DATA DESCRIPTION

The dataset contains 718 rows and 8 columns, though some columns have missing values.

It appears to be a record of midterm marks for students across different subjects.

Columns:

S.NO (float64) - Serial number of students (601 non-null values, some missing).

SECTION (object) - Section name (e.g., ALPHA) of the student (691 non-null values).

DV (object) - Marks in a subject named "DV" (716 non-null values).

M-II (object) - Marks in "M-II" subject (716 non-null values).

PP (object) - Marks in "PP" subject (716 non-null values).

BEEE (object) - Marks in "BEEE" subject (716 non-null values).

FL (object) - Marks in "FL" subject (715 non-null values).

FIMS (object) - Marks in "FIMS" subject (716 non-null values).

In [295…
```
import pandas as pd
```

In [297…
```
import matplotlib.pyplot as plt
```

In [299…
```
import seaborn as sns
```

In [301…
```
pip install openpyxl
```

```
Requirement already satisfied: openpyxl in c:\programdata\anaconda3\lib\site-packages (3.1.5)
Requirement already satisfied: et-xmlfile in c:\programdata\anaconda3\lib\site-packages (from openpyxl) (1.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [302…
```
df = pd.read_excel(r"C:\Users\rohit\OneDrive\Desktop\MIDMARKS-MINOR1-EXAM.xlsx")
```

In [ ]:
```
```

In [303…
```
df
```

Out[303…

	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 [307…
```
df['SECTION'] = df['SECTION'].fillna('ZETA')
df
```

Out[307…

	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 [311…
```
df['DV'] = df['DV'].replace(['MP'],0)
df
```

Out[311…

	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 [283…
```
df.replace(["AB","mp"],0, inplace=True)
```

In [293…
```
df
```

Out[293…

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

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

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

In [82]:
```
df.shape
```

Out[82]:
```
(480, 17)
```

In [84]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 480 entries, 0 to 479
Data columns (total 17 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   S.NO            480 non-null    int64  
 1   SECTION         480 non-null    object 
 2   DV              480 non-null    int32  
 3   M2              480 non-null    int32  
 4   PP              480 non-null    int32  
 5   BEEE            480 non-null    int32  
 6   FL              480 non-null    int32  
 7   FIMS            480 non-null    int32  
 8   Total           480 non-null    int32  
 9   Percentage      480 non-null    float64
 10  Grade           0 non-null      object 
 11  backlogs        480 non-null    int64  
 12  Coding-skills   480 non-null    object 
 13  PP_Grade        480 non-null    object 
 14  DV_Grade        480 non-null    object 
 15  skills          480 non-null    object 
 16  section         480 non-null    object 
dtypes: float64(1), int32(7), int64(2), object(7)
memory usage: 50.8+ KB
```

In [86]:
```
df.shape
df.size
```

Out[86]:
```
8160
```

In [88]:
```
df.size
```

Out[88]:
```
8160
```

In [90]:
```
df.rename(columns={'M-II': 'M2'}, inplace=True)
```

In [92]:
```
df
```

Out[92]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs	Coding-skills	PP_Grade	DV_Grade	skills
0	1	ALPHA	12	0	17	9	19	15	72	60.00	NaN	2	good	Good	Good	Good
1	2	ALPHA	19	12	16	16	18	3	84	70.00	NaN	1	good	Good	Good	Good
2	3	ALPHA	18	14	18	18	18	16	102	85.00	NaN	0	very good	Very Good	Very Good	Very Good
3	4	ALPHA	15	9	19	17	19	15	94	78.33	NaN	1	very good	Very Good	Very Good	Very Good
4	5	ALPHA	18	17	19	19	20	18	111	92.50	NaN	0	very good	Very Good	Very Good	Very Good
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
475	476	SIGMA	18	2	12	3	17	15	67	55.83	NaN	2	poor	poor	Poor	Poor
476	477	SIGMA	20	6	16	11	20	14	87	72.50	NaN	1	good	Good	Good	Good
477	478	SIGMA	20	0	18	13	20	18	89	74.17	NaN	1	very good	Very Good	Very Good	Very Good
478	479	SIGMA	20	20	5	19	18	14	96	80.00	NaN	1	poor	poor	Poor	Poor
479	480	SIGMA	20	16	18	19	20	19	112	93.33	NaN	0	very good	Very Good	Very Good	Very Good

480 rows × 17 columns

In [269…

```
df
```

Out[269…

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs	Coding-skills	PP_Grade	DV_Grade	skills
0	1	ALPHA	12	0	17	9	19	15	72	60.00	NaN	2	good	Good	Good	Good
1	2	ALPHA	19	12	16	16	18	3	84	70.00	NaN	1	good	Good	Good	Good
2	3	ALPHA	18	14	18	18	18	16	102	85.00	NaN	0	very good	Very Good	Very Good	Very Good
3	4	ALPHA	15	9	19	17	19	15	94	78.33	NaN	1	very good	Very Good	Very Good	Very Good
4	5	ALPHA	18	17	19	19	20	18	111	92.50	NaN	0	very good	Very Good	Very Good	Very Good
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
475	476	SIGMA	18	2	12	3	17	15	67	55.83	NaN	2	poor	poor	Poor	Poor
476	477	SIGMA	20	6	16	11	20	14	87	72.50	NaN	1	good	Good	Good	Good
477	478	SIGMA	20	0	18	13	20	18	89	74.17	NaN	1	very good	Very Good	Very Good	Very Good
478	479	SIGMA	20	20	5	19	18	14	96	80.00	NaN	1	poor	poor	Poor	Poor
479	480	SIGMA	20	16	18	19	20	19	112	93.33	NaN	0	very good	Very Good	Very Good	Very Good

480 rows × 17 columns

In [95]:

```
df['S.NO'] = range(1, len(df) + 1)
df['SECTION'] = df['SECTION'].fillna('SIGMA')
df['SECTION'] = df['SECTION'].replace('', 'SIGMA')
print("Updated DataFrame with missing sections replaced by 'SIGMA':")
df
```

```
Updated DataFrame with missing sections replaced by 'SIGMA':
```

Out[95]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs	Coding-skills	PP_Grade	DV_Grade	skills
0	1	ALPHA	12	0	17	9	19	15	72	60.00	NaN	2	good	Good	Good	Good
1	2	ALPHA	19	12	16	16	18	3	84	70.00	NaN	1	good	Good	Good	Good
2	3	ALPHA	18	14	18	18	18	16	102	85.00	NaN	0	very good	Very Good	Very Good	Very Good
3	4	ALPHA	15	9	19	17	19	15	94	78.33	NaN	1	very good	Very Good	Very Good	Very Good
4	5	ALPHA	18	17	19	19	20	18	111	92.50	NaN	0	very good	Very Good	Very Good	Very Good
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
475	476	SIGMA	18	2	12	3	17	15	67	55.83	NaN	2	poor	poor	Poor	Poor
476	477	SIGMA	20	6	16	11	20	14	87	72.50	NaN	1	good	Good	Good	Good
477	478	SIGMA	20	0	18	13	20	18	89	74.17	NaN	1	very good	Very Good	Very Good	Very Good
478	479	SIGMA	20	20	5	19	18	14	96	80.00	NaN	1	poor	poor	Poor	Poor
479	480	SIGMA	20	16	18	19	20	19	112	93.33	NaN	0	very good	Very Good	Very Good	Very Good

480 rows × 17 columns

In [97]:
```
df['DV'] = pd.to_numeric(df['DV'], errors='coerce').fillna(0).astype(int)
df['M2'] = pd.to_numeric(df['M2'], errors='coerce').fillna(0).astype(int)
df['PP'] = pd.to_numeric(df['PP'], errors='coerce').fillna(0).astype(int)
df['BEEE'] = pd.to_numeric(df['BEEE'], errors='coerce').fillna(0).astype(int)
df['FL'] = pd.to_numeric(df['FL'], errors='coerce').fillna(0).astype(int)
df['FIMS'] = pd.to_numeric(df['FIMS'], errors='coerce').fillna(0).astype(int)
df
```

Out[97]:

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs	Coding-skills	PP_Grade	DV_Grade	skills
0	1	ALPHA	12	0	17	9	19	15	72	60.00	NaN	2	good	Good	Good	Good
1	2	ALPHA	19	12	16	16	18	3	84	70.00	NaN	1	good	Good	Good	Good
2	3	ALPHA	18	14	18	18	18	16	102	85.00	NaN	0	very good	Very Good	Very Good	Very Good
3	4	ALPHA	15	9	19	17	19	15	94	78.33	NaN	1	very good	Very Good	Very Good	Very Good
4	5	ALPHA	18	17	19	19	20	18	111	92.50	NaN	0	very good	Very Good	Very Good	Very Good
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
475	476	SIGMA	18	2	12	3	17	15	67	55.83	NaN	2	poor	poor	Poor	Poor
476	477	SIGMA	20	6	16	11	20	14	87	72.50	NaN	1	good	Good	Good	Good
477	478	SIGMA	20	0	18	13	20	18	89	74.17	NaN	1	very good	Very Good	Very Good	Very Good
478	479	SIGMA	20	20	5	19	18	14	96	80.00	NaN	1	poor	poor	Poor	Poor
479	480	SIGMA	20	16	18	19	20	19	112	93.33	NaN	0	very good	Very Good	Very Good	Very Good

480 rows × 17 columns

In [99]:
```
df['Total'] = df['DV'] + df['M2'] + df['PP'] + df['BEEE'] + df['FL'] + df['FIMS']
df['Percentage'] = round((df['Total'] / 120) * 100, 2)
```

In [101…
```
df
```

Out[101…

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs	Coding-skills	PP_Grade	DV_Grade	skills
0	1	ALPHA	12	0	17	9	19	15	72	60.00	NaN	2	good	Good	Good	Good
1	2	ALPHA	19	12	16	16	18	3	84	70.00	NaN	1	good	Good	Good	Good
2	3	ALPHA	18	14	18	18	18	16	102	85.00	NaN	0	very good	Very Good	Very Good	Very Good
3	4	ALPHA	15	9	19	17	19	15	94	78.33	NaN	1	very good	Very Good	Very Good	Very Good
4	5	ALPHA	18	17	19	19	20	18	111	92.50	NaN	0	very good	Very Good	Very Good	Very Good
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
475	476	SIGMA	18	2	12	3	17	15	67	55.83	NaN	2	poor	poor	Poor	Poor
476	477	SIGMA	20	6	16	11	20	14	87	72.50	NaN	1	good	Good	Good	Good
477	478	SIGMA	20	0	18	13	20	18	89	74.17	NaN	1	very good	Very Good	Very Good	Very Good
478	479	SIGMA	20	20	5	19	18	14	96	80.00	NaN	1	poor	poor	Poor	Poor
479	480	SIGMA	20	16	18	19	20	19	112	93.33	NaN	0	very good	Very Good	Very Good	Very Good

