# Assessment 1: Data Transformaton and Management

Tima, Matthew

New Zealand Skills and Education College

**School of Tech**

**Bachelor of Business Information Management (Level 6)**

**Cover Sheet and Student Declaration**

This sheet must be signed by the student and attached to the submitted assessment.

| | | | |
|---|---|---|---|
| **Course Title:** | **Data Transformation and Management** | **Course code:** | **BBIM612** |
| **Student Name:** | Matthew Tima | **Student ID:** | 764705436 |
| **Assessment No & Type:** | Assessment 1 - Project 1 | **Cohort:** | **BBIM7123C** |
| **Due Date:** | 20/12/2024 | **Date Submitted:** | 20/12/2024 |
| **Tutor's Name:** | Giang Mai | | |
| **Assessment Weighting:** | 60% | | |
| **Total Marks:** | 100 | | |

## Student Declaration:

I declare that:

- I have read the New Zealand School of Education Ltd policies and regulations on assessments and understand what plagiarism is.
- I am aware of the penalties for cheating and plagiarism as laid down by the New Zealand School of Education Ltd.
- This is an original assessment and is entirely my own work.
- Where I have quoted or made use of the ideas of other writers, I have acknowledged the source.
- This assessment has been prepared exclusively for this course and has not been or will not be submitted as assessed work in any other course.
- It has been explained to me that this assessment may be used by NZSE Ltd, for internal and/or external moderation.

## Student signature:

**Date:**

| **Tutor only to complete** | | | |
|---|---|---|---|
| | **Part A** (max. 20 marks) | **Part B** (max. 35 marks) | **Part C** (max. 45 marks) |
| **Assessment result:** | **Marks: /100** | **Grade:** | |
| **LO1 Requirements** | **Met** <br> **Not Met** | **Assessor signature:** | |
| **LO2 Requirements** | **Met** <br> **Not Met** | | |

## Table of Contents:

# Business Objective:

Education is a critical component of human development, influencing both economic progress and societal well-being. The objective of the investigation as a company (Tec Trends Inc) is to examine youth literacy rates and education spending to identify which elements are most beneficial in driving educational success in the context of human development. To accomplish this, we'll use the following datasets:

# PART A - (Data Collection):

## Task 1. Identify Data Sources:

Progress in International Reading Literacy Study (PIRLS):

Every five years, students in participating nations take reading comprehension tests as part of the Progress in International Reading Literacy Study (PIRLS), which evaluates students' reading skills worldwide. Countries with strong literacy education systems are highlighted by their regular top rankings, which include Singapore, Russia, and Hong Kong. Tec Trends Inc. may find trends in high-performing countries and identify the elements that contribute to their success by analysing PIRLS data, which provide ratings on a scale of 0–1,000. With the use of this data, the business may compare literacy outcomes throughout the globe and identify best practices that can be modified to enhance basic literacy abilities, which will directly contribute to academic achievement (Wikipedia, 2024).

List of Countries by Spending on Education as Percentage of GDP:

This data source provides a comprehensive view of national investments in education relative to their economic output. Countries like Norway (7.6%), Denmark (7.0%), and Sweden (6.8%) demonstrate higher levels of spending, correlating with strong education systems and human development. By analysing these metrics, Tec Trends Inc can explore the relationship between financial investment and educational outcomes, identifying whether higher spending leads to improved literacy rates. This analysis helps the company evaluate optimal spending strategies that balance investment with measurable results, supporting the objective of determining impactful funding practices for educational success (Wikipedia, 2024).

List of Countries by Human Development Index (HDI):

Every year, the United Nations Development Programme publishes the Human Development Index (HDI), which assesses societal advancement along three main dimensions: income, life expectancy, and education. The highest-ranking nations, including Switzerland (0.962), Norway (0.961), and Iceland (0.959), demonstrate the importance of robust educational institutions in promoting human development. Metrics that are directly related to educational outcomes, such as mean and projected years of education, are expressly included in the HDI. This information gives Tec Trends Inc. a more comprehensive understanding of the ways in which education investment and literacy rates affect human development. The business can illustrate the long-term advantages of enhancing education by tying these educational metrics to societal advancement, assisting in the creation of significant policies that promote human development on a worldwide scale (Wikipedia, 2024).

By merging these datasets, the analysis will uncover key patterns and insights that link education spending, literacy rates, and human development. These findings will help inform effective global education strategies and highlight actionable solutions to drive educational success.

## Task 2. Web Scraping:

### A). Application of Web Scraping Techniques:

*Figure 1.*

```
[4]: # Importing necessary libraries for data manipulation and web scraping
import pandas as pd  # For data analysis and manipulation
import requests  # For making HTTP requests to fetch web pages
from bs4 import BeautifulSoup  # For parsing HTML content
from tabulate import tabulate  # For creating formatted tables
```

```
[6]: # Defining the URLs of the websites we want to scrape
url_literacy_rate = "https://en.wikipedia.org/wiki/Progress_in_International_Reading_Literacy_Study"  # URL for literacy rate data
url_education_spending = "https://en.wikipedia.org/wiki/List_of_countries_by_spending_on_education_as_percentage_of_GDP"  # URL for education
url_HDI = "https://en.wikipedia.org/wiki/List_of_countries_by_Human_Development_Index"  # URL for Human Development Index data

# Fetching the HTML content of the websites
response_literacy_rate = requests.get(url_literacy_rate)  # Getting the HTML content of the literacy rate page
response_education_spending = requests.get(url_education_spending)  # Getting the HTML content of the education spending page
response_HDI = requests.get(url_HDI)  # Getting the HTML content of the HDI page
```

```
[10]: # Checking connection to the data source (Website HTML)
response_literacy_rate
response_education_spending
response_HDI

[10]: <Response [200]>
```

*Connection Source (Tima,2024)*

The provided Python code snippet outlines the initial steps for web scraping data from three data sources mentioned above. It begins by importing essential libraries for data manipulation and web scraping: pandas for data analysis, requests for fetching web pages, BeautifulSoup4 for parsing HTML content, and tabulate for formatting tables.

Next, the code defines the URLs of the three target web pages: the Progress in International Reading Literacy Study, the List of Countries by Spending on Education as a Percentage of GDP, and the List of Countries by Human Development Index. It then uses the requests library to fetch the HTML content of these pages and stores it in the response variables.

The final step in this code snippet is to check the connection status to the data sources. The response objects for each URL contain information about the HTTP response, including the status code. A status code of 200 indicates a successful request, confirming that the HTML content has been fetched successfully.

