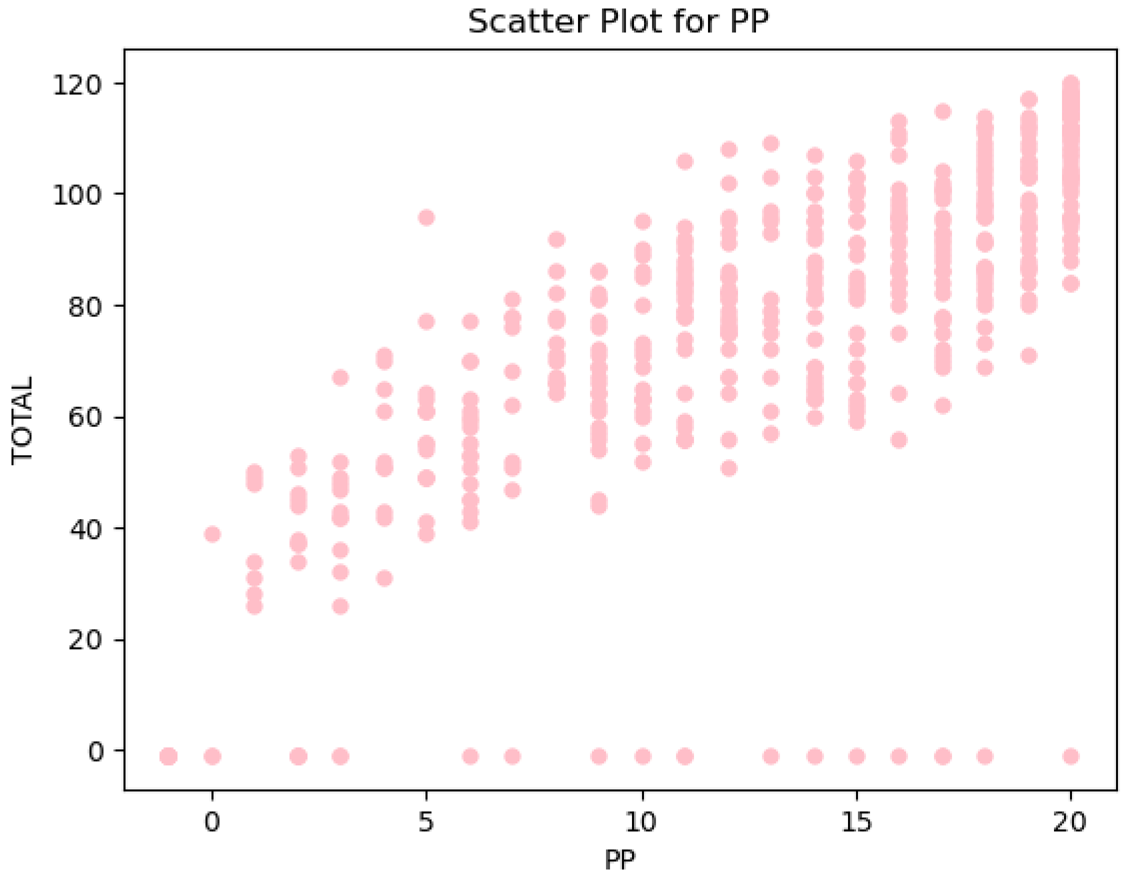
Creating scatter plot for 'DV' vs 'Total' values

In [32]:

```
p.plot.scatter(x = 'PP', y = 'TOTAL',color='pink',s=25)
plt.title("Scatter Plot for PP")
```

Out[32]:

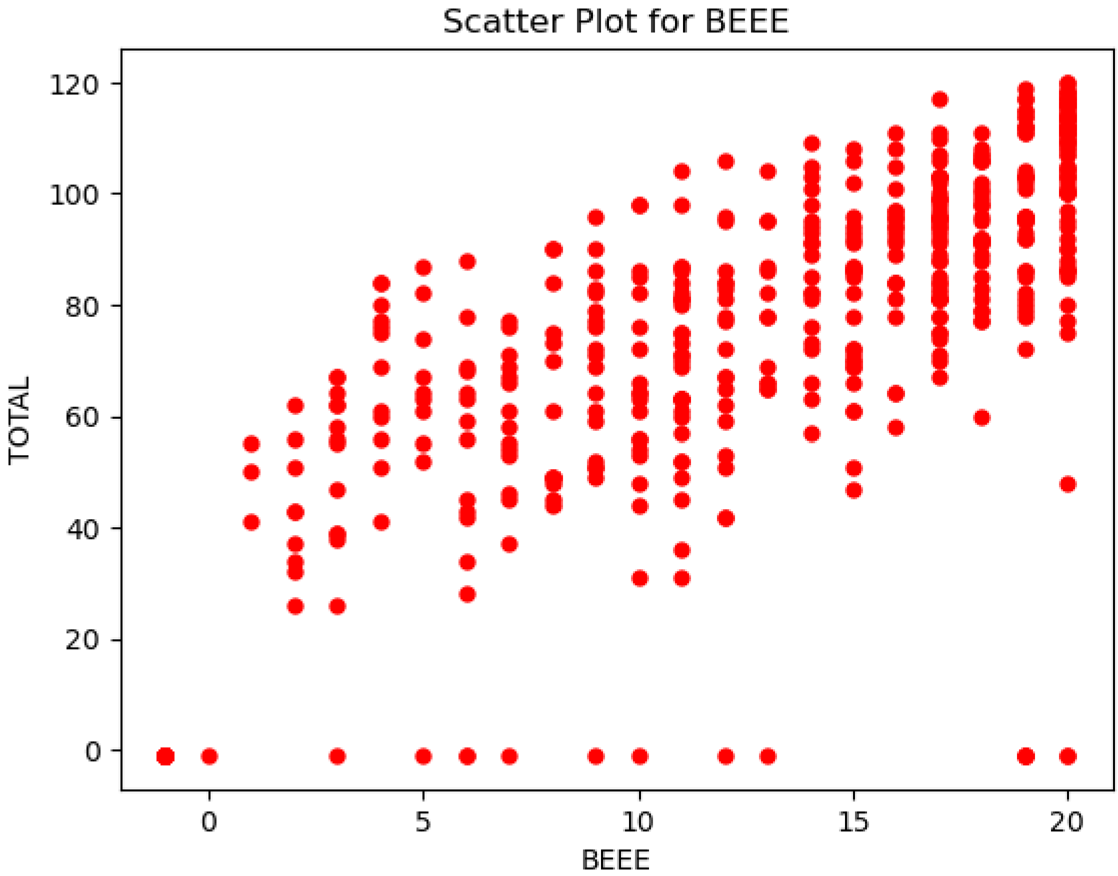
```
Text(0.5, 1.0, 'Scatter Plot for PP')
```



Creating scatter plot for 'PP' vs 'Total' values

```
In [33]: p.plot.scatter(x = 'BEEE', y = 'TOTAL',color='red',s=25)
         plt.title("Scatter Plot for BEEE")
```

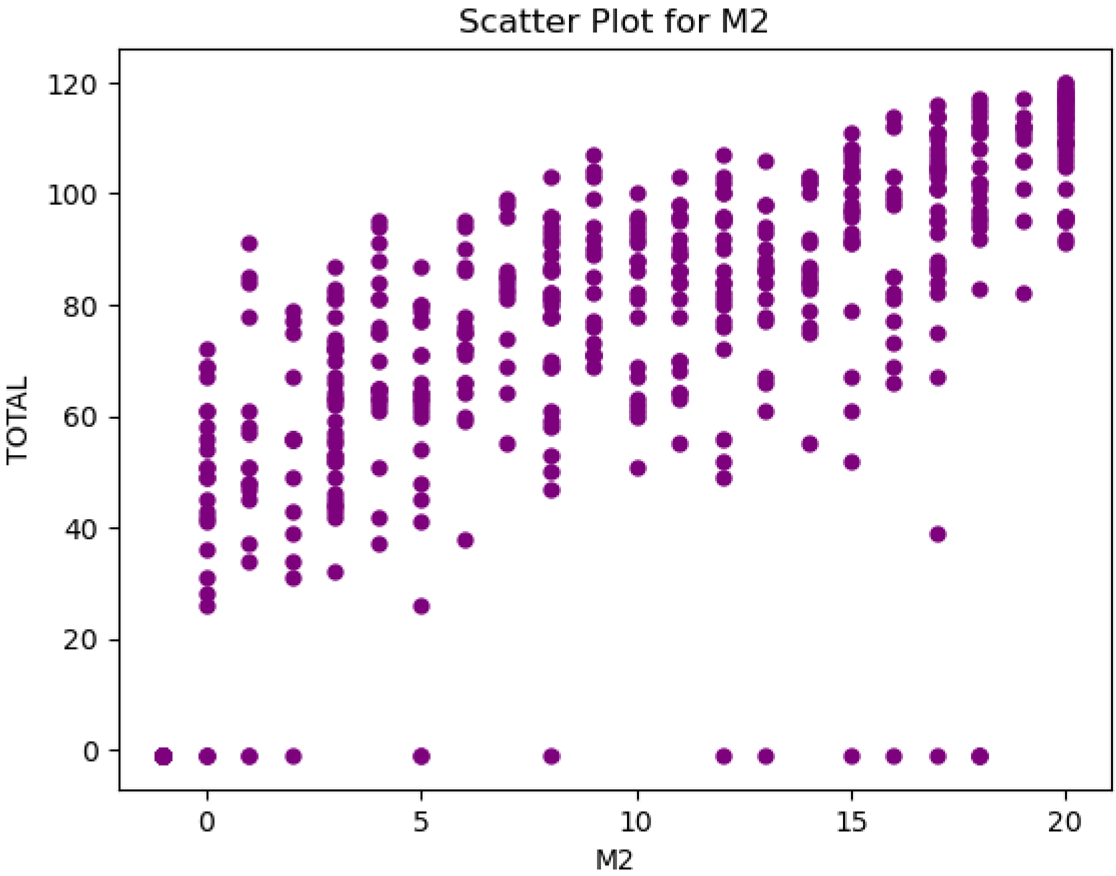
```
Out[33]: Text(0.5, 1.0, 'Scatter Plot for BEEE')
```



