FIT3163 Data Science Project Part 1

Data Analysis Report

An Automated Health Information System

MDS02

29th April 2024

Import required Libraries

In [1]:
```
import os
import cv2
import zipfile
import numpy as np
import pandas as pd
import seaborn as sns
from io import TextIOWrapper
from matplotlib import pyplot as plt
from collections import defaultdict
```

Set working directory

In [2]:
```
os.chdir("C:\\Users\\Nicol Foo\\Downloads")
os.getcwd()
```

Out[2]:
```
'C:\\Users\\Nicol Foo\\Downloads'
```

Open the dataset zip file

In [3]:
```
# Open the zip file
with zipfile.ZipFile("KaggleDataset.zip", "r") as zip_ref:
    # Get the list of files in the zip file
    file_list = zip_ref.namelist()

    # Loop through each file in the zip file
    for file_name in file_list:
        # Open the file within the zip file
        with zip_ref.open(file_name) as file:
            # Read the content of the file into memory
            file_content = file.read()
```

Obtain basic file information of the dataset zip file

In [4]:
```
# Zip file name
zip_file_path = "KaggleDataset.zip"

# Dictionary to store directory structure and top 5 files for display
directory_structure = defaultdict(list)

# Open the zip file
with zipfile.ZipFile(zip_file_path, "r") as zip_ref:
```

```
    # Get the list of files and directories in the zip file
    file_list = zip_ref.namelist()

    # Iterate over each file/directory
    for file_path in file_list:
        # Split the file path into directory and file name
        components = file_path.split("/")

        # If it's a file in the main directory
        if len(components) == 1:
            directory = ''  # Main directory
            filename = components[0]  # File name
        else:
            directory = '/'.join(components[:-1])  # Get the directory
            filename = components[-1]               # Get the filename

        # Add the file to the directory structure
        directory_structure[directory].append(filename)

# Print the directory structure and top 5 files in each directory
for directory, files in directory_structure.items():
    print(f"Directory: {directory}")
    print("Top 5 files:")
    for file_name in files[:5]:
        print(file_name)
    print()
```

```
Directory: test_v2/test
Top 5 files:
TEST_0001.jpg
TEST_0002.jpg
TEST_0003.jpg
TEST_0004.jpg
TEST_0005.jpg

Directory: train_v2/train
Top 5 files:
TRAIN_00001.jpg
TRAIN_00002.jpg
TRAIN_00003.jpg
TRAIN_00004.jpg
TRAIN_00005.jpg

Directory: validation_v2/validation
Top 5 files:
VALIDATION_0001.jpg
VALIDATION_0002.jpg
VALIDATION_0003.jpg
VALIDATION_0004.jpg
VALIDATION_0005.jpg

Directory:
Top 5 files:
written_name_test_v2.csv
written_name_train_v2.csv
written_name_validation_v2.csv
```

From the code output results above, we can see that there are 3 separate folders named "test_v2", "train_v2" and "validation_v2" and 3 CSV files inside the main directory named "written_name_test_v2.csv", "written_name_train_v2.csv" and "written_name_validation_v2.csv".

The folders "test_v2", "train_v2" and "validation_v2" further divide to one other folder each which are "test", "train" and "validation".

In [5]:
```
# List of CSV file names
csv_files_to_read = ["written_name_test_v2.csv", "written_name_train_v2.csv", "written_name_

# Open the zip file
with zipfile.ZipFile(zip_file_path, "r") as zip_ref:
    # List to store DataFrame for each CSV file
    csv_dataframes = []

    # Iterate over each CSV file name
    for csv_file_name in csv_files_to_read:
        # Open the CSV file within the zip file
        with zip_ref.open(csv_file_name) as csv_file:
            # Read the CSV file using Pandas
            df = pd.read_csv(TextIOWrapper(csv_file, 'utf-8'))
            # Append the DataFrame to the list
            csv_dataframes.append(df)

# Print the shape, datatypes and columns of each CSV file
for csv_file_name, df in zip(csv_files_to_read, csv_dataframes):
    print(f"File: {csv_file_name}")
    print(f"  Shape: {df.shape}")
    print("  Datatypes:")
    for column, dtype in df.dtypes.items():
        print(f"    {column}: {dtype}")
    print("  Column Names:")
    for column in df.columns:
        print(f"    {column}")
    print()
```

```
File: written_name_test_v2.csv
  Shape: (41370, 2)
  Datatypes:
    FILENAME: object
    IDENTITY: object
  Column Names:
    FILENAME
    IDENTITY

File: written_name_train_v2.csv
  Shape: (330961, 2)
  Datatypes:
    FILENAME: object
    IDENTITY: object
  Column Names:
    FILENAME
    IDENTITY

File: written_name_validation_v2.csv
  Shape: (41370, 2)
  Datatypes:
    FILENAME: object
    IDENTITY: object
  Column Names:
    FILENAME
    IDENTITY
```

The output of the code above gave the basic information of each of the CSV files which are the dimensions of the dataset, datatypes of cell under each column and the column names.

In [6]:
```
# Describe each CSV file
for csv_file_name, df in zip(csv_files_to_read, csv_dataframes):
    print(f"File: {csv_file_name}")
```

```
    print(df.describe())
    print()
```

```
File: written_name_test_v2.csv
             FILENAME IDENTITY
count           41370    41300
unique          41370    20279
top     TEST_0001.jpg   THOMAS
freq                1      227

File: written_name_train_v2.csv
               FILENAME IDENTITY
count            330961   330396
unique           330961   100539
top     TRAIN_00001.jpg   THOMAS
freq                  1     1825

File: written_name_validation_v2.csv
                 FILENAME IDENTITY
count                41370    41292
unique               41370    20227
top     VALIDATION_0001.jpg   THOMAS
freq                     1      219
```

Based on the results from the code output above, we can get the count, unique values, first data and the frequency of the unique names of the dataset.

In [7]:
```
# Basic information of each CSV file
for csv_file_name, df in zip(csv_files_to_read, csv_dataframes):
    print(f"File: {csv_file_name}")
    print(df.info())
    print()
```

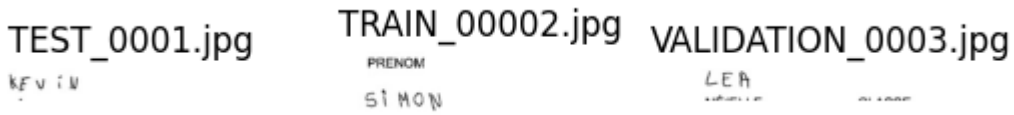
```
File: written_name_test_v2.csv
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41370 entries, 0 to 41369
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   FILENAME  41370 non-null  object
 1   IDENTITY  41300 non-null  object
dtypes: object(2)
memory usage: 646.5+ KB
None

File: written_name_train_v2.csv
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 330961 entries, 0 to 330960
Data columns (total 2 columns):
 #   Column    Non-Null Count   Dtype 
---  ------    --------------   ----- 
 0   FILENAME  330961 non-null  object
 1   IDENTITY  330396 non-null  object
dtypes: object(2)
memory usage: 5.1+ MB
None

File: written_name_validation_v2.csv
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41370 entries, 0 to 41369
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   FILENAME  41370 non-null  object
 1   IDENTITY  41292 non-null  object
dtypes: object(2)
memory usage: 646.5+ KB
None
```

View image dataset

In [8]:

```python
# Dictionary to store directory structure and top 5 files for display
directory_structure = defaultdict(list)

# Open the zip file
zip_ref = zipfile.ZipFile(zip_file_path, "r")
# Get the list of files and directories in the zip file
file_list = zip_ref.namelist()

# Iterate over each file/directory
for file_path in file_list:
    # Split the file path into directory and file name
    components = file_path.split("/")

    # If it's a file in the main directory
    if len(components) == 1:
        directory = ''  # Main directory
        filename = components[0]  # File name
    else:
        directory = '/'.join(components[:-1])  # Get the directory
        filename = components[-1]               # Get the filename

    # Add the file to the directory structure dictionary
    directory_structure[directory].append(filename)

# Display images
for i, (directory, files) in enumerate(directory_structure.items()):
```

```
    # Display up to 3 images
    if i >= 3:
        break

    ax = plt.subplot(2, 3, i+1)

    # Extract and read the image
    img_data = zip_ref.read(directory + "/" + files[i])
    img_array = cv2.imdecode(np.frombuffer(img_data, np.uint8), cv2.IMREAD_GRAYSCALE)

    plt.imshow(img_array, cmap='gray')
    plt.title(files[i], fontsize=12)
    plt.axis('off')

plt.subplots_adjust(wspace=0.2, hspace=-0.8)
plt.show()

# Close the zip file
zip_ref.close()
```



TEST_0001.jpg is the first testing image data from the testing dataset. We can see that the word written is kEViN but in mixed capital and small letters.

TRAIN_00002.jpg is the second training image data from the training dataset. We can see that the word written is SiMON but in mixed capital and small letters.

VALIDATION_0003.jpg is the third validation image data from the validation dataset. We can see that the word written is LEA in all capital letters.

Missing values / image dataset in each CSV file

In [9]:
```
# Missing files in each CSV file
for csv_file_name, df in zip(csv_files_to_read, csv_dataframes):
    print(f"Number of missing data in the {csv_file_name} dataset is: ", df['IDENTITY'].isnu
```

```
Number of missing data in the written_name_test_v2.csv dataset is:  70
Number of missing data in the written_name_train_v2.csv dataset is:  565
Number of missing data in the written_name_validation_v2.csv dataset is:  78
```

In [10]:
```
for csv_file_name, df in zip(csv_files_to_read, csv_dataframes):
    print(f"{csv_file_name} number of rows with 'IDENTITY' == 'UNREADABLE': ", (df['IDENTITY
```

```
written_name_test_v2.csv number of rows with 'IDENTITY' == 'UNREADABLE':  11
written_name_train_v2.csv number of rows with 'IDENTITY' == 'UNREADABLE':  102
written_name_validation_v2.csv number of rows with 'IDENTITY' == 'UNREADABLE':  12
```

Based on the results obtained above, we can know that the images that are not readable have the IDENTITY of 'UNREADABLE' and these images will not be used in the training, testing and validation dataset for the model.

Combine all csv dataframe together to one

In [11]:
```
df_test = csv_dataframes[0]
df_train = csv_dataframes[1]
df_validate = csv_dataframes[2]
```

```
# Combine all datasets (if needed)
all_data = pd.concat([df_test, df_train, df_validate])
```

Set Seaborn theme

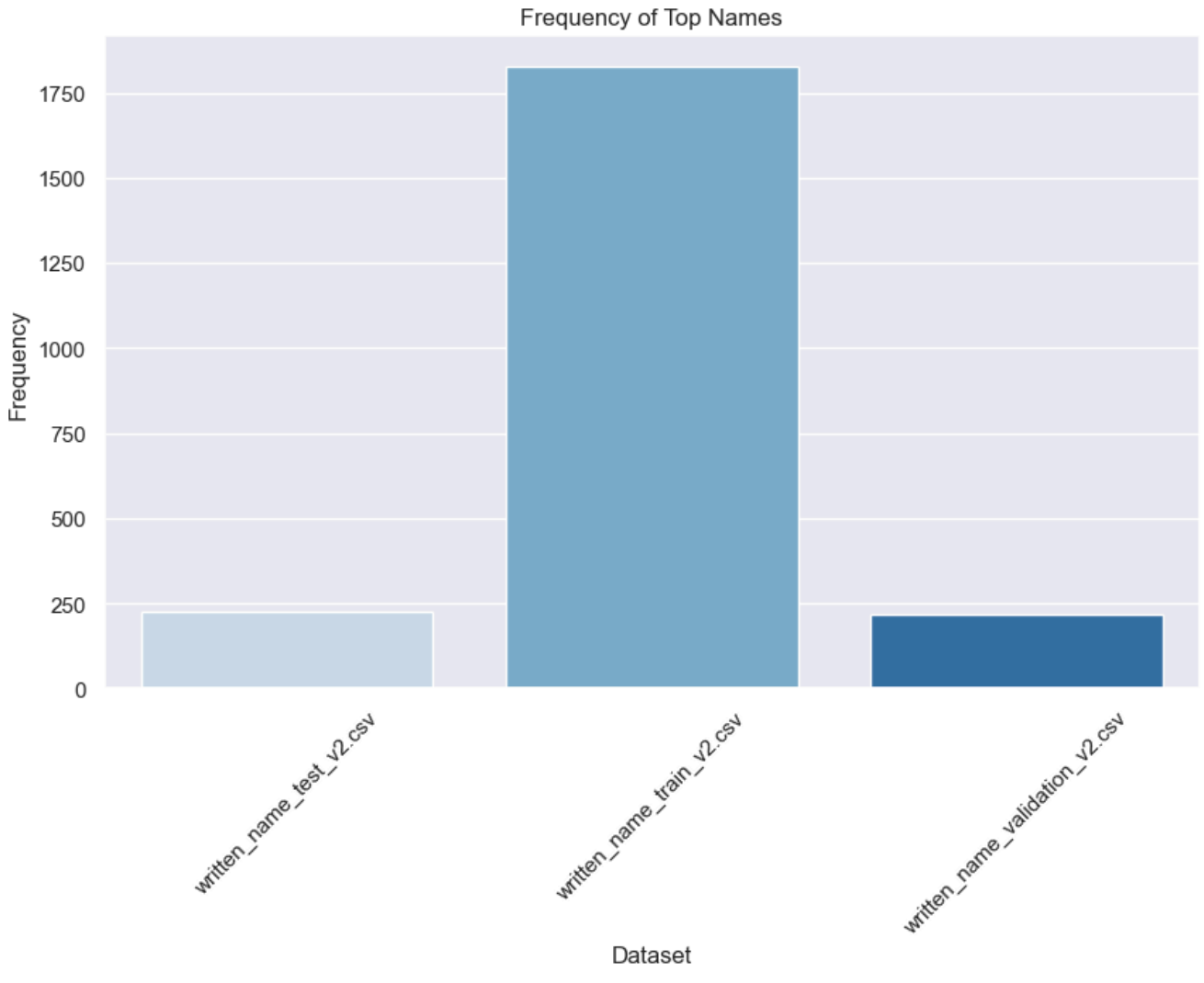
In [12]:
```
sns.set_theme()
```

Seaborn theme is set to ontrol the general style of plots that will be plotted below.

Plotting Graphs for visualisation

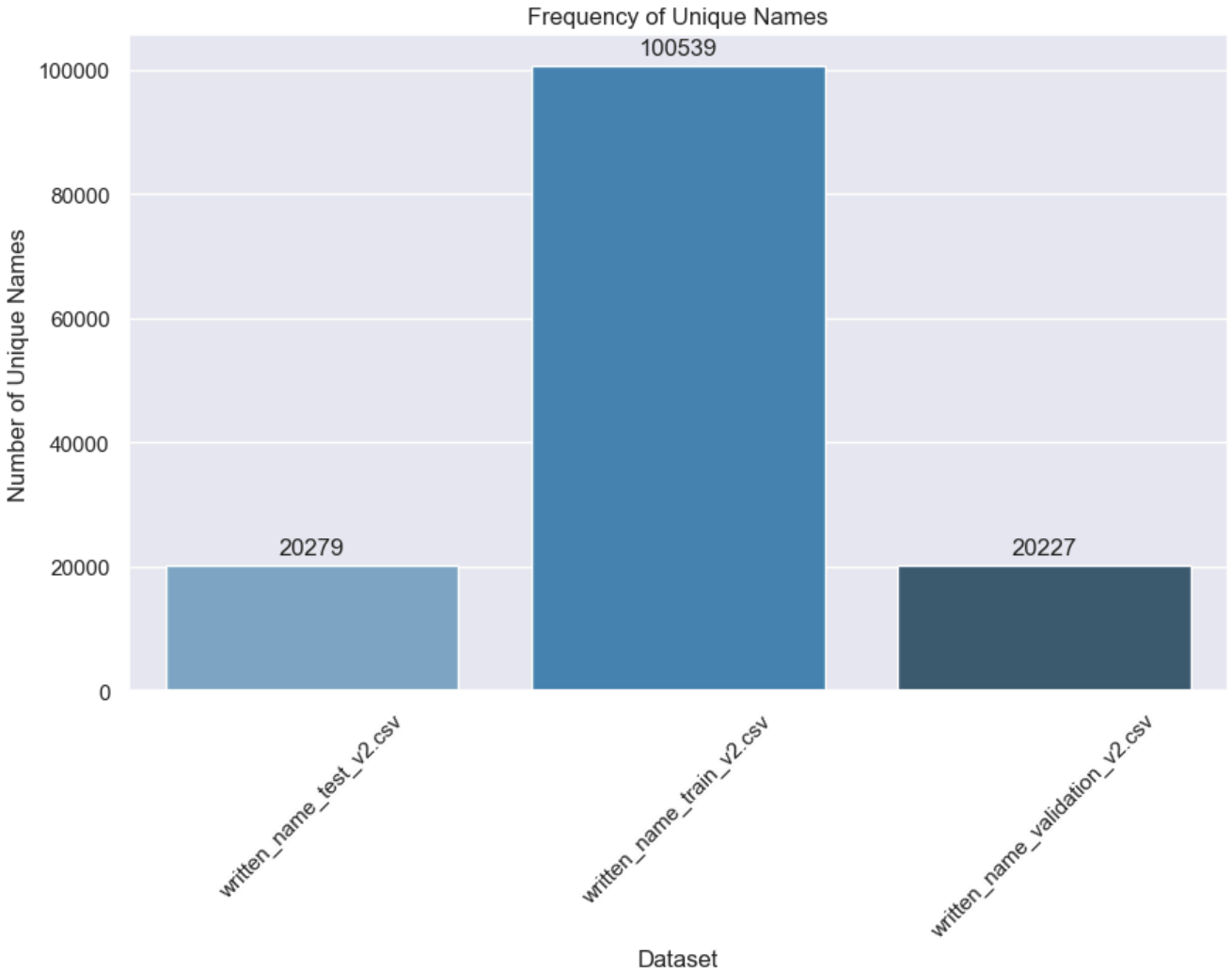
In [13]:
```
# Extracting necessary information for plotting
top_names_freq = [df['IDENTITY'].value_counts().max() for df in csv_dataframes]
name_lengths = [len(df['FILENAME'][df['IDENTITY'] == df['IDENTITY'].mode()[0]].iloc[0].split

# Plot Frequency of Top Names (Bar Plot)
plt.figure(figsize=(10, 6))
blue_palette = sns.color_palette("Blues", len(csv_files_to_read))
sns.barplot(x=csv_files_to_read, y=top_names_freq, palette=blue_palette)
plt.title('Frequency of Top Names')
plt.xlabel('Dataset')
plt.ylabel('Frequency')
plt.xticks(rotation=45)
plt.show()
```

```
C:\Users\Nicol Foo\AppData\Local\Temp\ipykernel_10284\3162245484.py:8: FutureWarning:

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

  sns.barplot(x=csv_files_to_read, y=top_names_freq, palette=blue_palette)
```

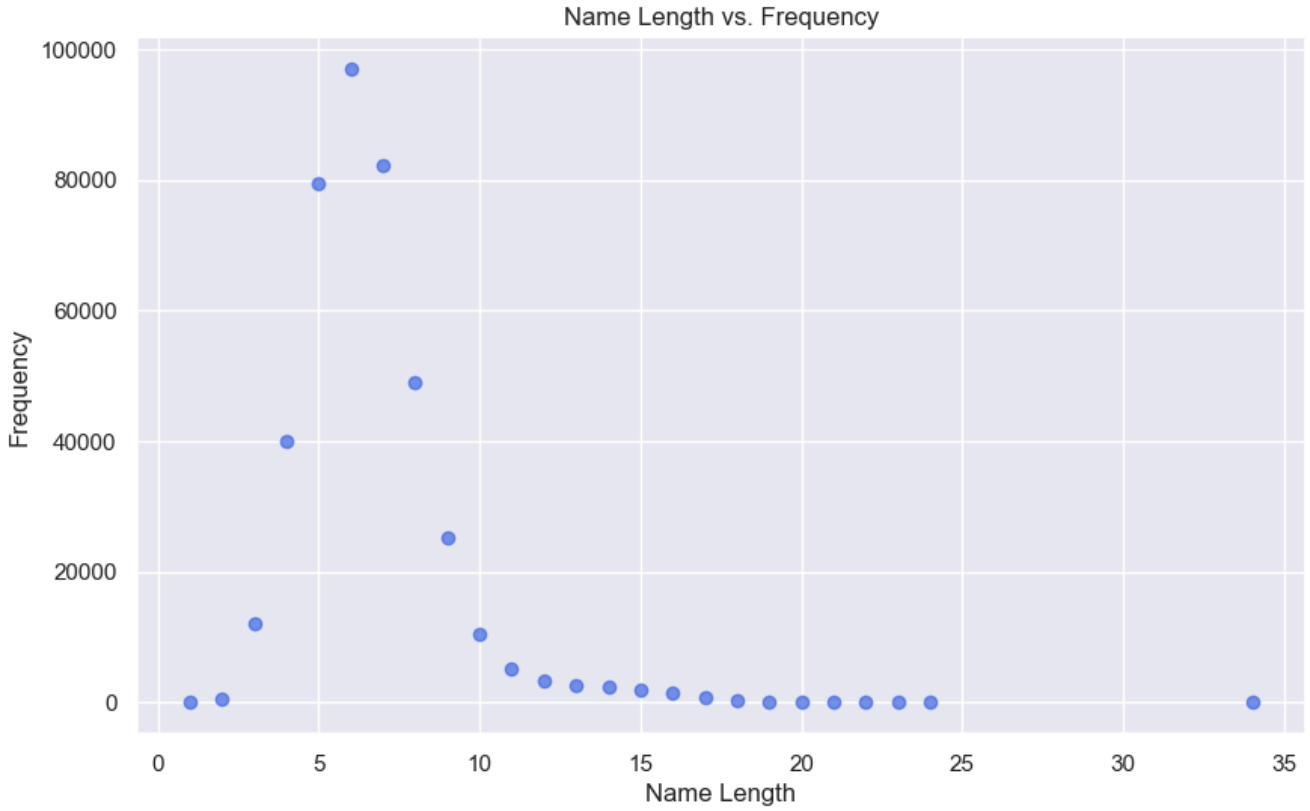


In [14]:
```
# Plot Frequency of unique names
top_names_unique = [df['IDENTITY'].nunique() for df in csv_dataframes]

# Create a DataFrame from the unique names count
unique_names_df = pd.DataFrame({
    'File': csv_files_to_read,
    'Unique_Names': top_names_unique
})

# Create the bar plot
plt.figure(figsize=(10, 6))
barplot_uniq_name = sns.barplot(x='File', y='Unique_Names', data=unique_names_df, palette="B

for p in barplot_uniq_name.patches:
    barplot_uniq_name.annotate(format(p.get_height(), '.0f'),
                       (p.get_x() + p.get_width() / 2., p.get_height()),
                       ha = 'center', va = 'center',
                       xytext = (0, 9),
                       textcoords = 'offset points')

plt.title('Frequency of Unique Names')
plt.xlabel('Dataset')
plt.ylabel('Number of Unique Names')
plt.xticks(rotation=45)
plt.show()
```

```
C:\Users\Nicol Foo\AppData\Local\Temp\ipykernel_10284\4132356596.py:12: FutureWarning:

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

  barplot_uniq_name = sns.barplot(x='File', y='Unique_Names', data=unique_names_df, palette
="Blues_d")
```



In [15]:
```
# Calculate name lengths
all_data['Name_Length'] = all_data['IDENTITY'].apply(lambda x: len(str(x)))

# Scatter plot of name length vs. frequency
name_frequencies = all_data['Name_Length'].value_counts()
plt.figure(figsize=(10, 6))
plt.scatter(name_frequencies.index, name_frequencies.values, alpha=0.7, c='royalblue')
plt.xlabel('Name Length')
plt.ylabel('Frequency')
plt.title('Name Length vs. Frequency')
plt.grid(True)
plt.show()
```



In [16]:
```
# Dictionary with directory names and number of files
directory_files = {
    'Testing': 41300,
    'Training': 330961,
    'Validation': 41292
}

df = pd.DataFrame(list(directory_files.items()), columns=['Dataset type', 'Number_of_Files'])

# Create the bar plot
plt.figure(figsize=(10, 6))
barplot = sns.barplot(x='Dataset type', y='Number_of_Files', data=df, palette="Blues_d")

# Annotate each bar with the number of files
for p in barplot.patches:
    barplot.annotate(format(p.get_height(), '.0f'),
                     (p.get_x() + p.get_width() / 2., p.get_height()),
                     ha = 'center', va = 'center',
                     xytext = (0, 9),
                     textcoords = 'offset points')

plt.title('Number of Files in Each Directory')
plt.xlabel('Dataset type')
plt.ylabel('Number of Files')
plt.xticks(rotation=45)
plt.show()
```

```
C:\Users\Nicol Foo\AppData\Local\Temp\ipykernel_10284\2213849944.py:12: FutureWarning:

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

  barplot = sns.barplot(x='Dataset type', y='Number_of_Files', data=df, palette="Blues_d")
```