Open in Colab

In [1]:
```
import pandas as pd
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
```

In [2]:
```
# Construct the raw GitHub URLs
url1 = "https://github.com/Kartavya-Jharwal/Kartavya_Business_Analytics2025/raw/mai
url2 = "https://github.com/Kartavya-Jharwal/Kartavya_Business_Analytics2025/raw/mai
url3 = "https://github.com/Kartavya-Jharwal/Kartavya_Business_Analytics2025/raw/mai
```

In [3]:
```
# Load the data into pandas DataFrames directly from the URLs
merge_data1_df = pd.read_excel(url1)
merge_data2_df = pd.read_excel(url2)
dirty_df = pd.read_excel(url3)
```

In [4]:
```
print("\nDataFrame Main:")
display(dirty_df.head())

print("\nDataFrame Info:")
dirty_df.info()

print("\nUnique values in DataFrame Dirty:")
for column in dirty_df.columns:
    unique_values = dirty_df[column].unique()
    print(f"\nColumn: {column}")
    print(f"Number of unique values: {len(unique_values)}")
    print(f"Unique values: {unique_values}")
```

```
DataFrame Main:
```

	ID	Gender	Age	Class	Major	Grad Intention	GPA
0	S1	Female	20.0	Junior	Other	Yes	2.88
1	S2	Male	23.0	Senior	Management	Yes	3.60
2	S3	Male	21.0	Junior	Other	Yes	2.50
3	S4	Male	21.0	Junior	CIS	Yes	2.50
4	S5	Mail	23.0	Senior	Other	Undecided	2.80

```
DataFrame Info:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 63 entries, 0 to 62
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   ID              63 non-null     object
 1   Gender          62 non-null     object
 2   Age             61 non-null     float64
 3   Class           63 non-null     object
 4   Major           63 non-null     object
 5   Grad Intention  63 non-null     object
 6   GPA             60 non-null     float64
dtypes: float64(2), object(5)
memory usage: 3.6+ KB

Unique values in DataFrame Dirty:

Column: ID
Number of unique values: 62
Unique values: ['S1' 'S2' 'S3' 'S4' 'S5' 'S6' 'S7' 'S8' 'S9' 'S10' 'S11' 'S12' 'S13'
 'S14' 'S15' 'S16' 'S17' 'S18' 'S19' 'S20' 'S21' 'S22' 'S23' 'S24' 'S25'
 'S26' 'S27' 'S28' 'S29' 'S30' 'S31' 'S32' 'S33' 'S34' 'S35' 'S36' 'S37'
 'S38' 'S39' 'S40' 'S41' 'S42' 'S43' 'S44' 'S45' 'S46' 'S47' 'S48' 'S49'
 'S50' 'S51' 'S52' 'S53' 'S54' 'S55' 'S56' 'S57' 'S58' 'S59' 'S60' 'S61'
 'S62']

Column: Gender
Number of unique values: 4
Unique values: ['Female' 'Male' 'Mail' nan]

Column: Age
Number of unique values: 9
Unique values: [20. 23. 21. 22. 24. 19. 18. 26. nan]

Column: Class
Number of unique values: 5
Unique values: ['Junior' 'Senior' 'Sophmore' 'Sophomore' 'Management']

Column: Major
Number of unique values: 9
Unique values: ['Other' 'Management' 'CIS' 'Economics/Finance' 'Economics/ Finance'
 'Undecided' 'International Business' 'Retailing/Marketing' 'Accounting']

Column: Grad Intention
Number of unique values: 6
Unique values: ['Yes' 'Undecided' 'No' 'Undecide' 'Undeicded' 'Accounting']

Column: GPA
Number of unique values: 26
Unique values: [ 2.88  3.6   2.5   2.8   2.34  3.    3.1   3.3   3.5   3.4   3.2
3.68
  3.17  3.19  2.95  2.6   2.9   3.33  2.57   nan  3.05  2.75  3.82 -3.
  7.3   2.4 ]
```

