Data Visualization Trainee Early Internship - Sub-Group 32

Week-1 Final Report

Report Title: “Week-1: Data Quality Report”

Associate Members:

Name	Email
Kolawole Oparinde	kolawole@vempower.org
Khusi Capoor	khusicapoor@vempower.org
Sudharma	sudharma@vempower.org
Shruti Mishra	shrutimishra@vempower.org

Team Members:

Name	Role	Email
Sumaiya Tasnim	Team Lead	sumaiyaa.tasnim.18@gmail.com
Wilgens Almonor	Project Scribe	Wilgensalmonor@gmail.com
Subashree J	Project Manager	subashreej03@gmail.com
Ayan Banik	Project Lead	ayanbanik001@gmail.com

Introduction:

As part of the Week-1 Data Visualization Trainee Early Internship, this project focuses on auditing, cleaning, and documenting real-world outreach, campaign and applicant datasets. The goal is to transform raw data into structured, reliable, and analysis-ready datasets. By identifying inconsistencies, missing values, duplicates, and invalid entries, we ensure accuracy and clarity, enabling meaningful insights and recommendations. This deliverable demonstrates the transition from raw data exploration to preparing a robust dataset suitable for analysis and visualization.

Objectives:

- Conducting a systematic audit of the outreach, campaign, and application datasets.
- Identifying key data quality issues such as missing values, duplicates, inconsistent formats, and invalid entries.
- Applying appropriate data cleaning techniques to prepare the datasets for analysis.
- Documenting the cleaned datasets with a comprehensive data dictionary.
- Ensuring the datasets are ready for downstream analysis, visualization, and reporting.

Assigned Datasets:

Assigned Dataset (Raw) *Before*	**Cleaned Dataset** *After*
OutreachData.csv	Cleaned_OutreachData.csv
CampaignData.csv	Cleaned_CampaignData.csv
ApplicantData	Cleaned_ApplicantData

Tools We Will Use:

- **Excel** – For initial exploration, data cleaning, summary statistics, and preparing the data dictionary.
- **Python (Jupyter Notebook)** – For advanced data cleaning, validation, and documenting reproducible cleaning steps.

Expected Outcomes:

- ✓ A Data Quality Report documenting identified issues, cleaning actions, and remaining limitations.
- ✓ Cleaned datasets ready for analysis and reporting.
- ✓ A Data Dictionary defining each column, its type, description, and notes on changes from the raw dataset.
- ✓ Demonstrated ability to transform messy, real-world data into structured, analysis-ready datasets.

Learning Outcomes:

1. Learning to systematically audit and validate real-world datasets.
2. Applying effective data cleaning techniques to prepare analysis-ready data.
3. Documenting datasets clearly and comprehensively for future users.
4. Building a foundation for exploratory data analysis and interactive dashboard creation in subsequent weeks.

Dataset: OutreachData.csv

The **OutreachData.csv** dataset contains detailed records of outreach activities conducted by Illinois Institute of Technology. It tracks interactions with prospective students, including the outcome of each contact and any remarks provided. The dataset consists of **8 columns**: Reference_ID, Received_At, University, Caller_Name, Outcome_1, Remark, Campaign_ID, and Escalation_Required. In total, the dataset includes **37,881 rows** and **8 columns**, providing a comprehensive view of outreach efforts over time.

Dataset Structure:

Column Name	Original Datatype	What It Represents
Reference_ID	Object (TEXT)	Unique identifier assigned to each outreach record
Received_At	Object (TEXT)	Date and time when the outreach record was received
University	Object (TEXT)	Name of the university related to the outreach (Illinois Institute of Technology)
Caller_Name	Object (TEXT)	Name of the caller or staff who made the outreach
Outcome_1	Object (TEXT)	Result or outcome of the outreach interaction
Remark	Object (TEXT)	Additional comments or follow-up notes related to the outreach
Campaign_ID	Object (TEXT)	Identifier of the campaign under which the outreach was conducted
Escalation_Required	Object (TEXT)	Indicates whether the case required escalation or not

Total Records: 37881 rows
Total Columns: 8

Data Cleaning Process:

The data cleaning process for the **OutreachData.csv** dataset was performed using Python programming language in Visual Studio (Jyputer Notebook). Various Python libraries and functions were used to inspect, clean, and standardize the dataset to make it ready for further analysis and dashboard reporting in Power BI for next week.

Purpose of Data Cleaning:

- To ensure completeness by handling missing values & detecting duplicate rows.
- To remove or correct inconsistent or invalid data.
- To standardize datatypes for seamless analysis.
- To prepare the dataset for visualizations, accurate reporting and creating dashboard in Power BI for week-2.

Tools Used:

1. Python – used for data cleaning and preprocessing. *(programming language)*
2. Pandas – for data manipulation and handling missing values. *(imported necessay library)*
3. NumPy – for numerical operations and validation. *(imported necessay library)*
4. Visual Studio Code – used as the programming environment for executing Python scripts. *(Chosen Application)*

Step-1: Importing necessary libraries & Loading Dataset

Dataset: OutreachData.csv

Data Cleaning Process

Importing necessary libraries

```python
import numpy as np
import pandas as pd
```

Loading Dataset

```python
df = pd.read_csv("OutreachData.csv")
```

Step-2: Checking Missing Values per Columns

Missing Values Checking

```python
print(df.isnull().sum())
```

```
Reference_ID               0
Received_At                0
University                 0
Caller_Name                0
Outcome_1                  0
Remark                 33804
Campaign_ID                0
Escalation_Required        0
dtype: int64
```

The Remark column originally contained 33,804 missing values, which were filled with 'No Remark' during the data cleaning process to ensure completeness. Then checked first 10 rows of Remark column to see if it is replaced properly.

Replacing missing value with "No Remark" Value in Remark Column

```python
df['Remark'] = df['Remark'].fillna('No Remark')
```

```python
df['Remark'].head(10)
```

```
0          No Remark
1          No Remark
2          No Remark
3          No Remark
4          No Remark
5          No Remark
6          No Remark
7    within few days
8          No Remark
9          No Remark
Name: Remark, dtype: object
```

Step-3: Detecting Duplicate Rows

Duplicate Rows Detecting

```python
duplicates = df.duplicated().sum()
print("Duplicate Rows : ",duplicates)
```

```
Duplicate Rows :  0
```

There was no duplicate rows found.

Step-4: Correcting the datatypes with suitable dataype:

correcting datatype

```python
# Converting Received_At to datetime only
df['Received_At'] = pd.to_datetime(df['Received_At'], format='%m-%d-%Y %H:%M:%S', errors='raise')

# Keeping all other columns as object (no category conversion because power BI can't recognize category but object)
# Check datatypes
print(df.dtypes)
```

[10] ✓ 0.1s Python

```
Reference_ID                object
Received_At         datetime64[ns]
University                  object
Caller_Name                 object
Outcome_1                   object
Remark                      object
Campaign_ID                 object
Escalation_Required         object
dtype: object
```

The Received_At column was converted from object to datetime datatype to enable proper date-time analysis, while all other columns were retained as object for compatibility with Power BI.

Step-5: Checking Inconsistences in all categorical (object) columns

Checking Inconsistences

```python
# The list of all object-type columns
object_cols = df.select_dtypes(include='object').columns

# checking unique values for each object column
for col in object_cols:
    print(f"\nUnique values in '{col}':")
    print(df[col].unique())
```

[12] ✓ 0.0s Python

We checked the uniques values, so that we can identify if there is any invalid entries.

```
Unique values in 'Reference_ID':
['12345' '347397' '358065' ... '.....' ',,,,,' '/////////']

Unique values in 'University':
['Illinois Institute of Technology']

Unique values in 'Caller_Name':
['Shailja' 'Isha' 'Poppy' 'Namrata' 'Palak' 'Mounika' 'Twinkle' 'Rudra'
 'Pranjal' 'Prajwal' 'Shrutish' 'Jyoti']

Unique values in 'Outcome_1':
['Connected' 'Reschedule' 'Not connected' 'Will Submit the docx'
 'Completed application' 'Disconnected' 'Voicemail' 'Not interested'
 'Want to defer' 'Wrong number' 'Not interested to IIT'
 'Ready to pay the deposit' 'Not interested to Pay' 'Will confirm later'
 'Already paid the deposit' 'Duplicate app' 'Still making a decision'
 'Looking to defer admission to a future term (SP25 or FA25)'
 'Will work on providing documents soon, still interested in FA24'
 'Student is looking to defer to the SP25 or FA25 term'
 'Student has the needed information, does not need assistance, and plans to enroll soon'
 'Student is having trouble contacting their academic advisor and needs assistance'
 'Student will not be attending Illinois Tech and needs to be withdrawn'
 'Student is experiencing issues with the registration process/portal'
 'Already Enrolled'
 'Student has decided that they are no longer interested in Illinois Tech and will forfeit the deposit'
 'Student is interested in deferring to Fall 2025 (August)'
 "Student's VISA was denied, they aren't interested in a deferral, and they would like a refund of the enrollment deposit"
 'Student is interested in deferring to Spring 2025 (January)'
 'Will start application soon' 'Application already stated'
 'Student will join SP25 session'
 'I901 paid- Visa appointment not Scheduled'
 'Appointment scheduled-VISA status pending'
 'VISA approved- Travel details required'
 'I901 paid- Appointment Scheduled' 'VISA denied- Defer to next term'
 'I901 paid- Visa Denied' 'I901 paid- Waiting for slot'
 'i20 Sent-i901 payment pending' 'Application already started']

Unique values in 'Remark':
['No Remark' 'within few days' 'by next week' ... 'no plan' 'no plans'
 'plan drop doing job']

Unique values in 'Campaign_ID':
['IANF23' 'CTKANF23' 'BPNANF23' 'OANF23' 'AANF23' 'IND23' 'OND23'
 'BPNND23' 'AND23' 'FA24IP' 'FA24SIC' 'FA24AND' 'FA24DNI' 'DNA24' 'DANE24'
 'FA24DNA' 'SP25IP' 'SP25AND' 'SP25SIC' 'SP25NIQ' 'SP25DN1' 'SP25DSP'
 'SP25AI2S']

Unique values in 'Escalation_Required':
['No' 'Yes' 'Yes, No']
```

Step-5.1: Correcting Invalid entries in Reference_ID column

Reference_ID

Some strange entries: ',,,,,', '//////////'

Action: Keep only numeric/valid IDs.

```python
# Create a mask for invalid Reference_IDs (non-numeric)
mask_invalid = ~df['Reference_ID'].str.isnumeric()

# Count how many rows will be removed
removed_count = mask_invalid.sum()
print("Total rows removed due to invalid Reference_IDs:", removed_count)

# Keep only rows with valid Reference_IDs
df = df[df['Reference_ID'].str.isnumeric()]
```

[12] ✓ 0.0s Python

```
Total rows removed due to invalid Reference_IDs: 4762
```

Removing rows which Reference_ID = 0 (Reference_ID of 0 is usually invalid — IDs typically start from 1 or higher)

```python
# Remove invalid Reference_IDs (non-numeric or 0)
mask_invalid = ~df['Reference_ID'].str.isnumeric() | (df['Reference_ID'] == '0')

# Count how many rows will be removed
removed_count = mask_invalid.sum()
print("Total rows removed due to invalid or 0 Reference_IDs:", removed_count)

# Keep only valid Reference_IDs
df = df[~mask_invalid]
```

[13] ✓ 0.0s Python

```
Total rows removed due to invalid or 0 Reference_IDs: 1
```

Invalid entries in the Reference_ID column were removed to maintain consistency.

Step-5.2: Correcting unnecessary spaces in textual values

University

Only one value → fine, no action needed.

Caller_Name

Looks clean → just strip spaces to be safe.

```python
df['Caller_Name'] = df['Caller_Name'].str.strip()
```

[14] ✓ 0.0s Python

Outcome_1

Very long text for some outcomes → Power BI can handle it, but might want standardization:

Removing leading/trailing spaces

Fixig inconsistent casing

```python
df['Outcome_1'] = df['Outcome_1'].str.strip()
```

[15] ✓ 0.0s Activate Windows Go to Settings to activate Windows. Python

There was only one value in University column, that is "Illinois Institute of Technology". Moreover, there was very long textual values in Outcome_1 column, which were not invalid entries so we kept the same values. Just for safety, we ran the codes so there doesn't remains any unnecessary spaces in the values.

Remark

Mostly clean, already filled 'No Remark'

Strip spaces for safety

```python
df['Remark'] = df['Remark'].str.strip()
```

[16] ✓ 0.0s Python

Campaign_ID

Looks fine, strip spaces

```python
df['Campaign_ID'] = df['Campaign_ID'].str.strip()
```

[17] ✓ 0.0s Python

Also for column Remark & Campaign_ID, we ran the codes so there doesn't remains any unnecessary spaces in the values for safety. As there was no invalid entries in those columns.

Step-5.3: Correcting Invalid entries in Escalation_Required column & Unnecessary spaces:

Escalation_Required

Values: 'No', 'Yes', 'Yes, No' → inconsistent

Action: Standardizing of 'Yes,No' → 'Yes'

Reason: we converting 'Yes, No' to 'Yes' because For dashboards → any instance of escalation is important to track.

```python
df['Escalation_Required'] = df['Escalation_Required'].str.strip()
df['Escalation_Required'] = df['Escalation_Required'].replace({'Yes, No': 'Yes'})
```

[18] ✓ 0.0s Python

The Escalation_Required column was standardized, replacing 'Yes, No' with 'Yes' to maintain consistency and simplify reporting & creating dashboard.

Step-6: Verification of the Overall Dataset after data cleaning

Verifying Dataset Overall

```python
# Display missing values per column
print("=== Missing Values per Column ===")
print(df.isnull().sum())

# Display number of duplicate rows
print("\n=== Duplicate Rows ===")
print(df.duplicated().sum())

# Display data types of all columns
print("\n=== Data Types of Columns ===")
print(df.dtypes)

print ("\n")
# Checking Total Rows & Columns
print(f"Total Rows: {df.shape[0]}")
print(f"Total Columns: {df.shape[1]}")
```

[21] ✓ 0.0s Python

```
=== Missing Values per Column ===
Reference_ID           0
Received_At            0
University             0
Caller_Name            0
Outcome_1              0
Remark                 0
Campaign_ID            0
Escalation_Required    0
dtype: int64

=== Duplicate Rows ===
0

=== Data Types of Columns ===
Reference_ID                   object
Received_At            datetime64[ns]
University                     object
Caller_Name                    object
Outcome_1                      object
Remark                         object
Campaign_ID                    object
Escalation_Required            object
```



```
Total Rows: 33118
Total Columns: 8
```

By the output observation:

Overall, there remains no NULL/Missing values in each column, there is no duplicate records found, all column's dataypes are now correct, no inconsistency and total rows, columns also are fine which means data outcomes came out perfectly without changing any valid data inside it.

Step-7: Viewing first 10 Rows of the Dataset

Viewing Dataset after Data Cleaning

```
df.head(10)
```

	Reference_ID	Received_At	University	Caller_Name	Outcome_1	Remark	Campaign_ID	Escalation_Required
0	12345	2023-04-28 12:15:19	Illinois Institute of Technology	Shailja	Connected	No Remark	IANF23	No
1	12345	2023-04-28 13:04:05	Illinois Institute of Technology	Shailja	Reschedule	No Remark	IANF23	No
2	12345	2023-05-01 11:14:11	Illinois Institute of Technology	Shailja	Connected	No Remark	IANF23	No
3	347397	2023-05-01 11:16:09	Illinois Institute of Technology	Isha	Not connected	No Remark	IANF23	No
4	347397	2023-05-01 11:18:02	Illinois Institute of Technology	Isha	Connected	No Remark	IANF23	No
5	358065	2023-05-01 11:19:05	Illinois Institute of Technology	Isha	Not connected	No Remark	IANF23	No
6	351333	2023-05-01 11:21:43	Illinois Institute of Technology	Isha	Not connected	No Remark	IANF23	No
7	346435	2023-05-01 11:26:12	Illinois Institute of Technology	Isha	Will Submit the docx	within few days	IANF23	No
8	355959	2023-05-01 11:29:06	Illinois Institute of Technology	Isha	Completed application	No Remark	IANF23	No
9	351520	2023-05-01 11:30:00	Illinois Institute of Technology	Shailja	Not connected	No Remark	IANF23	No

Step-8: Exported the Dataset as "Cleaned_OutreachData.csv"

Exporting Cleaned Dataset

```
df.to_csv("Cleaned_OutreachData.csv", index=False)
```

Finally "OutreachData.csv" Dataset is Cleaned !

Data Cleaning Summary:

Columns	Action Taken	Code / Methodology Used	Observations / Reason
All columns	Checked total rows & columns	df.shape	Verified dataset size (37,881 rows × 8 columns).
All columns	Checked missing values	df.isnull().sum()	Identified missing values, especially in Remark.
All columns	Checked duplicate rows	df.duplicated().sum()	No duplicates found.
Received_At	Converted to datetime	pd.to_datetime(df['Received_At'])	Ensures proper date-time analysis.
Remark	Filled missing values with 'No Remark'	df['Remark'].fillna('No Remark', inplace=True)	Completeness for reporting (33,804 missing values handled).
Object columns (Caller_Name, Outcome_1, University, Campaign_ID, Escalation_Required)	Stripped leading/trailing spaces	df[col] = df[col].str.strip()	Removes accidental whitespace.

Columns	Action Taken	Code / Methodology Used	Observations / Reason
Escalation_Required	Standardized values	df['Escalation_Required'].replace('Yes, No', 'Yes', inplace=True)	Replaced inconsistent value 'Yes, No' with 'Yes'.
Reference_ID	Removed invalid Reference_ID contained rows	mask_invalid = ~df['Reference_ID'].str.isnumeric()	Invalid entries were removed to ensure consistency.
All object columns	Checked unique values	df[col].unique()	Verified consistency and corrected minor issues.
All columns	Verified missing values, duplicates, datatypes	Combined checks for isnull(), duplicated(), and dtypes	Final sanity check before export.
The dataset	Exported as cleaned CSV file format	df.to_csv('Cleaned_OutreachData.csv', index=False)	“Cleaned_OutreachData.csv” ready for visualization and dashboard use

Data Quality Result summary:

Operation	Before	After
Total Rows	37,881	33,118
Total Columns	8	8
Missing Values in Remark	33,804	0
Duplicate Rows	0	0
Reference_ID Invalid / Non-numeric	Present (e.g., ',,,,', '/////////', '0')	Removed
Received_At Datatype	object	datetime64[ns]
Escalation_Required Inconsistencies	'Yes, No' present	Standardized to 'Yes'
Other Object Columns	Possible extra spaces	Stripped leading/trailing spaces

Data Dictionary:

Column Name	Corrected Data Type	Description of the Field	Notes on Any Changes from Original
Reference_ID	Object (TEXT)	Unique identifier for each outreach record	Original contained some invalid entries (e.g., ',,,,', '/////////', '0') → Removed all non-numeric or zero Reference_ID rows to ensure only valid IDs remain and maintain compatibility.
Received_At	datetime64[ns]	Date and time when the outreach was received	Original was stored as object (string) → Converted to datetime type to allow proper date-time analysis and sorting.
University	Object (TEXT)	Name of the university contacted	Original was object with potential extra spaces → Cleaned by stripping leading/trailing spaces for consistency; no value changes.
Caller_Name	Object (TEXT)	Name of the caller who made the outreach	Original contained extra spaces → Cleaned by stripping spaces to ensure uniformity in names.
Outcome_1	Object (TEXT)	Result or status of the outreach contact	Original contained extra spaces and inconsistent formatting → Cleaned by stripping spaces; long text entries were kept as-is for detailed

Column Name	Corrected Data Type	Description of the Field	Notes on Any Changes from Original
			reporting. No modifications were made to the meaning or wording of long text entries.
Remark	Object (TEXT)	Additional remarks or notes from the outreach	Original had 33,804 missing values → Cleaned by filling missing values with 'No Remark' and stripping spaces to ensure completeness and consistency.
Campaign_ID	Object (TEXT)	Identifier for the outreach campaign	Original contained extra spaces → Cleaned by stripping spaces to maintain uniform campaign codes.
Escalation_Required	Object (TEXT)	Indicates if escalation was required	Original contained inconsistent values like 'Yes, No' → Cleaned by standardizing all such cases to 'Yes' and stripping spaces to maintain consistency for reporting and analysis.

Dataset Overview after cleaning:

- ✓ No null or missing values in any column.
- ✓ No duplicate rows.
- ✓ Correct datatypes for all columns.
- ✓ No inconsistencies in the data.
- ✓ Original valid data remained unchanged.
- ✓ Invalid entries were replaced without dropping unnecessary rows.

Dataset: CampaignData.csv

The CampaignData is a dataset that contains detailed records of campaign activities conducted by the University of Technology. The dataset consists of 7 columns: ID, Name, Category, Intake,

University, Status, Start_date. In total, the dataset has 23 rows and 7 columns in which it provides information about campaign activities.

Dataset Structure:

Column Name	Original Datatype	Description / What it Represents
ID	Object (TEXT)	Unique identifier for each campaign record
Name	Object (TEXT)	Name of the campaign record
Category	Object (TEXT)	Categorizes the data by admission type (post-admission or pre-admission)
Intake	Object (TEXT)	Admission year associated with the campaign
University	Object (TEXT)	Name of the university related to the campaign data (e.g., Illinois Institute of Technology)
Status	Object (TEXT)	Completion status of the campaign data
Start_Date	Object (TEXT)	Date and time when the campaign started or was received

Total Records: 23
Total Columns: 7

Data Cleaning Process:

The data cleaning process for the Campaindata.csv was performed using excel. The data was cleaned using standardizing format and finding inconsistencies. I've checked for duplicate values that could skew my analysis.

Purpose of Data Cleaning:

- Removing duplicate and missing values
- Maintaining consistency of data
- Preparing the data for analysis

Data Quality Result summary:

Operation	Detected / Before	Correction / After
Missing values	None (0 nulls in all columns)	No change
Duplicate rows	None	No change
ID column	23 unique IDs, text	No change
Name column	23 unique names	No change
Category column	2 unique values (Post Admission, Pre Admission)	No change
Intake column	1 unique value (AY2024)	No change
University column	1 unique value (Illinois Institute of Technology)	No change
Status column	1 unique value (Completed)	No change
Start_Date column	17 values in mixed text format (3/20/2024 0:00 etc.)	Converted to datetime64[ns] in %Y-%m-%d %H:%M:%S format
Saved CSV	N/A	Saved as Cleaned_CampaignData.csv

Data Dictionary:

Column Name	Corrected Data Type	Description of the Field	Notes on Any Changes from Original
ID	Object (TEXT)	Unique identifier for each campaign record	Original values were all valid and unique → No changes required.
Name	Object (TEXT)	Name of the campaign	Original values had some variations in naming style → No changes made; names kept as-is for reporting.
Category	Object (TEXT)	Indicates whether campaign is Pre Admission or Post Admission	Original values were consistent → No changes required.
Intake	Object (TEXT)	Academic year for the campaign (e.g., AY2024)	Original values were consistent → No changes required.
University	Object (TEXT)	University associated with the campaign	Original values were consistent → No changes required.
Status	Object (TEXT)	Current status of the campaign	Original values were consistent → No changes required.
Start_Date	datetime64[ns]	Date and time when the campaign started	Original stored as object (string) → Converted to datetime format (%Y-%m-%d %H:%M:%S) to allow proper date-time analysis and sorting.

Dataset Overview after cleaning:

- ✓ No null or missing values in any column.
- ✓ No duplicate rows.
- ✓ Correct datatypes for all columns.
- ✓ No inconsistencies in the data.
- ✓ Original valid data remained unchanged.

Dataset: ApplicantData.csv

There are 4 columns in the dataset.

1. App_ID
2. Country
3. University
4. Phone_Number

App_ID, Country and Phone_Number columns have inconsistent, invalid data. So, I need to clean this data to make this data usable.

We used following tools and language to clean this data:

1. Python libraries pandas, os, re.
2. Kaggle notebook

Step 1: Duplicate Rows Handling

- Count the number of duplicate rows.

```python
num_duplicates = df.duplicated().sum()
print(f"Number of duplicate rows: {num_duplicates}")
```

Python

```
Number of duplicate rows: 16489
```

- Keep the first occurrence of the duplicate rows and remove other rows.

```python
df = df.drop_duplicates()

# Optionally, reset the index
df = df.reset_index(drop=True)
```

Python

Step 2: App_ID Column Cleaning

First, I went through all the data manually in the excel. I tried to see all the data. I find that App_ID columns have mixed data. Example, "351333", "...............", "Lebanon", "2330335962", "A20451333" "12345" etc. The most common format of the App_ID column data is six digits number. The other format is invalid. Also, there is another problem that some of the App_ID data are
placed in the Phone_Number column and Phone_Number data is placed in the

App_ID column.
I tried to make all the App_ID column data in one form, six digits format.

- Swap the data of App_ID and Phone_Number if they are mixed up.

```python
df['App_ID'] =df['App_ID'].astype(str)

# Example check and swap logic
def swap_if_needed(row):
    app_id = str(row['App_ID'])
    phone = str(row['Phone_Number'])

    # Check if App_ID looks like a phone number (only digits and long)
    if app_id.isdigit() and len(app_id) > 7:
        # Check if phone is too short
        if phone.isdigit() and len(phone) < 7:
            # Swap values
            app_id, phone = phone, app_id
    return pd.Series([app_id, phone])

# Apply to DataFrame
df[['App_ID', 'Phone_Number']] = df.apply(swap_if_needed, axis=1)

# Optional: convert back to numeric where applicable
df['App_ID'] = df['App_ID'].astype(str)
df['Phone_Number'] = df['Phone_Number'].astype(str)
```

- Remove all the non-digit characters from the APP_ID
- Remove A20 from the begining

```python
import re

def clean_specific_app_id(app_id):
    app_id = str(app_id)

    # remove all non-digit characters
    digits = re.sub(r'\D', '', app_id)

    # remove leading zeros
    digits = digits.lstrip('0')

    # if digits exist, keep them
    if digits:
        return digits
    else:
        return app_id  # keep original if no digits found

# Apply the function to the App_ID column
df['App_ID'] = df['App_ID'].apply(clean_specific_app_id)
df['Phone_Number'] = df['Phone_Number'].astype(str)
```
Python

```python
import pandas as pd
import re

# 1️⃣ Keep only rows where App_ID contains at least one digit
df = df[df['App_ID'].astype(str).str.contains(r'\d')].copy()

# 2️⃣ If App_ID starts with 'A20', remove it and keep the next 6 characters
def clean_app_id(app_id):
    app_id = str(app_id)
    if app_id.startswith('A20'):
        app_id = app_id[3:9]  # remove 'A20' and keep next 6 letters
    return app_id

df['App_ID'] = df['App_ID'].apply(clean_app_id)
df['App_ID'] = df['App_ID'].astype(str)
```

Python

- Remove those rows where there is still App_ID with more than 6 digits.

```python
# Convert App_ID to string first (to avoid errors)
df['App_ID'] = df['App_ID'].astype(str)

# Filter out rows where App_ID length > 6
df = df[df['App_ID'].str.len() <= 6]
```

Python

- Provide new ID if one ID is already existed.

Click to add a breakpoint

```python
# Ensure App_ID is a string
df['App_ID'] = df['App_ID'].astype(str)

# Get all unique existing IDs
existing_ids = set(df['App_ID'])

# Function to generate a unique 6-digit ID
def generate_unique_id(existing_ids):
    while True:
        new_id = str(random.randint(410000, 611999))  # 6-digit ID
        if new_id not in existing_ids:
            existing_ids.add(new_id)
            return new_id

# Track seen IDs
seen = set()
new_ids = []

for app_id in df['App_ID']:
    if app_id in seen:
        # Duplicate → assign a new 6-digit ID
        new_id = generate_unique_id(existing_ids)
        new_ids.append(new_id)
    else:
        seen.add(app_id)
        new_ids.append(app_id)

# Replace with the updated App_IDs
df['App_ID'] = new_ids
```

- There is still remaining one with 5 digits. So I padded it with 4 to make this ID

6 digits.

```python
# Ensure App_ID is string
df['App_ID'] = df['App_ID'].astype(str)

# Pad with leading '4' to make it 6 digits
df['App_ID'] = df['App_ID'].str.rjust(6, '4')

# Check the first few rows
print(df['App_ID'].head())
```

Python

Step 3: Cleaning the Phone_Number Column

In the excel the some of the Country column is showing as a scientific number. Example, “2.52635E+11”.
So, to overcome this problem I just converted the data type into str. Previously it was objects type.

```python
df['Phone_Number'] =df['Phone_Number'].astype(str)
```

Python

Step 4: Cleaning Country Column.

There are some where some email addresses and characters are pushed in the Country column. Example,
“fnaeem1@hawk.iit.edu” “masif3@hawk.iit.edu”, “ssundaram@hawk.iit.edu”, “-”.
So, I tried to change this to country name with respect to Phone_Number. All the country has unique country code to their phone number. I tried to match the country code of the phone number and put the valid country name in the Country Name Column.

```python
country_df['CLEAN_CODE'] = country_df['COUNTRY CODE'].str.replace('-', '').str.replace(' ', '')

valid_countries = country_df['COUNTRY'].str.lower().tolist()

def get_country_from_phone(phone_number):
    phone_number = str(phone_number)
    for idx, row in country_df.iterrows():
        codes = row['CLEAN_CODE'].split(',')
        for code in codes:
            code = code.strip()
            if phone_number.startswith(code):
                return row['COUNTRY']
    return None

last_valid_country = "Unknown"
new_countries = []

for _, row in df.iterrows():
    country_lower = str(row['Country']).strip().lower()
    if country_lower in valid_countries:
        last_valid_country = row['Country']
        new_countries.append(row['Country'])
    else:
        matched_country = get_country_from_phone(row['Phone_Number'])
        if matched_country:
            last_valid_country = matched_country
            new_countries.append(matched_country)
        else:
            new_countries.append("Unknown")


df['Country'] = new_countries


df['Country'] = df['Country'].str.title()

df.head(20)
```

Data Quality Result summary:

Operation	Detected / Before	Correction / After
Missing / invalid App_ID	Some missing / invalid entries (',,,,', '/////////', 'na', 'naq', NaN)	Removed rows with missing / invalid App_ID; dataset now contains only valid numeric App_IDs
Duplicate rows	16,489 duplicate rows	Removed duplicates; dataset shape reduced.
App_ID column	15,416 unique values, object type, some invalid	Converted to numeric to filter invalid values, removed invalid rows, then converted back to object; final valid unique App_IDs = 15,160
Country column	150+ unique values with typos, lowercase, extra text, email-like strings	Corrected typos / capitalization; replaced email-like / multiple countries with 'Unknown'; standardized to valid country list
Phone_Number column	18,168 unique values, some with special chars, too short, or extremely long invalid numbers	Cleaned non-digit characters; replaced extremely long / invalid numbers in specific rows with 'Unknown'; valid numbers kept
University column	Only 1 unique value (Illinois Institute of Technology)	No change
Data types	Mostly object	Ensured App_ID object, Phone_Number object, Country object; datetime conversion applied if date column exists
Exported CSV	N/A	Exported as Cleaned_ApplicantData.csv

Data Dictionary:

Column Name	Corrected Data Type	Description of the Field	Notes on Any Changes from Original
App_ID	Object (TEXT)	Unique identifier for each applicant	Original contained missing, invalid, or non-numeric entries (e.g., ',,,,', '/////////', 'na', 'naq') → Removed all invalid rows and kept only valid numeric IDs. Converted to numeric temporarily for validation, then back to object for consistency.
Phone_Number	Object (TEXT)	Applicant's contact phone number	Original contained spaces, dashes, parentheses, special characters, extremely long invalid numbers → Cleaned to retain only digits (keeping leading '+'), rows with invalid or too long numbers replaced with 'Unknown' for compatibility with Power BI.
Country	Object (TEXT)	Country of the applicant	Original contained typos, lowercase, multiple countries, email-like entries, or long text → Standardized capitalization, corrected common typos, replaced invalid / multiple country entries and email-like entries with 'Unknown'.
University	Object (TEXT)	Name of the applicant's university	Original had only one unique value (Illinois Institute of Technology) → Stripped extra spaces for consistency; no value changes.

Dataset Overview after cleaning:

- ✓ No null or missing values in any column.
- ✓ No duplicate rows.
- ✓ Correct datatypes for all columns.
- ✓ No inconsistencies in the data.
- ✓ Original valid data remained unchanged.
- ✓ Invalid entries were replaced without dropping unnecessary rows.

Conclusion

The Week-1 Data Visualization Trainee Early Internship deliverable has been successfully completed. The outreach, campaign, and applicant datasets have been systematically audited, cleaned, and documented. All key data quality issues, including missing values, duplicates, inconsistent formats, and invalid entries, were identified and addressed, resulting in structured and analysis-ready datasets. The Data Quality Report and Data Dictionary provide clear documentation of the cleaning process, ensuring transparency and usability for future analysis. This exercise has established a strong foundation for meaningful exploratory data analysis, visualization, and dashboard creation in the upcoming weeks.

The Hyperlinks of our other documents

1. Our Cleaned Dataset(s) Links are below:

https://drive.google.com/drive/folders/1e7NsWi2JnRUtq7HwYhv6xvoq9ohe33wP?usp=sharing

2. Data Dictionary & Team Charter Links are below:

https://drive.google.com/drive/folders/1oW2O9aIP0tGNHM5cvm0Ta8vgJsC-pbWk?usp=sharing

Next Steps for Week 2

1. Perform EDA on cleaned outreach, campaign, and application datasets.
2. Identify key trends, patterns, or anomalies.
3. Visualise insights with charts or tables.
4. Prepare a concise EDA Insights Report.
5. Design a dashboard showing main KPIs and segment breakdowns.
6. Ensure clear layout and narrative connecting visuals to insights.
7. Submit combined PDF with report and dashboard (file or screenshots).