AI Data Powered Analysis Early Internship - Sub-Group 7

Week-1 Final Report

Report Title: “Week-1: Data Cleaning and Feature Engineering Report”

Associate Members:

Name	Email
Kolawole Oparinde	kolawole@vempower.org
Akanksha Choudhary	akanksha@vempower.org
Sakshi Sahu	sakshisahu@vempower.org
Shishir Jaiswal	shishir@vempower.org
Shruti Mishra	shrutimishra@vempower.org

Team Members:

Name	Role	Email
Sumaiya Tasnim	Team Lead	sumaiyaa.tasnim.18@gmail.com
Saniya Dantal	Project Scribe	dantalsb04@gmail.com
Zaheer 123	Project Manager	mianzaheer4195@gmail.com
Devadharshini T	Project Lead	dharshinideva83@gmail.com
Peter Macharia	Team Member	mpeter778@gmail.com
Bokka Hamsini	Team Member	bokkahamsini@gmail.com

Introduction :

The first week of the AI Data Powered Analysis Early Internship with Excelerate will focus on building a strong foundation in data science through Python programming, data preprocessing, and feature engineering. The activities will include collecting and cleaning datasets, handling missing values, removing duplicates, and creating new features to capture deeper insights into learner engagement. Feature engineering will involve deriving new variables, transforming existing ones, and extracting trends to enhance the quality and utility of the data. These steps will help analyze engagement patterns, predict drop-offs, identify influencing factors, and optimize program strategies, preparing well-organized datasets for advanced analysis in the following weeks.

Objectives :

- Understand the fundamentals of data science and Python essentials.
- Explore the dataset and identify issues such as missing values, outliers, duplicates, and inconsistencies.
- Perform systematic data cleaning and validation to ensure dataset quality.
- Apply feature engineering techniques to create meaningful new features.
- Document the cleaning and preprocessing process in a structured report.

Assigned Dataset : "SLU_Opportunity_Wise_Data.csv"

Tools We Will Use :

- **Python & Libraries**: Pandas, NumPy, Scikit-learn, re, missingno etc (for cleaning and preprocessing) in Visual studio/Google Colab.
- **Excel**: For basic feature creation, normalization, and categorical encoding.
- **Documentation Tools**: Word/Google Docs for compiling reports.

Expected Outcomes

- ✓ A cleaned and preprocessed dataset free from errors, duplicates, and inconsistencies.
- ✓ Newly engineered features (e.g., age calculation, engagement duration, normalized metrics).
- ✓ A comprehensive Week 1 report outlining dataset structure, cleaning methods, and feature engineering steps.

Learning Outcomes

1. Gain familiarity with dataset exploration and preprocessing workflows.
2. Develop practical skills in Python and Excel for cleaning and transforming data.
3. Strengthen analytical thinking by solving real-world data issues.
4. Improve technical communication through structured reporting.
5. Lay the groundwork for advanced analytics and predictive modeling in subsequent weeks.

Understanding The Dataset

Exploring Dataset:

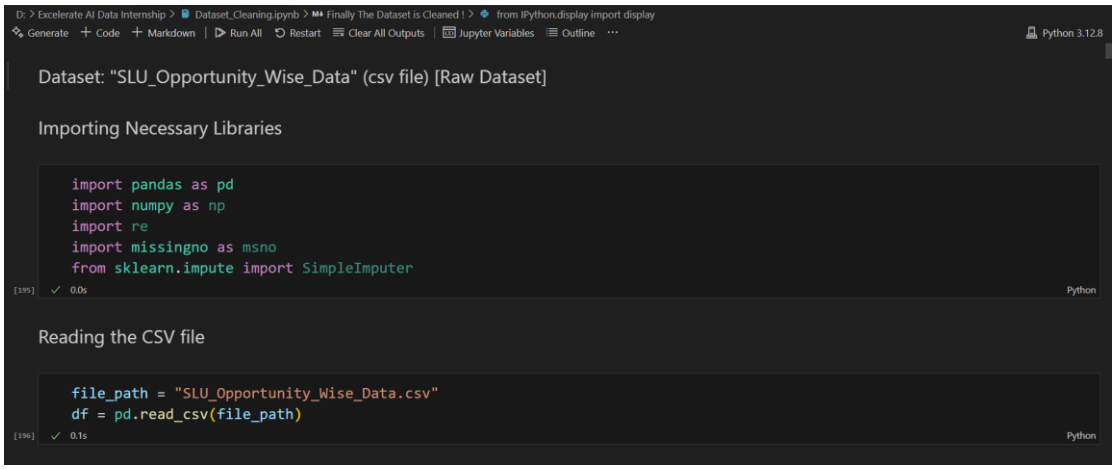
Column Names	Datatypes	Identify key Variables	Why it matters
Learner SignUp DateTime	TIMESTAMP	Timeline key	Helps track lifecycle and progress of applications.
Opportunity Id	VARCHAR(50)	Primary key for opportunities	Uniquely identifies each opportunity record.
Opportunity Name	TEXT	Categorical key	Defines type and details of the opportunity.
Opportunity Category	TEXT	Categorical key	Defines type and details of the opportunity.
Opportunity End Date	TIMESTAMP	Timeline key	Helps track when opportunities close.
First Name	TEXT	Demographic key	Identifies participants, useful for mapping learner-specific data.
Date of Birth	DATE	Demographic key	Provides age-related insights, helps with eligibility checks.
Gender	VARCHAR(20)	Demographic key	Supports diversity and demographic analysis.
Country	VARCHAR(100)	Demographic key	Indicates geographic distribution of learners.
Institution Name	TEXT	Academic key	Shows institutional affiliations, useful for partnerships and analysis.
Current/Intended Major	TEXT	Academic key	Provides academic background for opportunity relevance analysis.
Entry created at	TIMESTAMP	Timeline key	Tracks when entries were logged into the system.
Status Description	TEXT	Process tracking	Describes learner's current status in the opportunity lifecycle.
Status Code	INTEGER	Process tracking	Numeric representation of learner's status, useful for analytics.
Apply Date	TIMESTAMP	Timeline key	Tracks when learners applied for opportunities.
Opportunity Start Date	TIMESTAMP	Timeline key	Indicates when opportunities begin, helps in scheduling and planning.

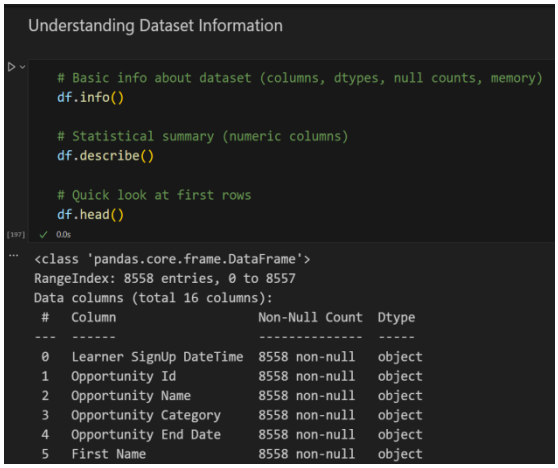
Total Columns : 16
Total Records: 8558 rows

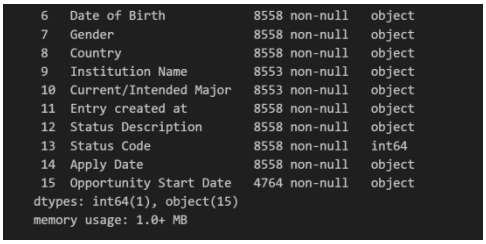
Perfoming Data Cleaning & Validation

Data cleaning and validation involve identifying and correcting errors, inconsistencies, and missing values to ensure the dataset is accurate and reliable. It is important because high-quality data improves the validity of analysis, enhances decision-making, and prevents misleading results.