*Figure 2.*

```
[14]: # Parsing the HTML content to extract the tables
soup_literacy_rate = BeautifulSoup(response_literacy_rate.text, 'html.parser')  # Parsing the literacy rate HTML content
soup_education_spending = BeautifulSoup(response_education_spending.text, 'html.parser')  # Parsing the education spending HTML content
soup_HDI = BeautifulSoup(response_HDI.text, 'html.parser')  # Parsing the HDI HTML content

# Finding the specific tables we want to extract data from
table_literacy_rate = soup_literacy_rate.find('table', {'class': 'wikitable'})  # Finding the literacy rate table
table_education_spending = soup_education_spending.find('table', {'class': 'wikitable'})  # Finding the education spending table
tables = soup_HDI.find_all('table', {'class': 'wikitable'})  # Finding all tables on the HDI page
```

*Connection Source (Tima,2024)*

The code snippet outlines the foundational steps for web scraping data from multiple the 3 data sources. It leverages the requests library to fetch the HTML content of each specified URL, and then employs Beautiful Soup code to parse the HTML structure. By targeting tables with the class "wiki table," the code efficiently identifies and extracts the relevant data from each page.

*Figure 3.*

```
[14]: # Parsing the HTML content to extract the tables
soup_literacy_rate = BeautifulSoup(response_literacy_rate.text, 'html.parser')  # Parsing the literacy rate HTML content
soup_education_spending = BeautifulSoup(response_education_spending.text, 'html.parser')  # Parsing the education spending HTML content
soup_HDI = BeautifulSoup(response_HDI.text, 'html.parser')  # Parsing the HDI HTML content

# Finding the specific tables we want to extract data from
table_literacy_rate = soup_literacy_rate.find('table', {'class': 'wikitable'})  # Finding the literacy rate table
table_education_spending = soup_education_spending.find('table', {'class': 'wikitable'})  # Finding the education spending table
tables = soup_HDI.find_all('table', {'class': 'wikitable'})  # Finding all tables on the HDI page
```

*Extracting Table Variables (Tima,2024)*

The next step, the Python code extracts data from an HTML table representing literacy rates. It iterates through each row in the table, extracts the rank, country name, average scale score, and change over 5 years from the respective cells, and appends this information to a list. Finally, it prints the extracted data in a formatted table with headers for better readability.

*Figure 4.*

```
[16]: # Extracting data from the literacy rate table
data_literacy_rate = []
for row in table_literacy_rate.find_all('tr')[1:]:  # Iterating through each row in the table
    cells = row.find_all('td')  # Finding the cells in each row
    if cells:  # Checking if the row has cells
        rank = cells[0].text.strip()  # Extracting the rank
        country = cells[1].text.strip()  # Extracting the country
        average_scale_score = cells[2].text.strip()  # Extracting the average scale score
        change_over_5_years = cells[3].text.strip()  # Extracting the change over 5 years
        data_literacy_rate.append([rank, country, average_scale_score, change_over_5_years])  # Appending the extracted data to a list

# Defining the headers for the table
headers = ['Rank', 'Country', 'Average scale score', 'Change over 5 years']

# Printing the extracted data in a formatted table
print(tabulate(data_literacy_rate, headers=headers))  # Printing the table

Rank    Country                              Average scale score    Change over 5 years
------  ---------------------------------  ---------------------  --------------------
1       Singapore                                              587  11 points
2       Ireland                                                577  10 points
3       Hong Kong                                              573  4 points
4       Russia                                                 567  14 points
5       Northern Ireland                                       566  1 point
6       England[a]                                             558  1 point
7       Croatia                                                557  N/A
8       Lithuania                                              552  4 points
9       Finland                                                549  17 points
```

*Extracting 1st Table (Tima,2024)*

The Python code effectively extracts data from an HTML table displaying literacy rates. It iterates through each row, identifies relevant cells, and extracts the rank, country name, average scale score, and change over 5 years. The extracted data is then appended to a list and finally printed in a formatted table for better readability.

*Figure 5.*

```
[18]: # Extracting data from the education spending table
data_education_spending = []
for row in table_education_spending.find_all('tr')[1:]:  # Iterating through each row in the table
    cells = row.find_all('td')  # Finding the cells in each row
    if len(cells) >= 4:  # Checking if the row has at least 4 cells
        location = cells[0].text.strip()  # Extracting the location
        percentage = cells[1].text.strip()  # Extracting the percentage of GDP
        year = cells[2].text.strip()  # Extracting the year
        source = cells[3].text.strip()  # Extracting the source
        data_education_spending.append([location, percentage, year, source])  # Appending the extracted data to a list

# Defining the headers for the education spending table
headers_education_spending = ['Location', 'Percentage of GDP', 'Year', 'Source']
# Printing the extracted education spending data in a formatted table
print(tabulate(data_education_spending, headers=headers_education_spending))
```

```
Location                          Percentage of GDP    Year    Source
--------------------------------  -------------------  ------  ----------------
Marshall Islands                  15.8                 2019    [1]
Cuba                              11.5                 2020    [2]
Micronesia                        10.5                 2020    [2]
Kiribati                          15.6                 2021    [2]
Somaliland                        9.6                  2021    [1]
Djibouti                          8.4                  2012    [1]
Namibia                           8.4                  2012    [1]
Norway                            8.0                  2016    [1]
Botswana                          7.8                  2012    [1]
Sweden                            7.7                  2016    [2]
```

*Extracting 2nd Table (Tima,2024)*

The code extracts data from an HTML table representing education spending. It iterates through each row in the table, extracts the location, percentage of GDP spent on education, year, and source from the respective cells, and appends this information to a list. Finally, it prints the extracted data in a formatted table with headers for better readability.

*Figure 6.*

```
[22]: # Extracting data from the HDI table
table_HDI = tables[1]
data_HDI = []
rows = table_HDI.find_all('tr')
for row in rows[1:]:  # Iterating through each row in the table
    cells = row.find_all(['th', 'td'])  # Finding the cells in each row
    if len(cells) >= 5:  # Checking if the row has at least 5 cells
        rank = cells[0].text.strip()  # Extracting the rank
        country = cells[2].text.strip().split('[')[0].strip()  # Extracting the country
        hdi_value = cells[3].text.strip()  # Extracting the HDI value
        annual_growth = cells[4].text.strip() if len(cells) > 4 else "N/A"  # Extracting the annual growth if available,
        data_HDI.append([rank, country, hdi_value, annual_growth])  # Appending the extracted data to a list

# Defining the headers for the HDI table
headers = ['Rank', 'Country', 'HDI Value', 'Annual Growth (2010-2021)']
# Printing the extracted HDI data in a formatted table
print(tabulate(data_HDI, headers=headers))
```

```
  Rank  Country                            HDI Value  Annual Growth (2010-2021)
------  -------------------------------  -----------  ---------------------------
     1  Switzerland                            0.967  0.24%
     2  Norway                                 0.966  0.25%
     3  Iceland                                0.959  0.28%
     4  Hong Kong                              0.956  0.38%
     5  Denmark                                0.952  0.35%
     7  Ireland                                0.95   0.38%
     9  Singapore                              0.949  0.25%
```

*Extracting 3rd Table (Tima,2024)*

The Python code extracts data from an HTML table displaying the Human Development Index (HDI). It iterates through each row, identifies relevant cells, and extracts the rank, country name, HDI value, and annual growth rate. The extracted data is then appended to a list and finally printed in a formatted table for a better display.