Creating scatter plot for 'BEEE' vs 'Total' values

```
In [34]: p.plot.scatter(x = 'M2', y = 'TOTAL',color='purple',s=25)
         plt.title("Scatter Plot for M2")
```

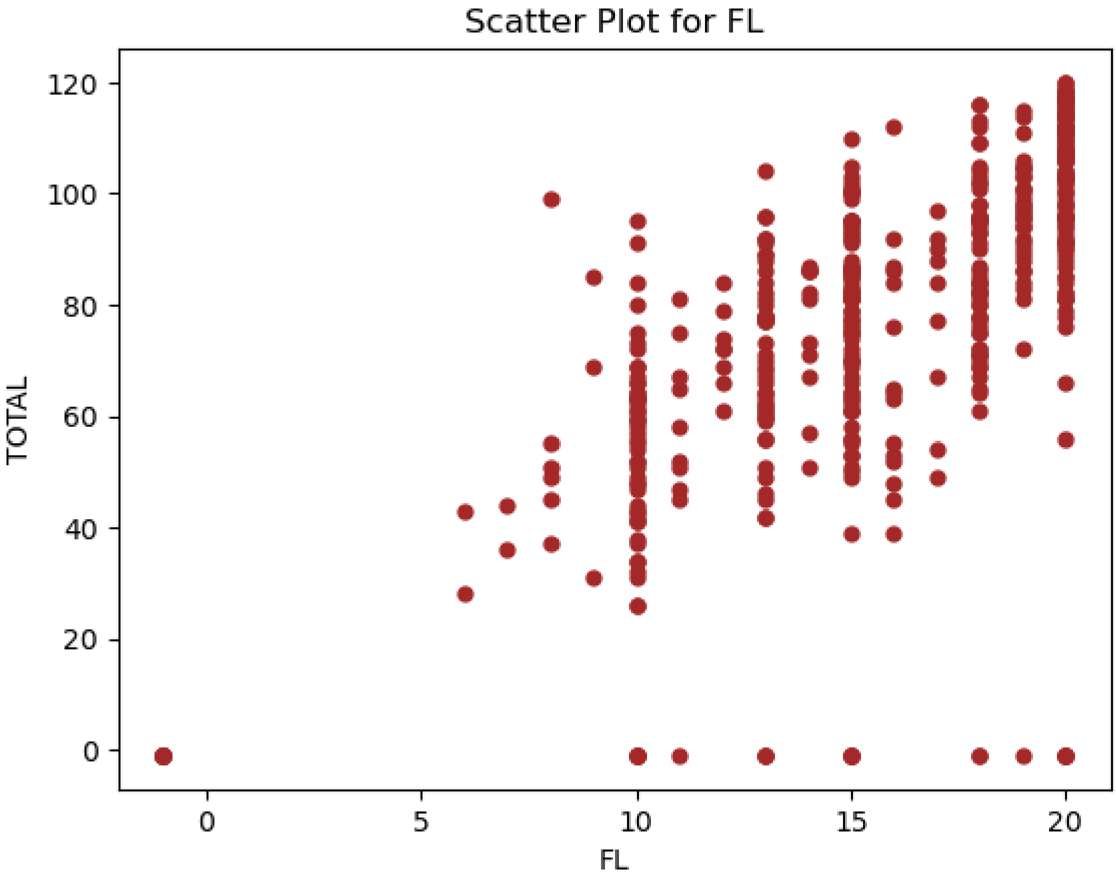
```
Out[34]: Text(0.5, 1.0, 'Scatter Plot for M2')
```



Creating scatter plot for 'M2' vs 'Total' values

In [35]:
```
p.plot.scatter(x = 'FL', y = 'TOTAL',color='brown',s=25)
plt.title("Scatter Plot for FL")
```

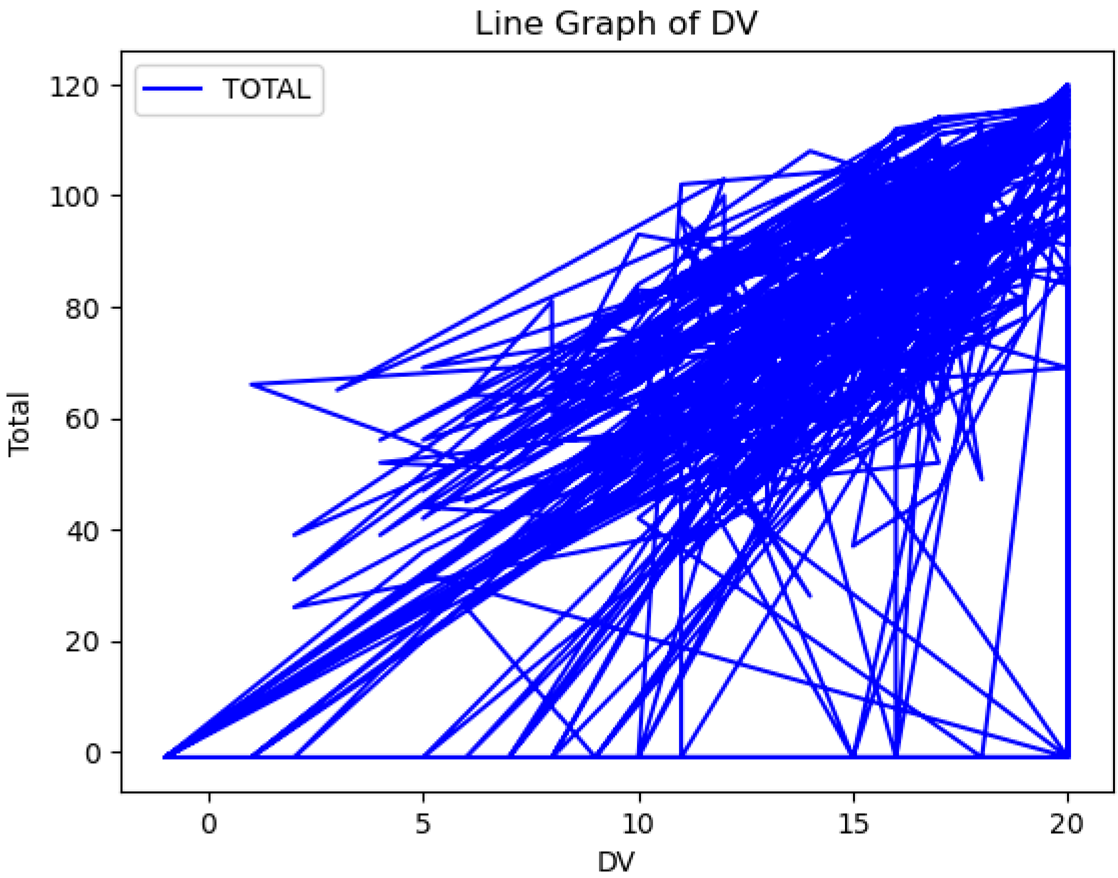
Out[35]:
```
Text(0.5, 1.0, 'Scatter Plot for FL')
```



Creating scatter plot for 'FL' vs 'Total' values

```
In [36]: p.plot.line(x='DV',y='TOTAL',color='blue')
         plt.title("Line Graph of DV")
         plt.ylabel("Total")
```