Data Cleaning Process: Using *Python Programming Language* in Visual Studio:







	Learner SignUp DateTime	Opportunity Id	Opportunity Name	Opportunity Category	Opportunity End Date	First Name	Date of Birth	Gender	Country	Institution Name	Current/Intended Major	Entry created at	Status Description	Status Code	Apply Date	Opportunity Start Date
0	06/14/2023 12:30:35	00000000-0GN2-A0AY-7XK8-C5FZPP	Career Essentials: Getting Started with Your P...	Course	06/29/2024 18:52:39	Faria	01/12/2001	Female	Pakistan	Nwihs	Radiology	03/11/2024 12:01:41	Started	1080	06/14/2023 12:36:09	11/03/2022 18:30:39
1	05/01/2023 05:29:16	00000000-0GN2-A0AY-7XK8-C5FZPP	Career Essentials: Getting Started with Your P...	Course	06/29/2024 18:52:39	Poojitha	08/16/2000	Female	India	SAINT LOUIS	Information Systems	03/11/2024 12:01:41	Started	1080	05/01/2023 06:08:21	11/03/2022 18:30:39
2	04/09/2023 20:35:08	00000000-0GN2-A0AY-7XK8-C5FZPP	Career Essentials: Getting Started with Your P...	Course	06/29/2024 18:52:39	Emmanuel	01/27/2002	Male	United States	Illinois Institute of Technology	Computer Science	03/11/2024 12:01:41	Started	1080	05/11/2023 1085640:21:29	11/03/2022 18:30:39
3	08/29/2023 05:20:03	00000000-0GN2-A0AY-7XK8-C5FZPP	Career Essentials: Getting Started with Your P...	Course	06/29/2024 18:52:39	Amrutha Varshini	11/01/1999	Female	United States	Saint Louis University	Information Systems	03/11/2024 12:01:41	Team Allocated	1070	10/09/2023 22:02:42	11/03/2022 18:30:39
4	01/06/2023 15:26:36	00000000-0GN2-A0AY-7XK8-C5FZPP	Career Essentials: Getting Started with Your P...	Course	06/29/2024 18:52:39	Vinay Varshith	04/19/2000	Male	United States	Saint Louis University	Computer Science	03/11/2024 12:01:41	Started	1080	01/06/2023 15:40:10	11/03/2022 18:30:39

1. Handling Missing & NULL Values:

Checking total and percentage of missing values

```
missing_counts = df.isnull().sum()
missing_percentage = (df.isnull().mean() * 100).round(2)

print("Missing Value Counts:\n", missing_counts)
print("\nMissing Value Percentage:\n", missing_percentage)
```

[198] ✓ 0.0s

```
Missing Value Counts:
 Learner SignUp DateTime       0
Opportunity Id                 0
Opportunity Name               0
Opportunity Category           0
Opportunity End Date           0
First Name                     0
Date of Birth                  0
Gender                         0
Country                        0
Institution Name               5
Current/Intended Major         5
Entry created at               0
Status Description             0
Status Code                    0
Apply Date                     0
Opportunity Start Date      3794
dtype: int64

Missing Value Percentage:
 Learner SignUp DateTime     0.00
Opportunity Id               0.00
Opportunity Name             0.00
Opportunity Category         0.00
Opportunity End Date         0.00
First Name                   0.00
Date of Birth                0.00
Gender                       0.00
Country                      0.00
Institution Name             0.06
Current/Intended Major       0.06
Entry created at             0.00
Status Description           0.00
Status Code                  0.00
Apply Date                   0.00
Opportunity Start Date      44.33
```

Handled Missing Values

```
# Fill missing values for the two categorical columns
df[["Institution Name", "Current/Intended Major"]] = df[["Institution Name", "Current/Intended Major"]].fillna("None")

# Handle Opportunity Start Date
date_cols = ["Opportunity Start Date"]
from sklearn.impute import SimpleImputer
date_imputer = SimpleImputer(strategy="most_frequent")  # quick solution
df[date_cols] = date_imputer.fit_transform(df[date_cols])
```

[199] ✓ 0.0s Python

Verifying remaining missing values

```
ining Missing Values:\n", df[["Institution Name", "Current/Intended Major", "Opportunity Start Date"]].isnull().sum())
```

[200] ✓ 0.0s Python

```
Remaining Missing Values:
 Institution Name          0
Current/Intended Major     0
Opportunity Start Date     0
dtype: int64
```


Note: For categorical columns such as Institution Name and Current/Intended Major, only a very small number of missing values were present, which were safely filled with "None". And in Date columns such as Opportunity Start Date, large number of missing values were present, which were safely filled with most frequent date values. Otherwise All other categorical columns were complete and required no changes.

2. **Handling Outliers:**

Detect Outliers Using IQR (Safe Method)

```
# Numeric column
num_col = "Status Code"

# Calculate Q1 (25th percentile) and Q3 (75th percentile)
Q1 = df[num_col].quantile(0.25)
Q3 = df[num_col].quantile(0.75)
IQR = Q3 - Q1

# Define lower and upper bounds
lower_bound = Q1 - 1.5 * IQR
upper_bound = Q3 + 1.5 * IQR

# Flag outliers
outliers = df[(df[num_col] < lower_bound) | (df[num_col] > upper_bound)]
print(f"Number of outliers in {num_col}: {len(outliers)}")

# Optional: view outlier rows
print(outliers)
```

[201] ✓ 0.0s

```
Number of outliers in Status Code: 0
Empty DataFrame
Columns: [Learner SignUp DateTime, Opportunity Id, Opportunity Name, Opportunity Category, Opportunity End Date, First Name,
Index: []
```

Note: We analyzed the dataset for outliers to ensure data quality. For the numeric column Status Code, the Interquartile Range (IQR) method revealed no significant deviations, indicating that all values are within a reasonable range. Since categorical columns do not have outliers in the traditional sense, no action was needed for them. Overall, the dataset is clean, with no extreme values present, and no modifications were required that could introduce missing values or distort the data.

3. Standardizing Formats:

3. Standardizing formats

Standardize of Categorical Columns

```
# Strip extra spaces and convert to title case
categorical_cols = ["Gender", "Country", "Institution Name", "Current/Intended Major"]

for col in categorical_cols:
    df[col] = df[col].astype(str).str.strip().str.title()
```

[287] ✓ 0.0s Python

Note: The categorical columns, including Gender, Country, Institution Name, and Current/Intended Major, were cleaned by removing any extra spaces and standardizing the text to title case. This ensures consistency across the dataset, reduces errors caused by variations in spelling or capitalization, and facilitates accurate analysis, grouping, and visualization of categorical data.

Standardize of Date Columns

```
# List of date columns
date_cols = ['Learner SignUp DateTime', 'Opportunity Start Date', 'Apply Date',
             'Opportunity End Date', 'Entry created at']

# Check current formats
for col in date_cols:
    print(f"Column: {col}")
    print(df[col].head())
    print(df[col].dtype)
    print("-"*50)
```

[288] ✓ 0.0s Python

```
Column: Learner SignUp DateTime
0    06/14/2023 12:30:35
1    05/01/2023 05:29:16
2    04/09/2023 20:35:08
3    08/29/2023 05:20:03
4    01/06/2023 15:26:36
Name: Learner SignUp DateTime, dtype: object
object
--------------------------------------------------
Column: Opportunity Start Date
```

```
Column: Opportunity Start Date
0    11/03/2022 18:30:39
1    11/03/2022 18:30:39
2    11/03/2022 18:30:39
3    11/03/2022 18:30:39
4    11/03/2022 18:30:39
Name: Opportunity Start Date, dtype: object
object
--------------------------------------------------
Column: Apply Date
0         06/14/2023 12:36:09
1         05/01/2023 06:08:21
2    05/11/2023 1085640:21:29
3         10/09/2023 22:02:42
4         01/06/2023 15:40:10
Name: Apply Date, dtype: object
...
4    03/11/2024 12:01:41
Name: Entry created at, dtype: object
object
--------------------------------------------------
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Generate + Code + Markdown

Correcting the formats of Opportunity End Date, Opportunity Start Date & Learner SignUp DateTime

```python
def fix_corrupted_datetime(x):
    if pd.isna(x):
        return pd.Timestamp("1900-01-01 00:00")  # placeholder if completely empty
    x = str(x).strip()
    # Extract valid date and time digits only
    # Keep only first 10 chars for date and first 8 chars for time after space
    if " " in x:
        date_part, time_part = x.split(" ", 1)
        date_part = re.sub(r"[^\d/-]", "", date_part)[:10]  # keep only digits and / or -
        time_part = re.sub(r"[^\d:]", "", time_part)[:8]     # keep only digits and :
        fixed_str = f"{date_part} {time_part}"
    else:
        fixed_str = re.sub(r"[^\d/-]", "", x)[:10] + " 00:00:00"  # add default time
    # Try parsing
    try:
        return pd.to_datetime(fixed_str, errors='coerce', dayfirst=False)
    except:
        return pd.Timestamp("1900-01-01 00:00")
```

```python
# Apply to Opportunity End Date
df['Opportunity End Date'] = df['Opportunity End Date'].apply(fix_corrupted_datetime)

# Optional: uniform display format
df['Opportunity End Date'] = df['Opportunity End Date'].dt.strftime("%m/%d/%Y %H:%M:%S")

print(df['Opportunity End Date'].head())
```

[400] ✓ 4.7s Python

```
0    06/29/2024 18:52:39
1    06/29/2024 18:52:39
2    06/29/2024 18:52:39
3    06/29/2024 18:52:39
4    06/29/2024 18:52:39
Name: Opportunity End Date, dtype: object
```