*Figure 7.*

```
[24]: # Creating DataFrames from the extracted data (df1 - DataFrame 1 etc..)
      df1 = pd.DataFrame(data_literacy_rate, columns=['Rank', 'Country', 'Average scale score', 'Change over 5 years'])
      df2 = pd.DataFrame(data_education_spending, columns=['Location', 'Percentage of GDP', 'Year', 'Source'])
      df3 = pd.DataFrame(data_HDI, columns=['Rank', 'Country', 'HDI Value', 'Annual Growth (2010-2021)'])

[28]: # Displaying the DataFrames
      print(df1.head())
      print(df2.head())
      print(df3.head())

  Rank            Country Average scale score Change over 5 years
0    1          Singapore                 587           11 points
1    2            Ireland                 577           10 points
2    3          Hong Kong                 573            4 points
3    4             Russia                 567           14 points
4    5   Northern Ireland                 566             1 point
           Location Percentage of GDP  Year Source
0  Marshall Islands              15.8  2019    [1]
1              Cuba              11.5  2020    [2]
2        Micronesia              10.5  2020    [2]
3          Kiribati              15.6  2021    [2]
4        Somaliland               9.6  2021    [1]
  Rank      Country HDI Value Annual Growth (2010-2021)
0    1  Switzerland     0.967                     0.24%
1    2       Norway     0.966                     0.25%
2    3      Iceland     0.959                     0.28%
3    4    Hong Kong     0.956                     0.38%
4    5      Denmark     0.952                     0.35%
```

*Three Data Frames (Tima,2024)*

The Python code effectively creates three Data Frames from the extracted data, each representing a different table: literacy rates, education spending, and Human Development Index (HDI). These Data Frames provide a structured and organized way to represent and analyse the data

*Figure 8.*

```
[30]: # Saving the DataFrames to CSV files
      df1.to_csv('Downloads\\CountriesLiteracyRate.csv', index=False)
      df2.to_csv('Downloads\\GovernmentEducationSpending.csv', index=False)
      df3.to_csv('Downloads\\HumanDevelopmentIndex.csv', index=False)
```

*Exporting Data Frames (Tima,2024)*

The code snippet demonstrates the process of saving the three Data Frames (literacy rates, education spending, and Human Development Index) to CSV files, making them accessible for further analysis or visualization.

## B). Adherence to Ethical Standards and Data Privacy:

The web scraping process described adheres to ethical standards by respecting the terms of service of the target websites and ensuring compliance with copyright laws. The data sources selected for scraping publicly available literacy rates, education spending, and HDI tables are openly accessible without authentication barriers or proprietary restrictions. By targeting data intended for public consumption, the process ensures that no intellectual property or licensing agreements are violated, aligning the activity with the permissible use of such resources for research purposes.

Furthermore, the scraping activity demonstrates a commitment to transparency by clearly defining its purpose as educational and research focused. The datasets are used exclusively to analyse literacy rates, educational funding, and their relationship to human development. This academic intent ensures the data is utilized responsibly, without exploitation for unauthorized commercial use or distribution. Such an approach aligns with ethical research practices by prioritizing data usage for societal benefit rather than personal or financial gain.

Finally, the process respects privacy laws and data protection regulations by avoiding the collection of sensitive or personally identifiable information (PII). The scraped content is restricted to aggregated statistical data at the country level, ensuring no individuals are identified or impacted by the analysis. By adhering to privacy-centric principles and targeting only macro-level information, the process complies with regulations like the General Data Protection Regulation (GDPR) and similar standards, ensuring ethical handling of the data throughout its lifecycle.

# PART B - (Data Preparation and Cleansing):

## Task 3. Data Preparation and Cleansing:

### Data Frame 1:

#### Correcting possible typos (Country Column):

*Figure 9.*

```
-       Alberta (Canada)                              539  N/A
-       British Columbia (Canada)                     535  N/A
-       Newfoundland and Labrador (Canada)            523  N/A
-       Abu Dhabi (United Arab Emirates)              440  26 points
-       South Africa[a][c]                            384  N/A
```

```
[61]: # Removing unnecessary text from the 'Country' column in df1
      df1["Country"] = df1["Country"].str.replace(r"\[.*?\]", "", regex=True).str.strip()
```

```
[67]: # I analysed that the tail end of the data had unnecessary text so I used tail funtion to display the newly clean text
      df1.tail(5)
```

[67]:

| | Rank | Country | Average scale score | Change over 5 years |
|---|---|---|---|---|
| 60 | – | Alberta (Canada) | 539 | N/A |
| 61 | – | British Columbia (Canada) | 535 | N/A |
| 62 | – | Newfoundland and Labrador (Canada) | 523 | N/A |
| 63 | – | Abu Dhabi (United Arab Emirates) | 440 | 26 points |
| 64 | – | South Africa | 384 | N/A |

*Df1 Data Typos (Tima,2024)*

#### Removing irrelevant data (Points):

*Figure 10.*

```
•[186]: # Removing irrelevant data (point and points) in this column
df1['Change over 5 years'] = (
    df1['Change over 5 years'].str.replace(' points', '', regex=False)  # Removing ' points' from the 'Change over 5 years' column
    .str.replace(' point', '', regex=False)  # Removing ' point' from the 'Change over 5 years' column
)
```

```
[201]: df1.head(5)
```

[201]:

| | Rank | Country | Average scale score | Change over 5 years |
|---|---|---|---|---|
| 0 | 1 | Singapore | 587 | 11 |
| 1 | 2 | Ireland | 577 | 10 |
| 2 | 3 | Hong Kong | 573 | 4 |
| 3 | 4 | Russia | 567 | 14 |
| 4 | 5 | Northern Ireland | 566 | 1 |

*Df1 Irrelevant Data (Tima,2024)*

## Data type conversion (Integer, Float):

*Figure 11.*

```
[183]: # Checking the data types of all columns to see if they are correct
df1.dtypes
```

```
[183]: Rank                   object
Country                object
Average scale score    object
Change over 5 years    object
dtype: object
```

```
[203]: # Fixing the 'Rank' column so it's a number
df1['Rank'] = (
    df1['Rank'].astype(str)  # Turning the 'Rank' column into text
    .apply(pd.to_numeric, errors='coerce')  # Trying to convert the 'Rank' column to integer, but if it fails, making it missing
    .fillna(0).astype(int) #Handles potential NaN values after conversion
)

# Fixing the 'Average scale score' column to a integer
df1['Average scale score'] = (
    df1['Average scale score'].astype(str)  # Turning the 'Average scale score' column into text
    .apply(pd.to_numeric, errors='coerce')  # Trying to convert the 'Average scale score' column to numbers, but if it fails, making it missing
    .fillna(0)  # Filling missing values in 'Average scale score' with 0
)

# Fixing the 'Change over 5 years' column so it's float type
df1['Change over 5 years'] = (
    df1['Change over 5 years'].astype(str)  # Turning the 'Change over 5 years' column into text
    .apply(pd.to_numeric, errors='coerce')  # Trying to convert the 'Change over 5 years' column to numbers, but if it fails, making it missing
    .fillna(0)  # Filling missing values in 'Change over 5 years' with 0
)
```

```
[205]: # Checking the data types of all columns again to make sure they are correct
df1.dtypes
```

```
[205]: Rank                     int32
Country                 object
Average scale score      int64
Change over 5 years    float64
dtype: object
```