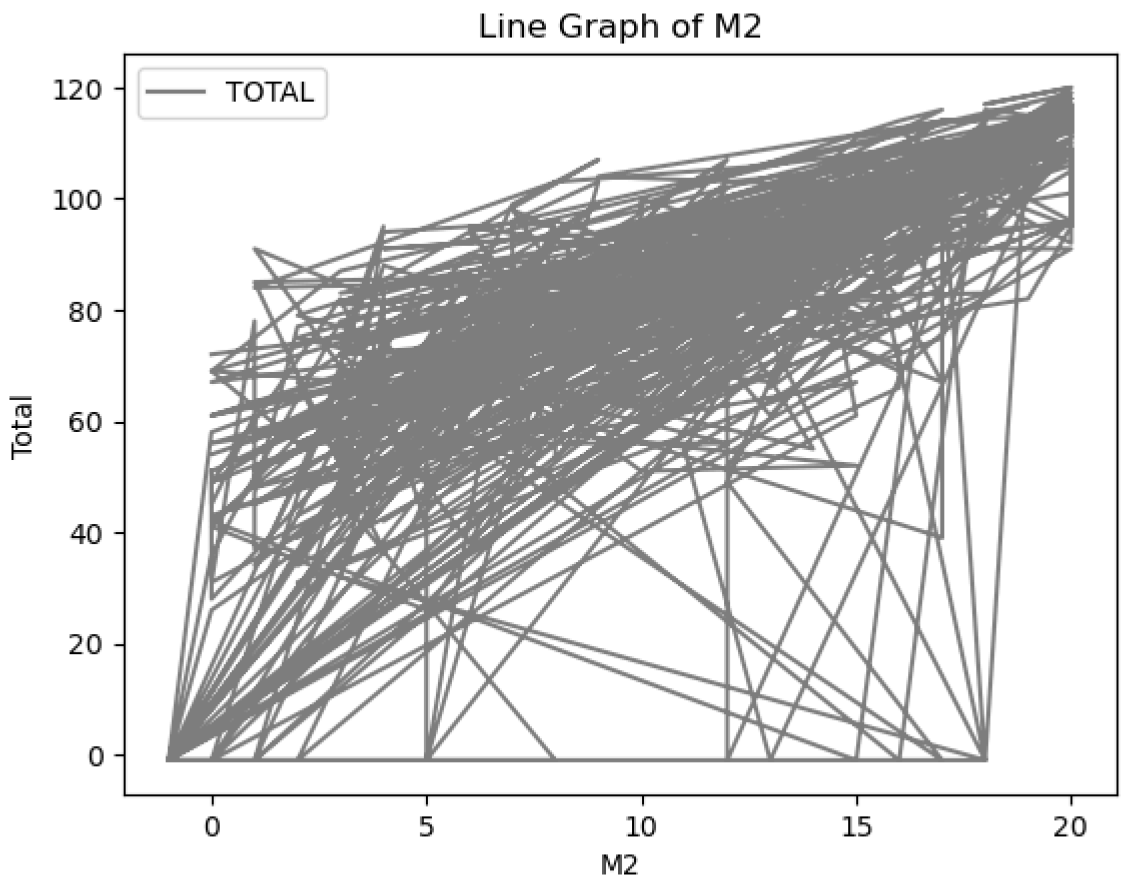
```
Out[36]: Text(0, 0.5, 'Total')
```



Plotting line graph of 'DV' vs 'Total' values

In [37]:
```
p.plot.line(x='M2',y='TOTAL',color='gray')
plt.title("Line Graph of M2")
plt.ylabel("Total")
```

Out[37]:
```
Text(0, 0.5, 'Total')
```



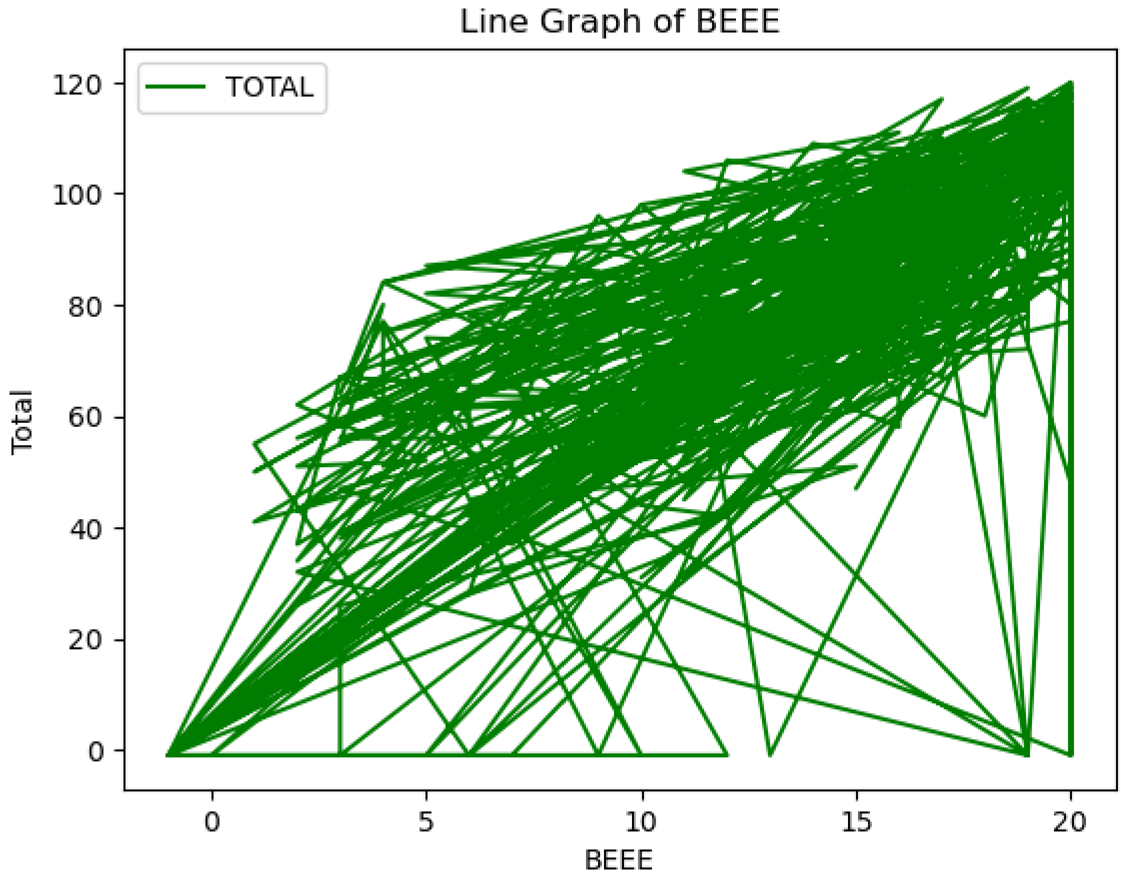
Plotting line graph of 'M2' vs 'Total' values

In [38]:

```
p.plot.line(x='BEEE',y='TOTAL',color='green')
plt.title("Line Graph of BEEE")
plt.ylabel("Total")
```

Out[38]:

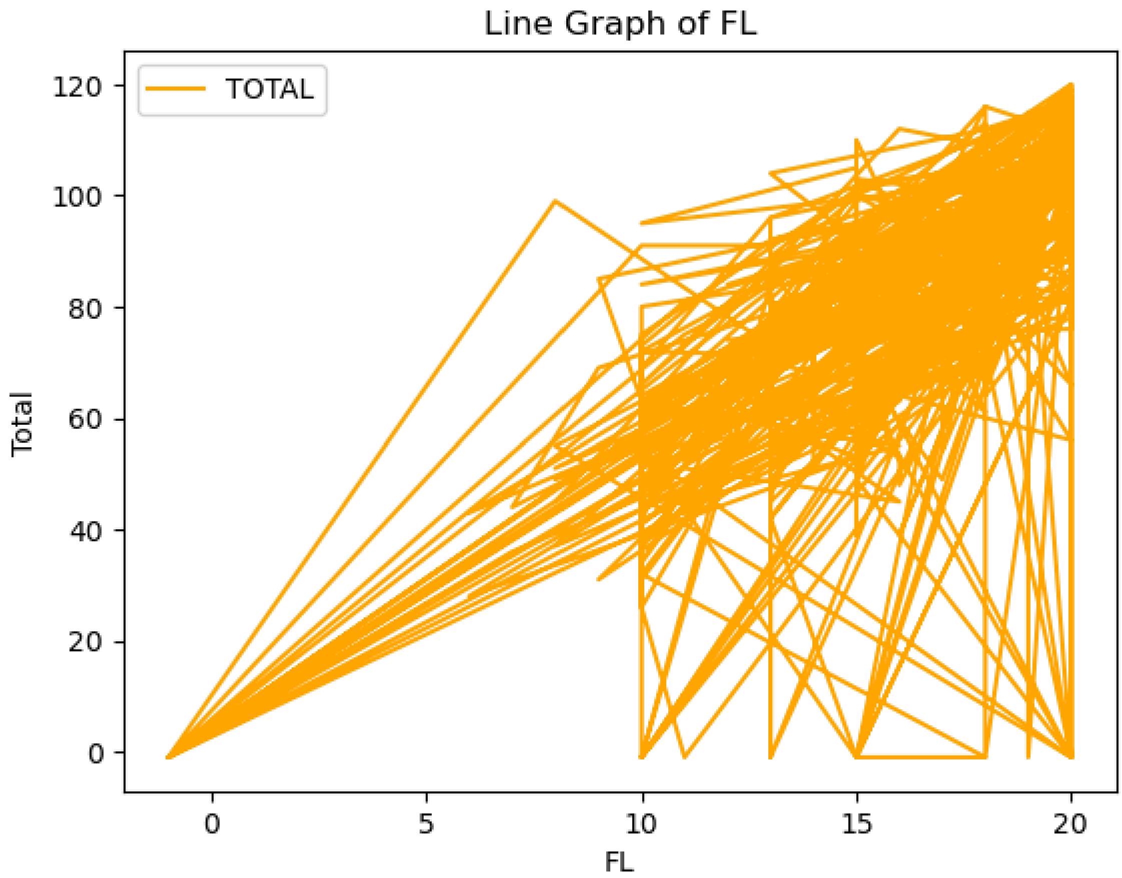
```
Text(0, 0.5, 'Total')
```



Plotting line graph of 'BEEE' vs 'Total' values

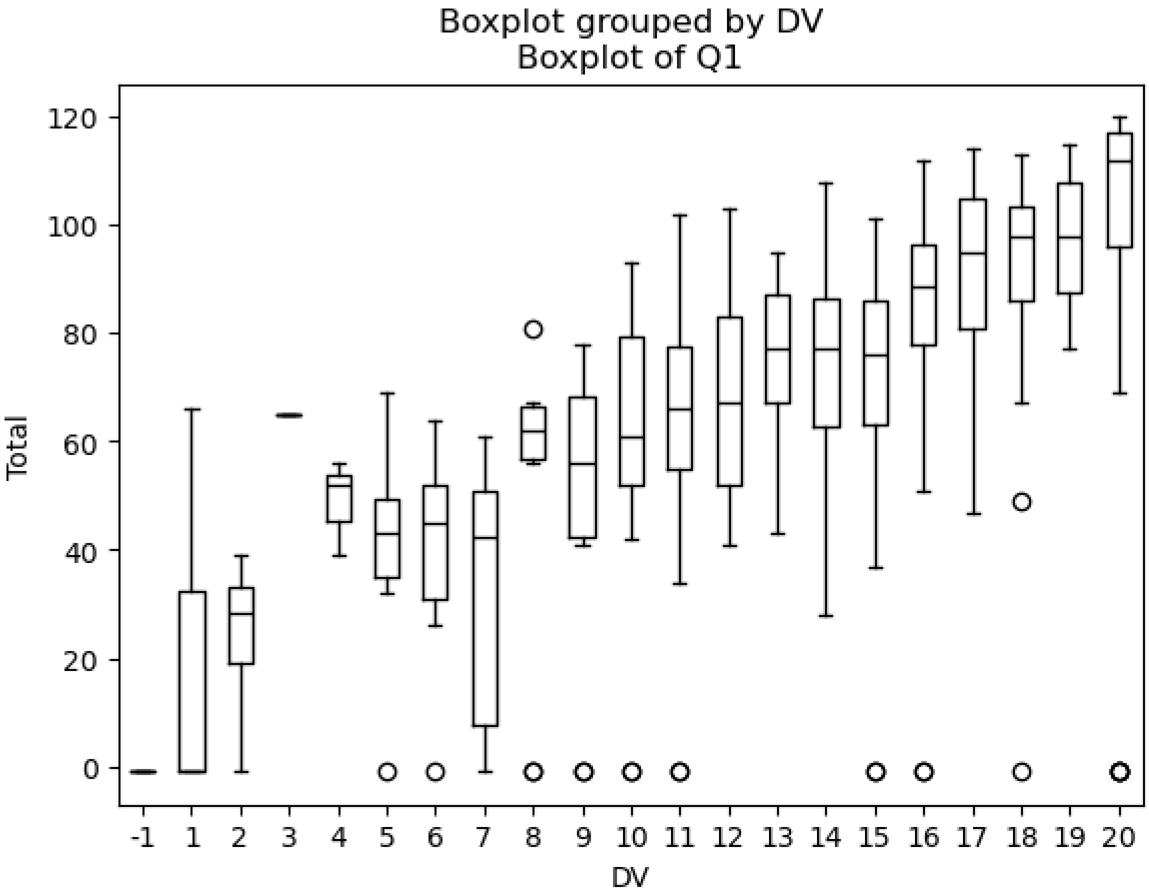
```
In [39]: p.plot.line(x='FL',y='TOTAL',color='orange')
         plt.title("Line Graph of FL")
         plt.ylabel("Total")
```

```
Out[39]: Text(0, 0.5, 'Total')
```



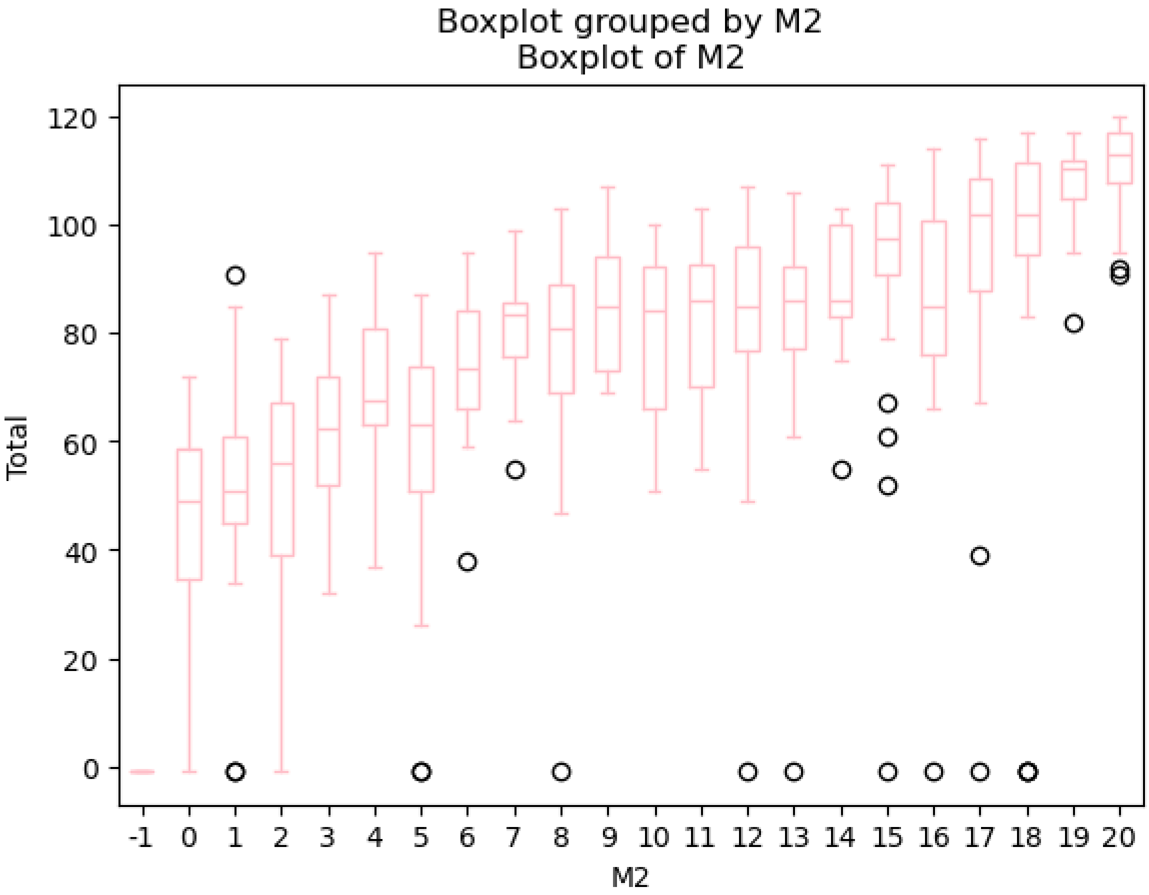
Plotting line graph of 'FL' vs 'Total' values

```
In [40]: p.boxplot(by='DV', column =['TOTAL'], grid = False,color='black')
         plt.title("Boxplot of Q1")
         plt.ylabel("Total")
         plt.show()
```



Creating boxplot to visualize 'Total' distribution by 'DV'

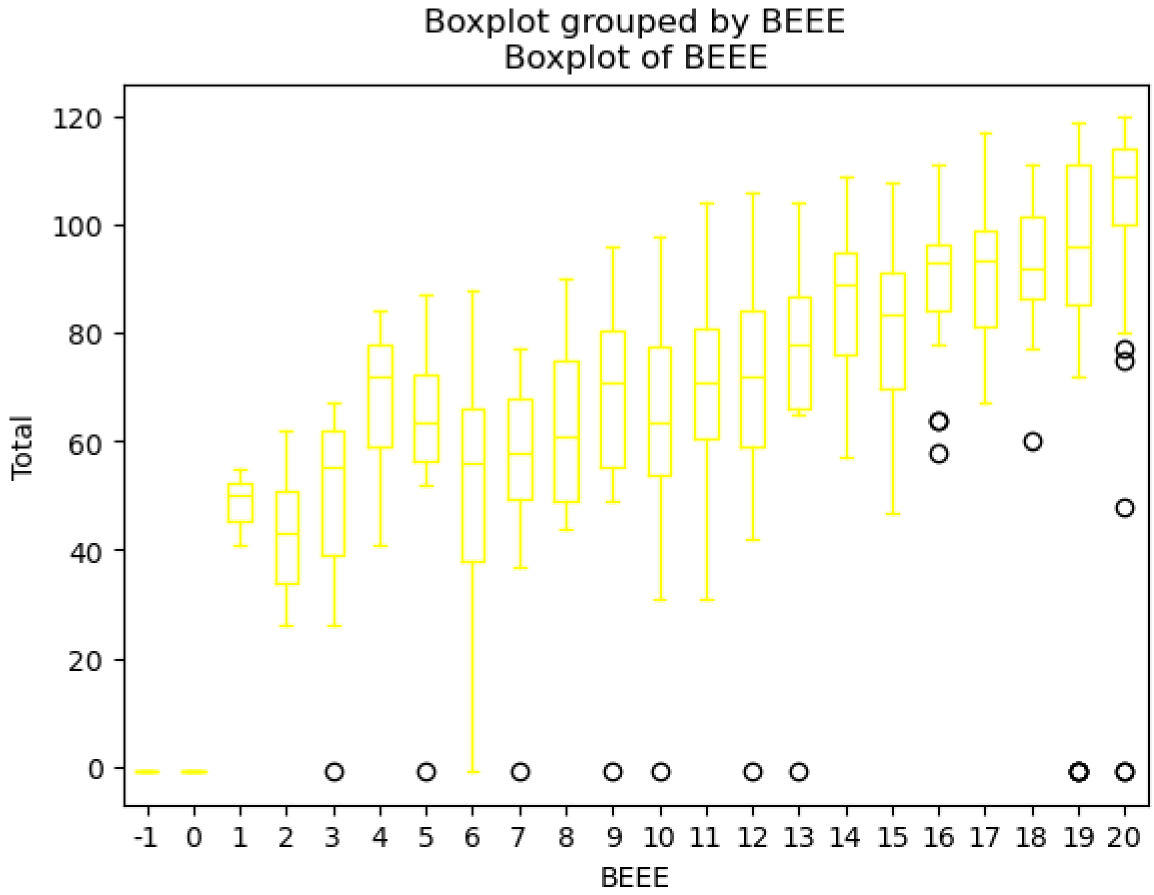
```
In [41]: p.boxplot(by='M2', column =['TOTAL'], grid = False,color='pink')
         plt.title("Boxplot of M2")
         plt.ylabel("Total")
         plt.show()
```



Creating boxplot to visualize 'Total' distribution by 'M2'

In [42]:

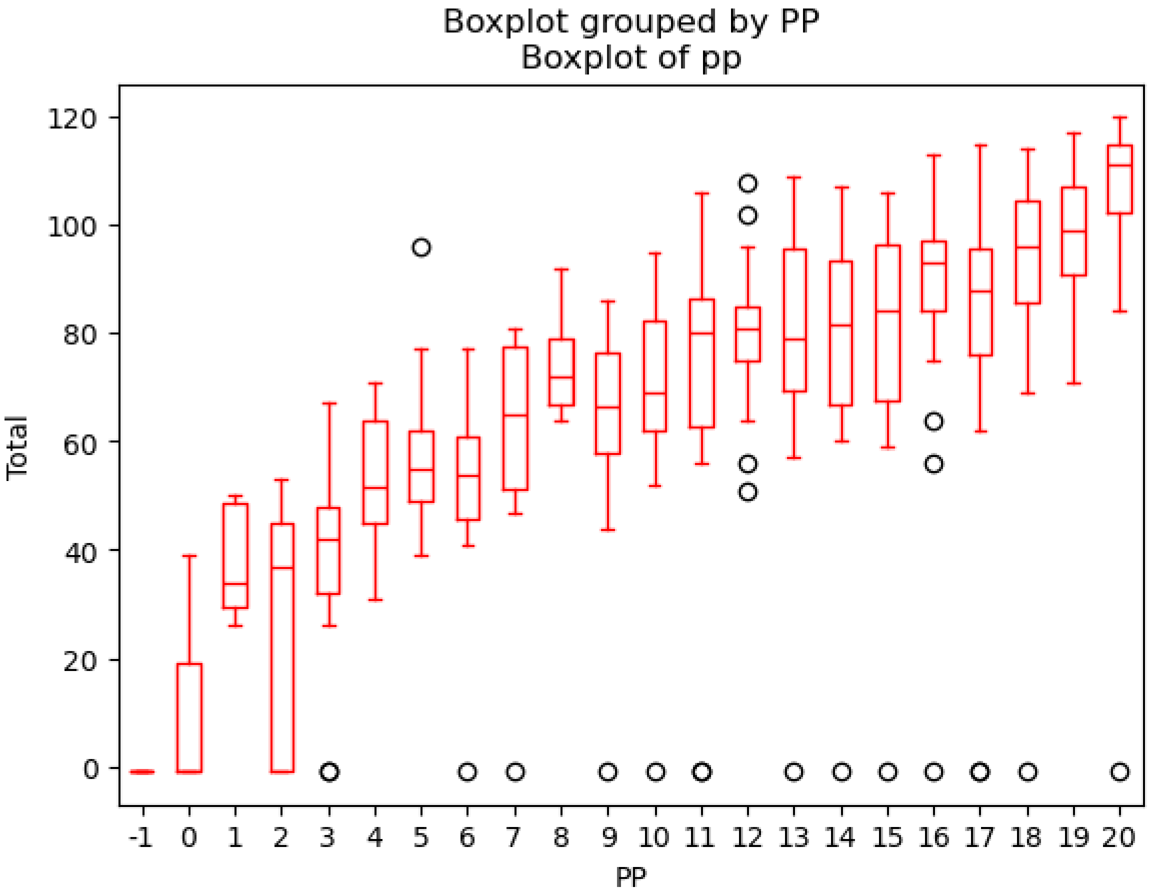
```
p.boxplot(by='BEEE', column =['TOTAL'], grid = False,color='yellow')
plt.title("Boxplot of BEEE")
plt.ylabel("Total")
plt.show()
```



Creating boxplot to visualize 'Total' distribution by 'BEEE'

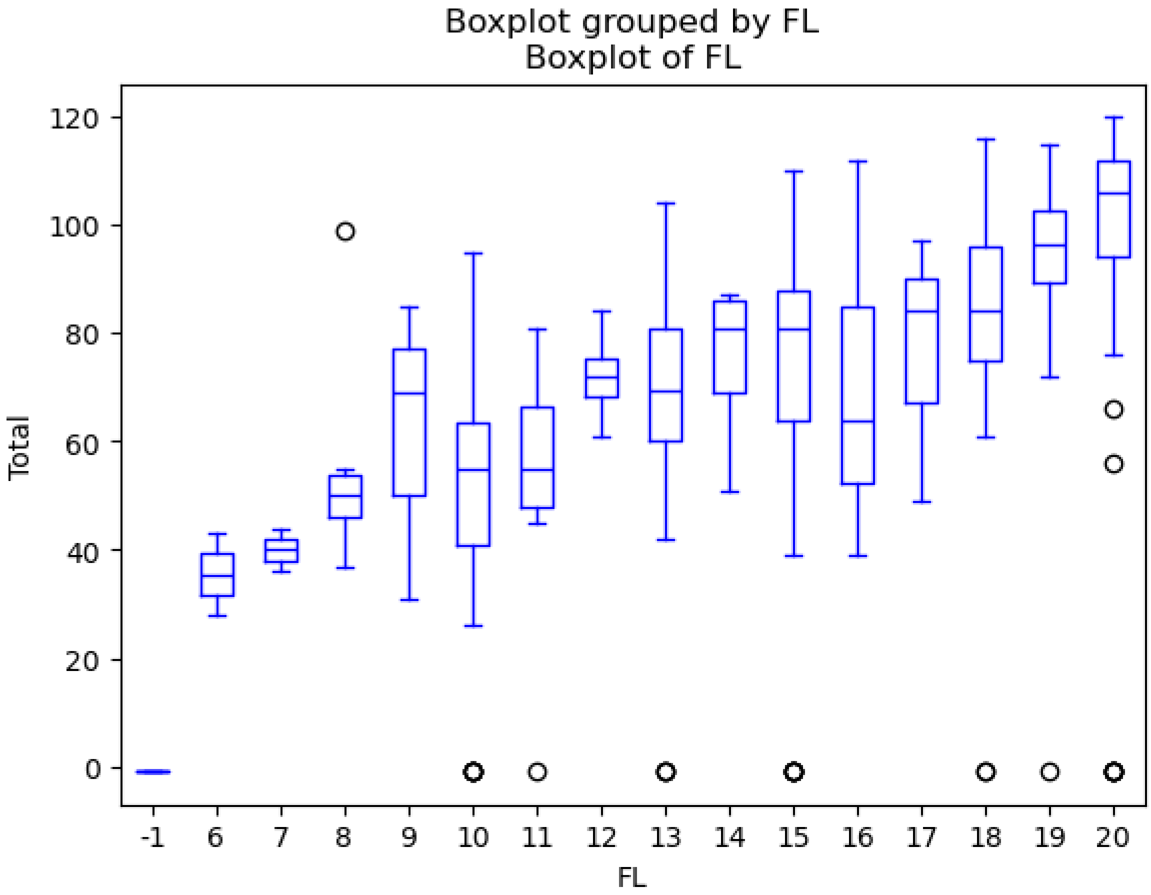
In [43]:

```
p.boxplot(by='PP', column =['TOTAL'], grid = False,color='red')
plt.title("Boxplot of pp")
plt.ylabel("Total")
plt.show()
```



Creating boxplot to visualize 'Total' distribution by 'PP'

In [44]:
```
p.boxplot(by='FL', column =['TOTAL'], grid = False,color='blue')
plt.title("Boxplot of FL")
plt.ylabel("Total")
plt.show()
```



Creating boxplot to visualize 'Total' distribution by 'Fl'

In [45]:

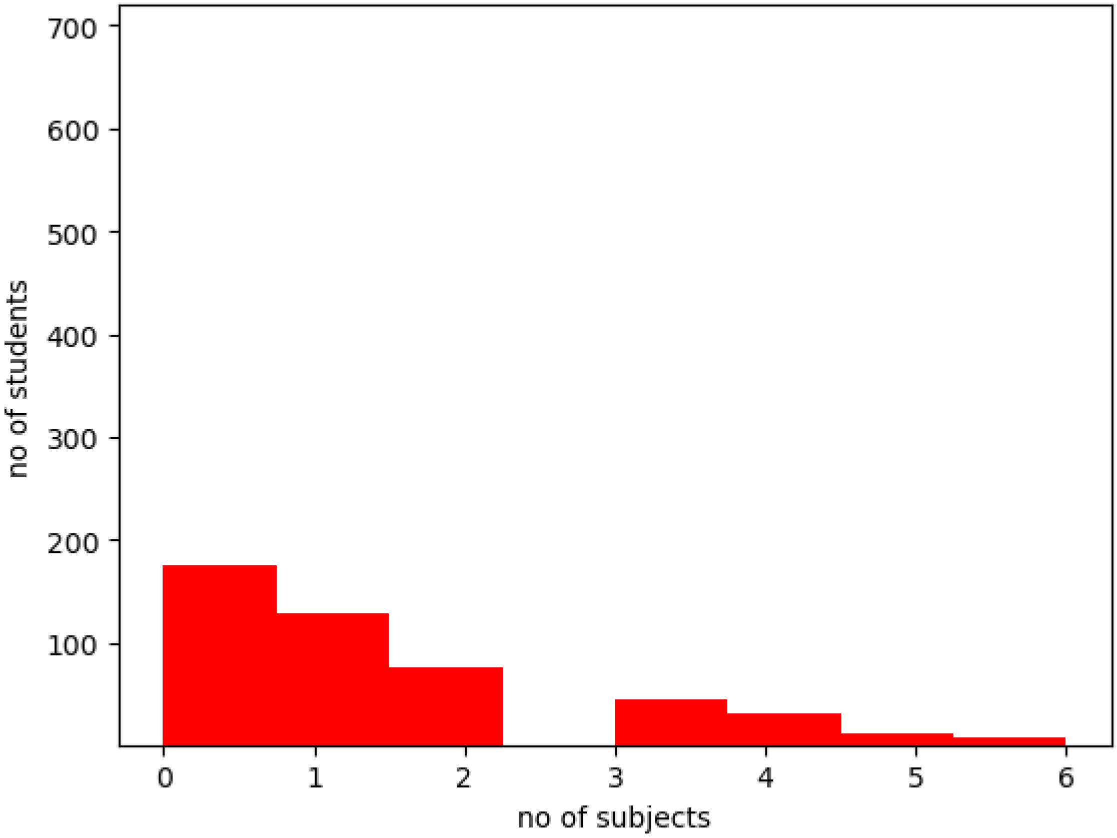
```
def failed(p):
     return ((p[['DV', 'M2', 'PP', 'BEEE', 'FL', 'FIMS']] < 10).sum(axis=1))

p['backlog'] = failed(p)


h = p.groupby('SECTION')['backlog'].sum()

print(h)
```

```
SECTION
-1          61
ALPHA       43
BETA        69
DELTA       73
EPSILON     85
GAMMA       93
OMEGA      103
SIGMA       97
ZETA        39
Name: backlog, dtype: Int64
```