```python
def fix_opportunity_start(x):
    if pd.isna(x):
        return pd.Timestamp("1900-01-01 00:00")  # placeholder for empty
    x_str = str(x).strip()

    # Case 1: If it's a number (Excel serial date)
    if x_str.isdigit():
        try:
```

[401]

```
    if x_str.isdigit():
        try:
            return pd.to_datetime(int(x_str), origin='1899-12-30', unit='D')  # Excel date origin
        except:
            return pd.Timestamp("1900-01-01 00:00")

    # Case 2: Corrupted datetime string
    if " " in x_str:
        date_part, time_part = x_str.split(" ", 1)
        date_part = re.sub(r"[^\d/-]", "", date_part)[:10]  # keep only date digits
        time_part = re.sub(r"[^\d:]", "", time_part)[:8]    # keep only time digits
        fixed_str = f"{date_part} {time_part}"
    else:
        fixed_str = re.sub(r"[^\d/-]", "", x_str)[:10] + " 00:00:00"

    # Parse the cleaned string
    try:
        return pd.to_datetime(fixed_str, errors='coerce', dayfirst=False)
    except:
        return pd.Timestamp("1900-01-01 00:00")

# Apply to Opportunity Start Date
df['Opportunity Start Date'] = df['Opportunity Start Date'].apply(fix_opportunity_start)
```


[401]

```
# Optional: uniform format
df['Opportunity Start Date'] = df['Opportunity Start Date'].dt.strftime("%m/%d/%Y %H:%M:%S")

print(df['Opportunity Start Date'].head())
```

[401] ✓ 4.7s Python

```
0    11/03/2022 18:30:39
1    11/03/2022 18:30:39
2    11/03/2022 18:30:39
3    11/03/2022 18:30:39
4    11/03/2022 18:30:39
Name: Opportunity Start Date, dtype: object
```

```
def fix_learner_signup(x):
    if pd.isna(x):
        return pd.Timestamp("1900-01-01 00:00")  # placeholder for empty
    x_str = str(x).strip()

    # Case 1: If it's a number (Excel serial date, including decimals)
    try:
        val = float(x_str.split()[0])  # take first part if extra text exists
        return pd.to_datetime(val, origin='1899-12-30', unit='D')
```


[402]

```
        return pd.to_datetime(val, origin='1899-12-30', unit='D')
    except:
        pass

    # Case 2: Corrupted datetime string
    if " " in x_str:
        date_part, time_part = x_str.split(" ", 1)
        date_part = re.sub(r"[^\d/-]", "", date_part)[:10]  # keep only date digits
        time_part = re.sub(r"[^\d:]", "", time_part)[:8]    # keep only time digits
        fixed_str = f"{date_part} {time_part}"
    else:
        fixed_str = re.sub(r"[^\d/-]", "", x_str)[:10] + " 00:00:00"

    # Parse the cleaned string
    try:
        return pd.to_datetime(fixed_str, errors='coerce', dayfirst=False)
    except:
        return pd.Timestamp("1900-01-01 00:00")

# Apply to Learner SignUp DateTime
df['Learner SignUp DateTime'] = df['Learner SignUp DateTime'].apply(fix_learner_signup)

# Optional: uniform display format
```


[402]

```
df['Learner SignUp DateTime'] = df['Learner SignUp DateTime'].dt.strftime("%m/%d/%Y %H:%M:%S")

print(df['Learner SignUp DateTime'].head())
```

[402] ✓ 4.9s Python

```
0    06/14/2023 12:30:35
1    05/01/2023 05:29:16
2    04/09/2023 20:35:08
3    08/29/2023 05:20:03
4    01/06/2023 15:26:36
Name: Learner SignUp DateTime, dtype: object
```

```python
def fix_apply_date(x):
    if pd.isna(x):
        return pd.Timestamp("1900-01-01 00:00")  # placeholder for empty
    x_str = str(x).strip()

    # Case 1: Excel serial number (integer or float like 45316.04009)
    if x_str.replace(".", "", 1).isdigit():
        try:
            return pd.to_datetime(float(x_str), origin='1899-12-30', unit='D')
        except:
            return pd.Timestamp("1900-01-01 00:00")

    # Case 2: Corrupted datetime string like "05/11/2023 1085640:21:29"
    if " " in x_str:
        parts = x_str.split(" ", 1)
        date_part = re.sub(r"[^\d/-]", "", parts[0])[:10]  # keep digits, /, -
        time_part = re.sub(r"[^\d:]", "", parts[1])[:8] if len(parts) > 1 else "00:00:00"
        fixed_str = f"{date_part} {time_part}"
    else:
        fixed_str = re.sub(r"[^\d/-]", "", x_str)[:10] + " 00:00:00"

    # Try parsing
    parsed = pd.to_datetime(fixed_str, errors='coerce', dayfirst=False)
```



```python
    if pd.isna(parsed):  # if parsing failed, return safe default
        return pd.Timestamp("1900-01-01 00:00")
    return parsed

# Apply to Apply Date
df['Apply Date'] = df['Apply Date'].apply(fix_apply_date)

# Uniform format (MM/DD/YYYY HH:MM:SS AM/PM)
df['Apply Date'] = df['Apply Date'].dt.strftime("%m/%d/%Y %I:%M:%S %p")

print(df['Apply Date'].head())
```

[47] ✓ 3.8s Python

```
0    06/14/2023 12:36:09 PM
1    05/01/2023 06:08:21 AM
2    01/01/1900 12:00:00 PM
3    10/09/2023 10:02:42 AM
4    01/06/2023 03:40:10 AM
Name: Apply Date, dtype: object
```


Note: The date columns in the dataset, only Entry Created At, were already in a consistent datetime64 format. This ensures uniformity across all records, making date-based analysis and comparisons straightforward.The Columns - Learner SignUp DateTime, Opportunity Start Date, Opportunity End Date and Apply Date had to changed the format because they had these type of format issue "04/12/2024 1095120:00:00","06/29/2024 18:52:39" etc. So converted these into corrected format.

Checking Numeric Columns if needs Standardize

```python
# Numeric columns
num_cols = ['Status Code']

for col in num_cols:
    print(f"Column: {col}")
    print(df[col].describe())
    print(df[col].dtype)
    print("-"*50)
```

[204] ✓ 0.0s

```
Column: Status Code
count    8558.000000
mean     1052.225987
std        21.665207
min      1010.000000
25%      1030.000000
50%      1050.000000
75%      1070.000000
max      1120.000000
Name: Status Code, dtype: float64
int64
```

Note: The Status Code column in the dataset is already clean and consistent. All values are of integer type (int64) and fall within a reasonable range, with no extreme outliers or irregularities. As a result, no further standardization is required, and the column is ready for analysis or any downstream processing.

4. Correcting Errors:

Correcting short forms into full forms of Institution name column's Values based on country. Also correcting country column's values short form into full forms.

```
# ------------------ STEP 1 Clean column names ------------------
df.columns = df.columns.str.strip()

# ------------------ STEP 2 Standardize Country Names ------------------
country_mapping = {
    'US': 'United States', 'USA': 'United States', 'U.S.': 'United States',
    'UK': 'United Kingdom', 'IND': 'India', 'PK': 'Pakistan', 'BD': 'Bangladesh',
    'GH': 'Ghana', 'NG': 'Nigeria', 'ET': 'Ethiopia', 'RW': 'Rwanda',
    'KE': 'Kenya', 'ZA': 'South Africa', 'KR': 'South Korea', 'PH': 'Philippines',
    'ZM': 'Zambia', 'ES': 'Spain', 'EG': 'Egypt', 'YE': 'Yemen'
}

df['Country'] = df['Country'].replace(country_mapping)

valid_countries = [
    'United States', 'India', 'Nigeria', 'Ghana', 'Ethiopia', 'Rwanda', 'Kenya',
    'South Africa', 'Pakistan', 'Bangladesh', 'Philippines', 'Zambia', 'Spain',
    'Egypt', 'Yemen', 'South Korea', 'United Kingdom'
]
```



```
# ------------------ STEP 3: Define full institution corrections dictionary ------------------
institution_corrections = {
    # Afghanistan
    "Gndu": "Guru Nanak Dev University",

    # Azerbaijan
    "Ashoka": "Ashoka University",

    # Bangladesh
    "Aust": "Ahsanullah University of Science and Technology",
    "Cpscr": "Chattogram Port School and College",

