**IFT-511-** **Analyzing Big Data**

**Project Step-3: Data Cleaning**

**Asmaa Elbadrawy**

**Date- 03/10/2024**

**Team – Gamma**

Kishore Prashanth – 1230355479

Shri Harri Priya Ramesh – 1229611021

Ashwani Kumar Guntupalli – 1230762236

Hemanth Kumar Guntupalli – 1230142149

Load the previously collected data(CSV) file from project step -2 into an IDE.

A screenshot of a computer

Description automatically generated

In this step of the project, we will be performing data cleaning keeping three steps as the agenda-

* + **Refining the data**
  + **Perform the transformations.**
  + **Remove the duplicates.**

**Refining the data**

In this step, we will be refining the data in the **CSV** we got during the data collection. Wherever the data is null or shown as zero we have tried fetching the data from other data sources to be as accurate as possible

We have removed the schools with populations lesser than 10.

**The data available in the data frame includes the following.**

School\_name: The name of the school

school\_type: The type of the school (e.g., Public, Private)

city: The city where the school is located.

zipcode: The zip code of the school's location

edu\_range: The educational range of the school (e.g., 9 - 12)

school\_grade: The grade assigned to the school.

title\_i\_status: Indicates whether the school is a Title I school (Yes/No)

total\_students: The total number of students in the school.

african\_american: The number of African American students in the school

african\_american\_percent: The percentage of African American students in the school

sports\_facilities: A list of sports facilities available at the school

music\_facilities: A list of music facilities available at the school

after\_school\_opportunities: A list of after-school opportunities available at the school

other\_facilities: A list of other facilities available at the school

student\_services: A list of student services provided by the school.

acad\_focus\_1, acad\_focus\_2, acad\_focus\_3, acad\_focus\_4, acad\_focus\_5: Academic focus areas of the school (details can be seen [here](https://azreportcards.azed.gov/schools/detail/10795)).

**Code**

school\_data['Dropout\_rate'].isnull().value\_counts()

school\_data.fillna({'Dropout\_rate': 0}, inplace=True)

school\_data['Dropout\_rate'].isnull().value\_counts()

school\_data['No\_of\_students\_adv\_placement\_course'].isnull().value\_counts()

school\_data['Students\_with\_Chronic\_Absenteeism'].isnull().value\_counts()

school\_data['crime\_rate'].isnull().value\_counts()

school\_data['total\_students\_bullied'].isnull().value\_counts()

school\_data.fillna({'No\_of\_students\_adv\_placement\_course': 0.0, 'Students\_with\_Chronic\_Absenteeism' : 0.0,'crime\_rate' : 0.0, 'total\_students\_bullied' : 0.0}, inplace=True)

school\_data['No\_of\_students\_adv\_placement\_course'].isnull().value\_counts()

school\_data['Students\_with\_Chronic\_Absenteeism'].isnull().value\_counts()

school\_data['crime\_rate'].isnull().value\_counts()

school\_data['total\_students\_bullied'].isnull().value\_counts()

for column in school\_data.columns:

if 'acad\_focus\_' in column:

print(school\_data[column].isnull().value\_counts())

for column in school\_data.columns:

if 'acad\_focus\_' in column:

school\_data.fillna({column: "No"}, inplace=True)

for column in school\_data.columns:

if 'acad\_focus\_' in column:

print(school\_data[column].isnull().value\_counts())

columns\_to\_replace = ['african\_american', 'african\_american\_percent',

'hispanic', 'hispanic\_percent',

'native\_american', 'native\_american\_percent',

'white', 'white\_percent',

'multiple\_race', 'multiple\_race\_percent',

'asian', 'asian\_percent',

'pacific\_islander', 'pacific\_islander\_percent']

for column in columns\_to\_replace:

print(school\_data[column].isnull().value\_counts())

print()

for column in columns\_to\_replace:

school\_data.fillna({column: 0}, inplace=True)

for column in columns\_to\_replace:

print(school\_data[column].isnull().value\_counts())

columns = ['personnel\_federal\_expenditure',

'personnel\_state\_and\_local\_expenditure',

'non\_personnel\_federal\_expenditure',

'non\_personnel\_state\_and\_local\_expenditure',

'total\_per\_pupil\_expenditure']

for i in columns:

print(school\_data[i].isnull().value\_counts())

for cols in columns:

school\_data.fillna({cols: school\_data[cols].dropna().min()}, inplace=True)

for cols in columns:

school\_data.fillna({cols: school\_data[cols].dropna().min()}, inplace=True)

for i in columns:

print(school\_data[i].isnull().value\_counts())

school\_data[school\_data['total\_students'] <= 10]

school\_data.drop(school\_data[school\_data['total\_students'] <= 10].index, inplace=True)

school\_data[school\_data['total\_students'] <= 10]

school\_data['school\_grade'].unique()

for i, row in school\_data.iterrows():

edu\_range = row['edu\_range']

a =edu\_range[0]

b =edu\_range[2:]

if int(edu\_range[0]) in range(0, 3):

school\_data.loc[i, 'elementary\_school'] = 1

if b!= ' ' and int(b) >= 3:

a = 3

if int(a) in range(3, 6):

school\_data.loc[i, 'intermediate\_school'] = 1

if b!= ' ' and int(b) >= 6:

a = 6

if int(a) in range(6, 9):

school\_data.loc[i, 'middle\_school'] = 1

if b!= ' ' and int(b) >= 9:

a = 9

if int(a) in range(9, 13):

school\_data.loc[i, 'high\_school'] = 1

**Explanation of code**

* First, the code checks if the "Dropout\_rate" column has any missing values by using the .isnull() function and then counts the number of True and False values using the .value\_counts() function.
* If there are missing values in the "Dropout\_rate" column, it fills them with 0 using the .fillna() function and sets the inplace parameter to True to modify the original DataFrame.
* The code then repeats the same process for other columns such as "No\_of\_students\_adv\_placement\_course", "Students\_with\_Chronic\_Absenteeism", "crime\_rate", "total\_students\_bullied", and any columns that contain "acad\_focus\_" in their names.
* Next, it creates a list called "columns\_to\_replace" which contains the names of columns that need to be checked for missing values. For each column in this list, it checks if there are missing values and prints the result using .isnull().value\_counts().
* Then, it fills the missing values in the columns listed in "columns\_to\_replace" with 0 using .fillna().
* The code then performs similar operations on a different set of columns listed in the "columns" list.
* After that, it identifies rows in the "school\_data" DataFrame where the "total\_students" column is less than or equal to 10 and drops those rows using the .drop() function.
* Lastly, the code iterates over each row in the DataFrame and performs the same operations based on the "edu\_range" column. It updates the values in the "elementary\_school", "intermediate\_school", "middle\_school", and "high\_school" columns based on the range of values in the "edu\_range" column.

**A screenshot of a computer

Description automatically generated**

****

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Transformation**

In this step, we will be transforming the data using strategies. The transformation steps

implemented include-**Removing** non-numeric characters (such as $, %, etc.) from numeric

values before transformation and apply appropriate transformations, such as scaling or

normalization, to numerical variables.

* School\_name - nominal
* city - nominal
* zipcode - nominal
* elementary\_school- nominal.
* intermediate\_school- nominal.
* middle\_school - nominal.
* high\_school - nominal.
* school\_grade- ordinal.
* title\_i\_status- nominal.
* total\_students -ratio.
* personnel\_federal\_expenditure -ratio
* personnel\_state\_and\_local\_expenditure -ratio
* non\_personnel\_federal\_expenditure -ratio
* non\_personnel\_state\_and\_local\_expenditure -ratio
* total\_per\_pupil\_expenditure -ratio.
* acad\_focus\_1 - nominal
* acad\_focus\_2 - nominal
* acad\_focus\_3 - nominal
* acad\_focus\_4 - nominal
* acad\_focus\_5 - nominal

**Code**

**#Removing '<' symbol**

def remove\_lt\_symbol(text):

if isinstance(text, str):

return text.replace('<', '')

else:

return text

**#Removing ','**

def remove\_comma(text):

if isinstance(text, str):

return text.replace(',', '')

else:

return text

**#Removing '$' sign**

def remove\_dollar(text):

if isinstance(text, str):

return text.replace('$', '')

else:

return text

**#Converting to int**

column\_numbers\_to\_convert = [7,8, 10, 12, 14, 16, 18, 20]

school\_data.iloc[:, column\_numbers\_to\_convert] = school\_data.iloc[:, column\_numbers\_to\_convert].astype(int)