In [5]:
```python
# 1. Clean 'Gender' column
# Replace 'Mail' with 'Male'
dirty_df['Gender'] = dirty_df['Gender'].replace('Mail', 'Male')

# Check unique values after cleaning
print("\nUnique values in 'Gender' after cleaning:")
print(dirty_df['Gender'].unique())
```

```
Unique values in 'Gender' after cleaning:
['Female' 'Male' nan]
```

In [6]:
```python
# 2. Clean 'Class' column
# Replace 'Sophmore' with 'Sophomore' and 'Management' with a more appropriate cate
# Based on the context, 'Management' seems like an outlier or incorrect entry in th
# For now, let's replace it with NaN and consider imputation later if needed.
dirty_df['Class'] = dirty_df['Class'].replace({'Sophmore': 'Sophomore', 'Management

# Check unique values after cleaning
print("\nUnique values in 'Class' after cleaning:")
print(dirty_df['Class'].unique())
```

```
Unique values in 'Class' after cleaning:
['Junior' 'Senior' 'Sophomore' nan]
```

In [7]:
```python
# 3. Clean 'Major' column
# Correct the inconsistency in 'Economics/Finance'
dirty_df['Major'] = dirty_df['Major'].replace('Economics/ Finance', 'Economics/Fina

# Check unique values after cleaning
print("\nUnique values in 'Major' after cleaning:")
print(dirty_df['Major'].unique())
```

```
Unique values in 'Major' after cleaning:
['Other' 'Management' 'CIS' 'Economics/Finance' 'Undecided'
 'International Business' 'Retailing/Marketing' 'Accounting']
```

In [8]:
```python
# 4. Clean 'Grad Intention' column
# Correct inconsistencies
dirty_df['Grad Intention'] = dirty_df['Grad Intention'].replace({'Undecide': 'Undec

# Check unique values after cleaning
print("\nUnique values in 'Grad Intention' after cleaning:")
print(dirty_df['Grad Intention'].unique())
```

```
Unique values in 'Grad Intention' after cleaning:
['Yes' 'Undecided' 'No' nan]
```

In [9]:
```python
# 5. Handle missing values in 'Age'
# For 'Age', we can fill missing values with the median age
dirty_df['Age'] = dirty_df['Age'].fillna(dirty_df['Age'].median())

# Check unique values after cleaning
print("\nUnique values in 'Age' after cleaning:")
print(dirty_df['Age'].unique())
```

```
Unique values in 'Age' after cleaning:
[20. 23. 21. 22. 24. 19. 18. 26.]
```

In [11]:
```
# Display the cleaned data info and unique values for all columns to verify the cha
print("\nDataFrame Info after cleaning:")
dirty_df.info()

print("\nUnique values in DataFrame after cleaning:")
for column in dirty_df.columns:
    unique_values = dirty_df[column].unique()
    print(f"\nColumn: {column}")
    print(f"Number of unique values: {len(unique_values)}")
    print(f"Unique values: {unique_values}")
```

```
DataFrame Info after cleaning:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 63 entries, 0 to 62
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   ID              63 non-null     object 
 1   Gender          62 non-null     object 
 2   Age             63 non-null     float64
 3   Class           62 non-null     object 
 4   Major           63 non-null     object 
 5   Grad Intention  62 non-null     object 
 6   GPA             63 non-null     float64
dtypes: float64(2), object(5)
memory usage: 3.6+ KB

Unique values in DataFrame after cleaning:

Column: ID
Number of unique values: 62
Unique values: ['S1' 'S2' 'S3' 'S4' 'S5' 'S6' 'S7' 'S8' 'S9' 'S10' 'S11' 'S12' 'S13'
 'S14' 'S15' 'S16' 'S17' 'S18' 'S19' 'S20' 'S21' 'S22' 'S23' 'S24' 'S25'
 'S26' 'S27' 'S28' 'S29' 'S30' 'S31' 'S32' 'S33' 'S34' 'S35' 'S36' 'S37'
 'S38' 'S39' 'S40' 'S41' 'S42' 'S43' 'S44' 'S45' 'S46' 'S47' 'S48' 'S49'
 'S50' 'S51' 'S52' 'S53' 'S54' 'S55' 'S56' 'S57' 'S58' 'S59' 'S60' 'S61'
 'S62']

Column: Gender
Number of unique values: 3
Unique values: ['Female' 'Male' nan]

Column: Age
Number of unique values: 8
Unique values: [20. 23. 21. 22. 24. 19. 18. 26.]

Column: Class
Number of unique values: 4
Unique values: ['Junior' 'Senior' 'Sophomore' nan]

Column: Major
Number of unique values: 8
Unique values: ['Other' 'Management' 'CIS' 'Economics/Finance' 'Undecided'
 'International Business' 'Retailing/Marketing' 'Accounting']

Column: Grad Intention
Number of unique values: 4
Unique values: ['Yes' 'Undecided' 'No' nan]

Column: GPA
Number of unique values: 23
Unique values: [2.88 3.6  2.5  2.8  2.34 3.   3.1  3.3  3.5  3.4  3.2  3.68 3.17 3.1
9
 2.95 2.6  2.9  3.33 2.57 3.05 2.75 3.82 2.4 ]
```