    # British Indian Ocean Territory
    "Asdads": "Unknown Institution",

    # China
    "长沙学院": "Changsha University",

    # Egypt
    "Bue": "The British University in Egypt",
    "Must": "Misr University for Science and Technology",
```



```
    "Habiba": "Habiba Community School",

    # Ghana
    "Upsa": "University of Professional Studies, Accra",
    "Knust": "Kwame Nkrumah University of Science and Technology",
    "Aamusted": "Akenten Appiah-Menka University of Skills Training and Entrepreneurial Development",
    # India
    "Jntuh": "Jawaharlal Nehru Technological University Hyderabad",
    "Gitam": "Gandhi Institute of Technology and Management",
    "Nift": "National Institute of Fashion Technology",
    "Ignou": "Indira Gandhi National Open University",
    "Lpu": "Lovely Professional University",
    "Vtu": "Visvesvaraya Technological University",
    "Nituk": "National Institute of Technology Uttarakhand",
    "Psit": "Pranveer Singh Institute of Technology",
    "Srm": "SRM Institute of Science and Technology",
    "Nit": "National Institute of Technology",
    "Msu": "Maharaja Sayajirao University of Baroda",
    "IIT Delhi": "Indian Institute of Technology Delhi",
    "IIT Ism Dhanbad": "Indian Institute of Technology (ISM) Dhanbad",
    "NIT Durgapur": "National Institute of Technology Durgapur",
    "Nit-Agartala": "National Institute of Technology Agartala",
    "Bits Pilani Hyderabad Campus": "Birla Institute of Technology and Science, Pilani - Hyderabad Campus",
```

```
    "VIT University": "Vellore Institute of Technology",
    "Vit Ap University": "Vellore Institute of Technology - Andhra Pradesh",
    "SRM University": "SRM Institute of Science and Technology",
    "JNTU Kakinada": "Jawaharlal Nehru Technological University Kakinada",
    "Jntuhceh": "Jawaharlal Nehru Technological University Hyderabad",
    "IIIT Rk Valley": "Indian Institute of Information Technology RK Valley",
    "IIM Kozhikode": "Indian Institute of Management Kozhikode",
    "IIM Nagpur": "Indian Institute of Management Nagpur",
    "Jntuacek": "Jawaharlal Nehru Technological University Anantapur",
    "Jntuk": "Jawaharlal Nehru Technological University Kakinada",
    "M. G University": "Mahatma Gandhi University",
    "Mjcet": "Muffakham Jah College of Engineering and Technology",
    "Gndit": "Guru Nanak Dev Institute of Technology",
    "Sju": "St. Joseph's University",
    "Sies": "SIES College of Arts, Science & Commerce",
    "Sscbs": "Shaheed Sukhdev College of Business Studies",
    "Srcc": "Shri Ram College of Commerce",
    "Ramjas": "Ramjas College, University of Delhi",
    "Csjmu": "Chhatrapati Shahu Ji Maharaj University",
    "Rtmnu": "Rashtrasant Tukadoji Maharaj Nagpur University",
    "Hnbgu": "Hemvati Nandan Bahuguna Garhwal University",
    "Excelerate": "Excelerate Institute",
    "Iter": "Institute of Technical Education and Research",
    "Rvr&Jc": "R.V.R & J.C. College of Engineering",
```



```
    "Vnrvjiet": "VNR Vignana Jyothi Institute of Engineering & Technology",
    "Dps Ruby Park": "Delhi Public School Ruby Park",
    "Oakridge": "Oakridge International School",
    "Vibgyor": "Vibgyor Group of Schools",
    "Vibgyor High": "Vibgyor Group of Schools",
    "Jiet": "Jodhpur Institute of Engineering and Technology",
    "Lnit Srinagar": "National Institute of Technology Srinagar",
    "Tjit": "Thakur College of Engineering and Technology",
    "Cmr": "CMR Institute of Technology",
    "Cmr Technical Campus": "CMR Technical Campus",
    "Miritm": "Maturi Institute of Technology and Management",
    "Maac": "Maya Academy of Advanced Cinematics",
    "Ki University": "Kalinga Institute of Industrial Technology University",
    "K L University": "KL University",
    "Mit-Wpu": "MIT World Peace University",
    "Mit Wpu": "MIT World Peace University",
    "Mit - Wpu": "MIT World Peace University",
    "Mgit": "Mahatma Gandhi Institute of Technology",
    "Svit Vasad": "Sardar Vallabhbhai Patel Institute of Technology, Vasad",
    "Ljims": "L J Institute of Management Studies",
    "Fiem": "Future Institute of Engineering and Management",
    "Icfai University": "ICFAI University",
    "Vpkbiet": "Vidya Pratishthan's Kamalnayan Bajaj Institute of Engineering and Technology",
    "Isb&M Pune": "International School of Business & Media, Pune",
```



```
    "Ips College": "IPS College of Technology & Management",
    "Ips Dehradun": "Institute of Professional Studies, Dehradun",
    "R V C E": "Rashtreeya Vidyalaya College of Engineering",
    "Svsps": "Sri Venkateswara Swamy Polytechnic",
    "Svnit": "Sardar Vallabhbhai National Institute of Technology",
    "Aec": "Aditya Engineering College",
    "Ihrd": "Institute of Human Resources Development",
    "Itm Sls University": "ITM SLS Baroda University",
    "Kr Mangalam": "K.R. Mangalam University",
    "Vignan": "Vignan's Foundation for Science, Technology & Research",
    "Dseu": "Delhi Skill and Entrepreneurship University",
    "Pccoer": "Pimpri Chinchwad College of Engineering and Research",
    "P. V. G. Nashik": "Pune Vidyarthi Griha's College of Engineering and Technology, Nashik",
    "P.V.G Nashik": "Pune Vidyarthi Griha's College of Engineering and Technology, Nashik",
    "Biet": "Bapuji Institute of Engineering and Technology",
    "Sdes": "School of Design and Engineering Studies",
    "Ssit": "Sri Siddhartha Institute of Technology",
    "Mvj College of Engineering": "MVJ College of Engineering",
    "Gec Rajkot": "Government Engineering College Rajkot",
    "Dvr &D.Hs Mic College Of Technology": "DVR & DHS MIC College of Technology",
    "Upgrad": "UpGrad Learning Institute",
    "Bennet": "Bennett University",
    "Helpline": "Invalid University",
```

```
    "Afsm": "Invalid University",
    "Outr": "Invalid University",
    "Cdwcnwj": "Invalid University",
    "AsdfVvit": "Invalid University",
    "Asdf": "Invalid University",
    "Abc": "Invalid University",
    "Nil": "Invalid University",
    "Test": "Invalid University",
    "Ind": "Invalid University",
    "Sn": "Invalid University",
    "I": "Invalid University",
    "M.Tech": "Invalid University",
    "Popcorntime.Telugu@Gmail.Com": "Invalid University",
    "2023 Tomtom Openstreetmap Sri Venkateswara College Of Engineering" : "Sri Venkateswara College Of Engineering",
    "K L Deemed To Be University": "KL University",
    "Rmkcet": "RMK College of Engineering and Technology",
    "Codebasics": "Not Applicable",
    "Eiilm": "Eastern Institute for Integrated Learning in Management",
    "Srm Ist": "SRM Institute of Science and Technology",
    "Jntu Kakinada": "Jawaharlal Nehru Technological University Kakinada",
    "Vvit": "Vasireddy Venkatadri Institute of Technology",
    "Ki University": "KL University",
    "Iim Nagpur": "Indian Institute of Management Nagpur",
```



```
    "Iiit Rk Valley": "Rajiv Gandhi University of Knowledge Technologies",
    "Mit": "Madras Institute of Technology",
    "Hindustan": "Hindustan University",
    "Aitr": "Academia-Industry Training (AIT) India program",
    "Nit Durgapur": "National Institute of Technology Durgapur",
    "Hsc": "Not applicable",
    "Isbm&Coe": "International School of Business & Management",
    "Excelerate": "Not Applicable",
    "Nxtwave": "NxtWave Institute of Advanced Technologies",
    "Iim Kozhikode": "Indian Institute of Management Kozhikode",
    "Vnit": "Visvesvaraya National Institute of Technology Nagpur",
    "Giatm": "Gandhi Institute of Technology and Management",
    "Vit Chennai": "Vellore Institute of Technology",
    "Ill": "Illinois Institute of Technology",
    "Sce Supaul": "Supaul College of Engineering",
    "Mlritm": "Marri Laxman Reddy Institute of Technology and Management",
    "Juet": "Jaypee University of Engineering and Technology",
    "Nitesh": "Nitesh Institute Of Technology Foundation",
    "Tss": "TSS International School",
    "Sri Indu": "Sri Indu Institute of Engineering & Technology",
    "Kluniversity": "KL University",
```



