Finding Missing Values

Estimated time needed: **30** minutes

Data wrangling is the process of cleaning, transforming, and organizing data to make it suitable for analysis. Finding and handling missing values is a crucial step in this process to ensure data accuracy and completeness. In this lab, you will focus exclusively on identifying and handling missing values in the dataset.

Objectives

After completing this lab, you will be able to:

- Identify missing values in the dataset.
- Quantify missing values for specific columns.
- Impute missing values using various strategies.

Hands on Lab

Setup: Install Required Libraries

In [1]:
```
!pip install pandas
!pip install matplotlib
!pip install seaborn
```

```
Requirement already satisfied: pandas in /opt/conda/lib/python3.12/site-packages (2.3.0)
Requirement already satisfied: numpy>=1.26.0 in /opt/conda/lib/python3.12/site-packages (from panda
s) (2.3.0)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/conda/lib/python3.12/site-packages (fr
om pandas) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.12/site-packages (from pandas)
(2024.2)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.12/site-packages (from panda
s) (2025.2)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-packages (from python-date
util>=2.8.2->pandas) (1.17.0)
Requirement already satisfied: matplotlib in /opt/conda/lib/python3.12/site-packages (3.10.3)
Requirement already satisfied: contourpy>=1.0.1 in /opt/conda/lib/python3.12/site-packages (from mat
plotlib) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.12/site-packages (from matplot
lib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.12/site-packages (from ma
tplotlib) (4.58.4)
Requirement already satisfied: kiwisolver>=1.3.1 in /opt/conda/lib/python3.12/site-packages (from ma
tplotlib) (1.4.8)
Requirement already satisfied: numpy>=1.23 in /opt/conda/lib/python3.12/site-packages (from matplotl
ib) (2.3.0)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.12/site-packages (from matp
lotlib) (24.2)
Requirement already satisfied: pillow>=8 in /opt/conda/lib/python3.12/site-packages (from matplotli
b) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in /opt/conda/lib/python3.12/site-packages (from mat
plotlib) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3.12/site-packages (from
matplotlib) (2.9.0.post0)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-packages (from python-date
util>=2.7->matplotlib) (1.17.0)
Collecting seaborn
  Downloading seaborn-0.13.2-py3-none-any.whl.metadata (5.4 kB)
Requirement already satisfied: numpy!=1.24.0,>=1.20 in /opt/conda/lib/python3.12/site-packages (from
seaborn) (2.3.0)
Requirement already satisfied: pandas>=1.2 in /opt/conda/lib/python3.12/site-packages (from seaborn)
(2.3.0)
Requirement already satisfied: matplotlib!=3.6.1,>=3.4 in /opt/conda/lib/python3.12/site-packages (f
rom seaborn) (3.10.3)
Requirement already satisfied: contourpy>=1.0.1 in /opt/conda/lib/python3.12/site-packages (from mat
plotlib!=3.6.1,>=3.4->seaborn) (1.3.2)
Requirement already satisfied: cycler>=0.10 in /opt/conda/lib/python3.12/site-packages (from matplot
lib!=3.6.1,>=3.4->seaborn) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in /opt/conda/lib/python3.12/site-packages (from ma
tplotlib!=3.6.1,>=3.4->seaborn) (4.58.4)
Requirement already satisfied: kiwisolver>=1.3.1 in /opt/conda/lib/python3.12/site-packages (from ma
tplotlib!=3.6.1,>=3.4->seaborn) (1.4.8)
Requirement already satisfied: packaging>=20.0 in /opt/conda/lib/python3.12/site-packages (from matp
lotlib!=3.6.1,>=3.4->seaborn) (24.2)
Requirement already satisfied: pillow>=8 in /opt/conda/lib/python3.12/site-packages (from matplotli
b!=3.6.1,>=3.4->seaborn) (11.2.1)
Requirement already satisfied: pyparsing>=2.3.1 in /opt/conda/lib/python3.12/site-packages (from mat
plotlib!=3.6.1,>=3.4->seaborn) (3.2.3)
Requirement already satisfied: python-dateutil>=2.7 in /opt/conda/lib/python3.12/site-packages (from
matplotlib!=3.6.1,>=3.4->seaborn) (2.9.0.post0)
Requirement already satisfied: pytz>=2020.1 in /opt/conda/lib/python3.12/site-packages (from pandas>
=1.2->seaborn) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /opt/conda/lib/python3.12/site-packages (from panda
s>=1.2->seaborn) (2025.2)
Requirement already satisfied: six>=1.5 in /opt/conda/lib/python3.12/site-packages (from python-date
util>=2.7->matplotlib!=3.6.1,>=3.4->seaborn) (1.17.0)
Downloading seaborn-0.13.2-py3-none-any.whl (294 kB)
Installing collected packages: seaborn
Successfully installed seaborn-0.13.2
```