480 rows × 17 columns

In [103…

```
df.loc[df['Total'] > 110, ['Grade']] = "A+"
df.loc[(df['Total'] > 90) & (df['Total'] <= 110), ['Grade']] = "A"
df.loc[(df['Total'] > 70) & (df['Total'] <= 90), ['Grade']] = "B+"
df.loc[(df['Total'] > 50) & (df['Total'] <= 70), ['Grade']] = "B"
df.loc[df['Total'] <= 50, ['Grade']] = "Fail"
failed_students_Total = df[df['Total'] < 50]
print("DataFrame of students who failed in subjects:")
print(failed_students_Total)
```

```
DataFrame of students who failed in subjects:
     S.NO  SECTION  DV  M2  PP  BEEE  FL  FIMS  Total  Percentage Grade  \
20     21    ALPHA   4   2   5     3  16     9     39       32.50  Fail
27     28    ALPHA   5   4   3    12  13     5     42       35.00  Fail
57     58    ALPHA   2   2   4    10  10     3     31       25.83  Fail
75     76     BETA   5   8   7    15  10     2     47       39.17  Fail
82     83     BETA   2   0   2     0   0     0      4        3.33  Fail
88     89     BETA   2  17   0     3  15     2     39       32.50  Fail
147   148    DELTA   9   5   6     8  13     4     45       37.50  Fail
160   161    DELTA   7   0  14     5  10     5     41       34.17  Fail
178   179    DELTA  13   0   3     2  10    15     43       35.83  Fail
180   181    SIGMA   9   3   4     2  10    15     43       35.83  Fail
197   198  EPSILON  11   1   1     2  10     9     34       28.33  Fail
206   207  EPSILON   6   6   2     3  10    11     38       31.67  Fail
210   211  EPSILON   0   0   0     0   0     0      0        0.00  Fail
226   227  EPSILON   8   0   3     0  10    14     35       29.17  Fail
229   230  EPSILON  13   2   5     9  10    10     49       40.83  Fail
235   236  EPSILON   9   5   5     1  10    11     41       34.17  Fail
255   256    GAMMA  14   0   5    11   8    11     49       40.83  Fail
260   261    GAMMA  12   2   6     6   6    11     43       35.83  Fail
263   264    GAMMA  14   3   5     8  13     6     49       40.83  Fail
278   279    GAMMA  10   3   6     7  11     8     45       37.50  Fail
288   289    GAMMA  18   0   3     8  17     3     49       40.83  Fail
298   299    GAMMA  15   1   2     7  10     2     37       30.83  Fail
299   300    GAMMA  17   1   3     3  11    12     47       39.17  Fail
302   303    OMEGA   0   0   0     0   0     0      0        0.00  Fail
303   304    OMEGA   5   0   3    11   7    10     36       30.00  Fail
309   310    OMEGA  12   1   3    10  16     6     48       40.00  Fail
311   312    OMEGA   6   0   1    11   9     4     31       25.83  Fail
316   317    OMEGA  14   0   1     6   6     1     28       23.33  Fail
318   319    OMEGA  11   0   2     6  16    10     45       37.50  Fail
323   324    SIGMA   9   0   2     3  11     1     26       21.67  Fail
324   325    SIGMA   6   0   3     3  10     4     26       21.67  Fail
325   326    SIGMA   9   3   3    12  10     5     42       35.00  Fail
326   327    SIGMA   5   3   9    10  10     7     44       36.67  Fail
334   335    SIGMA  10   0   4     6  13     9     42       35.00  Fail
340   341    SIGMA   7   0   3     0  13     8     31       25.83  Fail
356   357    SIGMA  14   1   6     8  10     9     48       40.00  Fail
359   360    SIGMA  12   3   2     7  13     9     46       38.33  Fail
361   362     ZETA   0   0   6     7  13     0     26       21.67  Fail
364   365     ZETA   5   3   3     2  10     9     32       26.67  Fail
368   369     ZETA   0   0   0     0   0     0      0        0.00  Fail
372   373     ZETA   7   2   2     6  10     7     34       28.33  Fail
378   379     ZETA   8   0   2     6  15     8     39       32.50  Fail
380   381    OMEGA   0   0   0     0   0     0      0        0.00  Fail
```

```
388   389    OMEGA  10  12   1     8  15     3     49       40.83  Fail
393   394    OMEGA   2   5   1     2  10     6     26       21.67  Fail
394   395    OMEGA  12   0   6     4  10     9     41       34.17  Fail
409   410    OMEGA  12   5   1    20  10     0     48       40.00  Fail
416   417    OMEGA   9   0   0     0   0     0      9        7.50  Fail
424   425    SIGMA   6   1   0     0   0     0      7        5.83  Fail
430   431    SIGMA   6   1   9    11   8    10     45       37.50  Fail
444   445    SIGMA   5   2  11     0  10     0     28       23.33  Fail
453   454    SIGMA   1   5   0     0   0     0      6        5.00  Fail
457   458    SIGMA  12   3   2     8   7    12     44       36.67  Fail
461   462    SIGMA   0   0   0     0   0     0      0        0.00  Fail
469   470    SIGMA   1   1   2     0  10     0     14       11.67  Fail
474   475    SIGMA  11   4   2     2   8    10     37       30.83  Fail
```

```
     backlogs Coding-skills PP_Grade DV_Grade   skills section
20          5          poor     poor     Poor     Poor   alpha
27          4          poor     poor     Poor     Poor   alpha
57          4          poor     poor     Poor     Poor   alpha
75          4          poor     poor     Poor     Poor   alpha
82          6          poor     poor     Poor     Poor   alpha
88          4          poor     poor     Poor     Poor   alpha
147         5          poor     poor     Poor     Poor   alpha
160         4       average  Average  Average  Average   alpha
178         3          poor     poor     Poor     Poor   alpha
180         4          poor     poor     Poor     Poor   alpha
197         4          poor     poor     Poor     Poor   alpha
206         4          poor     poor     Poor     Poor   alpha
210         6          poor     poor     Poor     Poor   alpha
226         4          poor     poor     Poor     Poor   alpha
229         3          poor     poor     Poor     Poor   alpha
235         4          poor     poor     Poor     Poor   alpha
255         3          poor     poor     Poor     Poor   alpha
260         4          poor     poor     Poor     Poor   alpha
263         4          poor     poor     Poor     Poor   alpha
278         4          poor     poor     Poor     Poor   alpha
288         4          poor     poor     Poor     Poor   alpha
298         4          poor     poor     Poor     Poor   alpha
299         3          poor     poor     Poor     Poor   alpha
302         6          poor     poor     Poor     Poor   alpha
303         4          poor     poor     Poor     Poor   alpha
309         3          poor     poor     Poor     Poor   alpha
311         5          poor     poor     Poor     Poor   alpha
316         5          poor     poor     Poor     Poor   alpha
318         3          poor     poor     Poor     Poor   alpha
323         5          poor     poor     Poor     Poor   alpha
324         5          poor     poor     Poor     Poor   alpha
325         4          poor     poor     Poor     Poor   alpha
326         4          poor     poor     Poor     Poor   alpha
334         4          poor     poor     Poor     Poor   alpha
340         5          poor     poor     Poor     Poor   alpha
356         4          poor     poor     Poor     Poor   alpha
359         4          poor     poor     Poor     Poor   alpha
361         5          poor     poor     Poor     Poor   alpha
364         5          poor     poor     Poor     Poor   alpha
368         6          poor     poor     Poor     Poor   alpha
372         5          poor     poor     Poor     Poor   alpha
378         5          poor     poor     Poor     Poor   alpha
380         6          poor     poor     Poor     Poor   alpha
388         3          poor     poor     Poor     Poor   alpha
393         5          poor     poor     Poor     Poor   alpha
394         4          poor     poor     Poor     Poor   alpha
409         3          poor     poor     Poor     Poor   alpha
416         6          poor     poor     Poor     Poor   alpha
424         6          poor     poor     Poor     Poor   alpha
430         4          poor     poor     Poor     Poor   alpha
444         4          poor     poor     Poor     Poor   alpha
453         6          poor     poor     Poor     Poor   alpha
457         4          poor     poor     Poor     Poor   alpha
461         6          poor     poor     Poor     Poor   alpha
469         5          poor     poor     Poor     Poor   alpha
474         4          poor     poor     Poor     Poor   alpha
```

In [105…

```
df.loc[df['Total'] > 110, ['Grade']] = "A+"
df.loc[(df['Total'] > 90) & (df['Total'] <= 110), ['Grade']] = "A"
df.loc[(df['Total'] > 70) & (df['Total'] <= 90), ['Grade']] = "B+"
df.loc[(df['Total'] > 50) & (df['Total'] <= 70), ['Grade']] = "B"
df.loc[df['Total'] <= 50, ['Grade']] = "Fail"
failed_students_Total = df[df['Total'] < 50]
print("DataFrame of students who failed in subjects:")
print(failed_students_Total)
```

```
DataFrame of students who failed in subjects:
     S.NO  SECTION  DV  M2  PP  BEEE  FL  FIMS  Total  Percentage Grade  \
20     21    ALPHA   4   2   5     3  16     9     39       32.50  Fail
```

27	28	ALPHA	5	4	3	12	13	5	42	35.00	Fail
57	58	ALPHA	2	2	4	10	10	3	31	25.83	Fail
75	76	BETA	5	8	7	15	10	2	47	39.17	Fail
82	83	BETA	2	0	2	0	0	0	4	3.33	Fail
88	89	BETA	2	17	0	3	15	2	39	32.50	Fail
147	148	DELTA	9	5	6	8	13	4	45	37.50	Fail
160	161	DELTA	7	0	14	5	10	5	41	34.17	Fail
178	179	DELTA	13	0	3	2	10	15	43	35.83	Fail
180	181	SIGMA	9	3	4	2	10	15	43	35.83	Fail
197	198	EPSILON	11	1	1	2	10	9	34	28.33	Fail
206	207	EPSILON	6	6	2	3	10	11	38	31.67	Fail
210	211	EPSILON	0	0	0	0	0	0	0	0.00	Fail
226	227	EPSILON	8	0	3	0	10	14	35	29.17	Fail
229	230	EPSILON	13	2	5	9	10	10	49	40.83	Fail
235	236	EPSILON	9	5	5	1	10	11	41	34.17	Fail
255	256	GAMMA	14	0	5	11	8	11	49	40.83	Fail
260	261	GAMMA	12	2	6	6	6	11	43	35.83	Fail
263	264	GAMMA	14	3	5	8	13	6	49	40.83	Fail
278	279	GAMMA	10	3	6	7	11	8	45	37.50	Fail
288	289	GAMMA	18	0	3	8	17	3	49	40.83	Fail
298	299	GAMMA	15	1	2	7	10	2	37	30.83	Fail
299	300	GAMMA	17	1	3	3	11	12	47	39.17	Fail
302	303	OMEGA	0	0	0	0	0	0	0	0.00	Fail
303	304	OMEGA	5	0	3	11	7	10	36	30.00	Fail
309	310	OMEGA	12	1	3	10	16	6	48	40.00	Fail
311	312	OMEGA	6	0	1	11	9	4	31	25.83	Fail
316	317	OMEGA	14	0	1	6	6	1	28	23.33	Fail
318	319	OMEGA	11	0	2	6	16	10	45	37.50	Fail
323	324	SIGMA	9	0	2	3	11	1	26	21.67	Fail
324	325	SIGMA	6	0	3	3	10	4	26	21.67	Fail
325	326	SIGMA	9	3	3	12	10	5	42	35.00	Fail
326	327	SIGMA	5	3	9	10	10	7	44	36.67	Fail
334	335	SIGMA	10	0	4	6	13	9	42	35.00	Fail
340	341	SIGMA	7	0	3	0	13	8	31	25.83	Fail
356	357	SIGMA	14	1	6	8	10	9	48	40.00	Fail
359	360	SIGMA	12	3	2	7	13	9	46	38.33	Fail
361	362	ZETA	0	0	6	7	13	0	26	21.67	Fail
364	365	ZETA	5	3	3	2	10	9	32	26.67	Fail
368	369	ZETA	0	0	0	0	0	0	0	0.00	Fail
372	373	ZETA	7	2	2	6	10	7	34	28.33	Fail
378	379	ZETA	8	0	2	6	15	8	39	32.50	Fail
380	381	OMEGA	0	0	0	0	0	0	0	0.00	Fail
388	389	OMEGA	10	12	1	8	15	3	49	40.83	Fail
393	394	OMEGA	2	5	1	2	10	6	26	21.67	Fail
394	395	OMEGA	12	0	6	4	10	9	41	34.17	Fail
409	410	OMEGA	12	5	1	20	10	0	48	40.00	Fail
416	417	OMEGA	9	0	0	0	0	0	9	7.50	Fail
424	425	SIGMA	6	1	0	0	0	0	7	5.83	Fail
430	431	SIGMA	6	1	9	11	8	10	45	37.50	Fail
444	445	SIGMA	5	2	11	0	10	0	28	23.33	Fail
453	454	SIGMA	1	5	0	0	0	0	6	5.00	Fail
457	458	SIGMA	12	3	2	8	7	12	44	36.67	Fail
461	462	SIGMA	0	0	0	0	0	0	0	0.00	Fail
469	470	SIGMA	1	1	2	0	10	0	14	11.67	Fail
474	475	SIGMA	11	4	2	2	8	10	37	30.83	Fail

	backlogs	Coding-skills	PP_Grade	DV_Grade	skills	section
20	5	poor	poor	Poor	Poor	alpha
27	4	poor	poor	Poor	Poor	alpha
57	4	poor	poor	Poor	Poor	alpha
75	4	poor	poor	Poor	Poor	alpha
82	6	poor	poor	Poor	Poor	alpha
88	4	poor	poor	Poor	Poor	alpha
147	5	poor	poor	Poor	Poor	alpha
160	4	average	Average	Average	Average	alpha
178	3	poor	poor	Poor	Poor	alpha
180	4	poor	poor	Poor	Poor	alpha
197	4	poor	poor	Poor	Poor	alpha
206	4	poor	poor	Poor	Poor	alpha
210	6	poor	poor	Poor	Poor	alpha
226	4	poor	poor	Poor	Poor	alpha
229	3	poor	poor	Poor	Poor	alpha
235	4	poor	poor	Poor	Poor	alpha
255	3	poor	poor	Poor	Poor	alpha
260	4	poor	poor	Poor	Poor	alpha
263	4	poor	poor	Poor	Poor	alpha
278	4	poor	poor	Poor	Poor	alpha
288	4	poor	poor	Poor	Poor	alpha
298	4	poor	poor	Poor	Poor	alpha
299	3	poor	poor	Poor	Poor	alpha
302	6	poor	poor	Poor	Poor	alpha
303	4	poor	poor	Poor	Poor	alpha
309	3	poor	poor	Poor	Poor	alpha

311	5	poor	poor	Poor	Poor	alpha
316	5	poor	poor	Poor	Poor	alpha
318	3	poor	poor	Poor	Poor	alpha
323	5	poor	poor	Poor	Poor	alpha
324	5	poor	poor	Poor	Poor	alpha
325	4	poor	poor	Poor	Poor	alpha
326	4	poor	poor	Poor	Poor	alpha
334	4	poor	poor	Poor	Poor	alpha
340	5	poor	poor	Poor	Poor	alpha
356	4	poor	poor	Poor	Poor	alpha
359	4	poor	poor	Poor	Poor	alpha
361	5	poor	poor	Poor	Poor	alpha
364	5	poor	poor	Poor	Poor	alpha
368	6	poor	poor	Poor	Poor	alpha
372	5	poor	poor	Poor	Poor	alpha
378	5	poor	poor	Poor	Poor	alpha
380	6	poor	poor	Poor	Poor	alpha
388	3	poor	poor	Poor	Poor	alpha
393	5	poor	poor	Poor	Poor	alpha
394	4	poor	poor	Poor	Poor	alpha
409	3	poor	poor	Poor	Poor	alpha
416	6	poor	poor	Poor	Poor	alpha
424	6	poor	poor	Poor	Poor	alpha
430	4	poor	poor	Poor	Poor	alpha
444	4	poor	poor	Poor	Poor	alpha
453	6	poor	poor	Poor	Poor	alpha
457	4	poor	poor	Poor	Poor	alpha
461	6	poor	poor	Poor	Poor	alpha
469	5	poor	poor	Poor	Poor	alpha
474	4	poor	poor	Poor	Poor	alpha