In [ ]:

In [46]:

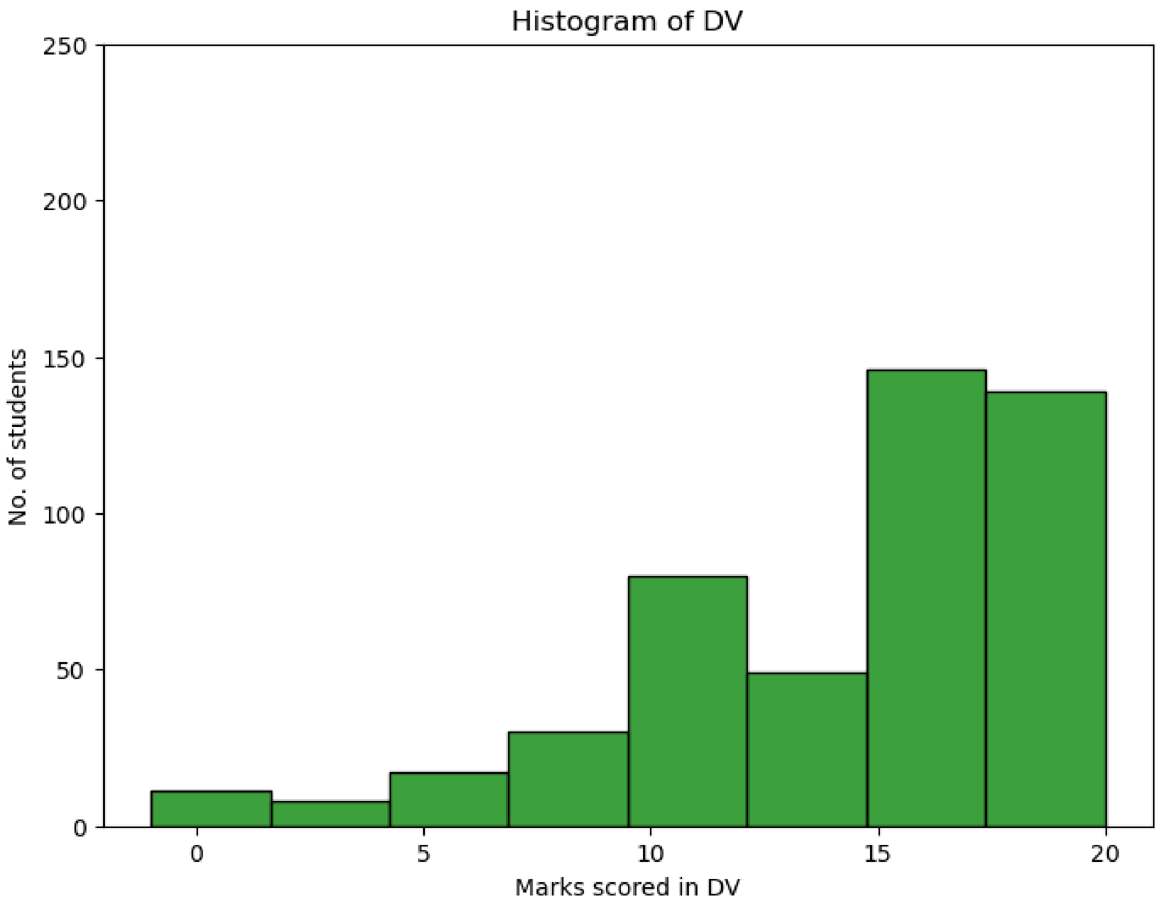
```
plt.hist(p['backlog'], color='red', bins=8)
plt.ylim(1, 720)
plt.ylabel("no of students")
plt.xlabel("no of subjects")
plt.title("")
plt.show()
```



using seaborn no of subjects who are failed

In [47]:

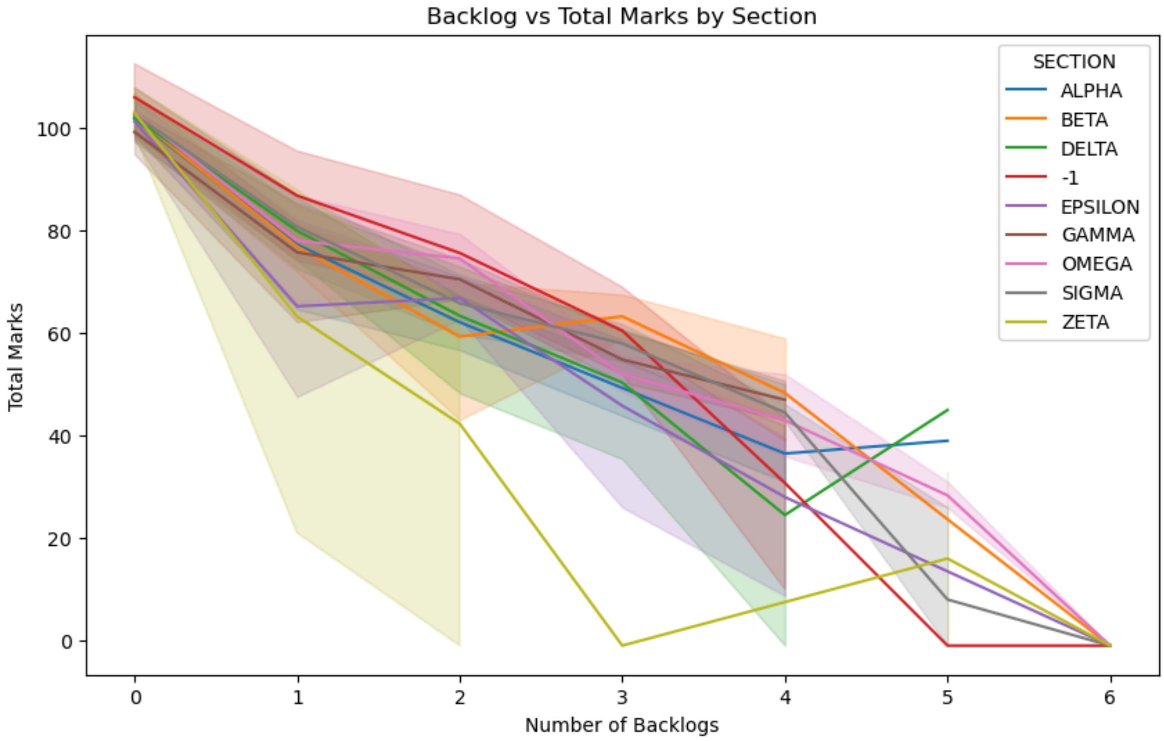
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt


plt.figure(figsize=(8, 6))
sns.histplot(p['DV'], color='green', bins=8, kde=False)

plt.ylim(0, 250)
plt.xlabel("Marks scored in DV")
plt.ylabel("No. of students")
plt.title("Histogram of DV")
plt.show()
```



Creating histogram to visualize 'Dv' sbject marks distribution

In [48]:

```
plt.figure(figsize=(10, 6))


sns.lineplot(x='backlog', y='TOTAL', hue='SECTION', data=p, markers=True)

plt.title('Backlog vs Total Marks by Section')
plt.xlabel('Number of Backlogs')
plt.ylabel('Total Marks')

plt.show()
```



by using seaborn we are showcasing Backlog vs Total Marks by Section

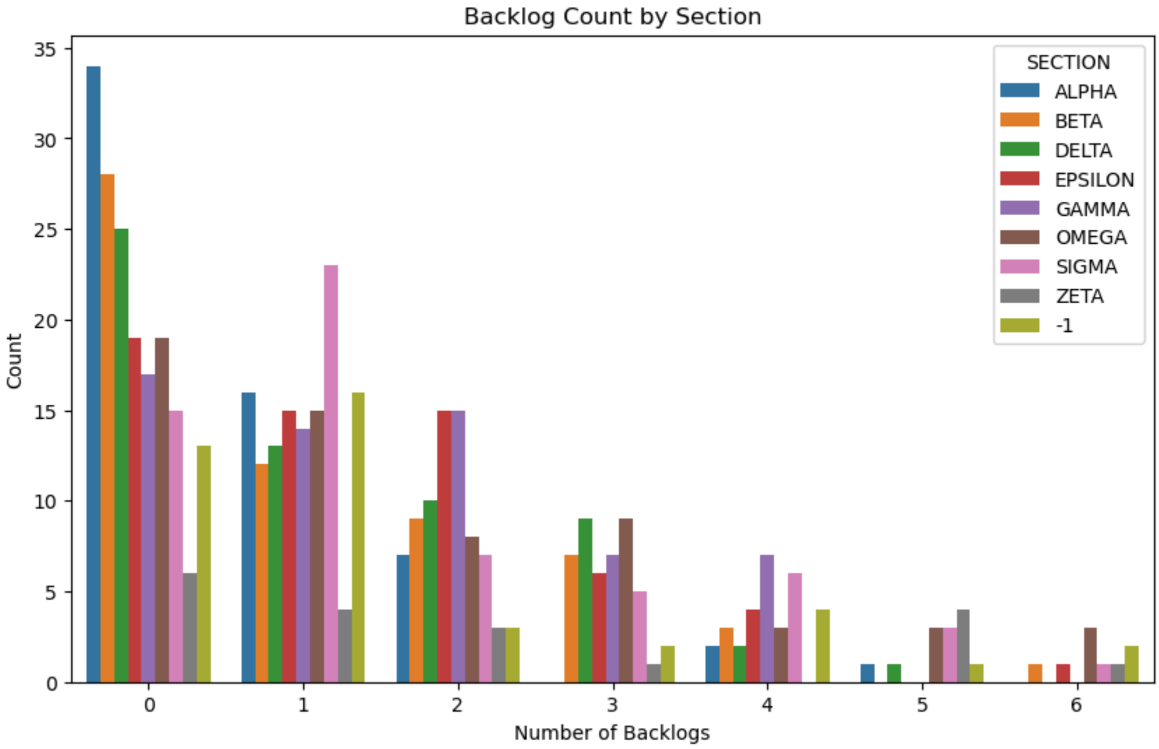
In [ ]:

```

```

by using seaborn we are showcasing Backlog vs Percentage by section

In [49]:

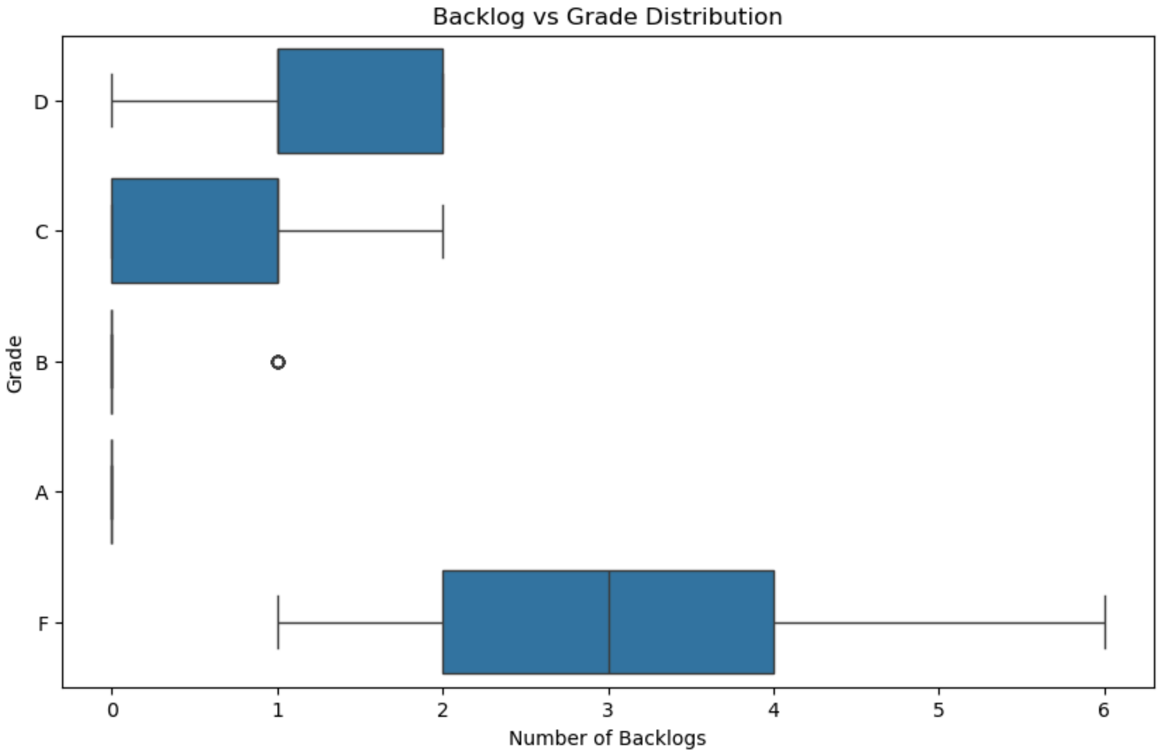
```
plt.figure(figsize=(10, 6))

sns.countplot(x='backlog', hue='SECTION', data=p)

plt.title('Backlog Count by Section')
plt.xlabel('Number of Backlogs')
plt.ylabel('Count')

plt.show()
```



by using seaborn we are showcasing Backlog count by Section

In [50]:

```
plt.figure(figsize=(10, 6))

sns.boxplot(x='backlog', y='Grade', data=p)

plt.title('Backlog vs Grade Distribution')
plt.xlabel('Number of Backlogs')
plt.ylabel('Grade')

plt.show()
```



by using seaborn we are showcasing backlog vs grade distribution

In [51]:

```
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]
```

In [52]:

```python
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]
```

In [53]:

```python
data.head(2)
```

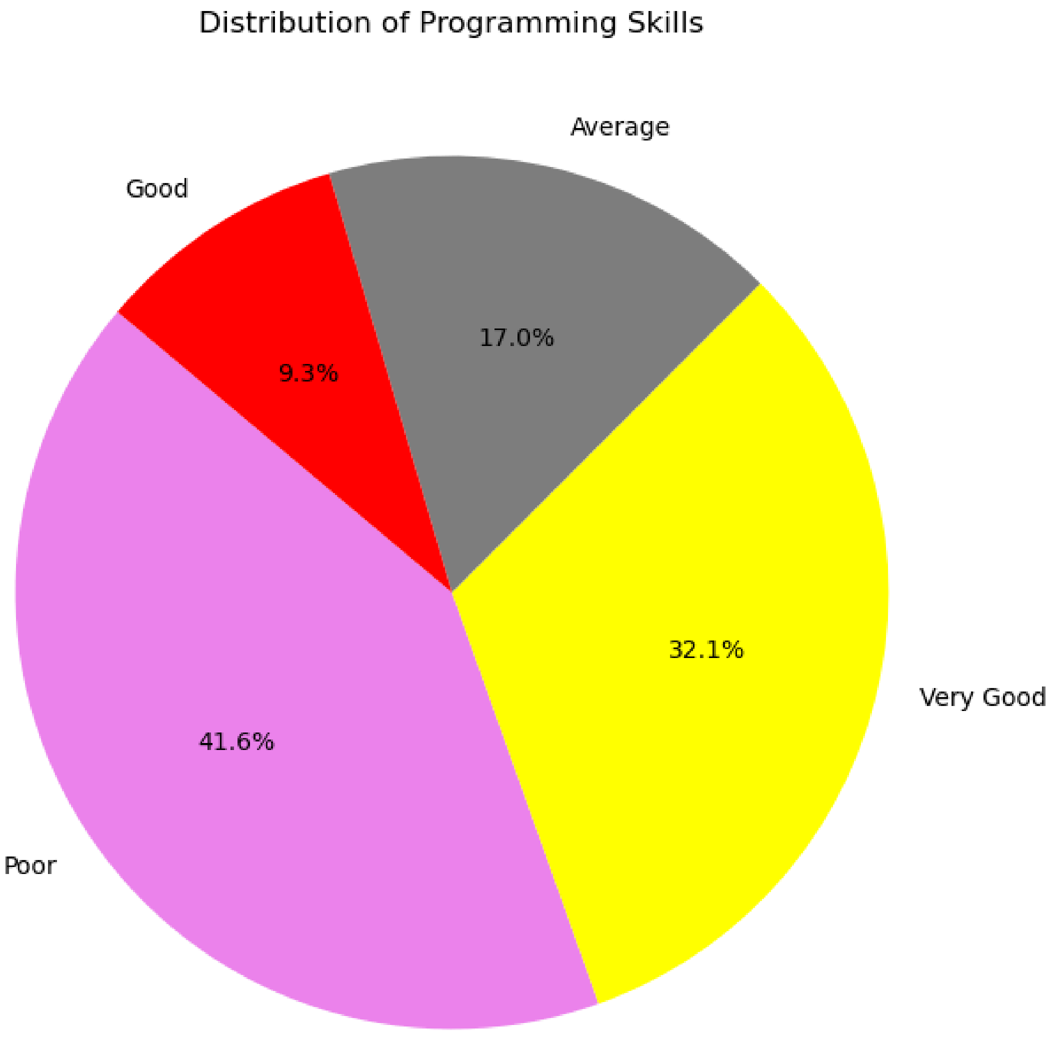
Out[53]:

	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

In [54]:

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

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

subjects = ["PP"]
df.dropna(inplace=True)  # Drop rows with missing values

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)

# Create a pie chart
skill_counts = df['Programming_skills'].value_counts()
plt.figure(figsize=(8, 8))
plt.pie(skill_counts, labels=skill_counts.index, autopct='%1.1f%%', startangle=1
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
```

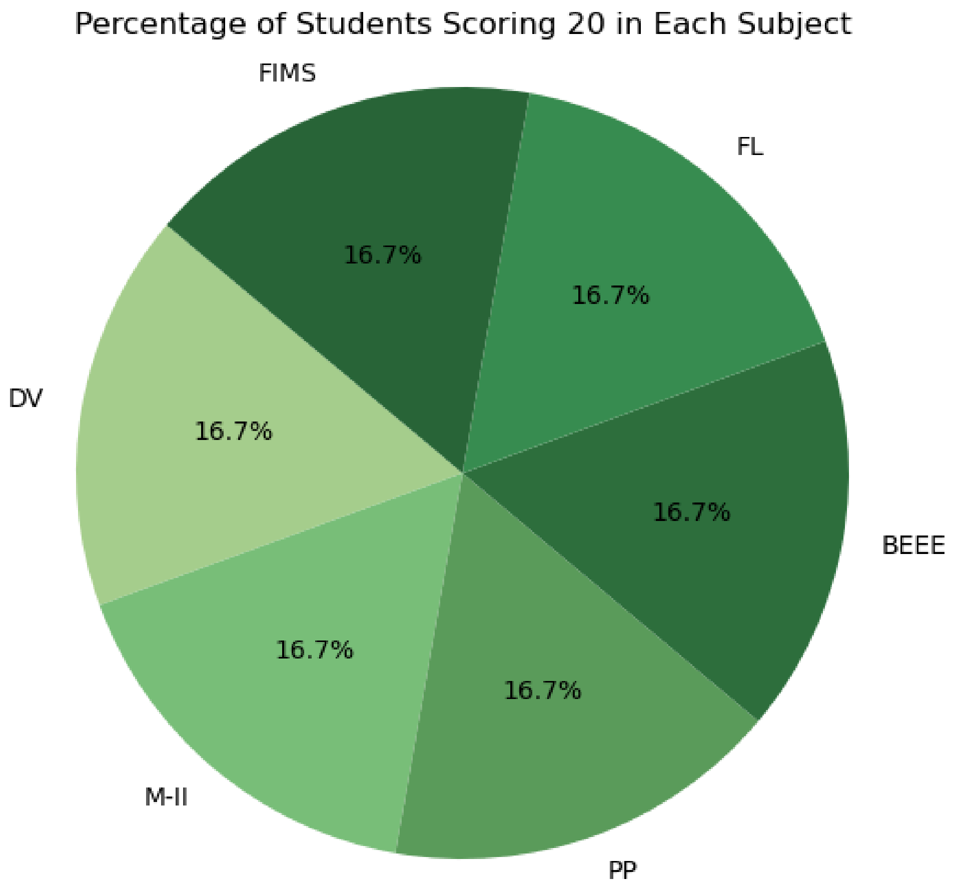


```
In [55]: subjects = ['DV', 'M-II', 'PP', 'BEEE', 'FL', 'FIMS']
         subset = df[df[subjects].eq(20).any(axis=1)]
         print("Subset of students who scored 20 in any subject:")
         print(subset)
         for subject in subjects:
             count_20 = (df[subject] == 20).sum()
             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
```

In [56]:

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

data = {
    'DV': [20, 15, 18, 20, 10],
    'M-II': [10, 20, 15, 14, 20],
    'PP': [20, 12, 20, 18, 16],
    'BEEE': [14, 20, 19, 12, 20],
    'FL': [18, 14, 20, 20, 15],
    'FIMS': [10, 20, 16, 14, 20]
}
df = pd.DataFrame(data)

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

counts = [df[subject].eq(20).sum() for subject in subjects]

green_shades = ['#a8d08d', '#7bbf7b', '#5a9b5a', '#2d703d', '#3b8c50', '#286838'

plt.figure(figsize=(8, 6))
plt.pie(counts, labels=subjects, autopct='%1.1f%%', startangle=140, colors=green

plt.title('Percentage of Students Scoring 20 in Each Subject')
plt.axis('equal')  # Equal aspect ratio ensures that pie is drawn as a circle.
plt.show()
```

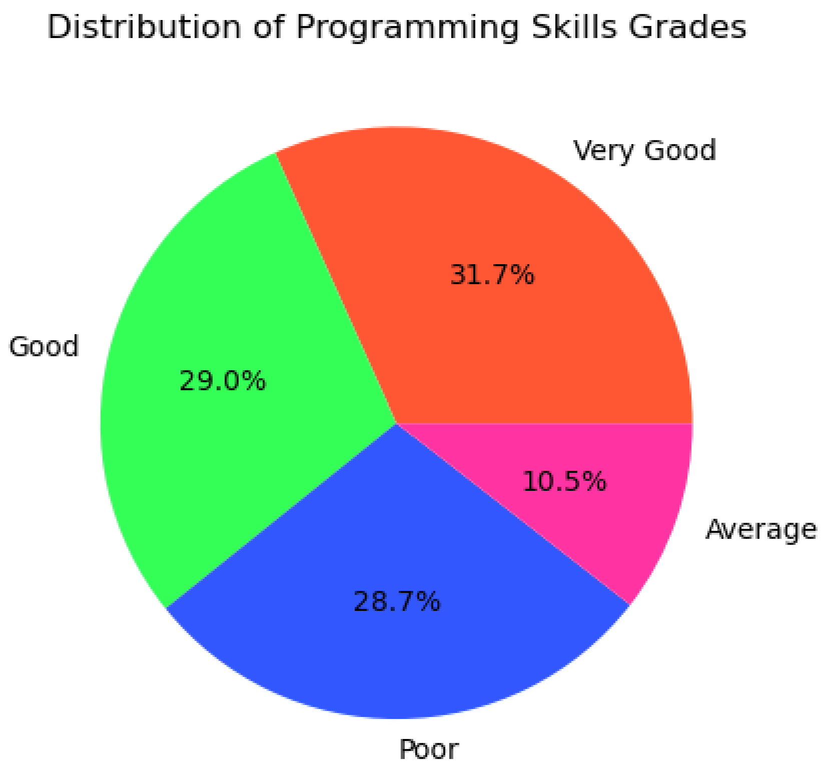


Percentage of Students Scoring 20 in Each Subject

In [57]:

```
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()
```

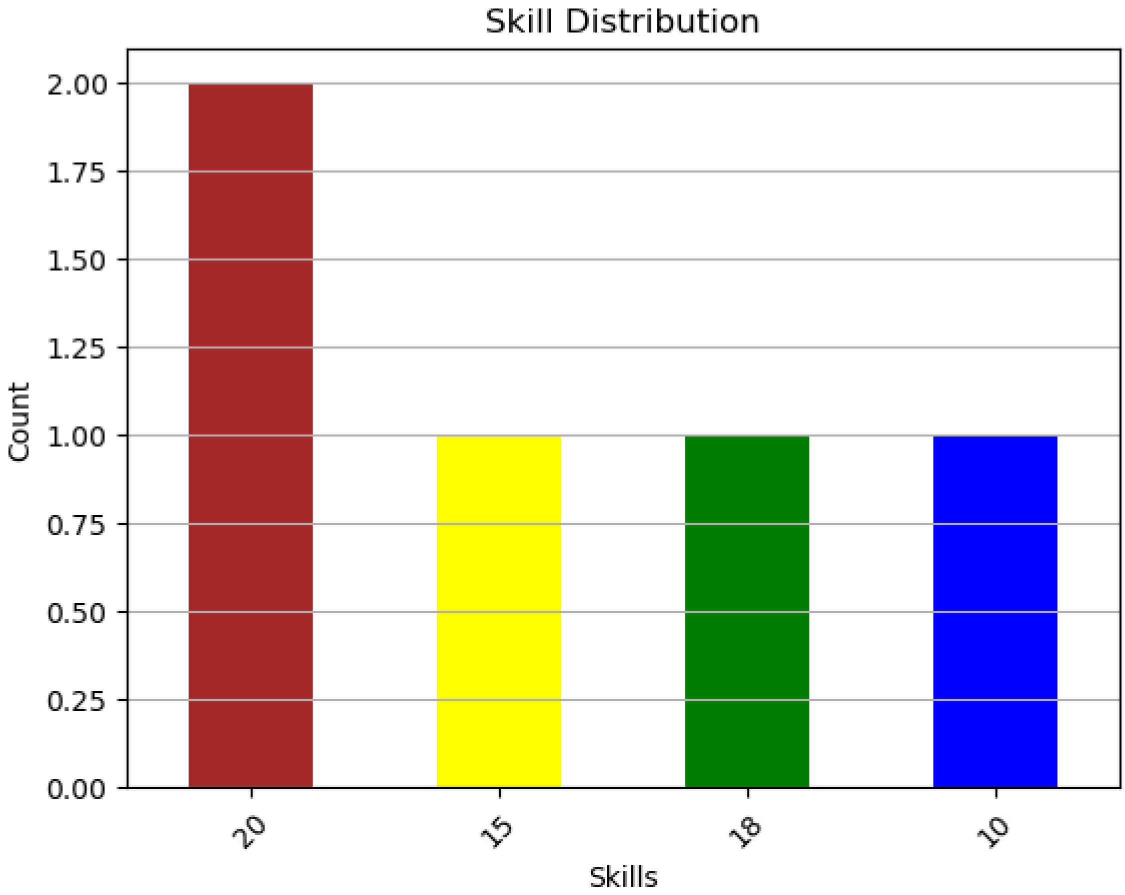
```
custom_colors = ['#FF5733', '#33FF57', '#3357FF', '#FF33A1']

plt.pie(grade_counts, labels=grade_counts.index, autopct='%1.1f%%', colors=custc
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]
```

In [58]:

```
import matplotlib.pyplot as plt
skill_counts = df['DV'].value_counts()
skill_counts.plot(kind='bar', color= ["brown" , "yellow" , "green", "blue"])
plt.title('Skill Distribution')
plt.xlabel('Skills')
plt.ylabel('Count')
plt.xticks(rotation=45)
plt.grid(axis='y')
plt.show()
print(skill_counts)
```



```
DV
20    2
15    1
18    1
10    1
Name: count, dtype: int64
```