Import Necessary Modules:

In [2]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Tasks

1. Load the Dataset

We use the `pandas.read_csv()` function for reading CSV files. However, in this version of the lab, which operates on JupyterLite, the dataset needs to be downloaded to the interface using the provided code below.

The functions below will download the dataset into your browser:

In [3]:
```
# Define the URL of the dataset
file_path = "https://cf-courses-data.s3.us.cloud-object-storage.appdomain.cloud/n01PQ9pSmiRX6520flu

# Load the dataset into a DataFrame
df = pd.read_csv(file_path)

# Display the first few rows to ensure it loaded correctly
print(df.head())
```

```
   ResponseId                      MainBranch                 Age  \
0           1  I am a developer by profession  Under 18 years old
1           2  I am a developer by profession    35-44 years old
2           3  I am a developer by profession    45-54 years old
3           4           I am learning to code    18-24 years old
4           5  I am a developer by profession    18-24 years old

            Employment RemoteWork   Check  \
0  Employed, full-time     Remote  Apples
1  Employed, full-time     Remote  Apples
2  Employed, full-time     Remote  Apples
3   Student, full-time        NaN  Apples
4   Student, full-time        NaN  Apples

                                  CodingActivities  \
0                                            Hobby
1  Hobby;Contribute to open-source projects;Other...
2  Hobby;Contribute to open-source projects;Other...
3                                              NaN
4                                              NaN

                                       EdLevel  \
0                       Primary/elementary school
1      Bachelor's degree (B.A., B.S., B.Eng., etc.)
2    Master's degree (M.A., M.S., M.Eng., MBA, etc.)
3  Some college/university study without earning ...
4  Secondary school (e.g. American high school, G...

                                    LearnCode  \
0                           Books / Physical media
1  Books / Physical media;Colleague;On the job tr...
2  Books / Physical media;Colleague;On the job tr...
3  Other online resources (e.g., videos, blogs, f...
4  Other online resources (e.g., videos, blogs, f...

                                  LearnCodeOnline  ... JobSatPoints_6  \
0                                             NaN  ...            NaN
1  Technical documentation;Blogs;Books;Written Tu...  ...            0.0
2  Technical documentation;Blogs;Books;Written Tu...  ...            NaN
3  Stack Overflow;How-to videos;Interactive tutorial  ...            NaN
4  Technical documentation;Blogs;Written Tutorial...  ...            NaN

  JobSatPoints_7 JobSatPoints_8 JobSatPoints_9 JobSatPoints_10  \
0            NaN            NaN            NaN             NaN
1            0.0            0.0            0.0             0.0
2            NaN            NaN            NaN             NaN
3            NaN            NaN            NaN             NaN
4            NaN            NaN            NaN             NaN

  JobSatPoints_11           SurveyLength SurveyEase ConvertedCompYearly JobSat
0             NaN                    NaN        NaN                 NaN    NaN
1             0.0                    NaN        NaN                 NaN    NaN
2             NaN  Appropriate in length       Easy                 NaN    NaN
3             NaN               Too long       Easy                 NaN    NaN
4             NaN             Too short       Easy                 NaN    NaN

[5 rows x 114 columns]
```

2. Explore the Dataset

Task 1: Display basic information and summary statistics of the dataset.