Display missing values

In [13]:
```python
# Check for missing values in dirty_df
print("\nMissing values in dirty_df after cleaning:")
print(dirty_df.isnull().sum())
dirty_df.info()
```

```
Missing values in dirty_df after cleaning:
ID                0
Gender            1
Age               0
Class             1
Major             0
Grad Intention    1
GPA               0
dtype: int64
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 63 entries, 0 to 62
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   ID              63 non-null     object 
 1   Gender          62 non-null     object 
 2   Age             63 non-null     float64
 3   Class           62 non-null     object 
 4   Major           63 non-null     object 
 5   Grad Intention  62 non-null     object 
 6   GPA             63 non-null     float64
dtypes: float64(2), object(5)
memory usage: 3.6+ KB
```

In [10]:
```python
# Handle missing values and outliers in 'GPA'
# For 'GPA', we can fill missing values with the median GPA. Also, address potentia
# Let's replace values outside a reasonable range (e.g., 0-4) with NaN and then imp
dirty_df['GPA'] = dirty_df['GPA'].fillna(dirty_df['GPA'].median())
dirty_df['GPA'] = dirty_df['GPA'].apply(lambda x: x if 0 <= x <= 4 else np.nan)

# Check unique values after cleaning
print("\nMissing values in dirty_df after filling Age and GPA:")
print(dirty_df.isnull().sum())
print("\nUnique values in 'GPA' after cleaning:")
print(dirty_df['GPA'].unique())
```

```
Unique values in 'GPA' after cleaning:
[2.88 3.6  2.5  2.8  2.34 3.   3.1  3.3  3.5  3.4  3.2  3.68 3.17 3.19
 2.95 2.6  2.9  3.33 2.57 3.05 2.75 3.82 2.4 ]
```

In [16]:
```python
# Fill missing values in object type columns with the mode
for column in ['Gender', 'Class', 'Grad Intention']:
    if dirty_df[column].isnull().any():
        mode_value = dirty_df[column].mode()[0]
        dirty_df[column] = dirty_df[column].fillna(mode_value)

# Check for missing values again to confirm
print("\nMissing values in dirty_df after filling object columns:")
print(dirty_df.isnull().sum())
```

```
Missing values in dirty_df after filling object columns:
ID                 0
Gender             0
Age                0
Class              0
Major              0
Grad Intention     0
GPA                0
dtype: int64
```