In [107…

```
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
```

Out[107…

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs	Coding-skills	PP_Grade	DV_Grade	skills
0	1	ALPHA	12	0	17	9	19	15	72	60.00	C+	2	good	Good	Good	Good
1	2	ALPHA	19	12	16	16	18	3	84	70.00	B	1	good	Good	Good	Good
2	3	ALPHA	18	14	18	18	18	16	102	85.00	B+	0	very good	Very Good	Very Good	Very Good
3	4	ALPHA	15	9	19	17	19	15	94	78.33	B	1	very good	Very Good	Very Good	Very Good
4	5	ALPHA	18	17	19	19	20	18	111	92.50	A	0	very good	Very Good	Very Good	Very Good
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
475	476	SIGMA	18	2	12	3	17	15	67	55.83	C	2	poor	poor	Poor	Poor
476	477	SIGMA	20	6	16	11	20	14	87	72.50	B	1	good	Good	Good	Good
477	478	SIGMA	20	0	18	13	20	18	89	74.17	B	1	very good	Very Good	Very Good	Very Good
478	479	SIGMA	20	20	5	19	18	14	96	80.00	B+	1	poor	poor	Poor	Poor
479	480	SIGMA	20	16	18	19	20	19	112	93.33	A	0	very good	Very Good	Very Good	Very Good

480 rows × 17 columns

In [109…

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

Out[109…

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs	Coding-skills	PP_Grade	DV_Grade	skills
0	1	ALPHA	12	0	17	9	19	15	72	60.00	C+	2	good	Good	Good	Good
1	2	ALPHA	19	12	16	16	18	3	84	70.00	B	1	good	Good	Good	Good
2	3	ALPHA	18	14	18	18	18	16	102	85.00	B+	0	very good	Very Good	Very Good	Very Good
3	4	ALPHA	15	9	19	17	19	15	94	78.33	B	1	very good	Very Good	Very Good	Very Good
4	5	ALPHA	18	17	19	19	20	18	111	92.50	A	0	very good	Very Good	Very Good	Very Good
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
475	476	SIGMA	18	2	12	3	17	15	67	55.83	C	2	poor	poor	Poor	Poor
476	477	SIGMA	20	6	16	11	20	14	87	72.50	B	1	good	Good	Good	Good
477	478	SIGMA	20	0	18	13	20	18	89	74.17	B	1	very good	Very Good	Very Good	Very Good
478	479	SIGMA	20	20	5	19	18	14	96	80.00	B+	1	poor	poor	Poor	Poor
479	480	SIGMA	20	16	18	19	20	19	112	93.33	A	0	very good	Very Good	Very Good	Very Good

480 rows × 17 columns

In [111…

```
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[111…

```
SECTION
SIGMA      73
GAMMA      43
EPSILON    41
OMEGA      41
DELTA      35
BETA       32
ALPHA      26
ZETA       13
Name: count, dtype: int64
```

In [113…

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

Out[113…

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs	Coding-skills	PP_Grade	DV_Grade	skills
0	1	ALPHA	12	0	17	9	19	15	72	60.00	C+	2	good	Good	Good	Good
1	2	ALPHA	19	12	16	16	18	3	84	70.00	B	1	good	Good	Good	Good
2	3	ALPHA	18	14	18	18	18	16	102	85.00	B+	0	very good	Very Good	Very Good	Very Good
3	4	ALPHA	15	9	19	17	19	15	94	78.33	B	1	very good	Very Good	Very Good	Very Good
4	5	ALPHA	18	17	19	19	20	18	111	92.50	A	0	very good	Very Good	Very Good	Very Good
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
475	476	SIGMA	18	2	12	3	17	15	67	55.83	C	2	poor	poor	Poor	Poor
476	477	SIGMA	20	6	16	11	20	14	87	72.50	B	1	good	Good	Good	Good
477	478	SIGMA	20	0	18	13	20	18	89	74.17	B	1	very good	Very Good	Very Good	Very Good
478	479	SIGMA	20	20	5	19	18	14	96	80.00	B+	1	poor	poor	Poor	Poor
479	480	SIGMA	20	16	18	19	20	19	112	93.33	A	0	very good	Very Good	Very Good	Very Good

480 rows × 17 columns

In [115…
```
df
```

Out[115…

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs	Coding-skills	PP_Grade	DV_Grade	skills
0	1	ALPHA	12	0	17	9	19	15	72	60.00	C+	2	good	Good	Good	Good
1	2	ALPHA	19	12	16	16	18	3	84	70.00	B	1	good	Good	Good	Good
2	3	ALPHA	18	14	18	18	18	16	102	85.00	B+	0	very good	Very Good	Very Good	Very Good
3	4	ALPHA	15	9	19	17	19	15	94	78.33	B	1	very good	Very Good	Very Good	Very Good
4	5	ALPHA	18	17	19	19	20	18	111	92.50	A	0	very good	Very Good	Very Good	Very Good
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
475	476	SIGMA	18	2	12	3	17	15	67	55.83	C	2	poor	poor	Poor	Poor
476	477	SIGMA	20	6	16	11	20	14	87	72.50	B	1	good	Good	Good	Good
477	478	SIGMA	20	0	18	13	20	18	89	74.17	B	1	very good	Very Good	Very Good	Very Good
478	479	SIGMA	20	20	5	19	18	14	96	80.00	B+	1	poor	poor	Poor	Poor
479	480	SIGMA	20	16	18	19	20	19	112	93.33	A	0	very good	Very Good	Very Good	Very Good

480 rows × 17 columns

In [117…
```
less_than_10 = df[
    (df['DV'] < 10) |
    (df['M2'] < 10) |
    (df['PP'] < 10) |
    (df['BEEE'] < 10) |
    (df['FL'] < 10) |
    (df['FIMS'] < 10)
]

print("Students with less than 10 marks in at least one subject:")
print(less_than_10)
```

```
Students with less than 10 marks in at least one subject:
     S.NO SECTION  DV  M2  PP  BEEE  FL  FIMS  Total  Percentage Grade  \
0       1   ALPHA  12   0  17     9  19    15     72       60.00    C+
1       2   ALPHA  19  12  16    16  18     3     84       70.00     B
3       4   ALPHA  15   9  19    17  19    15     94       78.33     B
5       6   ALPHA  17  16  18    10  15     9     85       70.83     B
8       9   ALPHA  10  18   0    20  19    15     82       68.33    C+
..    ...     ...  ..  ..  ..   ...  ..   ...    ...         ...   ...
474   475   SIGMA  11   4   2     2   8    10     37       30.83     F
475   476   SIGMA  18   2  12     3  17    15     67       55.83     C
476   477   SIGMA  20   6  16    11  20    14     87       72.50     B
477   478   SIGMA  20   0  18    13  20    18     89       74.17     B
478   479   SIGMA  20  20   5    19  18    14     96       80.00    B+

     backlogs Coding-skills   PP_Grade   DV_Grade     skills section
0           2          good       Good       Good       Good   alpha
1           1          good       Good       Good       Good   alpha
3           1     very good  Very Good  Very Good  Very Good   alpha
5           1     very good  Very Good  Very Good  Very Good   alpha
8           1          poor       poor       Poor       Poor   alpha
..        ...           ...        ...        ...        ...     ...
474         4          poor       poor       Poor       Poor   alpha
475         2          poor       poor       Poor       Poor   alpha
476         1          good       Good       Good       Good   alpha
477         1     very good  Very Good  Very Good  Very Good   alpha
478         1          poor       poor       Poor       Poor   alpha

[304 rows x 17 columns]
```

In [119…
```
j=df.sort_values('backlogs')
j
```

Out[119…

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs	Coding-skills	PP_Grade	DV_Grade	skills
239	240	EPSILON	17	17	14	20	18	17	103	85.83	B+	0	average	Average	Average	Average
194	195	EPSILON	18	11	19	10	20	20	98	81.67	B+	0	very good	Very Good	Very Good	Very Good
195	196	EPSILON	20	12	18	17	20	20	107	89.17	B+	0	very good	Very Good	Very Good	Very Good
201	202	EPSILON	18	17	15	18	19	19	106	88.33	B+	0	good	Good	Good	Good
202	203	EPSILON	16	11	16	14	13	19	89	74.17	B	0	good	Good	Good	Good
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
380	381	OMEGA	0	0	0	0	0	0	0	0.00	F	6	poor	poor	Poor	Poor
302	303	OMEGA	0	0	0	0	0	0	0	0.00	F	6	poor	poor	Poor	Poor
453	454	SIGMA	1	5	0	0	0	0	6	5.00	F	6	poor	poor	Poor	Poor
368	369	ZETA	0	0	0	0	0	0	0	0.00	F	6	poor	poor	Poor	Poor
461	462	SIGMA	0	0	0	0	0	0	0	0.00	F	6	poor	poor	Poor	Poor

480 rows × 17 columns

In [121…

```
df.iloc[:, 2:] = df.iloc[:, 2:].apply(pd.to_numeric, errors='coerce')
backlog_counts = (df.iloc[:, 2:] < 10).sum()
highest_backlog_subject = backlog_counts.idxmax()
highest_backlog_count = backlog_counts.max()
print(f"The subject with the highest number of backlogs is '{highest_backlog_subject}' with {highest_backlog_co
```

```
The subject with the highest number of backlogs is 'backlogs' with 480 backlogs.
```

In [123…

```
df.iloc[:, 2:] = df.iloc[:, 2:].apply(pd.to_numeric, errors='coerce')
backlog_counts = (df.iloc[:, 2:] < 10).sum()
print("Number of backlogs in each subject:")
print(backlog_counts)
```

```
Number of backlogs in each subject:
DV                66
M2               226
PP               135
BEEE             127
FL                23
FIMS              86
Total              9
Percentage         9
Grade              0
backlogs         480
Coding-skills      0
PP_Grade           0
DV_Grade           0
skills             0
section            0
dtype: int64
```

In [125…

```
import string
df.iloc[:, 2:] = df.iloc[:, 2:].apply(pd.to_numeric, errors='coerce')
absent_counts = df.iloc[:, 2:].isna().sum()
alpha_codes = {i: string.ascii_uppercase[i] for i in range(len(absent_counts))}
absent_alpha = absent_counts.map(alpha_codes)
print("Absentees in alphabetical codes for each subject:")
print(absent_alpha)
```

```
Absentees in alphabetical codes for each subject:
DV                A
M2                A
PP                A
BEEE              A
FL                A
FIMS              A
Total             A
Percentage        A
Grade           NaN
backlogs          A
Coding-skills   NaN
PP_Grade        NaN
DV_Grade        NaN
skills          NaN
section         NaN
dtype: object
```

In [127…

```
%matplotlib inline
```

In [129…
```
print(df.empty)
```

```
False
```

In [131…
```
print(df.columns)
```

```
Index(['S.NO', 'SECTION', 'DV', 'M2', 'PP', 'BEEE', 'FL', 'FIMS', 'Total',
       'Percentage', 'Grade', 'backlogs', 'Coding-skills', 'PP_Grade',
       'DV_Grade', 'skills', 'section'],
      dtype='object')
```

In [133…
```
print(df['Grade'].notna().sum())
```