In [4]:

```
## Write your code here
print("--- Task 1: Display basic information and summary statistics of the dataset ---")

# 1. Display basic information of the dataset (column names, non-null count, data types, memory usa
print("\n--- Basic Information (df.info()) ---")
df.info()

# 2. Display summary statistics of the dataset (count, mean, std, min, 25%, 50%, 75%, max for numer
print("\n--- Summary Statistics (df.describe()) ---")
print(df.describe())
```

```
--- Task 1: Display basic information and summary statistics of the dataset ---

--- Basic Information (df.info()) ---
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 65437 entries, 0 to 65436
Columns: 114 entries, ResponseId to JobSat
dtypes: float64(13), int64(1), object(100)
memory usage: 56.9+ MB

--- Summary Statistics (df.describe()) ---
         ResponseId      CompTotal       WorkExp  JobSatPoints_1  \
count  65437.000000   3.374000e+04  29658.000000    29324.000000
mean   32719.000000  2.963841e+145     11.466957       18.581094
std    18890.179119  5.444117e+147      9.168709       25.966221
min        1.000000   0.000000e+00      0.000000        0.000000
25%    16360.000000   6.000000e+04      4.000000        0.000000
50%    32719.000000   1.100000e+05      9.000000       10.000000
75%    49078.000000   2.500000e+05     16.000000       22.000000
max    65437.000000  1.000000e+150     50.000000      100.000000

       JobSatPoints_4  JobSatPoints_5  JobSatPoints_6  JobSatPoints_7  \
count    29393.000000    29411.000000    29450.000000     29448.00000
mean         7.522140       10.060857       24.343232        22.96522
std         18.422661       21.833836       27.089360        27.01774
min          0.000000        0.000000        0.000000         0.00000
25%          0.000000        0.000000        0.000000         0.00000
50%          0.000000        0.000000       20.000000        15.00000
75%          5.000000       10.000000       30.000000        30.00000
max        100.000000      100.000000      100.000000       100.00000

       JobSatPoints_8  JobSatPoints_9  JobSatPoints_10  JobSatPoints_11  \
count    29456.000000    29456.000000     29450.000000     29445.000000
mean        20.278165       16.169432        10.955713         9.953948
std         26.108110       24.845032        22.906263        21.775652
min          0.000000        0.000000         0.000000         0.000000
25%          0.000000        0.000000         0.000000         0.000000
50%         10.000000        5.000000         0.000000         0.000000
75%         25.000000       20.000000        10.000000        10.000000
max        100.000000      100.000000       100.000000       100.000000

       ConvertedCompYearly        JobSat
count         2.343500e+04  29126.000000
mean          8.615529e+04      6.935041
std           1.867570e+05      2.088259
min           1.000000e+00      0.000000
25%           3.271200e+04      6.000000
50%           6.500000e+04      7.000000
75%           1.079715e+05      8.000000
max           1.625660e+07     10.000000
```

3. Finding Missing Values

Task 2: Identify missing values for all columns.

In [6]:
```python
## Write your code here
print("--- Task 2: Identify missing values for all columns ---")

# Identify missing values for all columns by summing null (NaN) values
missing_values_count = df.isnull().sum()

print("\nNumber of missing values per column:")
print(missing_values_count)
```

```
--- Task 2: Identify missing values for all columns ---

Number of missing values per column:
ResponseId                 0
MainBranch                 0
Age                        0
Employment                 0
RemoteWork             10631
                       ...
JobSatPoints_11        35992
SurveyLength            9255
SurveyEase              9199
ConvertedCompYearly    42002
JobSat                 36311
Length: 114, dtype: int64
```