*Df1 Data Type Conversion (Tima,2024)*

## Removing unwanted observations (Duplicates):

*Figure 12.*

```
•[213]: # Making a Scenario if there were duplicate values in each 3 data frames-
# - we will know how to handle it using python
# - Process of cleaning duplicate values
# Implementing Duplicates for education rankings dataset (df1)
df_rankings_dup = pd.concat([df1, df1.iloc[:5]], ignore_index=True)
```

```
[215]: # Looking for duplicate rows in the education rankings dataset
print("Duplicate rows in education rankings dataset:")
print(df_rankings_dup[df_rankings_dup.duplicated()])
```

```
Duplicate rows in education rankings dataset:
    Rank           Country  Average scale score  Change over 5 years
65     1         Singapore                  587                 11.0
66     2           Ireland                  577                 10.0
67     3         Hong Kong                  573                  4.0
68     4            Russia                  567                 14.0
69     5  Northern Ireland                  566                  1.0
```

```
[217]: # Removing duplicate rows from the education rankings dataset
df_rankings_dup.drop_duplicates(inplace=True)
# Looking for duplicate rows again to make sure they're all gone
print("Duplicate rows in education rankings dataset:")
print(df_rankings_dup[df_rankings_dup.duplicated()])
```

```
Duplicate rows in education rankings dataset:
Empty DataFrame
Columns: [Rank, Country, Average scale score, Change over 5 years]
Index: []
```

*Df1 Duplicates Mitigation (Tima,2024)*

## Data Frame 2:

### Removing irrelevant data (Source Column):

*Figure 13.*

```
•[229]: # Displaying 1st row of DataFrame 2
df2.head(1)
```

[229]:

| | Location | Percentage of GDP | Year | Source |
|---|---|---|---|---|
| 0 | Marshall Islands | 15.8 | 2019 | [1] |

```
[231]: # Removing the 'Source' column-
# because it's not needed for objective
df2 = df2.drop(columns=['Source'], errors='ignore')
```

```
•[233]: # Displaying new Data Frame 2
df2.head(1)
```

[233]:

| | Location | Percentage of GDP | Year |
|---|---|---|---|
| 0 | Marshall Islands | 15.8 | 2019 |

*Removal Of Column (Tima,2024)*

## Correcting possible typos (Location Column):

*Figure 14.*

```
[ ]: # Removing unnecessary text from the 'Location' column in df2
df2["Location"] = df2["Location"].str.replace(r"\[.*?\]", "", regex=True).str.strip()
```

```
[235]: df2.head()
```

[235]:

| | Location | Percentage of GDP | Year |
|---|---|---|---|
| 0 | Marshall Islands | 15.8 | 2019 |
| 1 | Cuba | 11.5 | 2020 |
| 2 | Micronesia | 10.5 | 2020 |
| 3 | Kiribati | 15.6 | 2021 |
| 4 | Somaliland | 9.6 | 2021 |

*Df2 Data Typos (Tima,2024)*

## Data type conversion (Float, Integer):

*Figure 15.*

```
[18]: # Checking the data types of all columns to see if they are correct
df2.dtypes

[18]: Location            object
Percentage of GDP   object
Year                object
Source              object
dtype: object

•[19]: # Fixing the 'Percentage of GDP' column so it's a number
df2['Percentage of GDP'] = (
    df2['Percentage of GDP'].astype(str)  # Turning the 'Percentage of GDP' column into text
    .apply(pd.to_numeric, errors='coerce')  # Trying to convert the 'Percentage of GDP' column to numbers, but if it fails, making it missing
    .fillna(0)  # Filling missing values in 'Percentage of GDP' with 0
)

# Fixing the 'Year' column so it's a number
df2['Year'] = (
    df2['Year'].astype(str)  # Turning the 'Year' column into text
    .apply(pd.to_numeric, errors='coerce')  # Trying to convert the 'Year' column to numbers, but if it fails, making it missing
    .fillna(0).astype(int)  # Filling missing values in 'Year' with 0 and making sure it's a whole number
)

[20]: # Checking the data types of all columns again to make sure they are correct
df2.dtypes

[20]: Location             object
Percentage of GDP   float64
Year                  int32
dtype: object
```

*Df2 Data Conversion (Tima,2024)*

## Removing unwanted observations (Duplicates):

*Figure 16.*

```
[301]: # Making a Scenario if there were duplicate values in each 3 data frames-
       # - we will know how to handle it using python
       # - Process of cleaning duplicate values
       # Implementing Duplicates for education spending dataset (df2)
       df_spending_dup = pd.concat([df2, df2.iloc[:5]], ignore_index=True)

[303]: # Looking for duplicate rows in the education spending dataset
       print("\nDuplicate rows in education spending dataset:")
       print(df_spending_dup[df_spending_dup.duplicated()])

       Duplicate rows in education spending dataset:
                   Location  Percentage of GDP  Year
       198  Marshall Islands               15.8  2019
       199              Cuba               11.5  2020
       200        Micronesia               10.5  2020
       201          Kiribati               15.6  2021
       202        Somaliland                9.6  2021

[305]: # Removing duplicate rows from the education spending dataset
       df_spending_dup.drop_duplicates(inplace=True)
       # Looking for duplicate rows again to make sure they're all gone
       print("\nDuplicate rows in education spending dataset:")
       print(df_spending_dup[df_spending_dup.duplicated()])

       Duplicate rows in education spending dataset:
       Empty DataFrame
       Columns: [Location, Percentage of GDP, Year]
       Index: []
```

*Df2 Duplicates Mitigation (Tima,2024)*

# Data Frame 3:

## Correcting possible typos (Country Column):

*Figure 17.*

```
[442]: # Removing unnecessary text from the 'Location' column in df2
       df3["Country"] = df3["Country"].str.replace(r"\[.*?\]", "", regex=True).str.strip()

[480]: df3.head()
```

[480]:

| | Rank | Country | HDI Value | Annual Growth (2010-2021) |
|---|---|---|---|---|
| **0** | 1 | Switzerland | 0.967 | 0.24% |
| **1** | 2 | Norway | 0.966 | 0.25% |
| **2** | 3 | Iceland | 0.959 | 0.28% |
| **3** | 4 | Hong Kong | 0.956 | 0.38% |
| **4** | 5 | Denmark | 0.952 | 0.35% |

*Df3 Data Typos (Tima,2024)*

## Removing irrelevant data (% Mark):

*Figure 18.*

```
[484]: # Removing irrelevant data (% unit) in this column
df3['Annual Growth (2010-2021)'] = (
    df3['Annual Growth (2010-2021)'].str.replace('%', '', regex=True)  # Removing '%' from the 'Annual Growth (2010-2021)' column
)
```

```
[488]: df3.head()
```