**#Converting to float**

column\_numbers\_to\_convert = [9, 11, 13, 15, 17, 19, 21]

school\_data.iloc[:, column\_numbers\_to\_convert] = school\_data.iloc[:, column\_numbers\_to\_convert].astype(float)

column\_numbers\_to\_convert = [6,43,44,45,46,47]

**# for col\_num in column\_numbers\_to\_convert:**

school\_data.iloc[:, [6,43,44,45,46,47]] = school\_data.iloc[:, [6,43,44,45,46,47]].replace(to\_replace = 'No',value =0)

school\_data.iloc[:, [6,43,44,45,46,47]] = school\_data.iloc[:, [6,43,44,45,46,47]].replace(to\_replace = 'Yes',value =1)

school\_data

for column in school\_data.columns:

school\_data[column] = school\_data[column].apply(remove\_dollar)

for column in school\_data.columns:

school\_data[column] = school\_data[column].apply(remove\_comma)

for column in school\_data.columns:

school\_data[column] = school\_data[column].apply(remove\_lt\_symbol)

school\_data['edu\_range'] = school\_data['edu\_range'].replace({' - ': ' '},regex=True)

edu\_level\_categories = ['elementary\_school' ,'intermediate\_school','middle\_school','high\_school']

for column in edu\_level\_categories:

school\_data.fillna({column: 0}, inplace=True)

school\_data.insert(5, 'elementary\_school', None)

school\_data.insert(6, 'intermediate\_school', None)

school\_data.insert(7, 'middle\_school', None)

school\_data.insert(8, 'high\_school', None)

school\_data.drop('edu\_range',axis=1, inplace=True)

school\_data.drop('school\_type',axis=1, inplace=True)

school\_data.drop(school\_data.columns[38:45], axis=1,inplace=True)

school\_data['school\_grade'] =school\_data['school\_grade'].map(grade\_mapping)

school\_data['school\_grade']

grade\_mapping = {

'A': 5,

'B': 4,

'C': 3,

'D': 2,

'F': 1

}

school\_data['school\_grade'] =school\_data['school\_grade'].map(grade\_mapping)

school\_data['school\_grade']

school\_data['edu\_range'] = school\_data['edu\_range'].replace({'Preschool.': '0 ', 'Kindergarten.': '0 '},regex=True)

**Explanation**

* remove\_lt\_symbol(text): This function removes the '<' symbol from text values. It checks if the input text is a string, and if so, it replaces any occurrence of '<' with an empty string. If the input is not a string, it returns the input as it is.
* remove\_comma(text): This function removes the ',' symbol from text values. It follows a similar logic as the previous function, replacing any occurrence of ',' with an empty string in input strings.
* remove\_dollar(text): This function removes the '$' symbol from text values. It operates in the same way as the previous two functions, replacing '$' with an empty string in input strings.
* Conversion to int: The code specifies a list of column numbers column\_numbers\_to\_convert that need to be converted to integers. It selects those columns from the "school\_data" DataFrame using iloc and applies the astype(int) method to convert their data type to integer.
* Conversion to float: Similar to the previous step, this part converts specific columns to float data type by specifying their column numbers in column\_numbers\_to\_convert and applying astype(float) to those columns.
* Replacement of 'Yes' and 'No' values: This part replaces 'Yes' with 1 and 'No' with 0 in specific columns of the DataFrame. It uses replace() method and sets the to\_replace parameter to 'Yes' or 'No', and the value parameter to 1 or 0, respectively.
* Cleaning symbols: The code loops over each column in the DataFrame and applies the remove\_dollar, remove\_comma, and remove\_lt\_symbol functions to remove symbols from the column values.
* Replacement of ' - ' with space: The code replaces ' - ' with a space in the 'edu\_range' column of the DataFrame using the replace() method with the regex=True parameter.
* Filling NaN values: This part fills any NaN values in the columns listed in edu\_level\_categories with 0 using the fillna() method with the inplace=True parameter.
* Inserting new columns: The code inserts new columns with None values at specific positions in the DataFrame using the insert() method.
* Dropping columns: It drops the 'edu\_range', 'school\_type', and a range of columns using the drop() method with the axis=1 parameter.
* Mapping school grades: The code maps the values in the 'school\_grade' column to numerical values using the map() method and a defined mapping dictionary called grade\_mapping.
* Cleaning 'edu\_range' column: This part replaces specific strings in the 'edu\_range' column using the replace() method and regular expressions. It replaces 'Preschool.' with '0 ' and 'Kindergarten.' with '0 '.
* This transformation code performs various operations to clean and preprocess the "school\_data" DataFrame, including removing symbols, converting data types, replacing values, and filling missing values.