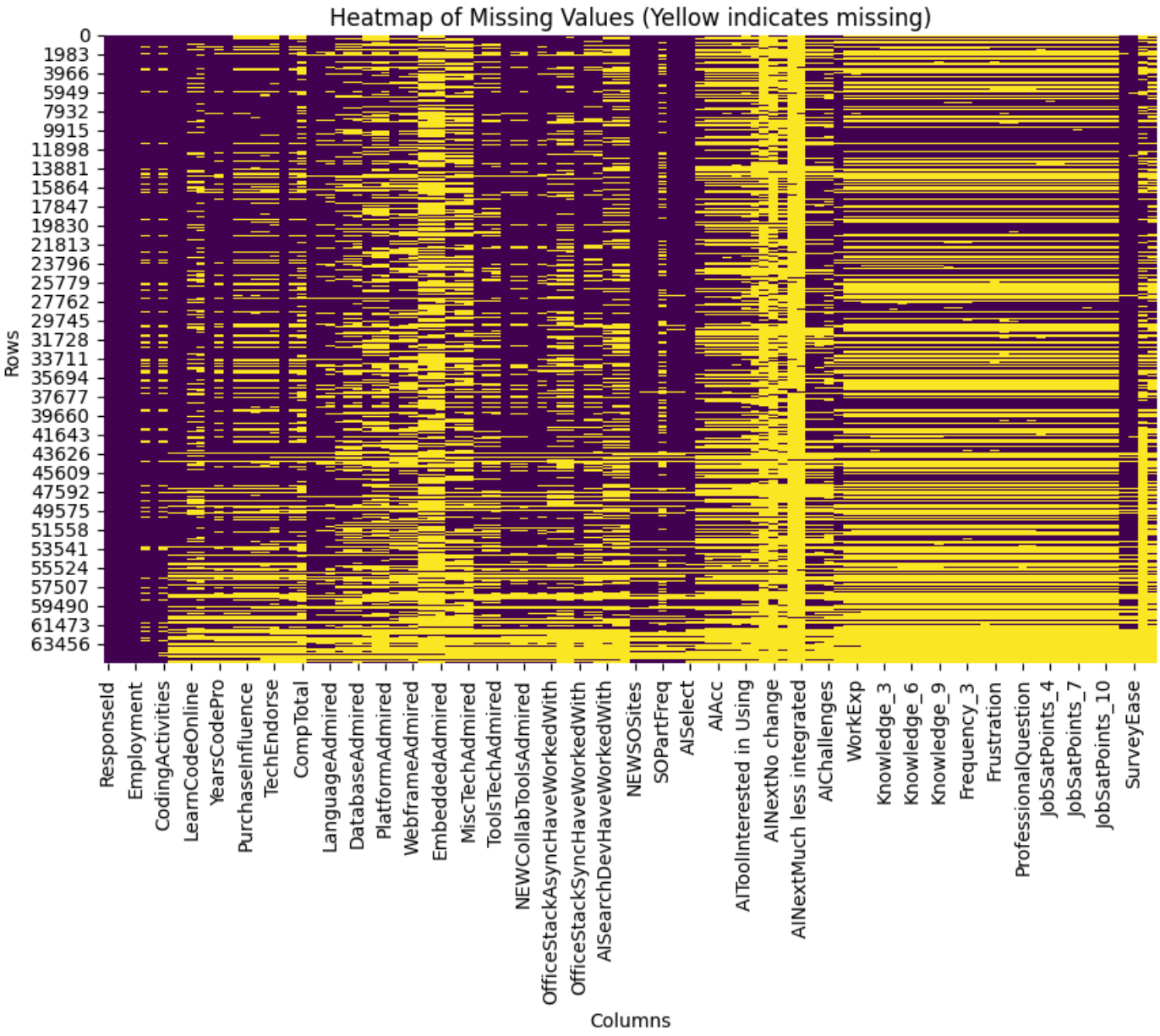
Task 3: Visualize missing values using a heatmap (Using seaborn library).