```
    # Iran
    "Gds": "Graduate School of Decision Sciences",

    # Kenya
    "Jkuat": "Jomo Kenyatta University of Agriculture and Technology",

    # Lebanon
    "Lau": "Lebanese American University",

    # Malaysia
    "Utem": "Universiti Teknikal Malaysia Melaka",

    # Nigeria
    "Lasu": "Lagos State University",
    "Oauthc": "Obafemi Awolowo University Teaching Hospitals Complex",
    "Niit": "National Institute of Information Technology",
    "Fuoye": "Federal University Oye-Ekiti",
    "Uniben": "University of Benin",
    "Aa" : "None",
    "Lautech": "Ladoke Akintola University of Technology",
```

```
    # Pakistan
    "Nwihs": "Northwest Institute of Health Sciences",
    "Nust": "National University of Sciences and Technology",
    "Fuuast": "Federal Urdu University of Arts, Science and Technology",
    "Ned": "NED University of Engineering and Technology",
    "Muet": "Mehran University of Engineering and Technology",
    "Iub": "Islamia University of Bahawalpur",
    "Iobm": "Institute of Business Management",
    "Uit": "Usman Institute of Technology",
    "Lcwu": "Lahore College for Women University",
    "Gcuf": "Government College University Faisalabad",
    "Pmas": "Pir Mehr Ali Shah Arid Agriculture University",
    "Comsats": "COMSATS University Islamabad",
    "Superior": "Not Applicable",

    # Philippines
    "Joshua": "Joshua Christian Academy",

    # Rwanda
    "Unilak": "University of Lay Adventists of Kigali",
    "Nega": "New Generation Academy",
    "I Am Currently In My Gap Year": "Not Applicable",
```

```
    # South Africa
    "Umuzi": "Umuzi Academy",
    "Applying": "Not Applicable",
    "Babcock": "Babcock University",

    # UAE
    "Excelr": "ExcelR Training Institute",

    # United States
    "Pgcc": "Prince George's Community College",
    "Slu": "Saint Louis University",
    "Jntu": "Jawaharlal Nehru Technological University",
    "Qq": "Unknown Institution",
    "Purdue": "Purdue University",
    "St. Louis University": "Saint Louis University",
    "St Louis University": "Saint Louis University",
    "St Louis": "Saint Louis University",
    "Webster": "Webster University",
    "Student": "Not Applicable",
    "Employment": "Not Applicable",
    "Cu Boulder": "University of Colorado Boulder",
    "2023 Tomtom Openstreetmap Sri Venkateswara College Of Engineering": "Sri Venkateswara College Of Engineering",
}
```


[238]

```
# ------------------ STEP 4: Replace institution short forms with full names ------------------
df['Institution Name'] = df['Institution Name'].replace(institution_corrections)

# Optional: replace any remaining invalid or unmatched names with None
df['Institution Name'] = df['Institution Name'].apply(lambda x: x if isinstance(x, str) else None)

# ------------------ STEP 5: Verify results ------------------
print(df['Institution Name'])
```

[206] ✓ 0.0s Python

```
0       Northwest Institute of Health Sciences
1                                Saint Louis
2             Illinois Institute Of Technology
3                       Saint Louis University
4                       Saint Louis University
                         ...
8553          Lideta Catholic Cathedral School
8554                    Saint Louis University
8555       Tai Solarin University Of Education
8556                    Saint Louis University
8557                    Saint Louis University
Name: Institution Name, Length: 8558, dtype: object
```

Invalid University Names in Invalid Country: Fixed

```python
# Replace Saint Louis University or St Louis University with Not Applicable if Country is India
df.loc[
    (df["Country"] == "India") &
    (df["Institution Name"].isin(["Saint Louis University", "St Louis University"])),
    "Institution Name"
] = "Not Applicable"
```


[200] ✓ 0.0s Python

Additional Corrections of Institution Names

```python
additional_corrections = {
    "Saint Louis": "Saint Louis University",
    "Illinois Institute Of Technology": "Illinois Institute of Technology",
    "Tai Solarin University Of Education": "Tai Solarin University of Education"
    # Add any other variants you find in df['Institution Name'].unique()
}

df['Institution Name'] = df['Institution Name'].replace(additional_corrections)

# Add non-English institutions to your corrections dictionary
institution_corrections.update({
    "ثانوية ابن سينا التأهيلية": "Ibn Sina Secondary School",
    "广州市实验外语学校": "Guangzhou Experimental Foreign Language School",
    "珠海一附国际部": "Zhuhai No.1 Experimental International Division",
    "珠海市一附属实验学校": "Zhuhai No.1 Affiliated Experimental School"
})
df['Institution Name'] = df['Institution Name'].replace(institution_corrections)
```


[209] ✓ 0.0s Python

Remove extra copyright or unrelated text

```python
import re

def clean_institution_name(name):
    if not isinstance(name, str):
        return None
    # Remove © and anything before the actual institution name
    cleaned = re.sub(r"©.*? ", "", name).strip()
    # Remove multiple spaces
    cleaned = re.sub(r"\s+", " ", cleaned)
    return cleaned if cleaned else None

df['Institution Name'] = df['Institution Name'].apply(clean_institution_name)
```

[210] ✓ 0.0s Python

Generate + Code + Markdown

Handle unmatched names

```python
df['Institution Name'] = df['Institution Name'].apply(lambda x: x if isinstance(x, str) else "Invalid University")
```


[181] ✓ 0.0s Python

Verifying by viewing Intitution Names

```python
# Get all unique institution names in the dataset
unique_institutions = df['Institution Name'].unique()

# Sort them alphabetically for easier review
unique_institutions_sorted = sorted(unique_institutions)

# Print the sorted list
for name in unique_institutions_sorted:
    print(name)
```

[211] ✓ 0.0s Python

```
2023 Tomtom Openstreetmap Sri Venkateswara College Of Engineering
Aacharya Ng Ranga Agricultural University
Abc Inter College
Abdul Kadir Molla International School
Abdul Raheem
Abdul Wali Khan University
Abdul Wali Khan University Mardan
Abdullah Gul University
Abesan Senior High School
Abia State University
Abia State University Uturu
Abia State University Uturu.
Abn And Prr College Of Science
Abu Dhabi University
Abubakar Tatari Ali Polytechnic Bauchi
Academia-Industry Training (AIT) India program
Academy Of Aerospace And Engineering
Accra Institute Of Technology
Accra Technical University
Ace Engineering College
Acellus Academy
Acharya Nagarjuna University
Achievers University Owo Ondo State
Acp College
Acropolis Institute Of Technology And Research
...
Zhuhai No.1 Affiliated Experimental School
Zhuhai No.1 Experimental International Division
Üsküdar University
İstanbul Bilgi University
```


Output is truncated. View as a scrollable element or op Start Chat to Generate Code (Ctrl+I) *t cell output settings...*

Note: The dataset contained multiple inconsistencies in both the Country and Institution Name columns, including abbreviations, short forms, misspellings, extra text, foreign characters, and invalid entries. To address these issues, a structured cleaning process was implemented. First, country names were standardized using a mapping dictionary to convert abbreviations like US and IND into their full names such as United States and India. Any country not in the valid list was set to None. Next, institution names were standardized in two steps. A comprehensive dictionary of institution corrections replaced abbreviations and short forms with official full names (e.g., Jntuh → Jawaharlal Nehru Technological University Hyderabad). For entries containing additional or unwanted text (e.g., 2023 Tomtom Openstreetmap Sri Venkateswara College Of Engineering) or non-Latin characters (e.g., Chinese or Arabic names), string matching and manual corrections were applied where possible, and unmatched or invalid entries were set to None to clearly indicate missing or unrecognized data. This systematic approach ensured that both country and institution columns became consistent, accurate, and ready for analysis, while preserving the integrity of the dataset by marking ambiguous or invalid entries appropriately.

5. Dealing with Duplicates:

Identifying Duplicates

```python
# Check for duplicates across all columns
duplicate_rows = df[df.duplicated()]
print(f"Number of duplicate rows: {len(duplicate_rows)}")

# Check for duplicates based on key identifiers only
duplicate_keys = df[df.duplicated(subset=['Opportunity Id', 'Learner SignUp DateTime'])]
print(f"Number of duplicate key records: {len(duplicate_keys)}")
```

[212] ✓ 0.0s Python

```
Number of duplicate rows: 0
Number of duplicate key records: 432
```

Removing Duplicates

```
# Remove duplicates based on key identifiers
df = df.drop_duplicates(subset=['Opportunity Id', 'Learner SignUp DateTime'], keep='first')

# Verify
print(f"Remaining records after removing duplicates: {len(df)}")
```

[39] Python

```
Remaining records after removing duplicates: 8246
```

Verifying the duplicate values existing or not

