In [124…

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.impute import SimpleImputer
```

In [125…

```python
import pandas as pd

def remove_outliers_iqr(df, column, threshold=1.5):
    q1 = df[column].quantile(0.25)
    q3 = df[column].quantile(0.75)
    iqr = q3 - q1

    lower_bound = q1 - threshold * iqr
    upper_bound = q3 + threshold * iqr

    outliers = df[(df[column] < lower_bound) | (df[column] > upper_bound)][column]

    df.drop(outliers.index, inplace=True)
```

In [126…

```python
def box_plot(dataframe, column):

    sns.set(style="whitegrid")
    plt.figure(figsize=(8, 6))
    sns.boxplot(x=dataframe[column])
    plt.title('Box Plot of {}'.format(column))
    plt.ylabel('Values')
    plt.xlabel('Column')
    plt.show()
```

In [127…

```python
import pandas as pd

def mean_inplace(df, column):
    mean_value = df[column].mean()
    df[column].fillna(mean_value, inplace=True)
```

In [128…

```python
academic_performance = pd.read_csv('Academic_Performance.csv')
academic_performance.head()
```

Out[128…

	STUDENT_ID	GENDER	PLACEMENT	HONOR_OPTED_OR_NOT	EDUCATION_TYPE	ACADEMIC_PROGRAM
0	SB11201210000129	F	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING
1	SB11201210000137	F	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING
2	SB11201210005154	M	No	Yes	ACADEMIC	ELECTRONIC ENGINEERING
3	SB11201210007504	F	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING
4	SB11201210007548	M	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING

In [129…

```python
academic_performance.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 12411 entries, 0 to 12410
Data columns (total 13 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   STUDENT_ID           12411 non-null  object 
 1   GENDER               12389 non-null  object 
 2   PLACEMENT            12396 non-null  object 
 3   HONOR_OPTED_OR_NOT   12397 non-null  object 
 4   EDUCATION_TYPE       12396 non-null  object 
 5   ACADEMIC_PROGRAM     12377 non-null  object 
 6   COURSE  1 MARKS      12400 non-null  float64
 7   COURSE  2 MARKS      12403 non-null  float64
 8   COURSE  3 MARKS      12397 non-null  float64
 9   COURSE  4 MARKS      12397 non-null  float64
 10  COURSE  5 MARKS      12389 non-null  float64
 11  PERCENTILE           12411 non-null  int64  
 12  OVEARLL_GRADE        12411 non-null  object 
dtypes: float64(5), int64(1), object(7)
memory usage: 1.2+ MB
```

In [130…

```
#Missing Value Percentage of Each Column
for column in academic_performance.columns:
    missing_percentage = (academic_performance[column].isnull().sum() / len(academic_performanc
    print(f"Percentage of missing values in {column}: {missing_percentage:.2f}%")
```

```
Percentage of missing values in STUDENT_ID: 0.00%
Percentage of missing values in GENDER: 0.18%
Percentage of missing values in PLACEMENT: 0.12%
Percentage of missing values in HONOR_OPTED_OR_NOT: 0.11%
Percentage of missing values in EDUCATION_TYPE: 0.12%
Percentage of missing values in ACADEMIC_PROGRAM: 0.27%
Percentage of missing values in COURSE  1 MARKS: 0.09%
Percentage of missing values in COURSE  2 MARKS: 0.06%
Percentage of missing values in COURSE  3 MARKS: 0.11%
Percentage of missing values in COURSE  4 MARKS: 0.11%
Percentage of missing values in COURSE  5 MARKS: 0.18%
Percentage of missing values in PERCENTILE: 0.00%
Percentage of missing values in OVEARLL_GRADE: 0.00%
```

In [131…

```
#Dealing With Missing Gender
imputer = SimpleImputer(strategy='most_frequent')
imputer.fit(academic_performance[['GENDER']])
academic_performance['GENDER'] = imputer.transform(academic_performance[['GENDER']])
```

In [132…

```
#Dealing with Missing Placement
academic_performance.dropna(subset=['PLACEMENT'], inplace=True)
```

In [133…

```
#Dealing with HONOR
imputer = SimpleImputer(strategy='most_frequent')
imputer.fit(academic_performance[['HONOR_OPTED_OR_NOT']])
academic_performance['HONOR_OPTED_OR_NOT'] = imputer.transform(academic_performance[['HONOR_OPT
```

In [134…