In [7]:
```python
## Write your code here
print("--- Task 3: Visualize missing values using a heatmap ---")

# Check if there are any missing values at all to decide if a heatmap is useful
if df.isnull().sum().sum() == 0:
    print("\nNo missing values found in the DataFrame. Heatmap will be entirely one color.")
else:
    plt.figure(figsize=(10, 6))
    # Create the heatmap: df.isnull() returns a boolean DataFrame (True for NaN, False otherwise)
    sns.heatmap(df.isnull(), cbar=False, cmap='viridis')
    plt.title('Heatmap of Missing Values (Yellow indicates missing)')
    plt.xlabel('Columns')
    plt.ylabel('Rows')
    plt.show()

print("\nMissing values count per column:")
print(df.isnull().sum())
```

```
--- Task 3: Visualize missing values using a heatmap ---
```




```
Missing values count per column:
ResponseId                 0
MainBranch                 0
Age                        0
Employment                 0
RemoteWork             10631
                       ...
JobSatPoints_11        35992
SurveyLength            9255
SurveyEase              9199
ConvertedCompYearly    42002
JobSat                 36311
Length: 114, dtype: int64
```

Task 4: Count the number of missing rows for a specific column (e.g., `Employment`).

In [8]:

```python
## Write your code here
print("--- Task 4: Count the number of missing rows for a specific column (Employment) ---")

# Count the number of missing values in the 'Employment' column
missing_employment_count = df['Employment'].isnull().sum()

print(f"\nNumber of missing values in the 'Employment' column: {missing_employment_count}")

# Optional: Display the rows with missing 'Employment' values
if missing_employment_count > 0:
    print("\nRows with missing 'Employment' values:")
    print(df[df['Employment'].isnull()])
else:
    print("\nNo missing values found in the 'Employment' column.")
```

```
--- Task 4: Count the number of missing rows for a specific column (Employment) ---

Number of missing values in the 'Employment' column: 0

No missing values found in the 'Employment' column.
```

4. Imputing Missing Values

Task 5: Identify the most frequent (majority) value in a specific column (e.g., `Employment`).

In [9]:
```python
## Write your code here
print("--- Task 5: Identify the most frequent (majority) value in a specific column (Employment) --

# Check if 'Employment' column exists
if 'Employment' in df.columns:
    # Get the most frequent value(s) in the 'Employment' column
    # .mode() returns a Series, so [0] gets the first mode if there are multiple.
    most_frequent_employment = df['Employment'].mode()[0]

    print(f"\nThe most frequent (majority) value in the 'Employment' column is: '{most_frequent_emp

    # Optional: Display all value counts to show distribution and confirm the mode
    print("\nValue counts for the 'Employment' column:")
    print(df['Employment'].value_counts())
else:
    print("Error: 'Employment' column not found in the DataFrame.")
```

```
--- Task 5: Identify the most frequent (majority) value in a specific column (Employment) ---

The most frequent (majority) value in the 'Employment' column is: 'Employed, full-time'

Value counts for the 'Employment' column:
Employment
Employed, full-time
39041
Independent contractor, freelancer, or self-employed
4846
Student, full-time
4709
Employed, full-time;Independent contractor, freelancer, or self-employed
3557
Not employed, but looking for work
2341

...
Not employed, but looking for work;Independent contractor, freelancer, or self-employed;Not employe
d, and not looking for work;Employed, part-time        1
Student, full-time;Retired
1
Employed, full-time;Not employed, but looking for work;Student, part-time
1
Not employed, and not looking for work;Student, part-time;Employed, part-time
1
Not employed, but looking for work;Independent contractor, freelancer, or self-employed;Student, par
t-time;Retired                                        1
Name: count, Length: 110, dtype: int64
```

Task 6: Impute missing values in the `Employment` column with the most frequent value.