```
0
```

In [135…
```
if df['Grade'].notna().sum() > 0:
    df['Grade'].value_counts().plot(kind='bar', figsize=(8, 6), color='skyblue', edgecolor='black')
    plt.title('Distribution of Grades', fontsize=16)
    plt.xlabel('Grades', fontsize=14)
    plt.ylabel('Count', fontsize=14)
    plt.xticks(rotation=0)
    plt.grid(axis='y', linestyle='--', alpha=0.7)
    plt.show()
else:
    print("The 'Grade' column is empty or contains no valid data.")
```

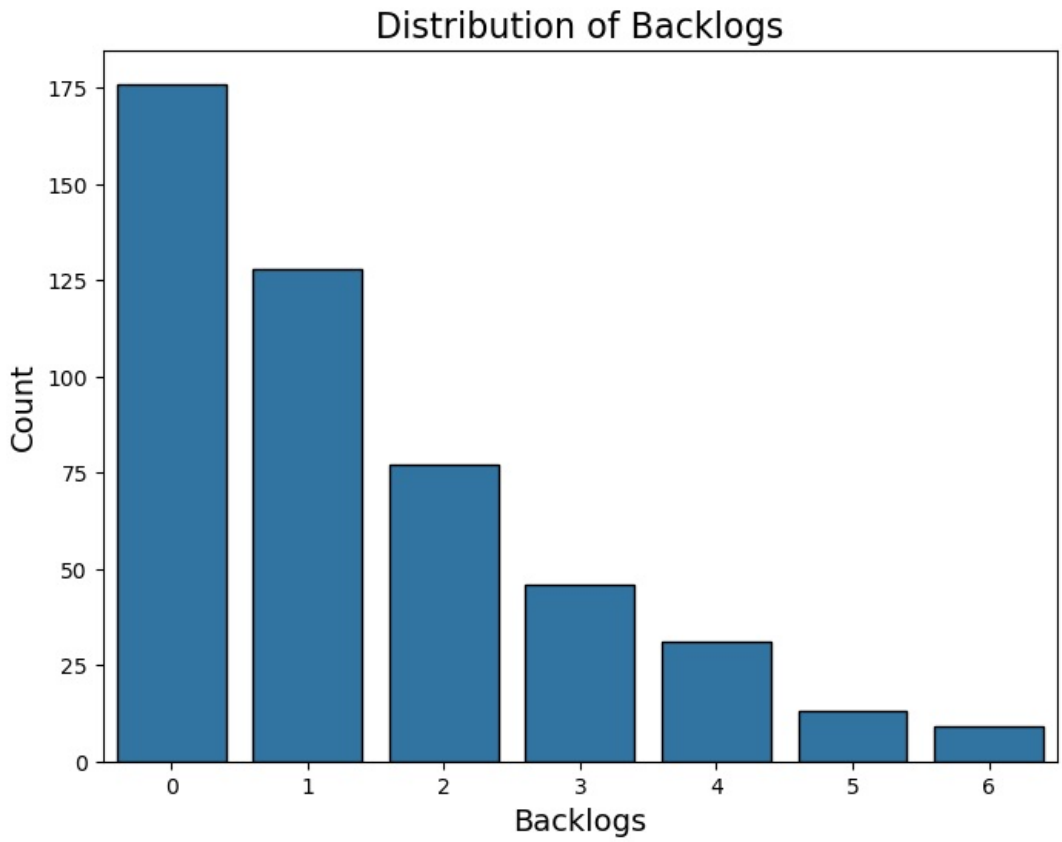
```
The 'Grade' column is empty or contains no valid data.
```

In [137…
```
print(df.head())
```

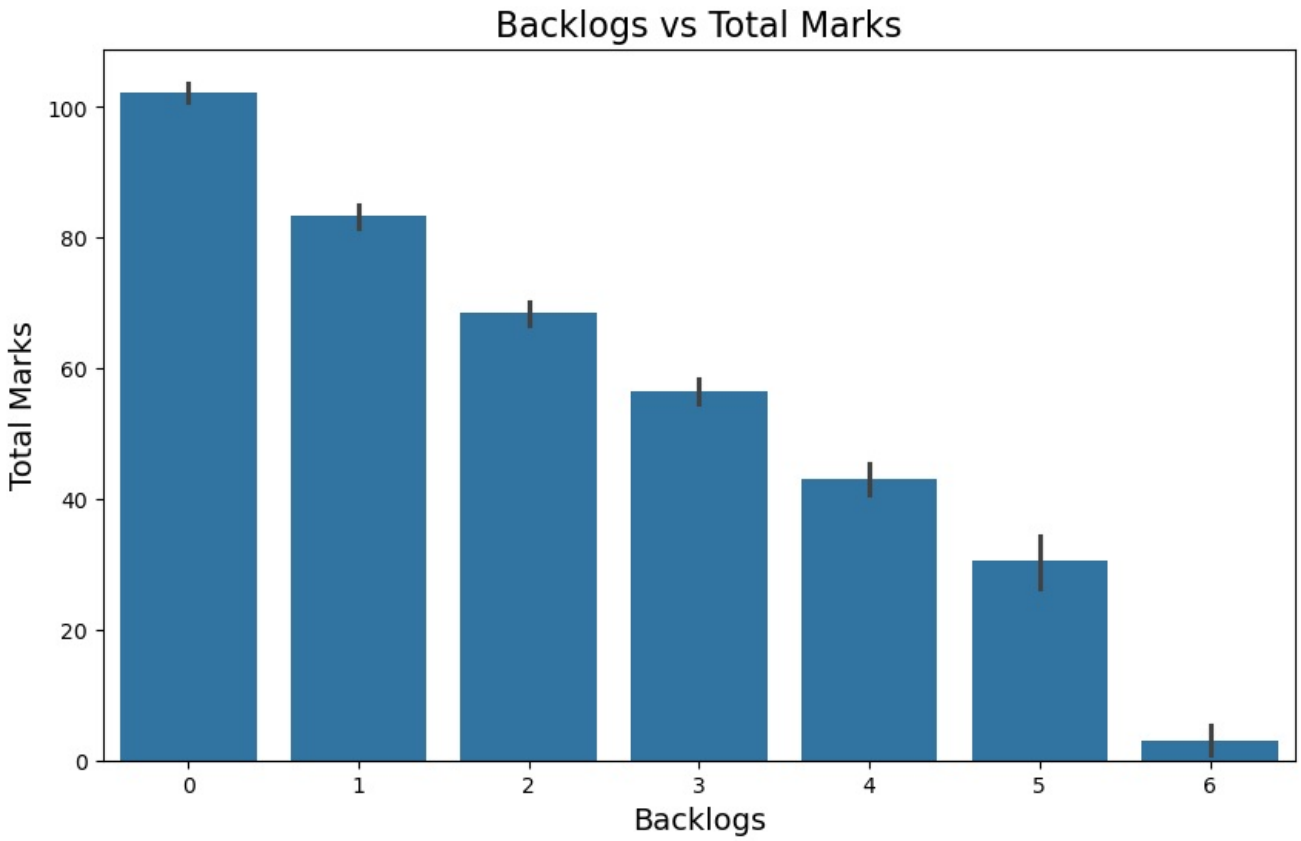
```
   S.NO SECTION  DV  M2  PP  BEEE  FL  FIMS  Total  Percentage Grade  \
0     1   ALPHA  12   0  17     9  19    15     72       60.00   NaN
1     2   ALPHA  19  12  16    16  18     3     84       70.00   NaN
2     3   ALPHA  18  14  18    18  18    16    102       85.00   NaN
3     4   ALPHA  15   9  19    17  19    15     94       78.33   NaN
4     5   ALPHA  18  17  19    19  20    18    111       92.50   NaN

   backlogs Coding-skills PP_Grade DV_Grade skills section
0         2           NaN      NaN      NaN    NaN     NaN
1         1           NaN      NaN      NaN    NaN     NaN
2         0           NaN      NaN      NaN    NaN     NaN
3         1           NaN      NaN      NaN    NaN     NaN
4         0           NaN      NaN      NaN    NaN     NaN
```

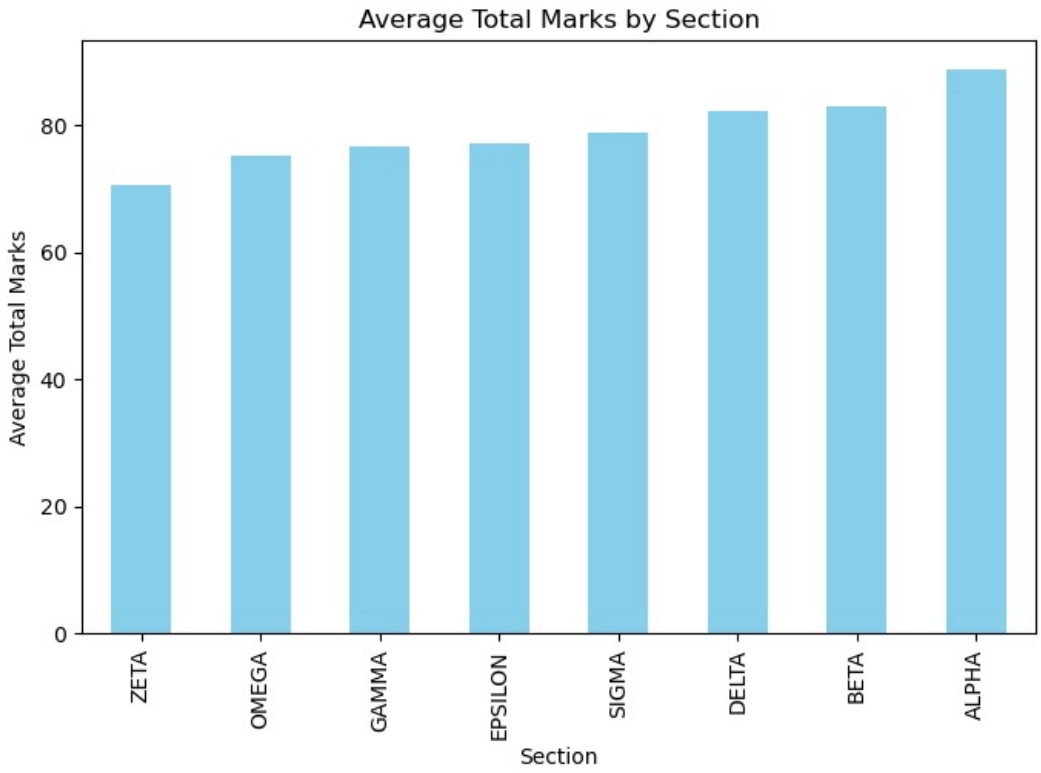
In [139…
```
plt.figure(figsize=(8, 6))
sns.countplot(x='backlogs', data=df, edgecolor='black')
plt.title('Distribution of Backlogs', fontsize=16)
plt.xlabel('Backlogs', fontsize=14)
plt.ylabel('Count', fontsize=14)
plt.show()
```



In [141…
```
plt.figure(figsize=(10, 6))
sns.barplot(x='backlogs', y='Total', data=df)
plt.title('Backlogs vs Total Marks', fontsize=16)
plt.xlabel('Backlogs', fontsize=14)
plt.ylabel('Total Marks', fontsize=14)
plt.show()
```

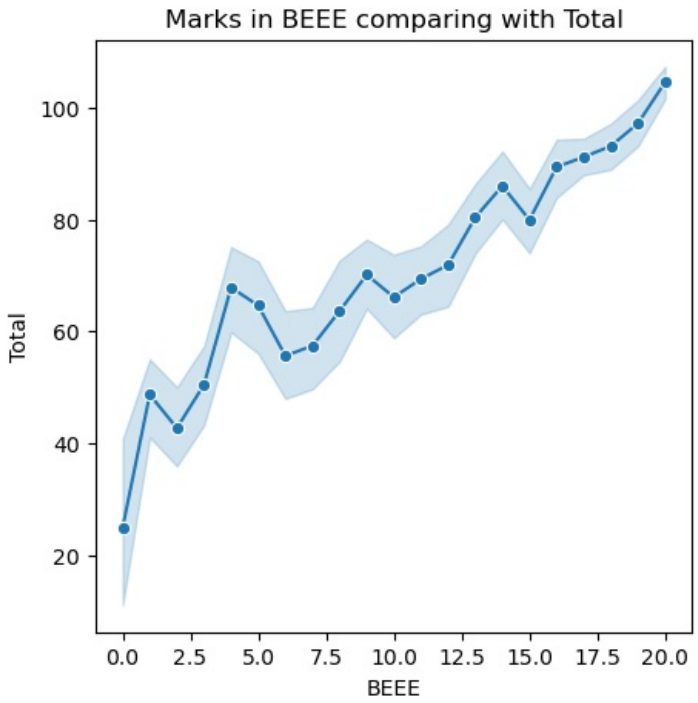


In [143…
```
avg_by_section = df.groupby("SECTION")["Total"].mean().sort_values()
plt.figure(figsize=(8, 5))
avg_by_section.plot(kind="bar", color="skyblue")
plt.title("Average Total Marks by Section")
plt.xlabel("Section")
plt.ylabel("Average Total Marks")
plt.show()
```

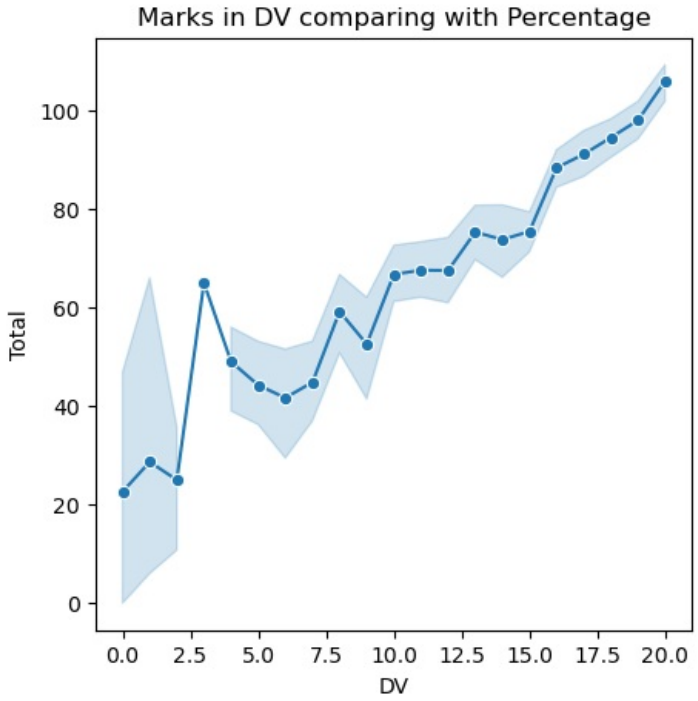


In [145…
```
plt.figure(figsize=(5, 5))
sns.lineplot(x="BEEE", y="Total", data=df, marker="o")
plt.xlabel("BEEE")
plt.ylabel("Total")
```