[488]:

| | Rank | Country | HDI Value | Annual Growth (2010-2021) |
|---|---|---|---|---|
| **0** | 1 | Switzerland | 0.967 | 0.24 |
| **1** | 2 | Norway | 0.966 | 0.25 |
| **2** | 3 | Iceland | 0.959 | 0.28 |
| **3** | 4 | Hong Kong | 0.956 | 0.38 |
| **4** | 5 | Denmark | 0.952 | 0.35 |

*Df3 Irrelevant Data (Tima,2024)*

## Data Conversion (Integer, Float):

*Figure 19.*

```
[343]: # Checking the data types of all columns to see if they are correct
df3.dtypes
```

```
[343]: Rank                         object
Country                      object
HDI Value                    object
Annual Growth (2010-2021)    object
dtype: object
```

```
•[344]: # Fixing the 'Rank' column so it's a number
df3['Rank'] = (
    df3['Rank'].astype(str)  # Turning the 'Rank' column into text
    .apply(pd.to_numeric, errors='coerce')  # Trying to convert the 'Rank' column to numbers, but if it fails, making it missing
    .fillna(0).astype(int)  # Filling missing values in 'Rank' with 0 and making sure it's a whole number
)
# Fixing the 'HDI Value' column so it's a number
df3['HDI Value'] = (
    df3['HDI Value'].astype(str)  # Turning the 'HDI Value' column into text
    .apply(pd.to_numeric, errors='coerce')  # Trying to convert the 'HDI Value' column to numbers, but if it fails, making it missing
    .fillna(0)  # Filling missing values in 'HDI Value' with 0
)
# Fixing the 'Annual Growth (2010-2021)' column so it's a number
df3['Annual Growth (2010-2021)'] = (
    df3['Annual Growth (2010-2021)'].astype(str)  # Turning the 'Annual Growth (2010-2021)' column into text
    .apply(pd.to_numeric, errors='coerce')  # Trying to convert the 'Annual Growth (2010-2021)' column to numbers, but if it fails, making it missing
    .fillna(0)  # Filling missing values in 'Annual Growth (2010-2021)' with 0
)
```

```
[345]: # Checking the data types of all columns again to make sure they are correct
df3.dtypes
```

```
[345]: Rank                           int32
Country                       object
HDI Value                    float64
Annual Growth (2010-2021)    float64
dtype: object
```

*Df3 Data Conversion (Tima,2024)*

## Removing unwanted observations (Duplicates):

*Figure 20.*

```
[498]: # Making a Scenario if there were duplicate values in each 3 data frames-
       # - we will know how to handle it using python
       # - Process of cleaning duplicate values
       # Implementing Duplicates Duplicates for HDI dataset (df3)
       df_hdi_dup = pd.concat([df3, df3.iloc[:5]], ignore_index=True)
```

```
[500]: # Looking for duplicate rows in the HDI dataset
       print("\nDuplicate rows in HDI dataset:")
       print(df_hdi_dup[df_hdi_dup.duplicated()])
```

```
Duplicate rows in HDI dataset:
     Rank      Country  HDI Value  Annual Growth (2010-2021)
166     1  Switzerland      0.967                       0.24
167     2       Norway      0.966                       0.25
168     3      Iceland      0.959                       0.28
169     4    Hong Kong      0.956                       0.38
170     5      Denmark      0.952                       0.35
```

```
[502]: # Removing duplicate rows from the HDI dataset
       df_hdi_dup.drop_duplicates(inplace=True)
       # Looking for duplicate rows again to make sure they're all gone
       print("\nDuplicate rows in HDI dataset:")
       print(df_hdi_dup[df_hdi_dup.duplicated()])
```

```
Duplicate rows in HDI dataset:
Empty DataFrame
Columns: [Rank, Country, HDI Value, Annual Growth (2010-2021)]
Index: []
```

*Df3 Duplicates Mitigation (Tima,2024)*

# Task 4. Documentation:

## A). Data Preparation and Cleansing Processes:

## Data Frame 1:

### Correcting possible typos (Country Column):

The code snippet demonstrates the removal of unnecessary text from the "Country" column. This step is essential for data cleaning as it ensures that the country names are consistent and accurate. By removing the extraneous text within square brackets, the code effectively standardizes the country names, making them suitable for further analysis and data manipulation.

### Removing irrelevant data (Points):

This code cleans up a column of numbers by removing the words "point" and "points" which are unnecessary for calculations. This data cleaning step is important because it ensures that only numerical data remains, allowing for accurate mathematical analysis and preventing errors that might occur if the words were included in calculations. The result is a cleaner dataset that's ready for reliable analysis of changes over a five-year period.

### Data type conversion:

The provided code snippet demonstrates several instances of data type conversion, a crucial step in data preparation for analysis. Data type conversion involves transforming data from one format to another, often from text-based formats to numerical formats. In this specific case, my code focuses on converting columns in DF1 (Date Frame 1) that contain numerical values but are currently stored as strings or objects.

To achieve this, the code first converts the target columns to strings. This step ensures that any text-based elements, such as commas or the word "points," can be removed. Once the columns are in string format, the code uses regular expressions to remove these extraneous characters. Finally, the cleaned strings are converted to the appropriate numeric data types, either integer or float, using the numeric function. Missing values are handled by filling them with 0. By performing these data type conversions, the code prepares the data for further analysis. Numeric data types allow for calculations, statistical analysis, and visualization, making the data more informative and actionable.

### Removing unwanted observations (Duplicates):

This code cleans the dataset of education rankings (df1) by identifying and removing duplicate rows. Duplicate rows are problematic because they can skew analysis and lead to inaccurate conclusions. The code first identifies these duplicates, then uses the drop duplicates function to remove them, and finally, it verifies that the removal was successful. This ensures the dataset is accurate and reliable for further analysis of education trends.

## Data Frame 2:

### Removing irrelevant data (Source Column):

The code displays the removal of the "Source" column from the Data Frame 2 (df2). This column was deemed irrelevant for the specific analysis. By dropping this column, the dataset is streamlined, focusing on the essential features: "Location," "Percentage of GDP," and "Year." This step helps to simplify the analysis and improve computational efficiency.

### Correcting possible typos (Location Column):

This code cleans the "Location" column of a dataset by removing unwanted characters (square brackets, periods, question marks, asterisks) and extra whitespace. This ensures that location names are consistent and standardized, improving data quality and making the data more suitable for analysis and other data processing tasks.

### Data type conversion:

This code ensures that the "Percentage of GDP" and "Year" columns contain numerical data. It first checks the data types. Then, it converts these columns to numeric, handling potential errors by replacing non-numeric values with missing values (Nan). Finally, it converts the "Year" column to integers, filling any remaining missing values with 0. This data cleaning step is crucial for performing numerical calculations and analyses on the dataset

### Removing unwanted observations (Duplicates):

This code demonstrates a data cleaning process for handling duplicate entries within education spending dataset (df2). The code uses a function to identify and display all rows that are exact duplicates. After showing which rows are duplicated, the code uses drop duplicates function to remove these duplicate rows. Finally, it checks again for duplicates to confirm that the cleaning process was successful. This entire sequence showcases a common data cleaning technique to ensure data integrity and accuracy before further analysis of education spending data.