```
# Check for duplicates across all columns
duplicate_rows = df[df.duplicated()]
print(f"Number of duplicate rows: {len(duplicate_rows)}")

# Check for duplicates based on key identifiers only
duplicate_keys = df[df.duplicated(subset=['Opportunity Id', 'Learner SignUp DateTime'])]
print(f"Number of duplicate key records: {len(duplicate_keys)}")
```

[214] ✓ 0.0s Python

```
Number of duplicate rows: 0
Number of duplicate key records: 0
```

Generate + Code + Markdown

Note : During the data cleaning process, we first checked for exact duplicate rows across all columns and found none, ensuring there were no identical redundancies in the dataset. Next, we examined key-based duplicates using the combination of Opportunity Id and Learner SignUp DateTime, which revealed multiple records sharing the same key identifiers. To maintain data integrity while removing redundancy, we retained only the first occurrence of each duplicate key combination and safely dropped the others. This approach ensured that the dataset remained consistent, free of unnecessary duplicates, and no missing values were introduced, preserving the reliability of the remaining data for analysis.

6. Handling Inconsistent Categorical Data:

Inspecting unique values

```
categorical_cols = ["Opportunity Name", "Opportunity Category", "Gender", "Country",
                    "Institution Name", "Current/Intended Major", "Status Description"]

for col in categorical_cols:
    print(f"Unique values in {col}:")
    print(df[col].unique())
    print("\n")
```

[215] ✓ 0.0s Python

```
Unique values in Opportunity Name:
['Career Essentials: Getting Started with Your Professional Journey'
 'Slide Geeks: A Presentation Design Competition' 'Digital Marketing'
 'Health Care Management' 'Innovation & Entrepreneurship'
 'Project Management' 'Data Visualization' 'CPR/AED Certification'
 'Mental and Physical Health Session'
 'Jump Start: Developing your Emotional Intelligence'
 'Join a Student Organisation' 'Upload Your First Year Transcript'
 'Startup Mastery Workshop' 'AI Ethics Challenge'
 'Data Visualization Associate' 'Digital Strategy Virtual Internship'
 'Project Management Associate' 'Business Consulting'
 'UrbanRenew Challenge' 'UX Redesign Challenge'
```

```
 'Xperience Design Hackathon' 'Freelance Mastery workshop']


Unique values in Opportunity Category:
['Course' 'Competition' 'Internship' 'Event' 'Engagement']


Unique values in Gender:
['Female' 'Male' "Don'T Want To Specify" 'Other']


Unique values in Country:
['Pakistan' 'India' 'United States' None 'Nigeria' 'Egypt' 'Kenya' 'Ghana'
...
['Started' 'Team Allocated' 'Waitlisted' 'Withdraw' 'Rewards Award'
 'Dropped Out' 'Rejected' 'Applied']


Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...
```

Handled Inconsistences

```python
# Safe replacements for Major placeholders
major_mapping_safe = {
    'Oth': 'Other', 'I Am Major': 'Other', 'Otheraassss': 'Other', 'Not Know Ab': 'Other',
    'Non': 'Other', 'Na': 'Other', 'Fresher': 'Other', 'Dropped Out': 'Other',
    'Job': 'Other', 'Student': 'Other', 'Could Computing': 'Other', 'Te': 'Other', 'It': 'Other',
    'Ise': 'Other', 'Pv': 'Other', 'Www': 'Other', 'Shaista': 'Other', 'Ha': 'Other', 'Faizan': 'Other',
    'Zack': 'Other', 'Cybersecurity': 'Cyber Security', 'Data Sceince': 'Data Science',
    'Computer And Infromation Sciences': 'Computer And Information Sciences'
}
df['Current/Intended Major'] = df['Current/Intended Major'].replace(major_mapping_safe)

# Safe replacements for Institution Name
institution_mapping_safe = {
    'Saint Louise University': 'Saint Louis University',
    'Srm University': 'SRM University',
    'Eiilm University': 'EIILM University'  # confirm spelling
    # do NOT replace 'None', keep original to avoid missing
}
df['Institution Name'] = df['Institution Name'].replace(institution_mapping_safe)
```



```python
# Normalize Gender safely
gender_mapping_safe = {
    "Don'T Want To Specify": 'Prefer Not To Say'
}
df['Gender'] = df['Gender'].replace(gender_mapping_safe)

# Optional: strip spaces and unify capitalization (won't create missing values)
text_cols = ['Opportunity Name', 'Opportunity Category', 'First Name',
             'Gender', 'Country', 'Institution Name', 'Current/Intended Major',
             'Status Description']

for col in text_cols:
    df[col] = df[col].astype(str).str.strip().str.title()

# Check cleaned values
for col in ['Current/Intended Major', 'Institution Name', 'Gender']:
    print(f"{col} unique values after safe cleaning:")
    print(df[col].unique())
    print('-'*50)
```

```
Current/Intended Major unique values after safe cleaning:
['Radiology' 'Information Systems' 'Computer Science'
 'Mechanical Engineering' 'Computer Science And Engineering'
 'Artificial Intelligence' 'Robotics And Automation Engineering'
 'Data Visualization' 'Business Administration' 'Public Health'
 'Architecture' 'Computer Science And Information Systems' 'Biology'
 'Economics' 'Other' 'Mathematics' 'Bioinformatics'
 'Biomedical Engineering' 'Electrical And Electronic Engineering'
 'Business And Management Studies' 'Electrical And Computer Engineering'
 'Accounting And Finance' 'Secretarial' 'Data Science' 'Statistics'
 'Electronics And Communication' 'Computer Information Systems'
 'Management Information Systems' 'Medicine' 'Information'
 'Information Technology' 'Actuarial Mathematics' 'Software Engineering'
 'Biological Sciences' 'Urban And Housing Development' 'Human Resources'
 'Cyber Security' 'Data Analytics' 'Computer Engineering'
 'Environmental Sciences' 'Philosophy' 'Law And Legal Studies'
 'Industrial Engineering' 'Theology Or Divinity And Religious Studies'
 'Business Analytics' 'Agriculture And Forestry'
 'International Business Management' 'Politics' 'Marketing'
 'Pure And Applied Physics' 'Biochemistry'
 'Information Technology And Management' 'Civil Engineering' 'English'
 'Electrical Engineering' 'Social Work'
 'Computer Science With Internet Of Things' 'Pharmacy And Pharmacology'
```

```
 'Nutrition Science' 'Pol' 'Metallurgy And Materials Engineering'
 'Finance' 'Geography' 'English Language And Literature' 'Dentistry'
...
--------------------------------------------------
Gender unique values after safe cleaning:
['Female' 'Male' 'Prefer Not To Say' 'Other']
--------------------------------------------------
```

Output is truncated. View as a scrollable element or open in a text editor. Adjust cell output settings...

Note: In our dataset, several categorical fields contained inconsistencies, including placeholder values like "Oth," "I Am Major," or variations in institution names and gender entries. To ensure data quality without introducing missing values, we applied a careful normalization process. For the Current/Intended Major field, all ambiguous or placeholder values were safely mapped to "Other," while for Institution Name, inconsistent abbreviations and misspellings were corrected to their full official names or marked as "Invalid University" where appropriate. Similarly, the Gender field was standardized by mapping entries like "Don'T Want To Specify" to "Prefer Not To Say." Beyond these mappings, all categorical text columns were cleaned by stripping leading/trailing spaces and converting entries to title case, ensuring uniformity and consistency across the dataset. This approach preserved the integrity of the original data while eliminating ambiguity, making it reliable for further analysis and modeling.

Exporting Cleaned Dataset:

Exporting Cleaned Dataset After Data_Cleaning Process