In [10]:
```python
## Write your code here
print("--- Task 6: Impute missing values in the Employment column with the most frequent value ---"

# Check current missing values in 'Employment'
initial_missing_count = df['Employment'].isnull().sum()
print(f"\nInitial number of missing values in 'Employment': {initial_missing_count}")

if initial_missing_count > 0:
    # 1. Identify the most frequent value (mode) in the 'Employment' column
    # .mode()[0] is used to get the first mode in case of ties.
    most_frequent_employment = df['Employment'].mode()[0]
    print(f"Most frequent value in 'Employment' column: '{most_frequent_employment}'")

    # 2. Impute missing values in 'Employment' with its most frequent value
    df['Employment'].fillna(most_frequent_employment, inplace=True)
```

```
    print(f"Filled missing values in 'Employment' with '{most_frequent_employment}'.")

    # 3. Verify the imputation
    missing_after_imputation = df['Employment'].isnull().sum()
    print(f"Number of missing values in 'Employment' after imputation: {missing_after_imputation}")

    if missing_after_imputation == 0:
        print("Verification successful: 'Employment' column imputed and no missing values remain.")
    else:
        print("Warning: 'Employment' column still has missing values. Review the imputation logic."
else:
    print("No missing values found in the 'Employment' column. No imputation needed.")

print("\nDataFrame 'Employment' column after Task 6:")
print(df['Employment'])
```

```
--- Task 6: Impute missing values in the Employment column with the most frequent value ---

Initial number of missing values in 'Employment': 0
No missing values found in the 'Employment' column. No imputation needed.

DataFrame 'Employment' column after Task 6:
0        Employed, full-time
1        Employed, full-time
2        Employed, full-time
3         Student, full-time
4         Student, full-time
               ...
65432    Employed, full-time
65433    Employed, full-time
65434    Employed, full-time
65435    Employed, full-time
65436     Student, full-time
Name: Employment, Length: 65437, dtype: object
```

5. Visualizing Imputed Data

Task 7: Visualize the distribution of a column after imputation (e.g., `Employment`).

In [12]:

```
## Write your code here
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Assume your DataFrame 'df' is already loaded and 'Employment' has been imputed.
# For demonstration purposes, let's create a sample DataFrame and impute 'Employment'
# as it would have been done in Task 6.
data = {
    'Respondent': [1, 2, 3, 4, 5, 6, 7],
    'Employment': ['Employed, full-time', 'Student', np.nan, 'Employed, part-time', np.nan, 'Retire
    'Country': ['USA', 'Canada', 'USA', 'Mexico', 'Canada', 'France', 'USA']
}
df = pd.DataFrame(data)

# Impute 'Employment' as per Task 6
if df['Employment'].isnull().any():
    most_frequent_employment = df['Employment'].mode()[0]
    df['Employment'].fillna(most_frequent_employment, inplace=True)
    print(f"Imputed 'Employment' column with '{most_frequent_employment}' for demonstration.")

print("--- Task 7: Visualize the distribution of a column after imputation (Employment) ---")

# Check if 'Employment' column exists and is not empty
if 'Employment' in df.columns and not df['Employment'].empty:
    plt.figure(figsize=(10, 6))
    sns.countplot(y='Employment', data=df, order=df['Employment'].value_counts().index, palette='vi
    plt.title('Distribution of Employment Statuses After Imputation')
    plt.xlabel('Count')
    plt.ylabel('Employment Status')
    plt.tight_layout()
    plt.show()

    print("\nValue counts of 'Employment' after imputation:")
```

```
    print(df['Employment'].value_counts())
else:
    print("The 'Employment' column is not available or is empty. Cannot visualize its distribution.")
```

```
Imputed 'Employment' column with 'Employed, full-time' for demonstration.
--- Task 7: Visualize the distribution of a column after imputation (Employment) ---
```