## DataFrame 3:

### Correcting possible typos (Country Column):

The code cleans the "Country" column in the data frame (df3) by removing unnecessary characters and extra whitespace using replace function with a regular expression and strip function. This ensures consistent formatting of country names, improving data quality and making it easier to work with for analysis

### Removing irrelevant data (% Mark):

This code removes the percentage symbol ("%") from the "Annual Growth (2010-2021)" column. This data cleaning step prepares the column for numerical calculations by ensuring that only numerical values remain. The removal of the "%" symbol is necessary to treat the growth values as numbers rather than text strings.

### Data Conversion (Integer, Float):

The code performs data type conversion and cleaning on the data frame (df3). It first checks the existing data types of all columns. Then, it iteratively converts the "Rank", "HDI Value", and "Annual Growth (2010-2021)" columns from their original (object) data types to numeric types. The code then converts the column to strings to handle potential errors during direct numeric conversion, it attempts to convert each element to a number; if conversion fails (due to non-numeric values), it replaces the element with a Nan (Not a Number) value. Finally, the code fills any remaining Nan values with 0. For the "Rank" column, an additional code ensures it's an integer. This comprehensive approach ensures data consistency and prepares the dataset for numerical analysis and calculations.

### Removing unwanted observations (Duplicates):

The provided code snippet demonstrates the removal of duplicate rows from Data Frame 3. To achieve this, the drop duplicate’s function is applied to the dataset, effectively eliminating any rows that are identical in all columns. By removing duplicate rows, the datasets are cleaned and ensure that each unique observation is considered in subsequent analysis. This step is crucial for maintaining data integrity and preventing potential biases in the results.

## B). Five (5) Challenges During Data Preparation/Cleansing Process and Solution:

### Data Frame 1 - Data Type Conversion:

One challenge I encountered while working with this section of the code was handling missing values within the target columns. While the code effectively removed unnecessary text and converted the cleaned strings to numeric data types, it also introduced missing values where the conversion process failed. To address this, I implemented a strategy of filling missing values with appropriate values, such as 0 for integer columns.

### Data Frame 1 – Correcting Possible Typos:

Initially, I mistakenly used strip function to remove unnecessary characters from the "Country" column. While strip function effectively removes leading and trailing whitespace, it doesn't address other characters that might be present, such as square brackets. To resolve this issue, I incorporated the string replace function to specifically target and remove the square brackets.

### Data Frame 2 – Removing irrelevant data (Source Column):

One challenge I faced while working with the code was determining the relevance of the "Source" column to the analysis. After careful consideration, I realized that the information contained within this column was not essential for the specific goals of the project, the column provided unnecessary links that can't be useful for further data manipulation. To streamline the dataset and improve computational efficiency, I decided to remove the "Source" column using the drop function.

### Data Frame 3 - Removing irrelevant data (% Mark):

During development, I encountered a challenge when attempting to clean the "Annual Growth (2010-2021)" column. My initial code had an indentation error in the line responsible for removing the "%" symbol using string replace function ('%',''). This incorrect indentation resulted in an Indentation Error, preventing the code from executing properly. The solution was to correct the indentation of the string replace function line, aligning it correctly within its code block. This simple fix resolved the Indentation Error, allowing the code to successfully remove the percentage symbols and clean the data.

### Data Frame 1, 2, 3 - Removing unwanted observations (Duplicates):

Biggest challenge I faced while working with the code was a syntax error in the drop duplicate's function. Initially, I mistakenly included the in place equals true argument within the print statement, which led to an error. To correct this, I moved the in place equals true argument to the correct position within the drop duplicate's function call.

# PART C - (Data Importation):

## Task 5. Store Datasets:

*Figure 21.*

```
[59]: # Saving the newly cleaned DataFrames to CSV files
df1.to_csv('Downloads\\CleanedCountriesLiteracyRateRefined.csv', index=False)
df2.to_csv('Downloads\\GovernmentEducationSpendingRefined.csv', index=False)
df3.to_csv('Downloads\\HumanDevelopmentIndexRefined.csv', index=False)
```

*Exporting Data Frames (Tima,2024)*

This code demonstrates how to save cleaned Data Frames (Three Data Frames named df1, df2, and df3, each containing cleaned data) to CSV files. It utilizes the CSV method provided by the panda's library to export the Data Frames into comma-separated value files. By setting the index equals False parameter, the row indices are excluded from the

CSV files. This ensures that the exported data is clean and ready for further analysis or visualization.

## Task 6. Merge Data:

*Figure 22.*

```
•[19]: # Importing pandas library for data manipulation
import pandas as pd

# Reading the three datasets from CSV files
df1 = pd.read_csv('Downloads\\CleanedCountriesLiteracyRateRefined.csv')
df2 = pd.read_csv('Downloads\\GovernmentEducationSpendingRefined.csv')
df3 = pd.read_csv('Downloads\\HumanDevelopmentIndexRefined.csv')
```

```
•[11]: # Renaming columns to have a consistent 'Country' column
df1 = df1.rename(columns={'Country': 'Country'})
df2 = df2.rename(columns={'Location': 'Country'})
df3 = df3.rename(columns={'Country': 'Country'})

# Merging all 3 DataFrames
merged_df = df1.merge(df2, on='Country', how='inner').merge(df3, on='Country', how='inner')

# Displaying the merged DataFrame
merged_df.head()
```

[11]:

| | Rank_x | Country | Average scale score | Change over 5 years | Percentage of GDP | Year | Rank_y | HDI Value | Annual Growth (2010-2021) |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | Singapore | 587 | 11.0 | 2.9 | 2013 | 9 | 0.949 | 0.25 |
| 1 | 2 | Ireland | 577 | 10.0 | 3.7 | 2017 | 7 | 0.950 | 0.38 |
| 2 | 3 | Hong Kong | 573 | 4.0 | 3.3 | 2018 | 4 | 0.956 | 0.38 |
| 3 | 4 | Russia | 567 | 14.0 | 3.7 | 2020 | 56 | 0.821 | 0.25 |
| 4 | 7 | Croatia | 557 | 0.0 | 4.6 | 2013 | 39 | 0.878 | 0.53 |

*Merged Data Frame (Tima,2024)*

This code snippet demonstrates the process of merging multiple Data Frames (df1, df2, df3) based on the chosen common column (Country and Location). First, it imports the panda's library for data manipulation. Then, it reads three CSV files, each containing data on literacy rates, education spending, and human development index, Data Frames.

To ensure consistent merging, the code renames the "Country" column in each DataFrame to have a uniform name. Then the code performs a series of inner joins on the "Country" column to combine the data from all three Data Frames. The resulting merged DataFrame contains information about countries, their literacy rates, education spending, and human development index. Finally, the code displays the first few rows of the merged DataFrame to visualize the combined data.