In [18]:

```python
# Check for duplicate rows
duplicate_rows = dirty_df.duplicated().sum()
print(f"\nNumber of duplicate rows before dropping: {duplicate_rows}")

# Drop duplicate rows
dirty_df = dirty_df.drop_duplicates()

# Check for duplicate rows after dropping
duplicate_rows_after = dirty_df.duplicated().sum()
print(f"Number of duplicate rows after dropping: {duplicate_rows_after}")

# Display the info and head of the dataframe after dropping duplicates
print("\nDataFrame Info after dropping duplicates:")
dirty_df.info()
print("\nDataFrame head after dropping duplicates:")
display(dirty_df.head())
```

```
Number of duplicate rows before dropping: 0
Number of duplicate rows after dropping: 0

DataFrame Info after dropping duplicates:
<class 'pandas.core.frame.DataFrame'>
Index: 62 entries, 0 to 62
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   ID              62 non-null     object
 1   Gender          62 non-null     object
 2   Age             62 non-null     float64
 3   Class           62 non-null     object
 4   Major           62 non-null     object
 5   Grad Intention  62 non-null     object
 6   GPA             62 non-null     float64
dtypes: float64(2), object(5)
memory usage: 3.9+ KB

DataFrame head after dropping duplicates:
```

	ID	Gender	Age	Class	Major	Grad Intention	GPA
0	S1	Female	20.0	Junior	Other	Yes	2.88
1	S2	Male	23.0	Senior	Management	Yes	3.60
2	S3	Male	21.0	Junior	Other	Yes	2.50
3	S4	Male	21.0	Junior	CIS	Yes	2.50
4	S5	Male	23.0	Senior	Other	Undecided	2.80

In [19]:

```
# Handle outliers using IQR
numerical_cols = ['Age', 'GPA']

for col in numerical_cols:
    Q1 = dirty_df[col].quantile(0.25)
    Q3 = dirty_df[col].quantile(0.75)
    IQR = Q3 - Q1

    lower_bound = Q1 - 1.5 * IQR
    upper_bound = Q3 + 1.5 * IQR

    # Identify outliers
    outliers = dirty_df[(dirty_df[col] < lower_bound) | (dirty_df[col] > upper_boun
    print(f"\nOutliers in '{col}' column:")
    print(outliers)

    # Option 1: Remove outliers (uncomment the following line to remove)
    # dirty_df = dirty_df[(dirty_df[col] >= lower_bound) & (dirty_df[col] <= upper_

    # Option 2: Cap outliers (replace outliers with the bounds - uncomment the foll
    dirty_df[col] = dirty_df[col].clip(lower=lower_bound, upper=upper_bound)


print("\nDataFrame info after handling outliers:")
dirty_df.info()
print("\nDataFrame head after handling outliers:")
display(dirty_df.head())
```

```
Outliers in 'Age' column:
     ID  Gender   Age   Class       Major Grad Intention  GPA
35  S36  Female  26.0  Junior  Accounting            Yes  3.1

Outliers in 'GPA' column:
Empty DataFrame
Columns: [ID, Gender, Age, Class, Major, Grad Intention, GPA]
Index: []

DataFrame info after handling outliers:
<class 'pandas.core.frame.DataFrame'>
Index: 62 entries, 0 to 62
Data columns (total 7 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   ID              62 non-null     object 
 1   Gender          62 non-null     object 
 2   Age             62 non-null     float64
 3   Class           62 non-null     object 
 4   Major           62 non-null     object 
 5   Grad Intention  62 non-null     object 
 6   GPA             62 non-null     float64
dtypes: float64(2), object(5)
memory usage: 3.9+ KB

DataFrame head after handling outliers:
```

	ID	Gender	Age	Class	Major	Grad Intention	GPA
0	S1	Female	20.0	Junior	Other	Yes	2.88
1	S2	Male	23.0	Senior	Management	Yes	3.60
2	S3	Male	21.0	Junior	Other	Yes	2.50
3	S4	Male	21.0	Junior	CIS	Yes	2.50
4	S5	Male	23.0	Senior	Other	Undecided	2.80

In [21]:

```python
# Perform a horizontal merge of dirty_df and merge_data1_df on the 'ID' column
final_merged_df = pd.merge(dirty_df, merge_data1_df, on='ID', how='inner')

# Display the head and info of the final merged dataframe
print("\nDataFrame Info after merging dirty_df and merge_data1_df:")
final_merged_df.info()
print("\nDataFrame head after merging dirty_df and merge_data1_df:")
display(final_merged_df.head())
```

```
DataFrame Info after merging dirty_df and merge_data1_df:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62 entries, 0 to 61
Data columns (total 8 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   ID                        62 non-null     object 
 1   Gender                    62 non-null     object 
 2   Age                       62 non-null     float64
 3   Class                     62 non-null     object 
 4   Major                     62 non-null     object 
 5   Grad Intention            62 non-null     object 
 6   GPA                       62 non-null     float64
 7   Average Household Income  62 non-null     int64  
dtypes: float64(2), int64(1), object(5)
memory usage: 4.0+ KB

DataFrame head after merging dirty_df and merge_data1_df:
```