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

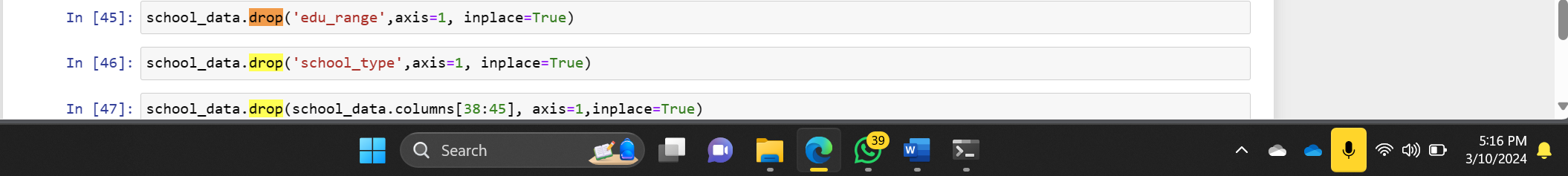
Description automatically generated**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

****

**Remove the duplicates.**

A screenshot of a computer

Description automatically generated

**Code**

original\_rows = len(df)

removed\_rows = df[df.duplicated(keep=False)]

df.drop\_duplicates(inplace=True)

print(f"Removed {len(removed\_rows)} duplicate rows:")

display(removed\_rows)

print(f"{original\_rows - len(df)} rows were removed from the original data.")

**Explanation:**

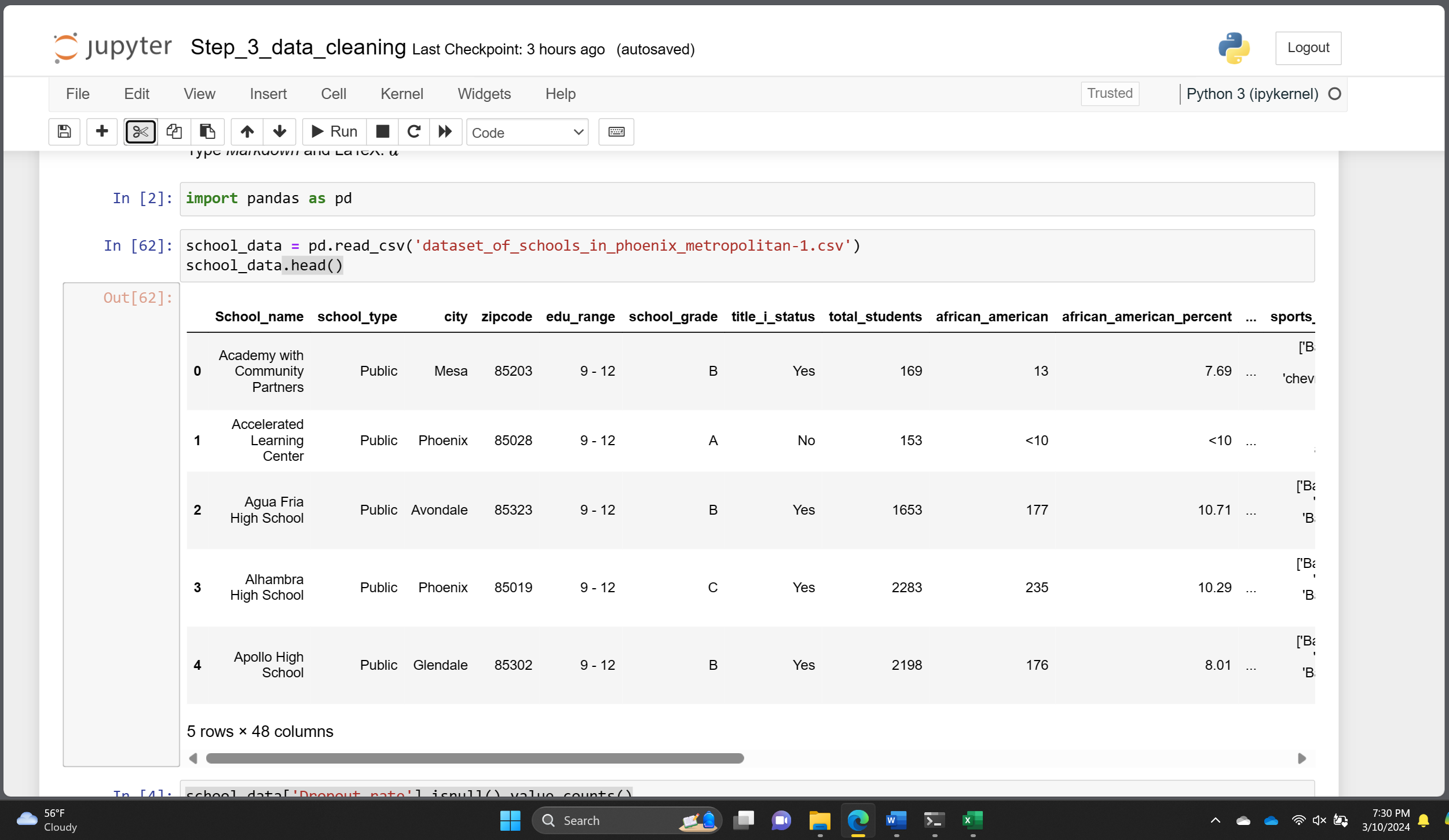
Since there was no duplicate data, as it was handled during data extraction; no rows were dropped.