## Task 7. Indexing:

### A. Set a specific column as the index:

*Figure 23.*

```
[14]: # a. Setting 'Country' column as the index
      merged_df.set_index('Country', inplace=True)
      print("\nDataFrame with 'Country' as index:\n")
      print(merged_df.head())
```

```
DataFrame with 'Country' as index:

           Rank_x  Average scale score  Change over 5 years  \
Country
Singapore       1                  587                 11.0
Ireland         2                  577                 10.0
Hong Kong       3                  573                  4.0
Russia          4                  567                 14.0
Croatia         7                  557                  0.0

           Percentage of GDP  Year  Rank_y  HDI Value  \
Country
Singapore                2.9  2013       9      0.949
Ireland                  3.7  2017       7      0.950
Hong Kong                3.3  2018       4      0.956
Russia                   3.7  2020      56      0.821
Croatia                  4.6  2013      39      0.878

           Annual Growth (2010-2021)
Country
Singapore                       0.25
Ireland                         0.38
Hong Kong                       0.38
Russia                          0.25
Croatia                         0.53
```

*Country Index (Tima,2024)*

### B. Reset the index to the default integer-based index:

*Figure 24.*

```
[16]: # b. Reseting the index to default integer-based index
merged_df.reset_index(inplace=True)
print("\nDataFrame with default integer index:\n")
print(merged_df.head())
```

```
DataFrame with default integer index:

     Country  Rank_x  Average scale score  Change over 5 years  \
0  Singapore       1                  587                 11.0   
1    Ireland       2                  577                 10.0   
2  Hong Kong       3                  573                  4.0   
3     Russia       4                  567                 14.0   
4    Croatia       7                  557                  0.0   

   Percentage of GDP  Year  Rank_y  HDI Value  Annual Growth (2010-2021)  
0                2.9  2013       9      0.949                       0.25  
1                3.7  2017       7      0.950                       0.38  
2                3.3  2018       4      0.956                       0.38  
3                3.7  2020      56      0.821                       0.25  
4                4.6  2013      39      0.878                       0.53  
```

*Resetting Index (Tima,2024)*

## C. Create a new DataFrame by selecting rows based on a conditional index:

*Figure 25.*

```
[18]: # c. Creating a new DataFrame by selecting rows based on a conditional index
high_hdi_countries = merged_df[merged_df['HDI Value'] > 0.8]
print("\nCountries with HDI Value > 0.8:\n")
print(high_hdi_countries[['Country', 'HDI Value']].head())
```

```
Countries with HDI Value > 0.8:

     Country  HDI Value
0  Singapore      0.949
1    Ireland      0.950
2  Hong Kong      0.956
3     Russia      0.821
4    Croatia      0.878
```

*Conditional Index (Tima,2024)*

## D. Perform multi-level indexing by setting multiple columns as the index:

*Figure 26.*

```
[20]: # d. multi-level indexing by setting multiple columns as the index
merged_df.set_index(['Country', 'Year'], inplace=True)
print("\nDataFrame with multi-level index:\n")
print(merged_df.head())
```

```
DataFrame with multi-level index:

                Rank_x  Average scale score  Change over 5 years  \
Country   Year
Singapore 2013       1                  587                 11.0
Ireland   2017       2                  577                 10.0
Hong Kong 2018       3                  573                  4.0
Russia    2020       4                  567                 14.0
Croatia   2013       7                  557                  0.0

                Percentage of GDP  Rank_y  HDI Value  \
Country   Year
Singapore 2013                2.9       9      0.949
Ireland   2017                3.7       7      0.950
Hong Kong 2018                3.3       4      0.956
Russia    2020                3.7      56      0.821
Croatia   2013                4.6      39      0.878

                Annual Growth (2010-2021)
Country   Year
Singapore 2013                       0.25
Ireland   2017                       0.38
Hong Kong 2018                       0.38
Russia    2020                       0.25
Croatia   2013                       0.53
```

*Multilevel Indexing (Tima,2024)*

# Task 8. Sorting:

*Figure 27.*

```
[16]: # Sorting Dataframe by 'Average scale score' in ascending order
merged_df_asc = merged_df.sort_values(by='Average scale score', ascending=True)

# Displaying the Top 10 Countries (first few rows of the sorted DataFrames)
print("\nSorted in ascending order:\n")
print(merged_df_asc[['Country', 'Average scale score', 'Percentage of GDP', 'HDI Value']].head(10))
```

```
Sorted in ascending order:

         Country  Average scale score  Percentage of GDP  HDI Value
35  South Africa                  288                6.2      0.717
34       Morocco                  372                5.4      0.698
33         Egypt                  378                3.9      0.728
32        Jordan                  381                3.6      0.736
36  South Africa                  384                6.2      0.717
31          Iran                  413                4.0      0.780
30          Oman                  429                6.8      0.819
29    Uzbekistan                  437                6.3      0.727
28    Azerbaijan                  440                2.5      0.760
27       Bahrain                  458                2.3      0.888
```

*Ascending Data (Tima,2024)*

*Figure 28.*

```
[18]: # Sorting by 'Average scale score' in descending order
merged_df_desc = merged_df.sort_values(by='Average scale score', ascending=False)

# Displaying the Top 10 Countries (first few rows of the sorted DataFrames)
print("\nSorted in descending order:\n")
print(merged_df_desc[['Country', 'Average scale score', 'Percentage of GDP', 'HDI Value']].head(10))
```

```
Sorted in descending order:

           Country  Average scale score  Percentage of GDP  HDI Value
0        Singapore                  587                2.9      0.949
1          Ireland                  577                3.7      0.950
2        Hong Kong                  573                3.3      0.956
3           Russia                  567                3.7      0.821
4          Croatia                  557                4.6      0.878
5           Poland                  549                4.6      0.881
8          Hungary                  539                4.7      0.851
10          Norway                  539                8.0      0.966
7   Czech Republic                  539                5.6      0.895
6         Bulgaria                  539                4.1      0.799
```

*Descending Data (Tima,2024)*

The two provided Python code snippets sort the merged DataFrame by the 'Average scale score' column in ascending and descending order. This enables the identification of countries with lower average scores and highest average scores in reading literacy. By analysing the sorted data, we can observe that countries like South Africa, Morocco, and Egypt exhibit lower performance. This suggests potential disparities in educational systems and socioeconomic factors within these countries.

Further analysis can involve correlating literacy scores with education spending and HDI values. This could reveal insights into the impact of education spending on literacy rates and the overall human development index. We as Tec Trends Inc. can leverage these insights to identify emerging trends, inform strategic planning, and provide valuable recommendations to stakeholders in the education sector.