```
#Dealing with Education_Type
#Dealing with HONOR
imputer = SimpleImputer(strategy='most_frequent')
imputer.fit(academic_performance[['EDUCATION_TYPE']])
academic_performance['EDUCATION_TYPE'] = imputer.transform(academic_performance[['EDUCATION_TYP
```

In [135…

```
#Dealing with ACADEMIC PROGRAM MISSING
academic_performance.dropna(subset=['ACADEMIC_PROGRAM'], inplace=True)
```

In [136…

```
missing_values = academic_performance.isnull().sum()
print(missing_values)
```

```
STUDENT_ID             0
GENDER                 0
PLACEMENT              0
HONOR_OPTED_OR_NOT     0
EDUCATION_TYPE         0
ACADEMIC_PROGRAM       0
COURSE 1 MARKS         9
COURSE 2 MARKS         7
COURSE 3 MARKS        14
COURSE 4 MARKS        11
COURSE 5 MARKS        19
PERCENTILE             0
OVEARLL_GRADE          0
dtype: int64
```

In [137…

```
#Remove Outliers
remove_outliers_iqr(academic_performance, 'COURSE 1 MARKS')
remove_outliers_iqr(academic_performance, 'COURSE 2 MARKS')
remove_outliers_iqr(academic_performance, 'COURSE 3 MARKS')
remove_outliers_iqr(academic_performance, 'COURSE 4 MARKS')
remove_outliers_iqr(academic_performance, 'COURSE 5 MARKS')
remove_outliers_iqr(academic_performance, 'PERCENTILE')
```

In [138…

```
#Remove Fill with NULL Values
mean_inplace(academic_performance, 'COURSE 1 MARKS')
mean_inplace(academic_performance, 'COURSE 2 MARKS')
mean_inplace(academic_performance, 'COURSE 3 MARKS')
mean_inplace(academic_performance, 'COURSE 4 MARKS')
mean_inplace(academic_performance, 'COURSE 5 MARKS')
```

In [139…

```
academic_performance
```

Out[139…

	STUDENT_ID	GENDER	PLACEMENT	HONOR_OPTED_OR_NOT	EDUCATION_TYPE	ACADEMIC_PROGRA
0	SB11201210000129	F	Yes	Yes	ACADEMIC	INDUSTR ENGINEERI
1	SB11201210000137	F	Yes	Yes	ACADEMIC	INDUSTR ENGINEERI
3	SB11201210007504	F	Yes	Yes	ACADEMIC	INDUSTR ENGINEERI
4	SB11201210007548	M	Yes	Yes	ACADEMIC	INDUSTR ENGINEERI
5	SB11201210007568	F	Yes	Yes	ACADEMIC	INDUSTR ENGINEERI
...	...	...	...	...	...	
12405	SB11201420565781	M	Yes	Yes	ACADEMIC	INDUSTR ENGINEERI
12406	SB11201420568705	M	Yes	Yes	ACADEMIC	MECHATRON ENGINEERI

	STUDENT_ID	GENDER	PLACEMENT	HONOR_OPTED_OR_NOT	EDUCATION_TYPE	ACADEMIC_PROGRA
12407	SB11201420573045	M	Yes	Yes	ACADEMIC	INDUSTR ENGINEERI
12408	SB11201420578809	M	Yes	No	ACADEMIC	INDUSTR ENGINEERI
12410	SB11201420583232	M	No	No	ACADEMIC	INDUSTR ENGINEERI

12071 rows × 13 columns

In [140…

```
performance_categorical = academic_performance.select_dtypes(exclude=[np.number])
```

In [141…

```
performance_categorical = performance_categorical.drop('STUDENT_ID',axis=1)
performance_categorical
```

Out[141…

	GENDER	PLACEMENT	HONOR_OPTED_OR_NOT	EDUCATION_TYPE	ACADEMIC_PROGRAM	OVEARLL_GRAD
0	F	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING	FIRST CLAS
1	F	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING	THIRD CLAS
3	F	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING	FIRST CLAS
4	M	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING	FIRST CLAS
5	F	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING	FIRST CLAS
...	...	...	...	...	...	
12405	M	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING	FIRST CLAS
12406	M	Yes	Yes	ACADEMIC	MECHATRONICS ENGINEERING	FIRST CLAS
12407	M	Yes	Yes	ACADEMIC	INDUSTRIAL ENGINEERING	FIRST CLAS
12408	M	Yes	No	ACADEMIC	INDUSTRIAL ENGINEERING	FIRST CLAS
12410	M	No	No	ACADEMIC	INDUSTRIAL ENGINEERING	THIRD CLAS

12071 rows × 6 columns

In [142…