**Final dataset dimensions**

Rows: 220

Columns: 43

**Before cleaning the data**



**After cleaning the data**

A screenshot of a computer

Description automatically generated

**Full code**

import pandas as pd

school\_data = pd.read\_csv('dataset\_of\_schools\_in\_phoenix\_metropolitan-1.csv')

school\_data.head()

school\_data['Dropout\_rate'].isnull().value\_counts()

school\_data.fillna({'Dropout\_rate': 0}, inplace=True)

school\_data['Dropout\_rate'].isnull().value\_counts()

school\_data['No\_of\_students\_adv\_placement\_course'].isnull().value\_counts()

school\_data['Students\_with\_Chronic\_Absenteeism'].isnull().value\_counts()

school\_data['crime\_rate'].isnull().value\_counts()

school\_data['total\_students\_bullied'].isnull().value\_counts()

school\_data.fillna({'No\_of\_students\_adv\_placement\_course': 0.0, 'Students\_with\_Chronic\_Absenteeism' : 0.0,'crime\_rate' : 0.0, 'total\_students\_bullied' : 0.0}, inplace=True)

school\_data['No\_of\_students\_adv\_placement\_course'].isnull().value\_counts()

school\_data['Students\_with\_Chronic\_Absenteeism'].isnull().value\_counts()

school\_data['crime\_rate'].isnull().value\_counts()

school\_data['total\_students\_bullied'].isnull().value\_counts()

for column in school\_data.columns:

if 'acad\_focus\_' in column:

print(school\_data[column].isnull().value\_counts())

for column in school\_data.columns:

if 'acad\_focus\_' in column:

school\_data.fillna({column: "No"}, inplace=True)

for column in school\_data.columns:

if 'acad\_focus\_' in column:

print(school\_data[column].isnull().value\_counts())

columns\_to\_replace = ['african\_american', 'african\_american\_percent',

'hispanic', 'hispanic\_percent',

'native\_american', 'native\_american\_percent',

'white', 'white\_percent',

'multiple\_race', 'multiple\_race\_percent',

'asian', 'asian\_percent',

'pacific\_islander', 'pacific\_islander\_percent']

for column in columns\_to\_replace:

print(school\_data[column].isnull().value\_counts())

print()

for column in columns\_to\_replace:

school\_data.fillna({column: 0}, inplace=True)

for column in columns\_to\_replace:

print(school\_data[column].isnull().value\_counts())

#Removing '<' symbol

def remove\_lt\_symbol(text):

if isinstance(text, str):

return text.replace('<', '')

else:

return text

for column in school\_data.columns:

school\_data[column] = school\_data[column].apply(remove\_lt\_symbol)