```
plt.title("Marks in BEEE comparing with Total")
plt.show()
```



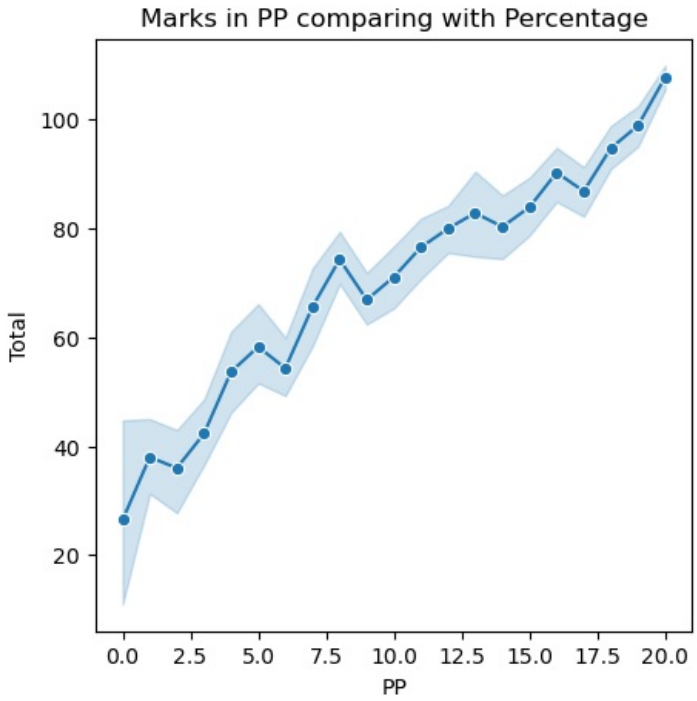
In [147…
```
df.sort_values(["Total"], ascending=True)
plt.figure(figsize=(5, 5))
sns.lineplot(x="DV", y="Total", data=df, marker="o")
plt.xlabel("DV")
plt.ylabel("Total")
plt.title("Marks in DV comparing with Percentage")
plt.show()
```

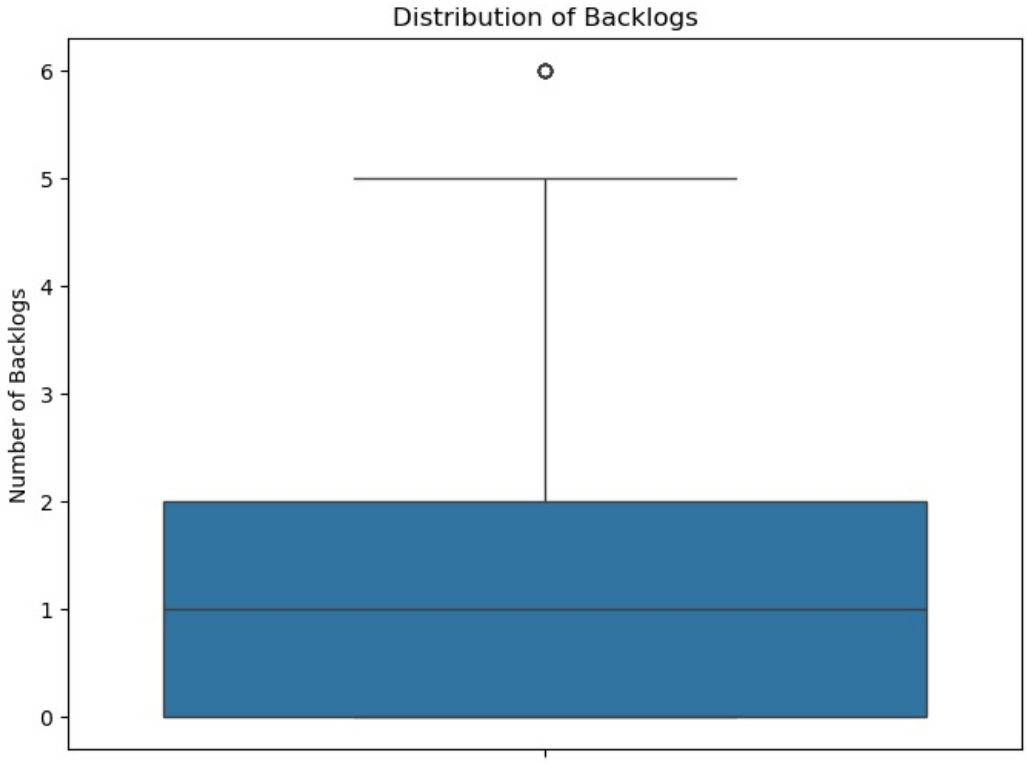


In [149…
```
plt.figure(figsize=(5, 5))
sns.lineplot(x="PP", y="Total", data=df, marker="o")
plt.xlabel("PP")
plt.ylabel("Total")
plt.title("Marks in PP comparing with Percentage")
```

Out[149… `Text(0.5, 1.0, 'Marks in PP comparing with Percentage')`

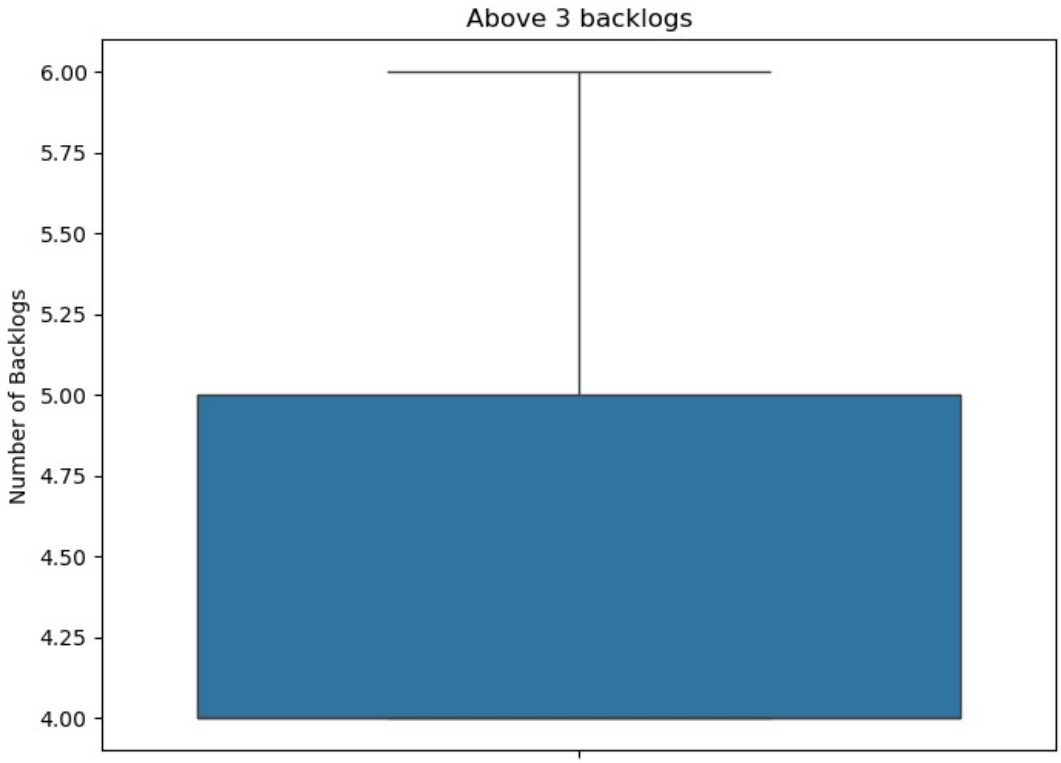
In [150…
```
plt.figure(figsize=(8, 6))
sns.boxplot(y=df['backlogs'])
plt.title("Distribution of Backlogs")
plt.ylabel("Number of Backlogs")
plt.show()
```



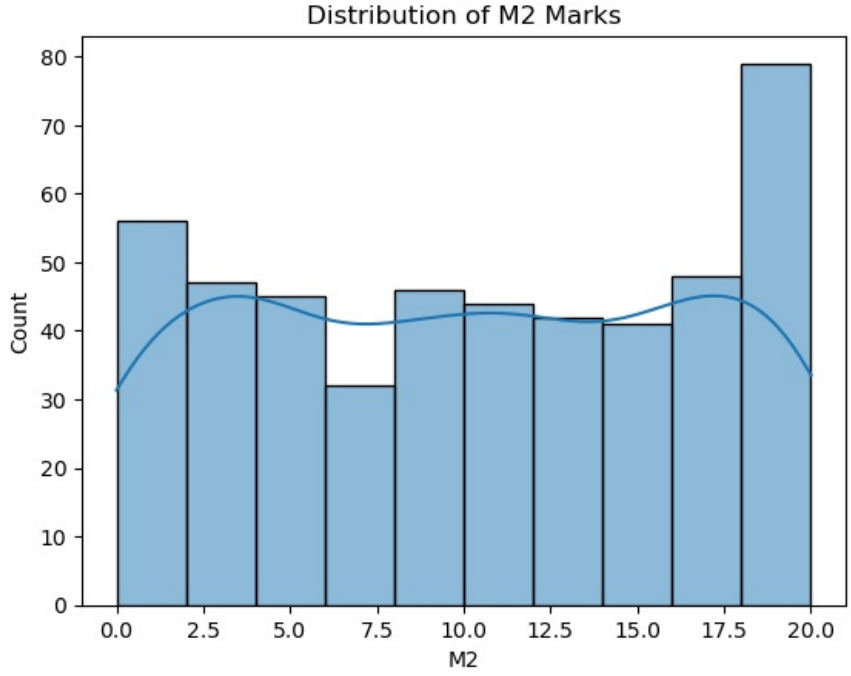


In [153…

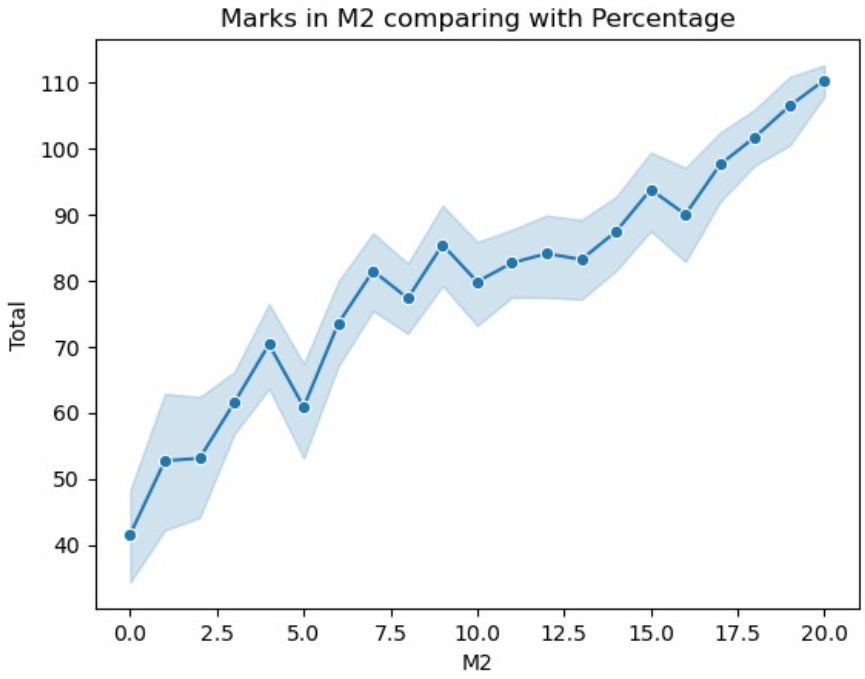
```
filtered_df = df[df['backlogs'] > 3]
plt.figure(figsize=(8, 6))
sns.boxplot(y=filtered_df['backlogs'])
plt.title("Above 3 backlogs")
plt.ylabel("Number of Backlogs")
plt.show()
```



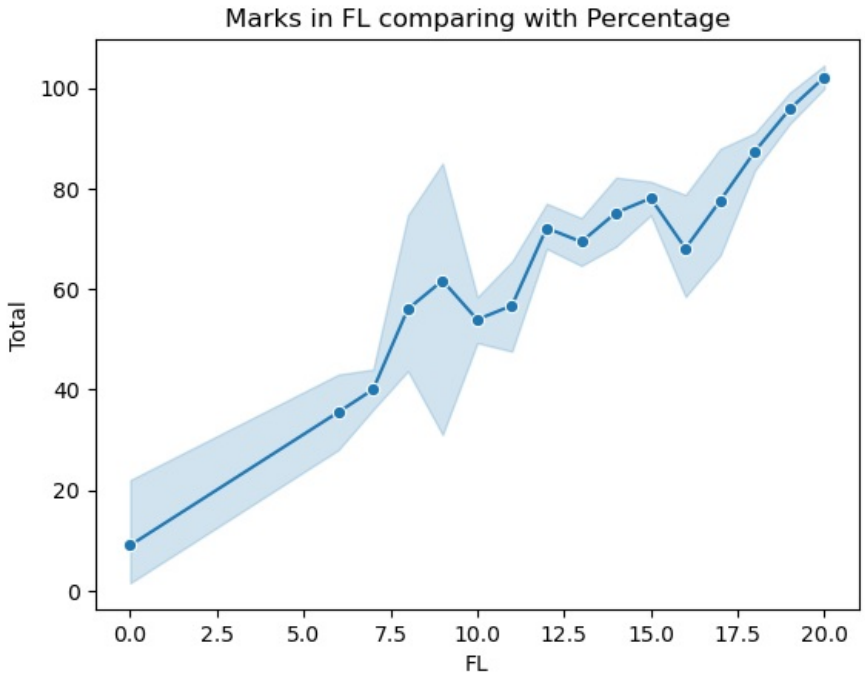
In [155…
```
sns.histplot(df['M2'], kde=True)
plt.title('Distribution of M2 Marks')
plt.show()
```



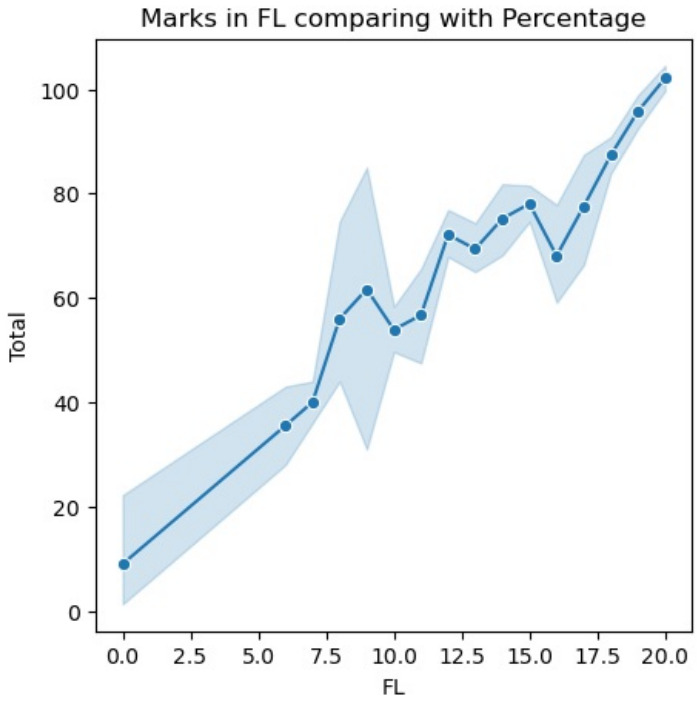
In [157…
```
sns.lineplot(x="M2", y="Total", data=df, marker="o")
plt.xlabel("M2")
plt.ylabel("Total")
plt.title("Marks in M2 comparing with Percentage")
plt.show()
```



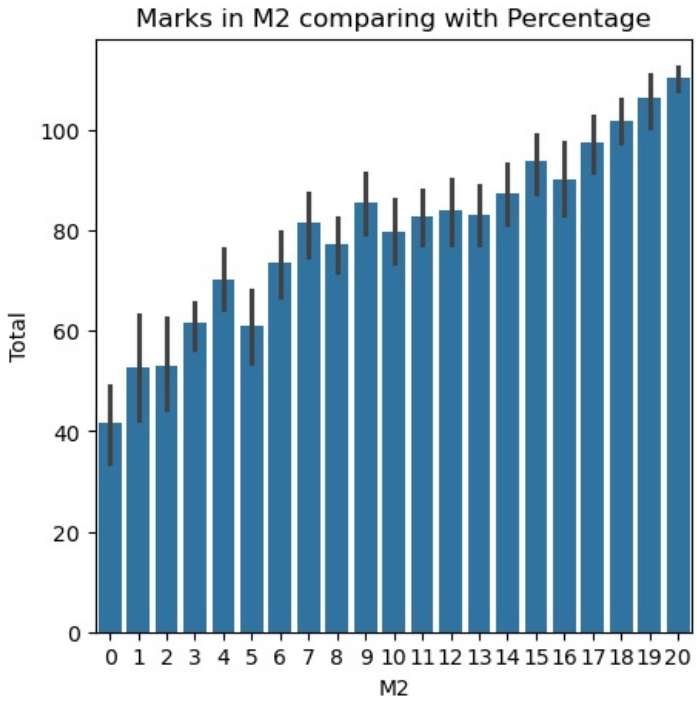
In [158…
```
sns.lineplot(x="FL", y="Total", data=df, marker="o")
plt.xlabel("FL")
plt.ylabel("Total")
plt.title("Marks in FL comparing with Percentage")
plt.show()
```



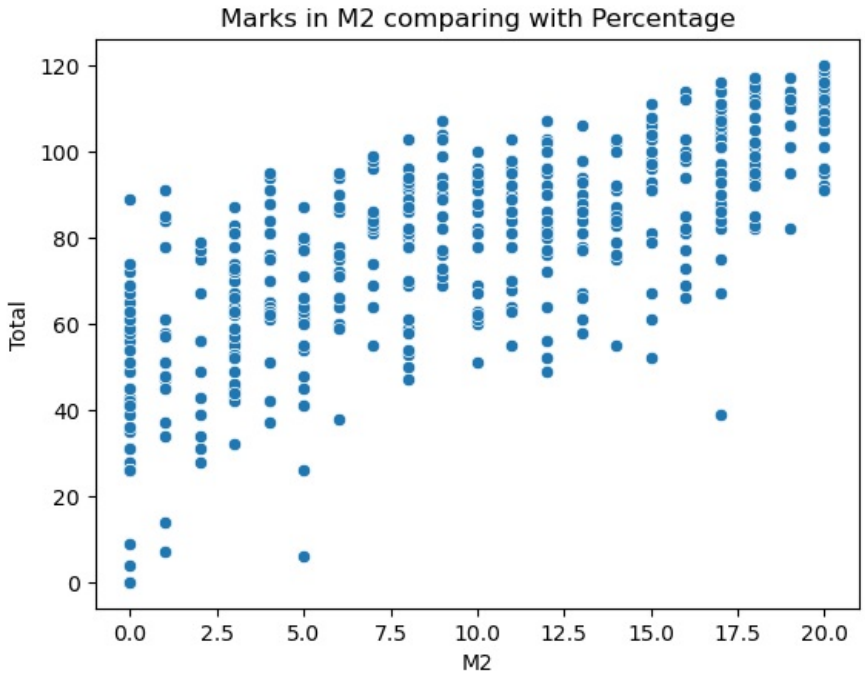
In [160…
```
plt.figure(figsize=(5, 5))
sns.lineplot(x="FL", y="Total", data=df, marker="o")
plt.xlabel("FL")
plt.ylabel("Total")
plt.title("Marks in FL comparing with Percentage")
plt.show()
```



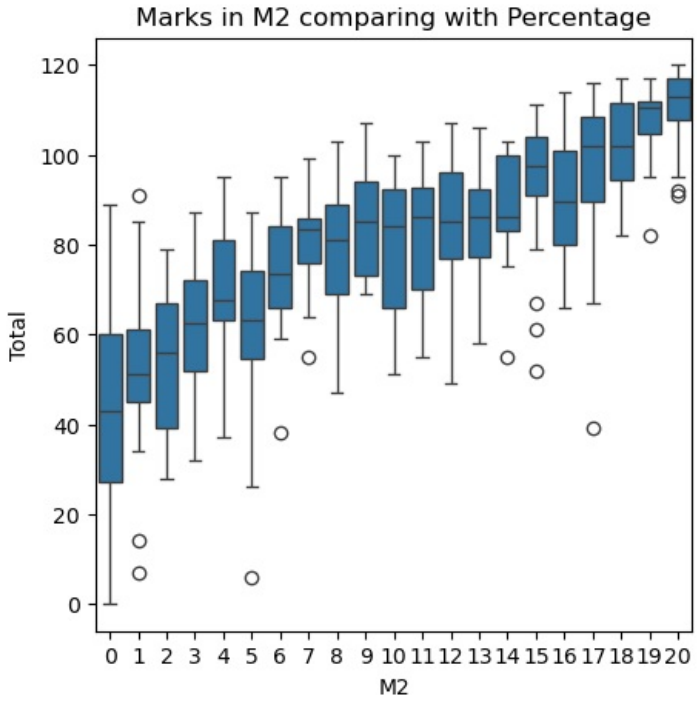
In [162…
```
plt.figure(figsize=(5, 5))
sns.barplot(x="M2", y="Total", data=df)
plt.xlabel("M2")
plt.ylabel("Total")
plt.title("Marks in M2 comparing with Percentage")
plt.show()
```



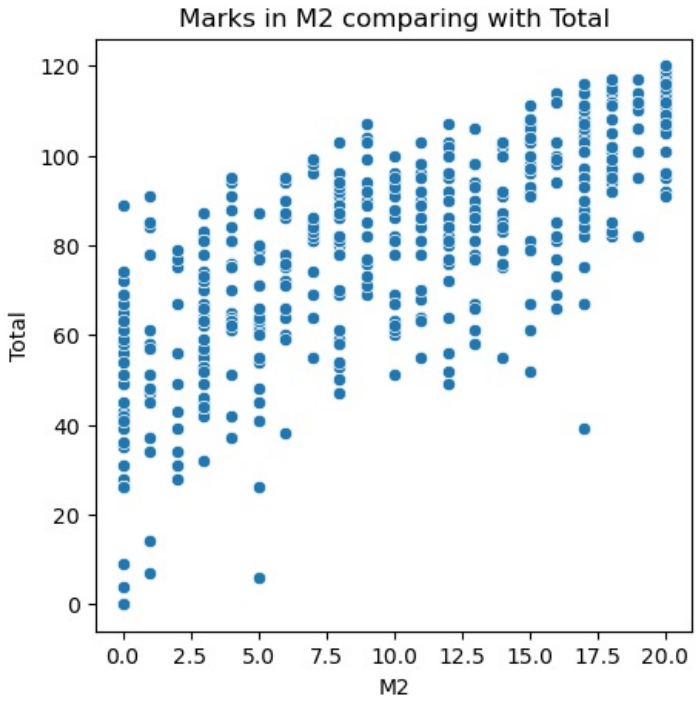
In [163…
```
sns.scatterplot(x="M2", y="Total", data=df, marker="o")
plt.xlabel("M2")
plt.ylabel("Total")
plt.title("Marks in M2 comparing with Percentage")
plt.show()
```



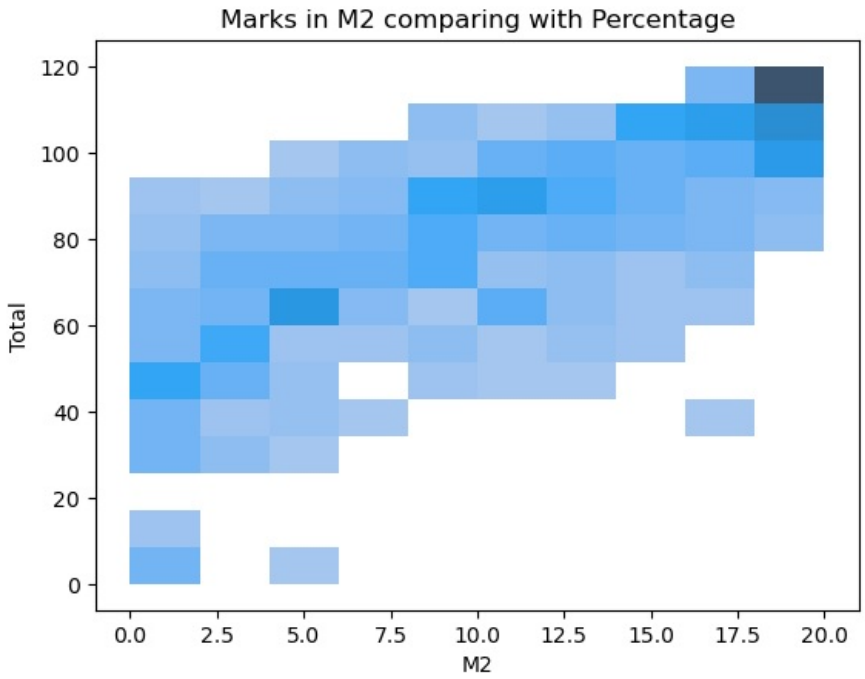
In [165…
```
plt.figure(figsize=(5, 5))
sns.boxplot(x="M2", y="Total", data=df)
plt.xlabel("M2")
plt.ylabel("Total")
plt.title("Marks in M2 comparing with Percentage")
plt.show()
```



In [166…
```
plt.figure(figsize=(5, 5))
sns.scatterplot(x="M2", y="Total", data=df, marker="o")
plt.xlabel("M2")
plt.ylabel("Total")
plt.title("Marks in M2 comparing with Total")
plt.show()
```



In [167…
```
sns.histplot(x="M2", y="Total", data=df)
plt.xlabel("M2")
plt.ylabel("Total")
plt.title("Marks in M2 comparing with Percentage")
plt.show()
```



In [170…
```
def assign_grade(PP):
    if PP >=18 :
        return "very good"