## Task 9. Summary Statistics:

*Figure 29.*

```
•[46]: # Executing summary statistics
summary_stats = merged_df.describe()

print("Summary Statistics:")
print(summary_stats)

# Calculating additional statistics
median_values = merged_df.median(numeric_only=True)
mode_values = merged_df.mode(numeric_only=True).iloc[0]

print("\nMedian Values:")
print(median_values)

print("\nMode Values:")
print(mode_values)
```

*Summary Statistics (Tima,2024)*

*Figure 30.*

```
Summary Statistics:
          Rank_x  Average scale score  Change over 5 years  Percentage of GDP  \
count  37.000000            37.000000            37.000000          37.000000
mean   29.000000           493.837838            12.864865           4.554054
std    17.227239            68.105928            12.660442           1.658244
min     0.000000           288.000000             0.000000           0.000000
25%    17.000000           458.000000             2.000000           3.700000
50%    29.000000           514.000000            11.000000           4.600000
75%    43.000000           539.000000            18.000000           5.600000
max    56.000000           587.000000            48.000000           8.000000

              Year      Rank_y  HDI Value  Annual Growth (2010-2021)
count    37.000000   37.000000  37.000000                  37.000000
mean   1961.054054   47.945946   0.856378                   0.416216
std     331.365353   34.349467   0.080582                   0.262672
min       0.000000    2.000000   0.698000                   0.090000
25%    2015.000000   25.000000   0.799000                   0.250000
50%    2016.000000   39.000000   0.878000                   0.350000
75%    2017.000000   70.000000   0.915000                   0.500000
max    2020.000000  120.000000   0.966000                   1.210000

Median Values:
Rank_x                         29.000
Average scale score           514.000
Change over 5 years            11.000
Percentage of GDP               4.600
Year                         2016.000
Rank_y                         39.000
HDI Value                       0.878
Annual Growth (2010-2021)       0.350
dtype: float64

Mode Values:
Rank_x                         17.000
Average scale score           539.000
Change over 5 years             0.000
Percentage of GDP               4.000
Year                         2016.000
Rank_y                        110.000
HDI Value                       0.717
Annual Growth (2010-2021)       0.250
Name: 0, dtype: float64
```

*Summary Statistics Output (Tima,2024)*

The provided summary statistics offer valuable insights into the distribution of key variables within the dataset. The "Average Scale Score" metric, for instance, reveals an average of approximately 493.84 with a standard deviation of 68.11, indicating a significant variation in reading literacy performance across countries. Countries with higher scores, such as Singapore and Ireland, demonstrate superior educational outcomes.

Additionally, the "Percentage of GDP" and "HDI Value" metrics provide context for understanding the factors influencing literacy rates. By analysing these metrics, Tec Trends Inc. can identify potential correlations between education spending, human development, and literacy rates. These insights can be used to inform strategic recommendations for improving educational outcomes and addressing disparities in literacy levels.

## Task 10. Slicing:

*Figure 31.*

```
[42]: # Slice the DataFrame
sliced_df = merged_df.loc[5:15, ['Country', 'Average scale score', 'Percentage of GDP', 'HDI Value']]

# Display the sliced DataFrame
sliced_df
```

[42]:

| | Country | Average scale score | Percentage of GDP | HDI Value |
|---|---|---|---|---|
| **5** | Poland | 549 | 4.6 | 0.881 |
| **6** | Bulgaria | 539 | 4.1 | 0.799 |
| **7** | Czech Republic | 539 | 5.6 | 0.895 |
| **8** | Hungary | 539 | 4.7 | 0.851 |
| **9** | Denmark | 539 | 7.6 | 0.952 |
| **10** | Norway | 539 | 8.0 | 0.966 |
| **11** | Italy | 537 | 3.8 | 0.906 |
| **12** | Latvia | 528 | 4.7 | 0.879 |
| **13** | Netherlands | 527 | 5.5 | 0.946 |
| **14** | New Zealand | 521 | 6.4 | 0.939 |
| **15** | Spain | 521 | 4.2 | 0.911 |

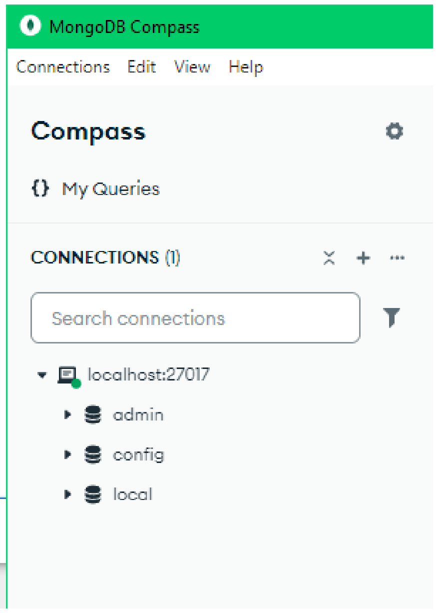
*Data Frame Slice (Tima,2024)*

The provided Python code snippet slices the merged DataFrame to extract specific rows and columns, focusing on countries with similar average scale scores. This analysis allows us to identify a group of high-performing countries, including Poland, Bulgaria, Czech Republic, Hungary, Denmark, Norway, Italy, Latvia, Netherlands, New Zealand, and Spain. By examining these countries, Tec Trends Inc. can gain insights into the factors contributing to their high literacy rates, such as education policies, cultural factors, and economic conditions. These insights can be used to inform strategies for improving literacy rates in other countries, ultimately leading to better educational outcomes and human development.

## Task 11. Data Import:

*Figure 32.*

```
[56]: from pymongo import MongoClient

# Step 1: Establishing a connection to MongoDB
client = MongoClient('mongodb://localhost:27017/')

# Step 2: Specifying the database and collection
db = client['education_database']
collection = db['country_stats']

# Step 3: Converting DataFrame to a list of dictionaries
data = merged_df.to_dict(orient='records')

# Step 4: Inserting data into the MongoDB collection
collection.insert_many(data)

# Step 5: Verifying the data insertion
document_count = collection.count_documents({})
print(f"Total records in the MongoDB collection: {document_count}")

# Step 6: Close the MongoDB connection
client.close()

Total records in the MongoDB collection: 37
```




*Data Frame Importation (Tima,2024)*

This piece of code clarifies how to import the merged Data Frame into my local MongoDB database. It initially connects to a MongoDB instance and chooses the "country stats" collection and my "education database" database. The DataFrame is transformed into a MongoDB-compatible collection of dictionaries. The insert many functions are then used to add the data to the collection. The successful import is confirmed by counting the number of documents in the collection. At last, the MongoDB database connection is closed. With this approach, the DataFrame data is stored in MongoDB, a format that facilitates effective data analysis and retrieval.

# References:


Wikipedia. (2024, December 11). *List of countries by Human Development Index*. Wikipedia: https://en.wikipedia.org/wiki/List_of_countries_by_Human_Development_Index

Wikipedia. (2024, December 6). *List of countries by spending on education as percentage of GDP*. Wikipedia: https://en.wikipedia.org/wiki/List_of_countries_by_spending_on_education_as_percentage_of_GDP

Wikipedia. (2024, October 16). *Progress in International Reading Literacy Study*. Wikipedia: https://en.wikipedia.org/wiki/Progress_in_International_Reading_Literacy_Study