```
performance_categorical.          EDUCATION_TYPE.value_counts()
```

Out[142…

```
ACADEMIC             7645
TECHNICAL/ACADEMIC   3395
TECHNICAL            1027
Not apply               4
Name: EDUCATION_TYPE, dtype: int64
```

In [143…

```python
performance_categorical.PLACEMENT.replace({"Yes":1, "No":-1}, inplace= True)
performance_categorical.GENDER.replace({"M":1, "F":-1}, inplace= True)
performance_categorical.HONOR_OPTED_OR_NOT.replace({"Yes":1, "No":-1}, inplace= True)
```

In [144…

```python
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
performance_categorical['OVEARLL_GRADE']=label_encoder.fit_transform(performance_categorical['O
```

In [153…

```python
from sklearn.preprocessing import OneHotEncoder

enc = OneHotEncoder(sparse=False)

encoded_column = enc.fit_transform(performance_categorical[['EDUCATION_TYPE']].values)

categories = enc.categories_[0]
new_columns = ['EDUCATION_TYPE_' + category for category in categories]

encoded_df = pd.DataFrame(encoded_column, columns=new_columns)
performance_categorical_encoded = pd.concat([performance_categorical.reset_index(drop=True), er
```

```
C:\Users\parth\anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:975: FutureWarnin
g: `sparse` was renamed to `sparse_output` in version 1.2 and will be removed in 1.4. `sparse_o
utput` is ignored unless you leave `sparse` to its default value.
  warnings.warn(
```

In [154…

```python
performance_categorical_encoded
```

Out[154…

	GENDER	PLACEMENT	HONOR_OPTED_OR_NOT	EDUCATION_TYPE	ACADEMIC_PROGRAM	OVEARLL_GRAD
0	-1	1	1	ACADEMIC	INDUSTRIAL ENGINEERING	
1	-1	1	1	ACADEMIC	INDUSTRIAL ENGINEERING	
2	-1	1	1	ACADEMIC	INDUSTRIAL ENGINEERING	
3	1	1	1	ACADEMIC	INDUSTRIAL ENGINEERING	
4	-1	1	1	ACADEMIC	INDUSTRIAL ENGINEERING	
...	...	...	...	...	...	
12066	1	1	1	ACADEMIC	INDUSTRIAL ENGINEERING	
12067	1	1	1	ACADEMIC	MECHATRONICS ENGINEERING	
12068	1	1	1	ACADEMIC	INDUSTRIAL ENGINEERING	
12069	1	1	-1	ACADEMIC	INDUSTRIAL ENGINEERING	
12070	1	-1	-1	ACADEMIC	INDUSTRIAL ENGINEERING	

12071 rows × 10 columns

a) Scan all variables for missing values and inconsistencies. If there are missing values and/or inconsistencies, use any of the suitable techniques to deal with them:-

Missing values below 10% were dropped for 'Placement' and 'Academic Program'. 'Gender', 'Honors', and 'Education Type' missing values were imputed with the most frequent strategy, while outliers in 'Course Marks' were removed and imputed with the mean to preserve data integrity.

b)Scan all numeric variables for outliers. If there are outliers, use any of the suitable techniques to deal with them. :-

IQR method was utilized to detect and manage outliers in the 'Courses' columns of the dataset. It identifies values lying beyond the interquartile range and allows for their appropriate treatment or removal, ensuring data integrity and robust analysis.

c) Apply data transformations on categorical variables to convert it into numerical variables. :-

Implemented a replacing strategy to transform binary category variables into numerical ones (Gender, Placement, etc), while leveraging label encoding for hierarchical Grade columns to maintain their inherent order. Additionally, applied one-hot encoding to columns with non-hierarchical, multiple categories, ensuring a suitable representation for classification purposes (Education Type, Academic Program etc).