    elif PP >=15 :
        return 'good'
    elif PP >= 13:
        return 'average'
    else:
        return 'poor'
df['Coding-skills'] = df['PP'].apply(assign_grade)
df
```

Out[170…

	S.NO	SECTION	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	Grade	backlogs	Coding-skills	PP_Grade	DV_Grade	skills
0	1	ALPHA	12	0	17	9	19	15	72	60.00	NaN	2	good	NaN	NaN	NaN
1	2	ALPHA	19	12	16	16	18	3	84	70.00	NaN	1	good	NaN	NaN	NaN
2	3	ALPHA	18	14	18	18	18	16	102	85.00	NaN	0	very good	NaN	NaN	NaN
3	4	ALPHA	15	9	19	17	19	15	94	78.33	NaN	1	very good	NaN	NaN	NaN
4	5	ALPHA	18	17	19	19	20	18	111	92.50	NaN	0	very good	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
475	476	SIGMA	18	2	12	3	17	15	67	55.83	NaN	2	poor	NaN	NaN	NaN
476	477	SIGMA	20	6	16	11	20	14	87	72.50	NaN	1	good	NaN	NaN	NaN
477	478	SIGMA	20	0	18	13	20	18	89	74.17	NaN	1	very good	NaN	NaN	NaN
478	479	SIGMA	20	20	5	19	18	14	96	80.00	NaN	1	poor	NaN	NaN	NaN
479	480	SIGMA	20	16	18	19	20	19	112	93.33	NaN	0	very good	NaN	NaN	NaN

480 rows × 17 columns

In [172…

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

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

```
     S.NO SECTION  DV  M2  PP  BEEE  FL  FIMS  Total  Percentage Grade  \
0       1   ALPHA  12   0  17     9  19    15     72       60.00   NaN
1       2   ALPHA  19  12  16    16  18     3     84       70.00   NaN
2       3   ALPHA  18  14  18    18  18    16    102       85.00   NaN
3       4   ALPHA  15   9  19    17  19    15     94       78.33   NaN
4       5   ALPHA  18  17  19    19  20    18    111       92.50   NaN
..    ...     ...  ..  ..  ..   ...  ..   ...    ...         ...   ...
475   476   SIGMA  18   2  12     3  17    15     67       55.83   NaN
476   477   SIGMA  20   6  16    11  20    14     87       72.50   NaN
477   478   SIGMA  20   0  18    13  20    18     89       74.17   NaN
478   479   SIGMA  20  20   5    19  18    14     96       80.00   NaN
479   480   SIGMA  20  16  18    19  20    19    112       93.33   NaN

     backlogs Coding-skills   PP_Grade DV_Grade skills section
0           2          good       Good      NaN    NaN     NaN
1           1          good       Good      NaN    NaN     NaN
2           0     very good  Very Good      NaN    NaN     NaN
3           1     very good  Very Good      NaN    NaN     NaN
4           0     very good  Very Good      NaN    NaN     NaN
..        ...           ...        ...      ...    ...     ...
475         2          poor       poor      NaN    NaN     NaN
476         1          good       Good      NaN    NaN     NaN
477         1     very good  Very Good      NaN    NaN     NaN
478         1          poor       poor      NaN    NaN     NaN
479         0     very good  Very Good      NaN    NaN     NaN