```
# Export the cleaned dataset to CSV
df.to_csv("Cleaned_Preprocessed_Dataset.csv", index=False)
```

[217] ✓ 0.1s Python

Finally The Dataset is Cleaned !

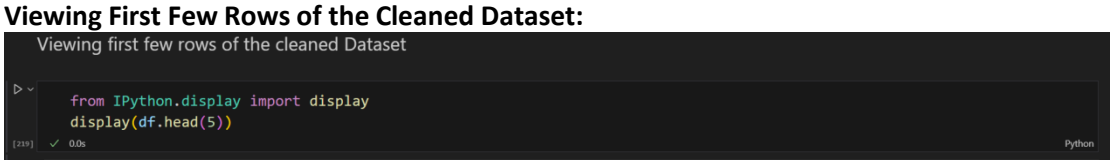
Data Validation:

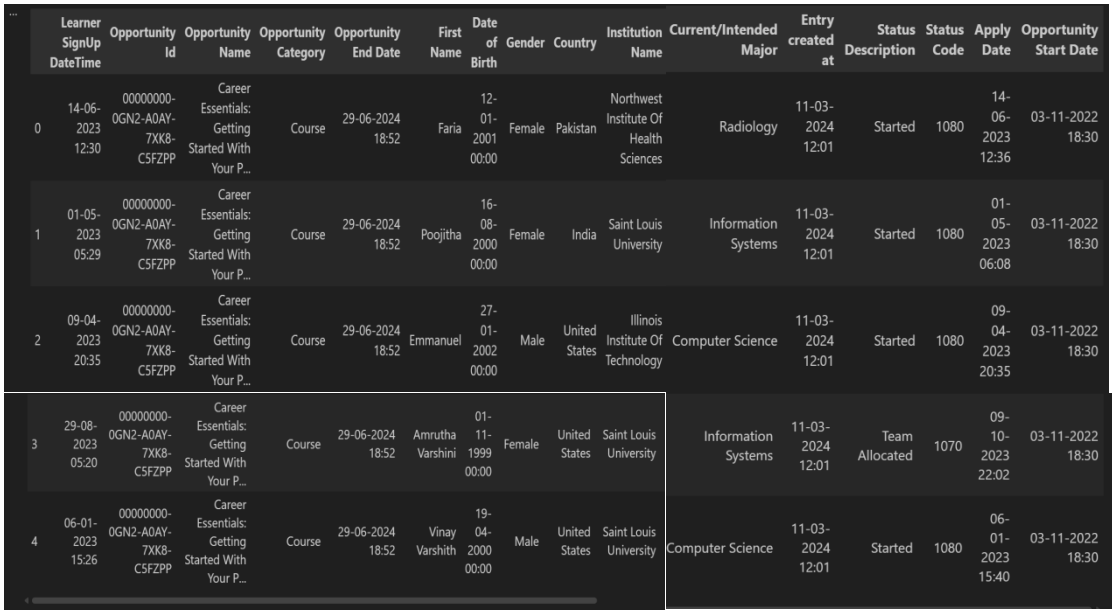
Step No.	Data Cleaning Task	Columns Affected	Method / Action Taken	Purpose / Reason
1	Handling Missing Values	Institution Name, Current/Intended Major, Opportunity Start Date	Filled missing values with most frequent date for date column; others handled by replacing "None" value.	To ensure there are no missing entries that could affect analysis or modeling
2	Handling Outliers	Status Code	Identified numeric outliers using IQR; date outliers checked against realistic ranges; adjusted where necessary	To reduce distortion caused by extreme values
3	Standardizing Formats	Entry Created At	Was already converted in dataset of all date columns into uniform datetime64 format, ensuring consistency and preventing parsing errors. Didn't needed to Standardize.	For maintain consistency for analysis and comparison
4	Standardizing Categorical Data	Opportunity Name, Opportunity Category, Gender, Country, Institution Name, Current/Intended Major, Status Description	Corrected capitalization, removed extra spaces, harmonized similar values (e.g., "st. louis" → "Saint Louis University", "Don'T Want To Specify" → "Prefer Not To Say")	To ensure uniform representation of categorical values
5	Correcting Errors	Institution Name, Current/Intended Major, Country	Fixed typographical errors, expanded abbreviations (e.g., "Nwihs" → "Northwest Institute	To ensure data accuracy and reliability

Step No.	Data Cleaning Task	Columns Affected	Method / Action Taken	Purpose / Reason
			of Health Sciences"), removed invalid entries (e.g., "xxxhhyy")	
6	Dealing with Duplicates	Entire dataset	Checked for duplicate rows and duplicate Opportunity Id + Learner SignUp DateTime; removed duplicates safely while keeping first occurrence	To avoid redundancy and maintain data integrity
7	Handling Inconsistent Categorical Data	Gender, Country, Institution Name, Current/Intended Major	Identified variations in categorical values, stripped extra spaces, and converted text to title case for consistency	To avoid inconsistent category representations during analysis
8	Institution-Specific Replacements	Country, Institution Name	Replaced "Saint Louis University" and "St Louis University" with "Not Applicable" when Country = India	To correct institution-country mismatches and ensure contextual accuracy
9	Fixing Corrupted Date Formats	Opportunity End Date, Opportunity Start Date, Learner SignUp DateTime,Apply Date	Applied custom parsing functions to handle mixed formats (e.g., Excel serial numbers like 45314, corrupted strings like 04/12/2024 1095120:00:00, and valid datetimes). Cleaned and standardized all values into datetime64[ns] with consistent "MM/DD/YYYY HH:MM:SS" format.	To ensure accurate temporal analysis, prevent errors in feature engineering (age, duration, engagement days), and maintain consistency across all date fields.
10	Verification	All columns	Checked final missing values, duplicates, date sequences, categorical consistency, and confirmed all date columns are in datetime64 format	To confirm dataset is clean, consistent, and ready for analysis

Total Records & Columns of Cleaned Dataset:

Viewing First Few Rows of the Cleaned Dataset:





	Learner SignUp DateTime	Opportunity Id	Opportunity Name	Opportunity Category	Opportunity End Date	First Name	Date of Birth	Gender	Country	Institution Name	Current/Intended Major	Entry created at	Status Description	Status Code	Apply Date	Opportunity Start Date
0	14-06-2023 12:30	00000000-0GN2-A0AY-7XK8-C5FZPP	Career Essentials: Getting Started With Your P...	Course	29-06-2024 18:52	Faria	12-01-2001 00:00	Female	Pakistan	Northwest Institute Of Health Sciences	Radiology	11-03-2024 12:01	Started	1080	14-06-2023 12:36	03-11-2022 18:30
1	01-05-2023 05:29	00000000-0GN2-A0AY-7XK8-C5FZPP	Career Essentials: Getting Started With Your P...	Course	29-06-2024 18:52	Poojitha	16-08-2000 00:00	Female	India	Saint Louis University	Information Systems	11-03-2024 12:01	Started	1080	01-05-2023 06:08	03-11-2022 18:30
2	09-04-2023 20:35	00000000-0GN2-A0AY-7XK8-C5FZPP	Career Essentials: Getting Started With Your P...	Course	29-06-2024 18:52	Emmanuel	27-01-2002 00:00	Male	United States	Illinois Institute Of Technology	Computer Science	11-03-2024 12:01	Started	1080	09-04-2023 20:35	03-11-2022 18:30
3	29-08-2023 05:20	00000000-0GN2-A0AY-7XK8-C5FZPP	Career Essentials: Getting Started With Your P...	Course	29-06-2024 18:52	Amrutha Varshini	01-11-1999 00:00	Female	United States	Saint Louis University	Information Systems	11-03-2024 12:01	Team Allocated	1070	09-10-2023 22:02	03-11-2022 18:30
4	06-01-2023 15:26	00000000-0GN2-A0AY-7XK8-C5FZPP	Career Essentials: Getting Started With Your P...	Course	29-06-2024 18:52	Vinay Varshith	19-04-2000 00:00	Male	United States	Saint Louis University	Computer Science	11-03-2024 12:01	Started	1080	06-01-2023 15:40	03-11-2022 18:30

You can view the cleaned and processed dataset at the following link:
https://drive.google.com/file/d/1kBs9lEZ9v4ffK6Jl3Jrd6YAPd755DHGD/view?usp=sharing

‘

Techniques For Feature Engineering

Feature engineering is the process of transforming raw data into meaningful features that improve a model's performance and predictive power. It is important because well-engineered features can significantly enhance accuracy and efficiency in machine learning tasks.

Tools for Feature Engineering Using Excel:

New Features	Section	Feature Name (Column Names)	Applied Formula	Formula Representation (Column meanings)	Why Used (Importance)
Creating New Features	Age of Learner	Age	=DATEDIF(G2, TODAY(), "Y")	G2 → Date of Birth	Calculates learner's current age. Helps analyze how engagement varies across age groups (e.g., younger vs. older learners). Useful for demographic insights and tailoring opportunities.
Creating New Features	Engagement Duration	Engagement Days	=ABS(O2 - P2)	O2 → Apply Date, P2 → Opportunity Start Date	Measures lag between application and opportunity start. Identifies quick vs delayed engagement, which helps uncover friction or motivation factors.
Creating New Features	Opportunity Duration	Opportunity Duration	=ABS(E2 - P2)	E2 → Opportunity End Date, P2 → Opportunity Start Date	Captures how long an opportunity lasts. Useful to assess whether longer/shorter opportunities affect learner participation and satisfaction.
Transforming Existing Features	Normalization of Metrics	Normalized Opportunity Duration	=ABS((R2 - MIN(R$2:R$100)) / (MAX(R$2:R$100) - MIN(R$2:R$100)))	R2 → Opportunity Duration	Scales durations between 0–1 for comparability. Prevents large numerical ranges from dominating analysis or models.
Transforming Existing Features	Normalization of Metrics	Normalized Status Code	=ABS((N2 - MIN(N$2:N$100)) / (MAX(N$2:N$100) - MIN(N$2:N$100)))	N2 → Status Code	Normalizes categorical status codes into a 0–1 range, enabling

New Features	Section	Feature Name (Column Names)	Applied Formula	Formula Representation (Column meanings)	Why Used (Importance)
					fair comparison and use in models without scale distortion.
Transforming Existing Features	Encoding Categorical Data	Encoded Gender	=IF(H2="Male", 1, IF(H2="Female", 2, 3))	H2 → Gender	Converts gender into numerical format. Prepares gender data for ML models and prevents text-related inconsistencies.
Transforming Existing Features	Encoding Categorical Data	Encoded Country	=IF(I2="India", 1, IF(I2="United States", 2, IF(I2="Canada", 3, 4)))	I2 → Country	Converts country into numeric codes. Allows easier grouping, analysis, and modeling of country-specific patterns.
Transforming Existing Features	Encoding Categorical Data	Encoded Opportunity Category	=IF(D2="Internship", 1, IF(D2="Training", 2, IF(D2="Scholarship", 3, 4)))	D2 → Opportunity Category	Represents opportunity types numerically. Enables predictive analysis of how different categories impact engagement.
Extracting Useful Components	Date-Based Features	Extracted Month	=MONTH(A2)	A2 → Learner SignUp DateTime	Extracts month of learner sign-up. Useful for trend and seasonality analysis.
Extracting Useful Components	Date-Based Features	Extracted Year	=YEAR(A2)	A2 → Learner SignUp DateTime	Captures the year of sign-up. Helps in long-term trend analysis.
Extracting Useful Components	Date-Based Features	Extracted Day	=DAY(A2)	A2 → Learner SignUp DateTime	Extracts day of the month. Enables fine-grained daily engagement insights.
Extracting Useful Components	Date-Based Features	Weekly Patterns	=WEEKDAY(A2,2)	A2 → Learner SignUp DateTime	Captures day of the week (1=Mon…7=Sun). Helps detect weekday vs weekend sign-up patterns.
Combining Features	Interaction Features	Duration × Age	=ABS(Q2 * R2)	Q2 → Age, R2 → Opportunity Duration	Multiplies learner's age with opportunity duration. Explores

New Features	Section	Feature Name (Column Names)	Applied Formula	Formula Representation (Column meanings)	Why Used (Importance)
					combined effects of demographic and opportunity length on engagement.
Combining Features	Engagement Scores	Engagement Score	=S2*0.4 + Q2*0.2 + AA2*0.3 + AB2*0.1	S2 → Normalized Opportunity Duration, Q2 → Age, AA2 → Engagement Days, AB2 → Duration × Age	Weighted composite score combining multiple engagement metrics. Provides a single holistic measure of overall engagement.
Temporal Analysis	Seasonal Patterns	Seasonal Pattern	=IF(OR(X2=12,X2<=2),"Winter", IF(X2<=5,"Spring", IF(X2<=8,"Summer","Fall")))	X2 → Extracted Month	Groups months into four seasons. Helps identify seasonal peaks/troughs in engagement and optimize scheduling of opportunities.

Note : In few formulas, we used the ABS() function to prevent negative values from appearing in the results. Throughout the process, we ensured that no original data values were altered; instead, we focused on correcting formats and standardizing the dataset through data cleaning.

You can view the featured dataset at the following link:
https://drive.google.com/file/d/1ltU9enu7mf7U72PUcTcBFUWvRRttFFUB/view?usp=sharing

Viewing First Few Rows of the "Featured_Dataset.csv" :
(Only those columns where features are used)

Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE
Age	Opportunit	Normalized Opp	Normalizec	Encoded G	Encoded O	Encoded C	Extracted M	Extracted Y	Extracted D	Weekly Pat	Engagemer	Duration x	Engagemer	Seasonal P
24	222.7538	0.004865498	0.571429	2	4	4	6	2023	14	3	222.7538	5346.092	27.97733	Summer
25	178.4845	0.003878738	0.571429	2	4	1	5	2023	1	1	178.4845	4462.113	23.15	Spring
23	44867.77	1	0.571429	1	4	2	4	2023	9	7	44867.77	1031959	4493.877	Spring
25	340.1473	0.00748219	0.428571	2	4	2	8	2023	29	2	340.1473	8503.681	39.61772	Summer
25	63.88161	0.001324247	0.571429	1	4	2	1	2023	6	5	63.88161	1597.04	12.88869	Winter
29	485.0833	0.010712804	0	1	4	1	3	2024	2	6	485.0833	14067.42	56.11261	Spring
24	222.4577	0.004858896	1	1	4	1	5	2023	31	3	222.4577	5338.984	27.94771	Spring
19	260.5512	0.005707999	0.428571	2	4	4	7	2023	22	6	260.5512	4950.473	31.6574	Summer
19	201.801	0.004398462	0.571429	1	4	1	3	2023	20	1	201.801	3834.22	24.28186	Spring
27	408.3528	0.009002488	0.428571	2	4	1	5	2023	11	4	408.3528	11025.53	47.43888	Spring
26	305.1051	0.006701102	0.428571	2	4	4	9	2023	4	1	305.1051	7932.733	36.01319	Fall
28	88.19416	0.001866172	0.571429	2	4	4	1	2023	28	6	88.19416	2469.436	16.22016	Winter
27	447.0184	0.009864342	0.428571	1	4	2	8	2023	18	5	447.0184	12069.5	51.60579	Summer
28	222.9819	0.00487058	0.571429	1	4	4	6	2023	14	3	222.9819	6243.492	28.80013	Summer
27	6.516424	4.55803E-05	0.571429	1	4	4	1	2023	5	4	6.516424	175.9434	7.251661	Winter
21	149.9918	0.003243637	0.571429	2	4	1	3	2023	16	4	149.9918	3149.828	20.40048	Spring
19	362.1626	0.007972911	0.428571	2	4	2	10	2023	31	2	362.1626	6881.09	40.61945	Fall
26	277.5366	0.006086602	0.428571	2	4	1	8	2023	8	2	277.5366	7215.952	33.5561	Summer
23	301.6541	0.006624179	0.428571	2	4	4	8	2023	15	2	301.6541	6938.043	35.36805	Summer
31	146.2029	0.003159183	0.571429	2	4	4	3	2023	29	3	146.2029	4532.29	21.72155	Spring
31	247.7862	0.005423468	0.428571	2	4	4	7	2023	8	6	247.7862	7681.372	32.78079	Summer
24	349.442	0.007689368	0.428571	1	4	2	10	2023	16	1	349.442	8386.607	40.04727	Fall
28	212.357	0.004633754	0.571429	1	4	1	5	2023	5	5	212.357	5945.997	28.33756	Spring
19	186.494	0.00405727	0.571429	2	4	1	5	2023	9	2	186.494	3543.387	23.05103	Spring
24	4.471539	0	0.571429	2	4	1	1	2023	5	4	4.471539	107.3169	6.447154	Winter

Conclusion

In the first week of the AI Data Powered Analysis Early Internship, we successfully built a strong foundation in data preprocessing and feature engineering. The raw dataset, *"SLU_Opportunity_Wise_Data.csv"*, was carefully explored to identify and address missing values, outliers, duplicates, and inconsistencies. Through systematic data cleaning steps, including handling corrupted date formats, standardizing categorical fields, and ensuring logical consistency in timelines, the dataset was transformed and exported as *"Cleaned_Preprocessed_Dataset.csv"*.

Building on this cleaned dataset, we applied feature engineering techniques to derive meaningful new insights, such as learner age, opportunity duration, engagement days, and seasonal patterns. Normalization and categorical encoding were also implemented to ensure features were consistent and analysis-ready. The engineered dataset was then saved as *"Featured_Dataset.csv"*, providing a structured and enriched resource for advanced engagement analysis.

Overall, Week 1 provided valuable experience in practical data cleaning and transformation using both Python and Excel. The outcomes of this stage—clean data, enriched features, and well-documented processes—serve as a strong foundation for deeper analysis and predictive modeling in the subsequent phases of the internship.

Next Steps for week-2:

- We will examine the cleaned dataset to understand its structure and key features.

- We will identify patterns, trends, and any unusual data points.
- We will create charts and graphs to represent the data visually.
- We will extract important insights and observations from the information.
- We will propose possible explanations or predictions based on the findings.
- We will compile all results, visuals, and hypotheses into an EDA report.
- We will get the dataset ready for AI analysis and predictive modeling in the upcoming weeks.