	ID	Gender	Age	Class	Major	Grad Intention	GPA	Average Household Income
0	S1	Female	20.0	Junior	Other	Yes	2.88	64976
1	S2	Male	23.0	Senior	Management	Yes	3.60	56240
2	S3	Male	21.0	Junior	Other	Yes	2.50	50466
3	S4	Male	21.0	Junior	CIS	Yes	2.50	114406
4	S5	Male	23.0	Senior	Other	Undecided	2.80	93589

In [22]:

```
# Perform a vertical merge by concatenating final_merged_df and merge_data2_df
final_merged_df = pd.concat([final_merged_df, merge_data2_df], ignore_index=True)

# Display the head and info of the final merged dataframe after vertical merge
print("\nDataFrame Info after vertical merge with merge_data2_df:")
final_merged_df.info()
print("\nDataFrame head after vertical merge with merge_data2_df:")
display(final_merged_df.head())
```

```
DataFrame Info after vertical merge with merge_data2_df:
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65 entries, 0 to 64
Data columns (total 8 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   ID                        65 non-null     object 
 1   Gender                    65 non-null     object 
 2   Age                       65 non-null     float64
 3   Class                     65 non-null     object 
 4   Major                     65 non-null     object 
 5   Grad Intention            65 non-null     object 
 6   GPA                       65 non-null     float64
 7   Average Household Income  62 non-null     float64
dtypes: float64(3), object(5)
memory usage: 4.2+ KB

DataFrame head after vertical merge with merge_data2_df:
```

	ID	Gender	Age	Class	Major	Grad Intention	GPA	Average Household Income
0	S1	Female	20.0	Junior	Other	Yes	2.88	64976.0
1	S2	Male	23.0	Senior	Management	Yes	3.60	56240.0
2	S3	Male	21.0	Junior	Other	Yes	2.50	50466.0
3	S4	Male	21.0	Junior	CIS	Yes	2.50	114406.0
4	S5	Male	23.0	Senior	Other	Undecided	2.80	93589.0

In [24]:

```python
# Perform basic EDA on the final_merged_df

# Display the shape of the dataframe
print("Shape of the final merged DataFrame:")
print(final_merged_df.shape)

# Display the head of the dataframe
print("\nHead of the final merged DataFrame:")
display(final_merged_df.head())

# Display the tail of the dataframe
print("\nTail of the final merged DataFrame:")
display(final_merged_df.tail())
```

```
Shape of the final merged DataFrame:
(65, 8)

Head of the final merged DataFrame:
```

	ID	Gender	Age	Class	Major	Grad Intention	GPA	Average Household Income
0	S1	Female	20.0	Junior	Other	Yes	2.88	64976.0
1	S2	Male	23.0	Senior	Management	Yes	3.60	56240.0
2	S3	Male	21.0	Junior	Other	Yes	2.50	50466.0
3	S4	Male	21.0	Junior	CIS	Yes	2.50	114406.0
4	S5	Male	23.0	Senior	Other	Undecided	2.80	93589.0

```
Tail of the final merged DataFrame:
```

	ID	Gender	Age	Class	Major	Grad Intention	GPA	Average Household Income
60	S61	Female	23.0	Senior	Accounting	Yes	3.5	83765.0
61	S62	Female	23.0	Senior	Economics/Finance	No	3.2	102497.0
62	S63	Female	20.0	Sophomore	CIS	No	2.5	NaN
63	S64	Male	21.0	Junior	Accounting	Yes	3.7	NaN
64	S65	Female	23.0	Senior	Economics/Finance	No	3.2	NaN