[480 rows x 17 columns]
```

In [174…

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

df['DV_Grade'] = df['PP'].apply(assign_grade)
print(df)
```

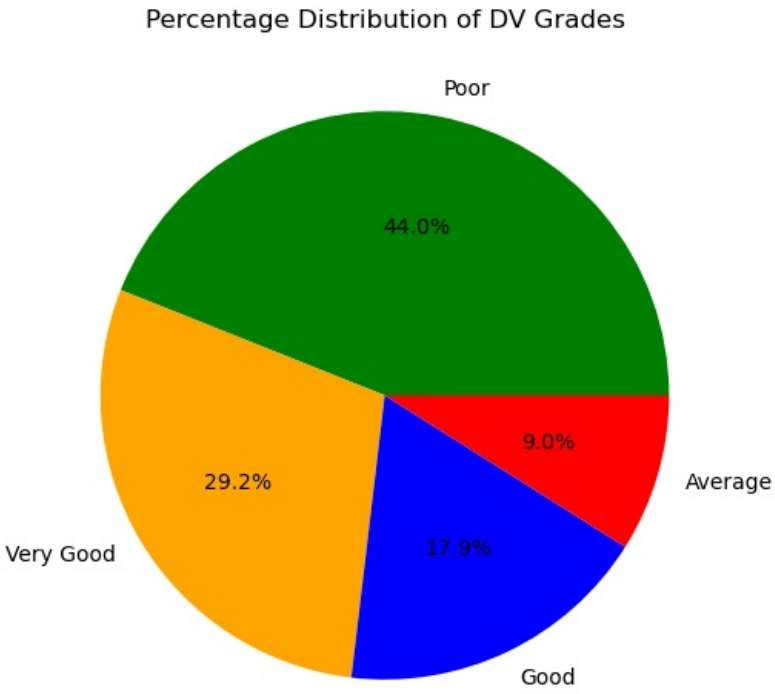
```
     S.NO SECTION  DV  M2  PP  BEEE  FL  FIMS  Total  Percentage Grade  \
0       1   ALPHA  12   0  17     9  19    15     72       60.00   NaN
1       2   ALPHA  19  12  16    16  18     3     84       70.00   NaN
2       3   ALPHA  18  14  18    18  18    16    102       85.00   NaN
3       4   ALPHA  15   9  19    17  19    15     94       78.33   NaN
4       5   ALPHA  18  17  19    19  20    18    111       92.50   NaN
..    ...     ...  ..  ..  ..   ...  ..   ...    ...         ...   ...
475   476   SIGMA  18   2  12     3  17    15     67       55.83   NaN
476   477   SIGMA  20   6  16    11  20    14     87       72.50   NaN
477   478   SIGMA  20   0  18    13  20    18     89       74.17   NaN
478   479   SIGMA  20  20   5    19  18    14     96       80.00   NaN
479   480   SIGMA  20  16  18    19  20    19    112       93.33   NaN

     backlogs Coding-skills   PP_Grade   DV_Grade skills section
0           2          good       Good       Good    NaN     NaN
1           1          good       Good       Good    NaN     NaN
2           0     very good  Very Good  Very Good    NaN     NaN
3           1     very good  Very Good  Very Good    NaN     NaN
4           0     very good  Very Good  Very Good    NaN     NaN
..        ...           ...        ...        ...    ...     ...
475         2          poor       poor       poor    NaN     NaN
476         1          good       Good       Good    NaN     NaN
477         1     very good  Very Good  Very Good    NaN     NaN
478         1          poor       poor       poor    NaN     NaN
479         0     very good  Very Good  Very Good    NaN     NaN

[480 rows x 17 columns]
```

In [179…

```python
def assign_grade(DV):
    if 18 <= DV <= 20:
        return 'Very Good'
    elif 13 <= DV <= 14:
        return 'Average'
    elif 15 <= DV <= 17:
        return 'Good'
    else:
        return 'Poor'
df['DV_Grade'] = df['PP'].apply(assign_grade)
grade_counts = df['DV_Grade'].value_counts()
plt.figure(figsize=(6, 6))
grade_counts.plot(kind='pie', autopct='%1.1f%%', colors=['green', 'orange', 'blue', 'red'])
plt.title("Percentage Distribution of DV Grades")
plt.ylabel("")
plt.show()
```



In [181…

```python
very_good_count = (df['PP'] == 'Very Good').sum()

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

```
Number of 'Very Good': 0
```

In [183…

```python
subjects = ['DV','M2', 'PP', 'BEEE', 'FL', 'FIMS']
subset = []
for index, row in df.iterrows():
```

```
    if any(row[subject] == 20 for subject in subjects):
        subset.append(row)
subset_df = pd.DataFrame(subset)
print(subset_df)
```

```
     S.NO SECTION  DV  M2  PP  BEEE  FL  FIMS  Total  Percentage  Grade  \
4       5   ALPHA  18  17  19    19  20    18    111       92.50    NaN
6       7   ALPHA  15  10  20    20  15    14     94       78.33    NaN
7       8   ALPHA  17  17  19    20  19    13    105       87.50    NaN
8       9   ALPHA  10  18   0    20  19    15     82       68.33    NaN
9      10   ALPHA  18  19  20    20  20    15    112       93.33    NaN
..    ...     ...  ..  ..  ..   ...  ..   ...    ...         ...    ...
473   474   SIGMA  20  20  20    20  20    20    120      100.00    NaN
476   477   SIGMA  20   6  16    11  20    14     87       72.50    NaN
477   478   SIGMA  20   0  18    13  20    18     89       74.17    NaN
478   479   SIGMA  20  20   5    19  18    14     96       80.00    NaN
479   480   SIGMA  20  16  18    19  20    19    112       93.33    NaN

     backlogs Coding-skills   PP_Grade   DV_Grade  skills  section
4           0     very good  Very Good  Very Good     NaN      NaN
6           0     very good  Very Good  Very Good     NaN      NaN
7           0     very good  Very Good  Very Good     NaN      NaN
8           1          poor       poor       Poor     NaN      NaN
9           0     very good  Very Good  Very Good     NaN      NaN
..        ...           ...        ...        ...     ...      ...
473         0     very good  Very Good  Very Good     NaN      NaN
476         1          good       Good       Good     NaN      NaN
477         1     very good  Very Good  Very Good     NaN      NaN
478         1          poor       poor       Poor     NaN      NaN
479         0     very good  Very Good  Very Good     NaN      NaN

[185 rows x 17 columns]
```

In [184…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 480 entries, 0 to 479
Data columns (total 17 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   S.NO           480 non-null    int64
 1   SECTION        480 non-null    object
 2   DV             480 non-null    int32
 3   M2             480 non-null    int32
 4   PP             480 non-null    int32
 5   BEEE           480 non-null    int32
 6   FL             480 non-null    int32
 7   FIMS           480 non-null    int32
 8   Total          480 non-null    int32
 9   Percentage     480 non-null    float64
 10  Grade          0 non-null      object
 11  backlogs       480 non-null    int64
 12  Coding-skills  480 non-null    object
 13  PP_Grade       480 non-null    object
 14  DV_Grade       480 non-null    object
 15  skills         0 non-null      object
 16  section        0 non-null      object
dtypes: float64(1), int32(7), int64(2), object(7)
memory usage: 50.8+ KB
```

In [185…
```
subjects = ['DV', 'M2', '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}")
```

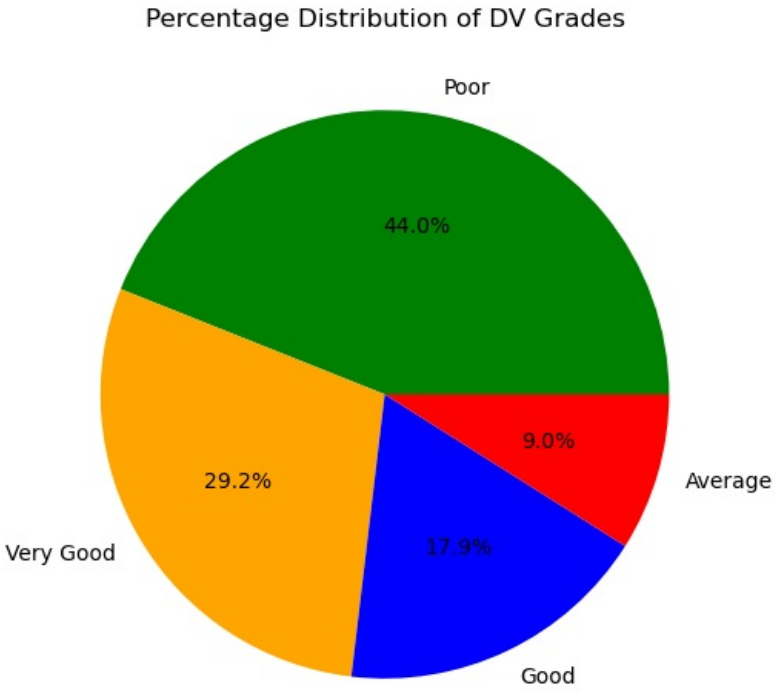
```
Subset of students who scored 20 in any subject:
     S.NO SECTION  DV  M2  PP  BEEE  FL  FIMS  Total  Percentage Grade  \
4       5   ALPHA  18  17  19    19  20    18    111       92.50   NaN
6       7   ALPHA  15  10  20    20  15    14     94       78.33   NaN
7       8   ALPHA  17  17  19    20  19    13    105       87.50   NaN
8       9   ALPHA  10  18   0    20  19    15     82       68.33   NaN
9      10   ALPHA  18  19  20    20  20    15    112       93.33   NaN
..    ...     ...  ..  ..  ..   ...  ..   ...    ...         ...   ...
473   474   SIGMA  20  20  20    20  20    20    120      100.00   NaN
476   477   SIGMA  20   6  16    11  20    14     87       72.50   NaN
477   478   SIGMA  20   0  18    13  20    18     89       74.17   NaN
478   479   SIGMA  20  20   5    19  18    14     96       80.00   NaN
479   480   SIGMA  20  16  18    19  20    19    112       93.33   NaN

     backlogs Coding-skills   PP_Grade   DV_Grade skills section
4           0     very good  Very Good  Very Good    NaN     NaN
6           0     very good  Very Good  Very Good    NaN     NaN
7           0     very good  Very Good  Very Good    NaN     NaN
8           1          poor       poor       Poor    NaN     NaN
9           0     very good  Very Good  Very Good    NaN     NaN
..        ...           ...        ...        ...    ...     ...
473         0     very good  Very Good  Very Good    NaN     NaN
476         1          good       Good       Good    NaN     NaN
477         1     very good  Very Good  Very Good    NaN     NaN
478         1          poor       poor       Poor    NaN     NaN
479         0     very good  Very Good  Very Good    NaN     NaN

[185 rows x 17 columns]
Students who scored 20 in DV: 53
Students who scored 20 in M2: 44
Students who scored 20 in PP: 70
Students who scored 20 in BEEE: 76
Students who scored 20 in FL: 121
Students who scored 20 in FIMS: 12
```

In [186…

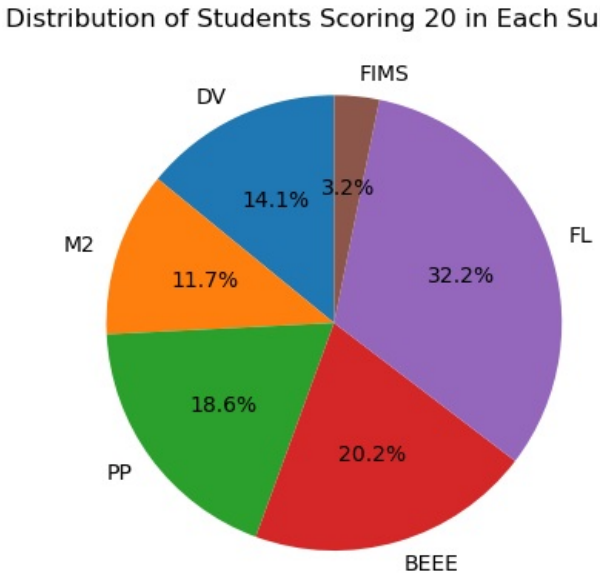
```python
def assign_grade(DV):
    if 18 <= DV <= 20:
        return 'Very Good'
    elif 13 <= DV <= 14:
        return 'Average'
    elif 15 <= DV <= 17:
        return 'Good'
    else:
        return 'Poor'
df['DV_Grade'] = df['PP'].apply(assign_grade)
grade_counts = df['DV_Grade'].value_counts()
plt.figure(figsize=(6, 6))
grade_counts.plot(kind='pie', autopct='%1.1f%%', colors=['green', 'orange', 'blue', 'red'])
plt.title("Percentage Distribution of DV Grades")
plt.ylabel("")
plt.show()
```