#Removing ','

def remove\_comma(text):

if isinstance(text, str):

return text.replace(',', '')

else:

return text

for column in school\_data.columns:

school\_data[column] = school\_data[column].apply(remove\_comma)

#Removing '$' sign

def remove\_dollar(text):

if isinstance(text, str):

return text.replace('$', '')

else:

return text

for column in school\_data.columns:

school\_data[column] = school\_data[column].apply(remove\_dollar)

#Converting to int

column\_numbers\_to\_convert = [7,8, 10, 12, 14, 16, 18, 20]

school\_data.iloc[:, column\_numbers\_to\_convert] = school\_data.iloc[:, column\_numbers\_to\_convert].astype(int)

#Converting to float

column\_numbers\_to\_convert = [9, 11, 13, 15, 17, 19, 21]

school\_data.iloc[:, column\_numbers\_to\_convert] = school\_data.iloc[:, column\_numbers\_to\_convert].astype(float)

column\_numbers\_to\_convert = [6,43,44,45,46,47]

# for col\_num in column\_numbers\_to\_convert:

school\_data.iloc[:, [6,43,44,45,46,47]] = school\_data.iloc[:, [6,43,44,45,46,47]].replace(to\_replace = 'No',value =0)

school\_data.iloc[:, [6,43,44,45,46,47]] = school\_data.iloc[:, [6,43,44,45,46,47]].replace(to\_replace = 'Yes',value =1)

school\_data

columns = ['personnel\_federal\_expenditure',

'personnel\_state\_and\_local\_expenditure',

'non\_personnel\_federal\_expenditure',

'non\_personnel\_state\_and\_local\_expenditure',

'total\_per\_pupil\_expenditure']

for i in columns:

print(school\_data[i].isnull().value\_counts())

for cols in columns:

school\_data.fillna({cols: school\_data[cols].dropna().min()}, inplace=True)

for i in columns:

print(school\_data[i].isnull().value\_counts())

school\_data[school\_data['total\_students'] <= 10]

school\_data.drop(school\_data[school\_data['total\_students'] <= 10].index, inplace=True)

school\_data[school\_data['total\_students'] <= 10]

school\_data['school\_grade'].unique()

grade\_mapping = {

'A': 5,

'B': 4,

'C': 3,

'D': 2,

'F': 1

}

school\_data['school\_grade'] =school\_data['school\_grade'].map(grade\_mapping)

school\_data['school\_grade']

school\_data['edu\_range'] = school\_data['edu\_range'].replace({' - ': ' '},regex=True)

school\_data

school\_data['edu\_range'] = school\_data['edu\_range'].replace({'Preschool.': '0 ', 'Kindergarten.': '0 '},regex=True)

school\_data.insert(5, 'elementary\_school', None)

school\_data.insert(6, 'intermediate\_school', None)

school\_data.insert(7, 'middle\_school', None)

school\_data.insert(8, 'high\_school', None)

school\_data

for i, row in school\_data.iterrows():

edu\_range = row['edu\_range']

a =edu\_range[0]

b =edu\_range[2:]

if int(edu\_range[0]) in range(0, 3):

school\_data.loc[i, 'elementary\_school'] = 1

if b!= ' ' and int(b) >= 3:

a = 3

if int(a) in range(3, 6):

school\_data.loc[i, 'intermediate\_school'] = 1

if b!= ' ' and int(b) >= 6:

a = 6

if int(a) in range(6, 9):

school\_data.loc[i, 'middle\_school'] = 1

if b!= ' ' and int(b) >= 9:

a = 9

if int(a) in range(9, 13):

school\_data.loc[i, 'high\_school'] = 1

school\_data

edu\_level\_categories = ['elementary\_school' ,'intermediate\_school','middle\_school','high\_school']

for column in edu\_level\_categories:

school\_data.fillna({column: 0}, inplace=True)

school\_data

school\_data.drop('edu\_range',axis=1, inplace=True)

school\_data.drop('school\_type',axis=1, inplace=True)

school\_data.drop(school\_data.columns[38:45], axis=1,inplace=True)

# Remove duplicates from the DataFrame

school\_data.drop\_duplicates(inplace=True)

school\_data

school\_data.to\_csv('511\_project\_Step\_3.csv', index=False)