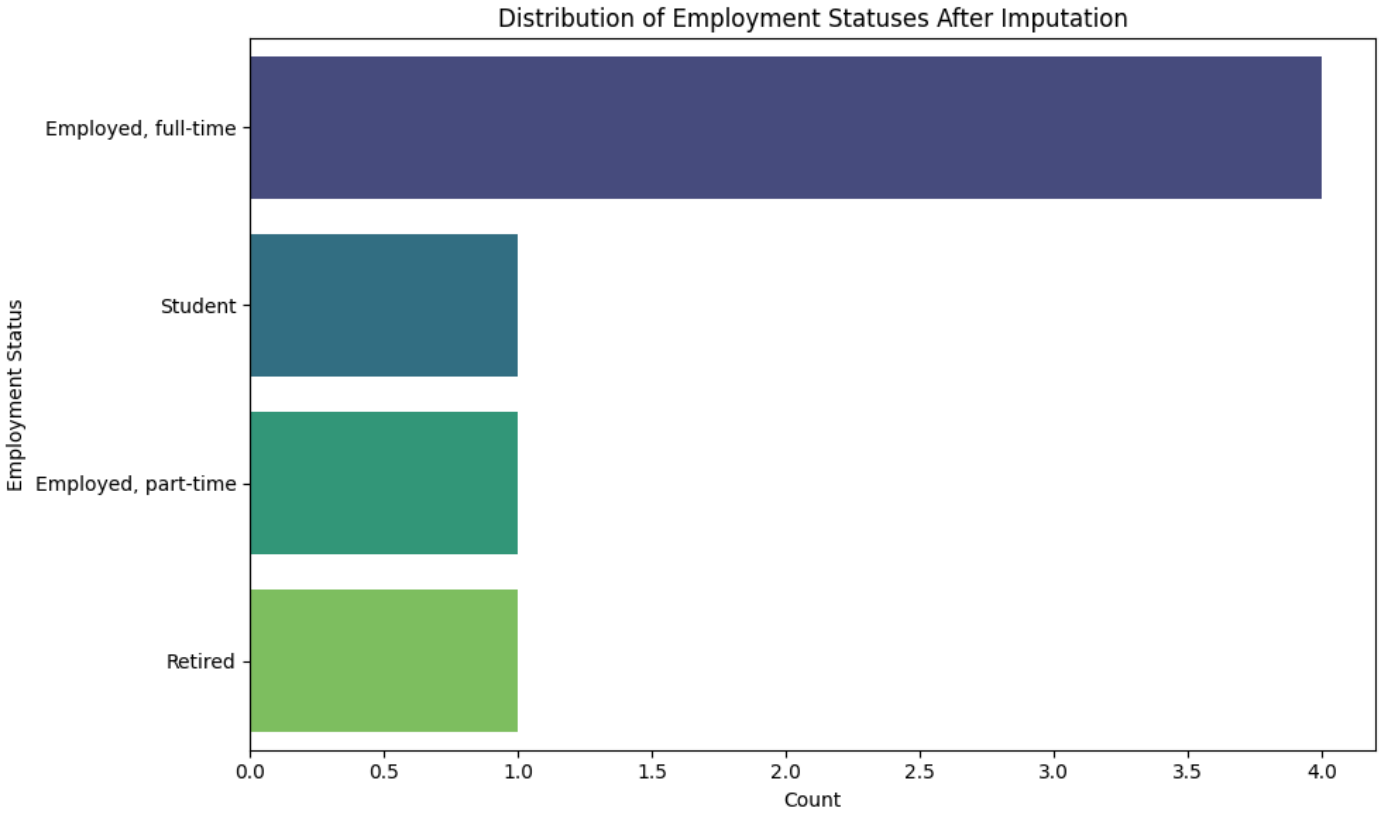
```
/tmp/ipykernel_1252/3584491578.py:20: FutureWarning: A value is trying to be set on a copy of a Data
Frame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate
object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, in
place=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the ori
ginal object.


  df['Employment'].fillna(most_frequent_employment, inplace=True)
/tmp/ipykernel_1252/3584491578.py:28: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in v0.14.0. Assign the `
y` variable to `hue` and set `legend=False` for the same effect.

  sns.countplot(y='Employment', data=df, order=df['Employment'].value_counts().index, palette='virid
is')
```




```
Value counts of 'Employment' after imputation:
Employment
Employed, full-time    4
Student                1
Employed, part-time    1
Retired                1
Name: count, dtype: int64
```

Summary

In this lab, you:

- Loaded the dataset into a pandas DataFrame.
- Identified missing values across all columns.
- Quantified missing values in specific columns.
- Imputed missing values in a categorical column using the most frequent value.
- Visualized the imputed data for better understanding.