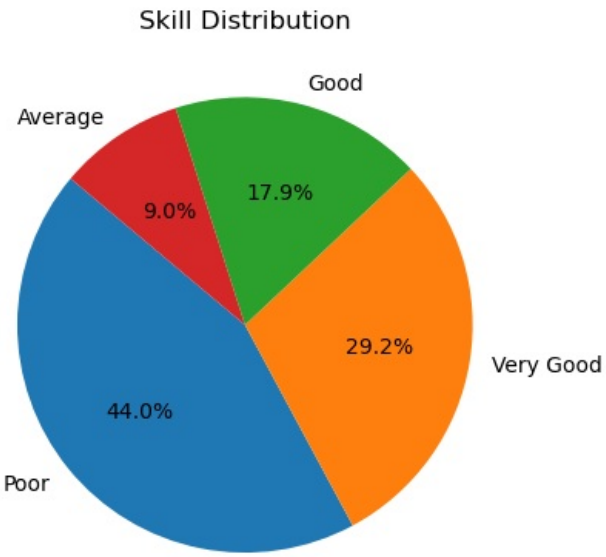
In [191…

```python
subjects = ['DV', 'M2', 'PP', 'BEEE', 'FL', 'FIMS']
```

```
counts = [(df[subject] == 20).sum() for subject in subjects]
plt.pie(counts, labels=subjects, autopct='%1.1f%%', startangle=90)
plt.title('Distribution of Students Scoring 20 in Each Subject')
plt.show()
```



In [193…
```
df['skills'] = df['PP'].apply(assign_grade)
skill_counts = df['skills'].value_counts()
plt.pie(skill_counts, labels=skill_counts.index, autopct='%1.1f%%', startangle=140)
plt.title('Skill Distribution')
plt.show()
df.skills.value_counts()
```



Out[193…
```
skills
Poor         211
Very Good    140
Good          86
Average       43
Name: count, dtype: int64
```

In [195…
```
df.describe()
```

Out[195…

	S.NO	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	backlogs
count	480.000000	480.000000	480.000000	480.000000	480.000000	480.000000	480.000000	480.000000	480.000000	480.00000
mean	240.500000	14.383333	10.093750	12.885417	13.220833	15.535417	13.645833	79.764583	66.470313	1.38125
std	138.708327	4.644674	6.433664	5.874238	5.904351	4.279728	4.716115	24.723878	20.603304	1.50103
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000
25%	120.750000	12.000000	4.000000	9.000000	9.000000	13.000000	11.000000	64.000000	53.330000	0.00000
50%	240.500000	15.500000	10.000000	14.000000	15.000000	15.000000	15.000000	83.000000	69.170000	1.00000
75%	360.250000	18.000000	16.000000	18.000000	18.000000	20.000000	17.000000	98.000000	81.670000	2.00000
max	480.000000	20.000000	20.000000	20.000000	20.000000	20.000000	20.000000	120.000000	100.000000	6.00000

In [197…
```
df["section"] = "alpha"
```

In [199…
```
df[df["SECTION"] == "ALPHA"].mean(numeric_only=True)
```

Out[199…
```
S.NO          30.500000
DV            14.033333
M2            13.733333
PP            16.066667
BEEE          15.616667
FL            16.550000
FIMS          12.850000
Total         88.850000
Percentage    74.041167
backlogs       0.716667
dtype: float64
```

In [201…
```
df.info
```

Out[201…
```
<bound method DataFrame.info of      S.NO SECTION  DV  M2  PP  BEEE  FL  FIMS  Total  Percentage Grade  \
0       1   ALPHA  12   0  17     9  19    15     72       60.00   NaN
1       2   ALPHA  19  12  16    16  18     3     84       70.00   NaN
2       3   ALPHA  18  14  18    18  18    16    102       85.00   NaN
3       4   ALPHA  15   9  19    17  19    15     94       78.33   NaN
4       5   ALPHA  18  17  19    19  20    18    111       92.50   NaN
..    ...     ...  ..  ..  ..   ...  ..   ...    ...         ...   ...
475   476   SIGMA  18   2  12     3  17    15     67       55.83   NaN
476   477   SIGMA  20   6  16    11  20    14     87       72.50   NaN
477   478   SIGMA  20   0  18    13  20    18     89       74.17   NaN
478   479   SIGMA  20  20   5    19  18    14     96       80.00   NaN
479   480   SIGMA  20  16  18    19  20    19    112       93.33   NaN

     backlogs Coding-skills   PP_Grade   DV_Grade     skills section
0           2          good       Good       Good       Good   alpha
1           1          good       Good       Good       Good   alpha
2           0     very good  Very Good  Very Good  Very Good   alpha
3           1     very good  Very Good  Very Good  Very Good   alpha
4           0     very good  Very Good  Very Good  Very Good   alpha
..        ...           ...        ...        ...        ...     ...
475         2          poor       poor       Poor       Poor   alpha
476         1          good       Good       Good       Good   alpha
477         1     very good  Very Good  Very Good  Very Good   alpha
478         1          poor       poor       Poor       Poor   alpha
479         0     very good  Very Good  Very Good  Very Good   alpha

[480 rows x 17 columns]>
```

In [203…
```
df.groupby("SECTION")[df.select_dtypes(include=["number"]).columns].mean()
```

Out[203…

SECTION	S.NO	DV	M2	PP	BEEE	FL	FIMS	Total	Percentage	backlogs
ALPHA	30.500000	14.033333	13.733333	16.066667	15.616667	16.550000	12.850000	88.850000	74.041167	0.716667
BETA	90.500000	12.083333	13.683333	15.666667	12.716667	15.833333	12.983333	82.966667	69.138500	1.150000
DELTA	150.500000	13.483333	11.466667	15.016667	11.050000	15.916667	15.350000	82.283333	68.570000	1.216667
EPSILON	214.816667	14.333333	9.716667	12.750000	9.700000	14.566667	16.116667	77.183333	64.318833	1.416667
GAMMA	270.500000	15.933333	7.616667	9.400000	14.866667	15.716667	13.050000	76.583333	63.818833	1.550000
OMEGA	369.833333	14.600000	8.000000	10.216667	14.900000	15.350000	12.066667	75.133333	62.611000	1.716667
SIGMA	404.069307	15.683168	8.178218	12.534653	13.504950	15.376238	13.603960	78.881188	65.734257	1.564356
ZETA	370.000000	13.263158	8.736842	9.052632	13.210526	14.105263	12.157895	70.526316	58.773158	2.052632

In [205…
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 480 entries, 0 to 479
Data columns (total 17 columns):
 #   Column         Non-Null Count  Dtype
---  ------         --------------  -----
 0   S.NO           480 non-null    int64
 1   SECTION        480 non-null    object
 2   DV             480 non-null    int32
 3   M2             480 non-null    int32
 4   PP             480 non-null    int32
 5   BEEE           480 non-null    int32
 6   FL             480 non-null    int32
 7   FIMS           480 non-null    int32
 8   Total          480 non-null    int32
 9   Percentage     480 non-null    float64
 10  Grade          0 non-null      object
 11  backlogs       480 non-null    int64
 12  Coding-skills  480 non-null    object
 13  PP_Grade       480 non-null    object
 14  DV_Grade       480 non-null    object
 15  skills         480 non-null    object
 16  section        480 non-null    object
dtypes: float64(1), int32(7), int64(2), object(7)
memory usage: 50.8+ KB
```

In [207…
```
from scipy.stats import ttest_ind
```

In [209…
```
df[df['SECTION']=='BETA']['DV']
```

Out[209…
```
60     19
61      8
62     12
63     11
64     12
65      9
66     12
67     12
68     16
69     20
70      4
71     17
72      7
73     10
74     17
75      5
76     17
77     13
78     19
79     19
80     19
81     18
82      2
83     10
84     12
85      3
86     17
87     13
88      2
89     10
90     17
91     14
92     11
93     14
94     12
95     16
96      8
97      8
98      6
99      9
100    10
101    13
102    10
103    11
104    17
105    12
106     9
107    11
108    10
109    13
110     8
111    10
112    16
113    15
114    11
115    20
116    13
117    12
118     9
119    15
Name: DV, dtype: int32
```

In [211…
```
from scipy.stats import ttest_ind
df[df['SECTION'] == 'ALPHA']['DV']
```

Out[211…

```
0     12
1     19
2     18
3     15
4     18
5     17
6     15
7     17
8     10
9     18
10    17
11    20
12    16
13    17
14    19
15    13
16    15
17    11
18    14
19    19
20     4
21    14
22    17
23    20
24    15
25     6
26    17
27     5
28    19
29     8
30    11
31    12
32    17
33    14
34    17
35     8
36    11
37    15
38    19
39    20
40    18
41    16
42    16
43    11
44    18
45    11
46    14
47    16
48    16
49    15
50     1
51     6
52    17
53     8
54    14
55    15
56    10
57     2
58    10
59    19
Name: DV, dtype: int32
```

In [213…

```
df[df['SECTION'] == 'BETA']['DV']
```

Out[213…
```
60     19
61      8
62     12
63     11
64     12
65      9
66     12
67     12
68     16
69     20
70      4
71     17
72      7
73     10
74     17
75      5
76     17
77     13
78     19
79     19
80     19
81     18
82      2
83     10
84     12
85      3
86     17
87     13
88      2
89     10
90     17
91     14
92     11
93     14
94     12
95     16
96      8
97      8
98      6
99      9
100    10
101    13
102    10
103    11
104    17
105    12
106     9
107    11
108    10
109    13
110     8
111    10
112    16
113    15
114    11
115    20
116    13
117    12
118     9
119    15
Name: DV, dtype: int32
```

In [215…
```
ttest_ind(df[df['SECTION'] == 'ALPHA']['DV'] , df[df['SECTION'] == 'BETA']['DV'])
```

Out[215…
```
TtestResult(statistic=2.3418185924318102, pvalue=0.020866453244001094, df=118.0)
```

In [217…
```
from scipy.stats import ttest_rel
ttest_rel(df[df['SECTION'] == 'ALPHA']['DV'] , df[df['SECTION'] == 'BETA']['DV'])
```

Out[217…
```
TtestResult(statistic=2.3172456109384103, pvalue=0.023979527821469917, df=59)
```

In [219…
```
ALPHA_DV = df[df["SECTION"] == "ALPHA"]['DV'].dropna()
BETA_DV = df[df["SECTION"] == "BETA"]['DV'].dropna()
ttest_ind(ALPHA_DV,BETA_DV)
```

Out[219…
```
TtestResult(statistic=2.3418185924318102, pvalue=0.020866453244001094, df=118.0)
```

In [230…
```
from scipy.stats import chi2_contingency
```

In [240…
```
df.DV.mean()
```

Out[240…
```
14.383333333333333
```

In [242…
```
df[df['SECTION']=='ALPHA'].DV.mean()
```

Out[242… 14.033333333333333

In [244…
```
import scipy.stats as stats
```

In [246…
```
t_statistics, p_value = stats.ttest_1samp(df[df['SECTION'] == 'BETA']['PP'], popmean=14.41)
print(t_statistics, p_value)
```

```
1.8778523020441942 0.06534654049350333
```

In [248…
```
t_statistics, p_value = stats.ttest_1samp(df[df['SECTION'] == 'BETA']['DV'], popmean=14.41)
print(t_statistics, p_value)
```

```
-4.035751834264198 0.0001588940914138618
```

In [250…
```
t_statistics, p_value = stats.ttest_1samp(df[df['SECTION'] == 'ALPHA']['DV'] ,df.DV.mean())
print(t_statistics, p_value)
```

```
-0.5825263515793191 0.5624319157350715
```

In [252…
```
t_statistics, p_value = stats.ttest_1samp(df[df['SECTION'] == 'GAMMA']['DV'] ,df.DV.mean())
print(t_statistics, p_value)
```

```
5.436735948684493 1.087970538399695e-06
```

In [254…
```
t_statistics, p_value = stats.ttest_1samp(df[df['SECTION'] == 'DELTA']['DV'] ,df.DV.mean())
print(t_statistics, p_value)
```

```
-1.6332146803454848 0.10774976960815406
```

In [256…
```
t_statistics, p_value = stats.ttest_1samp(df[df['SECTION'] == 'SIGMA']['DV'] ,df.DV.mean())
print(t_statistics, p_value)
```

```
2.6489850039194636 0.009384060119324023
```

In [258…
```
t_statistics, p_value = stats.ttest_1samp(df[df['SECTION'] == 'EPSILON']['DV'] ,df.DV.mean())
print(t_statistics, p_value)
```

```
-0.09486832980504935 0.9247408819314669
```

In [260…
```
sample_alpha = df[df["SECTION"] == "ALPHA"]["DV"]
sample_beta = df[df["SECTION"] == "BETA"]["DV"]
t_statistic, p_value = stats.ttest_ind(sample_alpha, sample_beta)
print("T-statistic:", t_statistic)
print("P-value:", p_value)
```

```
T-statistic: 2.3418185924318102
P-value: 0.020866453244001094
```

In [262…
```
from scipy.stats import chi2_contingency
```

In [264…
```
data = [df[df["SECTION"] == "ALPHA"]['DV'],df[df["SECTION"] == "BETA"]['DV']]
stat, p, dof, expected = chi2_contingency(data)
alpha = 0.05
print("p value is " + str(p))
if p <= alpha:
    print('Dependent (reject H0)')
else:
    print('Independent (H0 holds true)')
```

```
p value is 2.3496708155645757e-05
Dependent (reject H0)
```

DATASET OBSERVATION

The S.NO column has missing values, which might indicate improper data entry.

The SECTION column has some missing values, but it mostly contains section labels (e.g., ALPHA).

Marks columns (DV, M-II, PP, BEEE, FL, FIMS) are stored as objects instead of numerical values, suggesting potential formatting issues (e.g., extra spaces or non-numeric entries).

Most marks columns have minor missing values (only 1-3 missing entries per column).

The dataset likely contains students' marks in multiple subjects, with scores ranging from 0 to 20.

In [ ]:

In [ ]: