```
!pip install --upgrade gspread google-auth
```

```
Requirement already satisfied: gspread in /usr/local/lib/python3.10/dist-packages (6.0.2)
Collecting gspread
  Downloading gspread-6.1.2-py3-none-any.whl.metadata (11 kB)
Requirement already satisfied: google-auth in /usr/local/lib/python3.10/dist-packages (2.27.0)
Collecting google-auth
  Downloading google_auth-2.34.0-py2.py3-none-any.whl.metadata (4.7 kB)
Requirement already satisfied: google-auth-oauthlib>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from gspread) (1.2.1)
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth) (5.4.0)
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth) (0.4.0)
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth) (4.9)
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib>=0.4.1->g
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-auth
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-auth-o
Requirement already satisfied: requests>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-auth-o
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests>=2.0.0->requests-oaut
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests>=2.0.0->requests-oauthlib>=0.7.0-
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests>=2.0.0->requests-oauthlib>=
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests>=2.0.0->requests-oauthlib>=
Downloading gspread-6.1.2-py3-none-any.whl (57 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 57.5/57.5 kB 1.8 MB/s eta 0:00:00
Downloading google_auth-2.34.0-py2.py3-none-any.whl (200 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 200.9/200.9 kB 7.2 MB/s eta 0:00:00
Installing collected packages: google-auth, gspread
  Attempting uninstall: google-auth
    Found existing installation: google-auth 2.27.0
    Uninstalling google-auth-2.27.0:
      Successfully uninstalled google-auth-2.27.0
  Attempting uninstall: gspread
    Found existing installation: gspread 6.0.2
    Uninstalling gspread-6.0.2:
      Successfully uninstalled gspread-6.0.2
ERROR: pip's dependency resolver does not currently take into account all the packages that are installed. This behaviour is the source
google-colab 1.0.0 requires google-auth==2.27.0, but you have google-auth 2.34.0 which is incompatible.
Successfully installed google-auth-2.34.0 gspread-6.1.2
```

```
from google.colab import auth
auth.authenticate_user()

import gspread
from google.auth import default
creds, _ = default()

gc = gspread.authorize(creds)
```

```
# Open a sheet from a spreadsheet by name
spreadsheet = gc.open('DPAM Data Perspective')

# If you have the URL, you can do it this way:
# spreadsheet = gc.open_by_url('Your_Spreadsheet_URL')
```

Data preparation and Cleaning

```
# Get a worksheet by name
worksheet = spreadsheet.worksheet('Zimbabwe_children_under5_interv')

# Get all values from the worksheet
rows = worksheet.get_all_values()
#Convert data to pandas DataFrame (if needed)
import pandas as pd
data = pd.DataFrame.from_records(rows[1:],columns=rows[0])  # Skip header row
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2517 entries, 0 to 2516
Data columns (total 13 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   interview_date   2517 non-null   object
 1   child_age_years  2517 non-null   object
 2   child_birthday   2517 non-null   object
 3   EC6              2517 non-null   object
```

```
 4   EC7             2517 non-null   object
 5   EC8             2517 non-null   object
 6   EC9             2517 non-null   object
 7   EC10            2517 non-null   object
 8   EC11            2517 non-null   object
 9   EC12            2517 non-null   object
 10  EC13            2517 non-null   object
 11  EC14            2517 non-null   object
 12  EC15            2517 non-null   object
dtypes: object(13)
memory usage: 255.8+ KB
```

```
# Step 1: Check for missing values
missing_values = data.isnull().sum()

# Step 2: Convert date columns to datetime format
data['interview_date'] = pd.to_datetime(data['interview_date'])
data['child_birthday'] = pd.to_datetime(data['child_birthday'], errors='coerce')

# Step 3: Standardize column names (lowercase and replace spaces with underscores)
data.columns = data.columns.str.lower().str.replace(' ', '_')

# Check if there are still issues with date parsing
parsed_issues = data['child_birthday'].isnull().sum()



# Display the missing values and the first few rows after changes
missing_values, data.head()
```

```
<ipython-input-5-be84d2bf90bb>:5: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `da
  data['interview_date'] = pd.to_datetime(data['interview_date'])
<ipython-input-5-be84d2bf90bb>:6: UserWarning: Could not infer format, so each element will be parsed individually, falling back to `da
  data['child_birthday'] = pd.to_datetime(data['child_birthday'], errors='coerce')
(interview_date     0
 child_age_years    0
 child_birthday     0
 EC6                0
 EC7                0
 EC8                0
 EC9                0
 EC10               0
 EC11               0
 EC12               0
 EC13               0
 EC14               0
 EC15               0
 dtype: int64,
   interview_date child_age_years child_birthday ec6 ec7 ec8 ec9 ec10 ec11  \
 0     2019-03-31               3     2016-02-20   2   2   1   1    2    1
 1     2019-03-14               3     2015-10-19   2   2   2   1    2    2
 2     2019-03-17               4     2014-10-26   2   2   2   1    1    1
 3     2019-03-14               3     2015-03-21   1   2   2   1    2    1
 4     2019-03-14               3     2015-05-18   2   1   1   1    2    1

   ec12 ec13 ec14 ec15
 0    1    1    2    2
 1    2    1    1    1
 2    2    1    2    1
 3    1    1    2    1
 4    2    1    2    1  )
```

```
# Convert specified columns to int64
columns_to_convert = ['ec6', 'ec7', 'ec8', 'ec9', 'ec10', 'ec11', 'ec12', 'ec13', 'ec14', 'ec15']
data[columns_to_convert] = data[columns_to_convert].astype('int64')

# Replace values: 2 -> 0, 8 -> 0
data[columns_to_convert] = data[columns_to_convert].replace({2: 0, 8: 0})

# Display the first few rows to confirm the changes
data.head()
# Get distinct values for each of the specified columns
distinct_values = {col: data[col].unique() for col in columns_to_convert}

distinct_values
```

```
{'ec6': array([0, 1, 9]),
 'ec7': array([0, 1, 9]),
 'ec8': array([1, 0, 9]),
 'ec9': array([1, 0, 9]),
 'ec10': array([0, 1, 9]),
 'ec11': array([1, 0, 9]),
 'ec12': array([1, 0, 9]),
 'ec13': array([1, 0, 9]),
 'ec14': array([0, 1, 9]),
 'ec15': array([0, 1, 9])}
```

```
# Filter the data to exclude rows where any of the specified columns contain the value 9
filtered_data = data[~data[columns_to_convert].isin([9]).any(axis=1)]

# Display the shape and the first few rows of the filtered data
distinct_values = {col: filtered_data[col].unique() for col in columns_to_convert}

distinct_values
```

```
{'ec6': array([0, 1]),
 'ec7': array([0, 1]),
 'ec8': array([1, 0]),
 'ec9': array([1, 0]),
 'ec10': array([0, 1]),
 'ec11': array([1, 0]),
 'ec12': array([1, 0]),
 'ec13': array([1, 0]),
 'ec14': array([0, 1]),
 'ec15': array([0, 1])}
```

```
# Rename the columns with the provided new names
new_column_names = {
    'ec6': "Can (name) identify or name at least ten letters of the alphabet?",
    'ec7': "Can (name) read at least four simple, popular words?",
    'ec8': "Does (name) know the name and recognize the symbol of all numbers from 1 to 10?",
    'ec9': "Can (name) pick up a small object with two fingers, like a stick or a rock from the ground?",
    'ec10': "Is (name) sometimes too sick to play?",
    'ec11': "Does (name) follow simple directions on how to do something correctly?",
    'ec12': "When given something to do, is (name) able to do it independently?",
    'ec13': "Does (name) get along well with other children?",
    'ec14': "Does (name) kick, bite, or hit other children or adults?",
    'ec15': "Does (name) get distracted easily?"
}

filtered_data.rename(columns=new_column_names, inplace=True)

# Display the first few rows to confirm the column name changes
filtered_data.head()
```

```
<ipython-input-8-daeb1d166ab3>:15: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-c
  filtered_data.rename(columns=new_column_names, inplace=True)
```

	interview_date	child_age_years	child_birthday	Can (name) identify or name at least ten letters of the alphabet?	Can (name) read at least four simple, popular words?	Does (name) know the name and recognize the symbol of all numbers from 1 to 10?	Can (name) pick up a small object with two fingers, like a stick or a rock from the ground?	Is (name) sometimes too sick to play?	Does (name) follow simple directions on how to do something correctly?	When given something to do, is (name) able to do it independently?	(get well chil
0	2019-03-31	3	2016-02-20	0	0	1	1	0	1	1	
1	2019-03-14	3	2015-10-19	0	0	0	1	0	0	0	
2	2019-03-17	4	2014-10-26	0	0	0	1	1	1	0	
3	2019-03-14	3	2015-03-21	1	0	0	1	0	1	1	
4	2019-03-14	3	2015-05-18	0	1	1	1	0	1	0	

Extract cleaned data countries dataset to new Google sheet

```
# Add a new sheet with a name and specify rows and columns
worksheet_title = 'Zimbabwe_children_under5_Cleaned'

ws = spreadsheet.worksheet(worksheet_title)

# Use the gspread_dataframe to set the DataFrame to the sheet
from gspread_dataframe import set_with_dataframe
# Set the DataFrame to the worksheet
set_with_dataframe(ws, filtered_data)
```

Data Analysis

Descriptive Analytics: What Happened?

Objective: To understand the historical data and summarize it into meaningful patterns or trends.

Statistics for percent correct (Mean) by child age in years

```
filtered_data = filtered_data.rename(columns=new_column_names)

# Explicitly select only the numeric columns (excluding datetime columns)
numeric_columns = filtered_data.select_dtypes(include='number').columns

#calculate the summary statistics for percent correct by child age in years
summary_stats = filtered_data.groupby('child_age_years')[numeric_columns].mean() * 100

# Display the summary statistics DataFrame directly

print(summary_stats)
```

```
                 Can (name) identify or name at least ten letters of the alphabet?  \
child_age_years
3                                                         6.377760
4                                                        14.153132

                 Can (name) read at least four simple, popular words?  \
child_age_years
3                                                         7.686018
4                                                        13.379737

                 Does (name) know the name and recognize the symbol of all numbers from 1 to 10?  \
child_age_years
3                                                        11.120196
4                                                        24.903326

                 Can (name) pick up a small object with two fingers, like a stick or a rock from the ground?  \
child_age_years
3                                                        90.923957
4                                                        92.807425

                 Is (name) sometimes too sick to play?  \
child_age_years
3                                           39.820114
4                                           37.664346

                 Does (name) follow simple directions on how to do something correctly?  \
child_age_years
3                                                        83.973835
4                                                        89.095128

                 When given something to do, is (name) able to do it independently?  \
child_age_years
3                                                        71.054783
4                                                        76.720804

                 Does (name) get along well with other children?  \
child_age_years
3                                                      95.829926
4                                                      96.365043
```

```
                 Does (name) kick, bite, or hit other children or adults?  \
child_age_years
3                                                        50.695012
4                                                        47.177108

                 Does (name) get distracted easily?
child_age_years
3                                         37.939493
4                                         36.736272
```

Statistics for (Count) by child age in years

```
filtered_data = filtered_data.rename(columns=new_column_names)

# Explicitly select only the numeric columns (excluding datetime columns)
numeric_columns = filtered_data.select_dtypes(include='number').columns

# Recalculate the summary statistics for percent correct by child age in years
summary_stats_corrected = filtered_data.groupby('child_age_years')[numeric_columns].count()

# Display the summary statistics DataFrame directly
print(summary_stats_corrected)
```

```
                 Can (name) identify or name at least ten letters of the alphabet?  \
child_age_years
3                                                             1223
4                                                             1293

                 Can (name) read at least four simple, popular words?  \
child_age_years
3                                                             1223
4                                                             1293

                 Does (name) know the name and recognize the symbol of all numbers from 1 to 10?  \
child_age_years
3                                                             1223
4                                                             1293

                 Can (name) pick up a small object with two fingers, like a stick or a rock from the ground?  \
child_age_years
3                                                             1223
4                                                             1293

                 Is (name) sometimes too sick to play?  \
child_age_years
3                                           1223
4                                           1293

                 Does (name) follow simple directions on how to do something correctly?  \
child_age_years
3                                                             1223
4                                                             1293

                 When given something to do, is (name) able to do it independently?  \
child_age_years
3                                                             1223
4                                                             1293

                 Does (name) get along well with other children?  \
child_age_years
3                                                           1223
4                                                           1293

                 Does (name) kick, bite, or hit other children or adults?  \
child_age_years
3                                                             1223
4                                                             1293

                 Does (name) get distracted easily?
child_age_years
3                                       1223
4                                       1293
```

Statistics for (Standard Deviation) by child age in years

```
filtered_data = filtered_data.rename(columns=new_column_names)

# Explicitly select only the numeric columns (excluding datetime columns)
numeric_columns = filtered_data.select_dtypes(include='number').columns

# Recalculate the summary statistics for percent correct by child age in years
summary_stats_corrected = filtered_data.groupby('child_age_years')[numeric_columns].std()

# Display the summary statistics DataFrame directly
print(summary_stats_corrected)
```

```
                 Can (name) identify or name at least ten letters of the alphabet?  \
child_age_years
3                                                         0.244456
4                                                         0.348704

                 Can (name) read at least four simple, popular words?  \
child_age_years
3                                                         0.266478
4                                                         0.340566

                 Does (name) know the name and recognize the symbol of all numbers from 1 to 10?  \
child_age_years
3                                                         0.314511
4                                                         0.432620

                 Can (name) pick up a small object with two fingers, like a stick or a rock from the ground?  \
child_age_years
3                                                         0.287386
4                                                         0.258465

                 Is (name) sometimes too sick to play?  \
child_age_years
3                                            0.489728
4                                            0.484732

                 Does (name) follow simple directions on how to do something correctly?  \
child_age_years
3                                                         0.366999
4                                                         0.311821

                 When given something to do, is (name) able to do it independently?  \
child_age_years
3                                                         0.453694
4                                                         0.422774

                 Does (name) get along well with other children?  \
child_age_years
3                                                       0.199986
4                                                       0.187231

                 Does (name) kick, bite, or hit other children or adults?  \
child_age_years
3                                                         0.500156
4                                                         0.499396

                 Does (name) get distracted easily?
child_age_years
3                                         0.485435
4                                         0.482273
```

educational areas

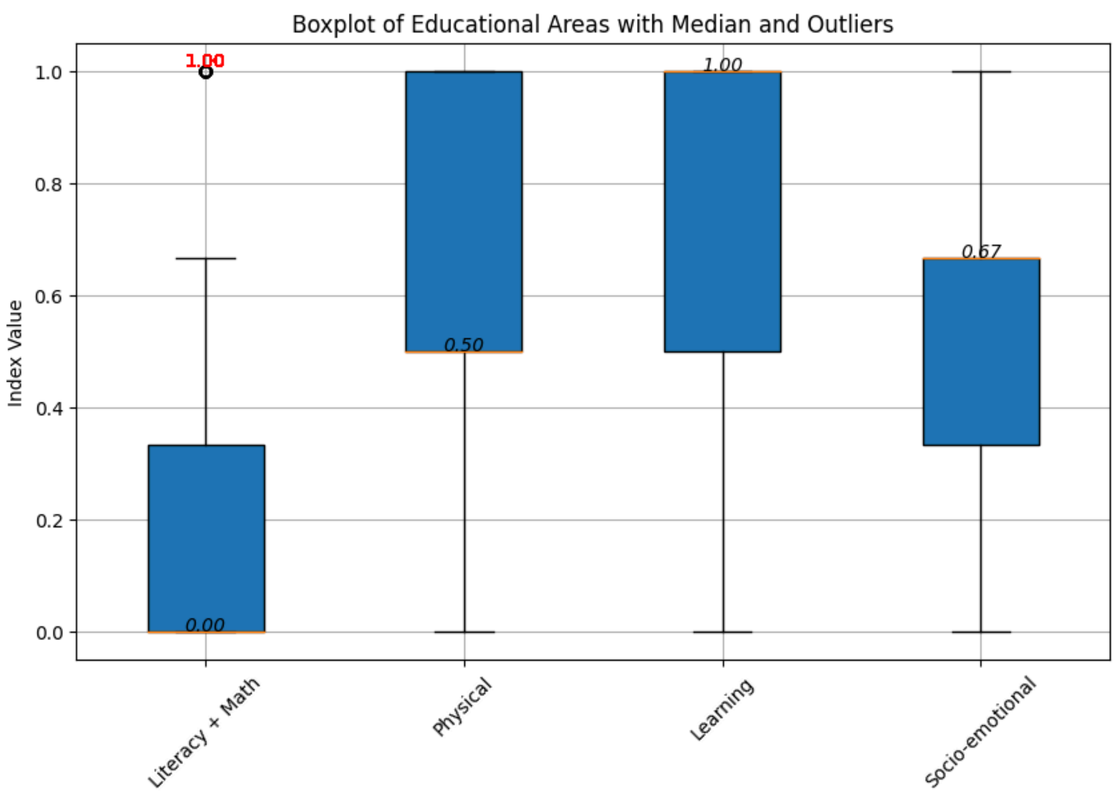
```
# Map the questions to the educational areas
educational_areas_mapping = {
    "Literacy + Math": ["Can (name) identify or name at least ten letters of the alphabet?",
                        "Can (name) read at least four simple, popular words?",
                        "Does (name) know the name and recognize the symbol of all numbers from 1 to 10?"],
    "Physical": ["Can (name) pick up a small object with two fingers, like a stick or a rock from the ground?",
                 "Is (name) sometimes too sick to play?"],
    "Learning": ["Does (name) follow simple directions on how to do something correctly?",
                 "When given something to do, is (name) able to do it independently?"],
    "Socio-emotional": ["Does (name) get along well with other children?",
                        "Does (name) kick, bite, or hit other children or adults?",
                        "Does (name) get distracted easily?"]
}

# Create a new DataFrame to display the mapping
mapping_df = pd.DataFrame.from_dict(educational_areas_mapping, orient='index').transpose()
# Display the mapping_df results
from IPython.display import display

display(mapping_df)
```

	Literacy + Math	Physical	Learning	Socio-emotional
0	Can (name) identify or name at least ten lette...	Can (name) pick up a small object with two fin...	Does (name) follow simple directions on how to...	Does (name) get along well with other children?
1	Can (name) read at least four simple, popular ...	Is (name) sometimes too sick to play?	When given something to do, is (name) able to ...	Does (name) kick, bite, or hit other children ...
2	Does (name) know the name and recognize the sy...	None	None	Does (name) get distracted easily?

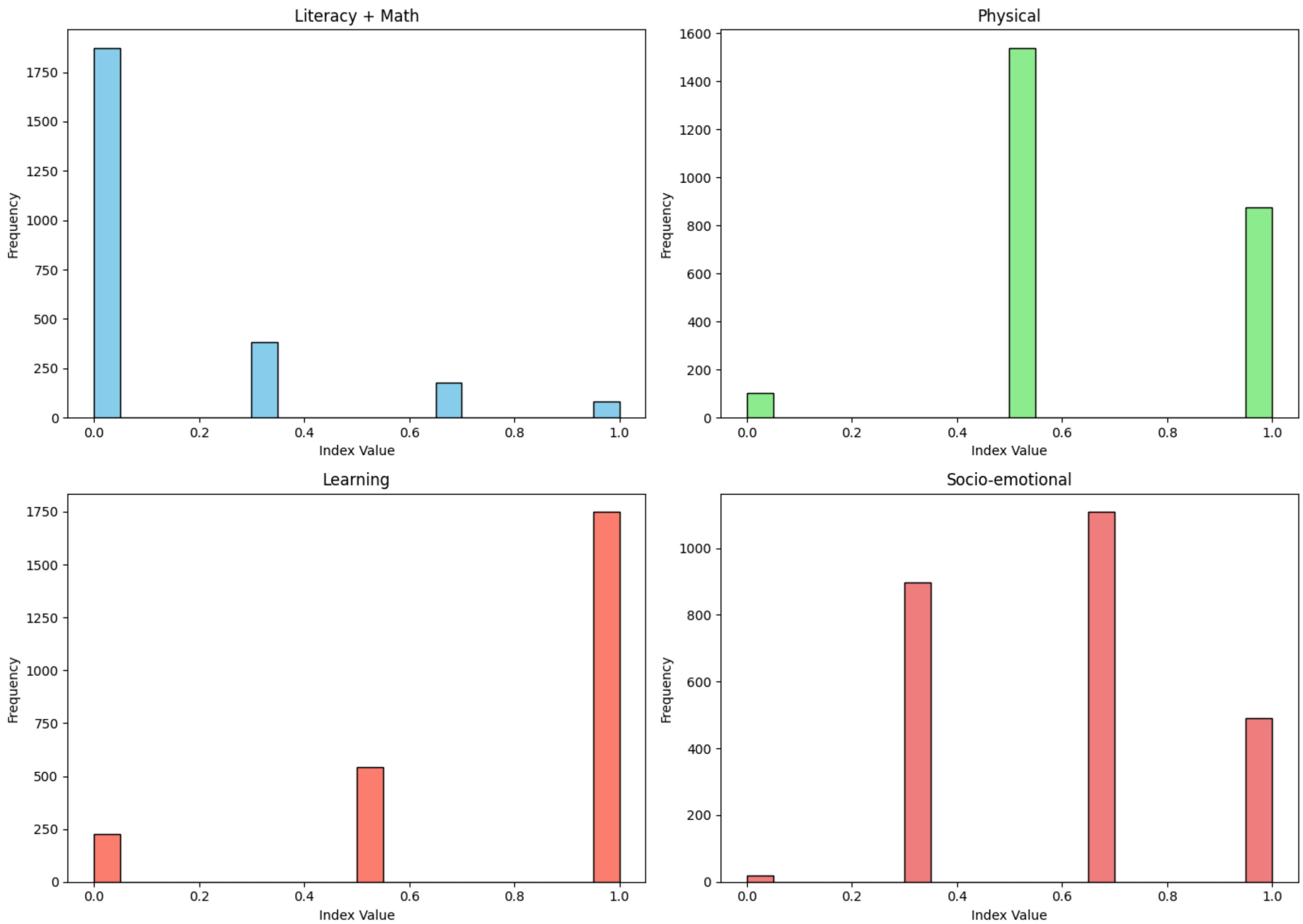
```
# Calculate the index for each educational area by taking the arithmetic average of the relevant columns

# First, create a dictionary to hold the calculated indexes
index_columns = {}

# Calculate the arithmetic average for each area
for area, questions in educational_areas_mapping.items():
    index_columns[area] = filtered_data[questions].mean(axis=1)

# Combine these indexes into a new DataFrame
indexes_df = pd.DataFrame(index_columns)

# Add the child age years to the DataFrame for reference
indexes_df['child_age_years'] = filtered_data['child_age_years']
indexes_df['interview_date'] = filtered_data['interview_date']
indexes_df['child_birthday'] = filtered_data['child_birthday']

# Reorder the columns for better readability
indexes_df = indexes_df[['interview_date', 'child_birthday', 'child_age_years',
                         'Literacy + Math', 'Physical', 'Learning', 'Socio-emotional']]
# Display the first few rows of the indexes DataFrame to the user
print(indexes_df.head())
```

```
  interview_date child_birthday child_age_years  Literacy + Math  Physical  \
0     2019-03-31     2016-02-20               3         0.333333       0.5
1     2019-03-14     2015-10-19               3         0.000000       0.5
2     2019-03-17     2014-10-26               4         0.000000       1.0
3     2019-03-14     2015-03-21               3         0.333333       0.5
4     2019-03-14     2015-05-18               3         0.666667       0.5

   Learning  Socio-emotional
0       1.0         0.333333
1       0.0         1.000000
2       0.5         0.666667
3       1.0         0.666667
4       0.5         0.666667
```

```
import matplotlib.pyplot as plt

# Data preparation for boxplot
data_to_plot = indexes_df[['Literacy + Math', 'Physical', 'Learning', 'Socio-emotional']]

# Create the boxplot
plt.figure(figsize=(10, 6))
boxprops = plt.boxplot(data_to_plot.values, labels=data_to_plot.columns, patch_artist=True, showmeans=False)

# Annotate medians and outliers
for i, line in enumerate(boxprops['medians']):
    # Get the median value
    median_value = line.get_ydata()[0]
    plt.text(i + 1, median_value, f'{median_value:.2f}', horizontalalignment='center', style='italic')

for i, line in enumerate(boxprops['fliers']):
    # Annotate each outlier
    for flier in line.get_ydata():
        plt.text(i + 1, flier, f'{flier:.2f}', horizontalalignment='center', verticalalignment='bottom', color='red')

# Set plot title and labels
plt.title('Boxplot of Educational Areas with Median and Outliers')
plt.ylabel('Index Value')
plt.xticks(rotation=45)
plt.grid(True)

# Show the plot
plt.show()
```



```
import matplotlib.pyplot as plt

# Plotting histograms for each educational area
plt.figure(figsize=(14, 10))

# Plotting histogram for Literacy + Math
plt.subplot(2, 2, 1)
plt.hist(indexes_df['Literacy + Math'], bins=20, color='skyblue', edgecolor='black')
plt.title('Literacy + Math')
plt.xlabel('Index Value')
plt.ylabel('Frequency')

# Plotting histogram for Physical
plt.subplot(2, 2, 2)
plt.hist(indexes_df['Physical'], bins=20, color='lightgreen', edgecolor='black')
plt.title('Physical')
plt.xlabel('Index Value')
plt.ylabel('Frequency')

# Plotting histogram for Learning
plt.subplot(2, 2, 3)
plt.hist(indexes_df['Learning'], bins=20, color='salmon', edgecolor='black')
plt.title('Learning')
plt.xlabel('Index Value')
plt.ylabel('Frequency')

# Plotting histogram for Socio-emotional
plt.subplot(2, 2, 4)
plt.hist(indexes_df['Socio-emotional'], bins=20, color='lightcoral', edgecolor='black')
plt.title('Socio-emotional')
plt.xlabel('Index Value')
plt.ylabel('Frequency')

plt.tight_layout()
plt.show()
```




```
# Add a new sheet with a name and specify rows and columns
worksheet_title = 'educational_areas_mapping'

ws = spreadsheet.worksheet(worksheet_title)

# Use the gspread_dataframe to set the DataFrame to the sheet
from gspread_dataframe import set_with_dataframe
# Set the DataFrame to the worksheet
set_with_dataframe(ws, indexes_df)
```

Diagnostic Analytics: Why Did It Happen?

Objective: To investigate and understand the causes behind the patterns or anomalies identified in descriptive analytics.

*T-Test and ANOVA *

```
from scipy.stats import ttest_ind, f_oneway
import pandas as pd

# Define age groups for comparison
age_groups = {
    '3 years': indexes_df[indexes_df['age_in_months'].between(36, 47)],
    '4 years': indexes_df[indexes_df['age_in_months'].between(48, 59)]
}

# Initialize a dictionary to store test results
test_results = {}

# Perform t-tests for each educational area between the age groups
for area in ['Literacy + Math', 'Physical', 'Learning', 'Socio-emotional']:
    group_3 = age_groups['3 years'][area]
    group_4 = age_groups['4 years'][area]

    t_stat, p_val = ttest_ind(group_3, group_4, nan_policy='omit')

    test_results[area] = {
        'T-Statistic': t_stat,
        'P-Value': p_val
    }

# Perform ANOVA to assess differences across multiple age groups (e.g., 4 years, 5 years)
anova_results = {}
for area in ['Literacy + Math', 'Physical', 'Learning', 'Socio-emotional']:
    group_3 = age_groups['3 years'][area]
    group_4 = age_groups['4 years'][area]

    f_stat, p_val = f_oneway(group_3, group_4)

    anova_results[area] = {
        'F-Statistic': f_stat,
        'P-Value': p_val
    }

# Combine the results into a DataFrame for display
t_test_results_df = pd.DataFrame(test_results).transpose()
anova_results_df = pd.DataFrame(anova_results).transpose()

# Displaying the t-test and ANOVA results
print("T-Test Results:")
print(t_test_results_df)

print("\nANOVA Results:")
print(anova_results_df)
```

```
T-Test Results:
                 T-Statistic       P-Value
Literacy + Math    -6.542320  7.600573e-11
Physical            0.801411  4.229851e-01
Learning           -2.590187  9.659248e-03
Socio-emotional     2.508550  1.219820e-02

ANOVA Results:
                 F-Statistic       P-Value
Literacy + Math    42.801947  7.600573e-11
Physical            0.642260  4.229851e-01
Learning            6.709071  9.659248e-03
Socio-emotional     6.292825  1.219820e-02
```

Summary:

- **Literacy + Math**: Both the T-Test and ANOVA show highly significant differences, indicating that literacy and math skills vary significantly between the groups being compared Age 3 and 4 Years. This suggests a critical period or intervention opportunity to support these skills.
- **Physical**: No significant differences are found in physical abilities, suggesting that these abilities are stable across the groups.
- **Learning**: Significant differences are observed, indicating that learning skills differ between the groups, which could reflect developmental milestones or varying educational experiences.
- **Socio-emotional**: Significant differences are observed in socio-emotional skills, indicating that these skills may develop differently across the groups, possibly due to varying social environments or interventions.
- *Actionable Insights:**

- **Focus on Literacy + Math**: Given the significant differences, educational programs should target the identified age group or groups where Literacy + Math skills show improvement or decline, to optimize learning outcomes.
- **Monitor Learning and Socio-emotional Development**: These areas also show significant differences, suggesting that targeted interventions or support might be necessary to help children who may be lagging in these areas.
- **Consistency in Physical Development:** Since no significant differences were observed in physical skills, consistent support and monitoring should suffice to ensure steady development.

Calculate the Cronbach’s Alpha

```
from scipy.stats import tmean
import numpy as np

# Function to calculate Cronbach's Alpha
def cronbach_alpha(items_scores):
    items_scores = np.array(items_scores)
    item_variances = items_scores.var(axis=1, ddof=1)
    total_score_variances = items_scores.sum(axis=0).var(ddof=1)
    n_items = items_scores.shape[0]
    return (n_items / (n_items - 1)) * (1 - item_variances.sum() / total_score_variances)

# Calculate Cronbach's Alpha for each educational area
cronbach_alphas = {}
num_observations = {}

for area, questions in educational_areas_mapping.items():
    items_scores = filtered_data[questions].transpose()
    cronbach_alphas[area] = cronbach_alpha(items_scores)
    num_observations[area] = len(filtered_data[questions].dropna())

# Create a summary table with Cronbach's Alpha and the number of observations
summary_table = pd.DataFrame({
    'Cronbach\'s Alpha': cronbach_alphas,
    'Number of Observations': num_observations
})

# Display the summary table to the user
print(summary_table.head())
```

```
                 Cronbach's Alpha  Number of Observations
Literacy + Math          0.620896                    2516
Physical                -0.113300                    2516
Learning                 0.519670                    2516
Socio-emotional          0.081009                    2516
```

Interpretation of Results: Literacy + Math (Alpha = 0.6209)

Interpretation: A Cronbach's Alpha of 0.62 suggests moderate internal consistency among the questions related to Literacy and Math. This indicates that the questions in this group are reasonably consistent and measure a related underlying concept, but there might still be room for improvement in reliability.

Physical (Alpha = -0.1133)

Interpretation: A negative Cronbach's Alpha indicates a lack of internal consistency, and it might suggest that the two items in this category (EC9 and EC10) are not measuring the same concept or are inversely related. This could imply that the items in the Physical category might not be well-suited for combining into a single index, or there could be issues with how the items are coded or interpreted.

Learning (Alpha = 0.5197)

Interpretation: An Alpha of 0.52 is on the lower side, suggesting that the questions related to Learning have some degree of internal consistency, but they are not very strongly correlated. This means the items may not be capturing the same concept consistently.

Socio-emotional (Alpha = 0.0810)

Interpretation: A very low Cronbach's Alpha (0.08) indicates poor internal consistency among the items related to Socio-emotional skills. This suggests that these items may not be measuring the same underlying concept, or there could be significant variability in how these questions are interpreted.

Number of Observations: For each category, the number of observations (2,516) reflects the number of respondents (children) for whom data was available for all questions in that category. Since the number is consistent across all categories, it indicates that the entire dataset was used for each calculation.

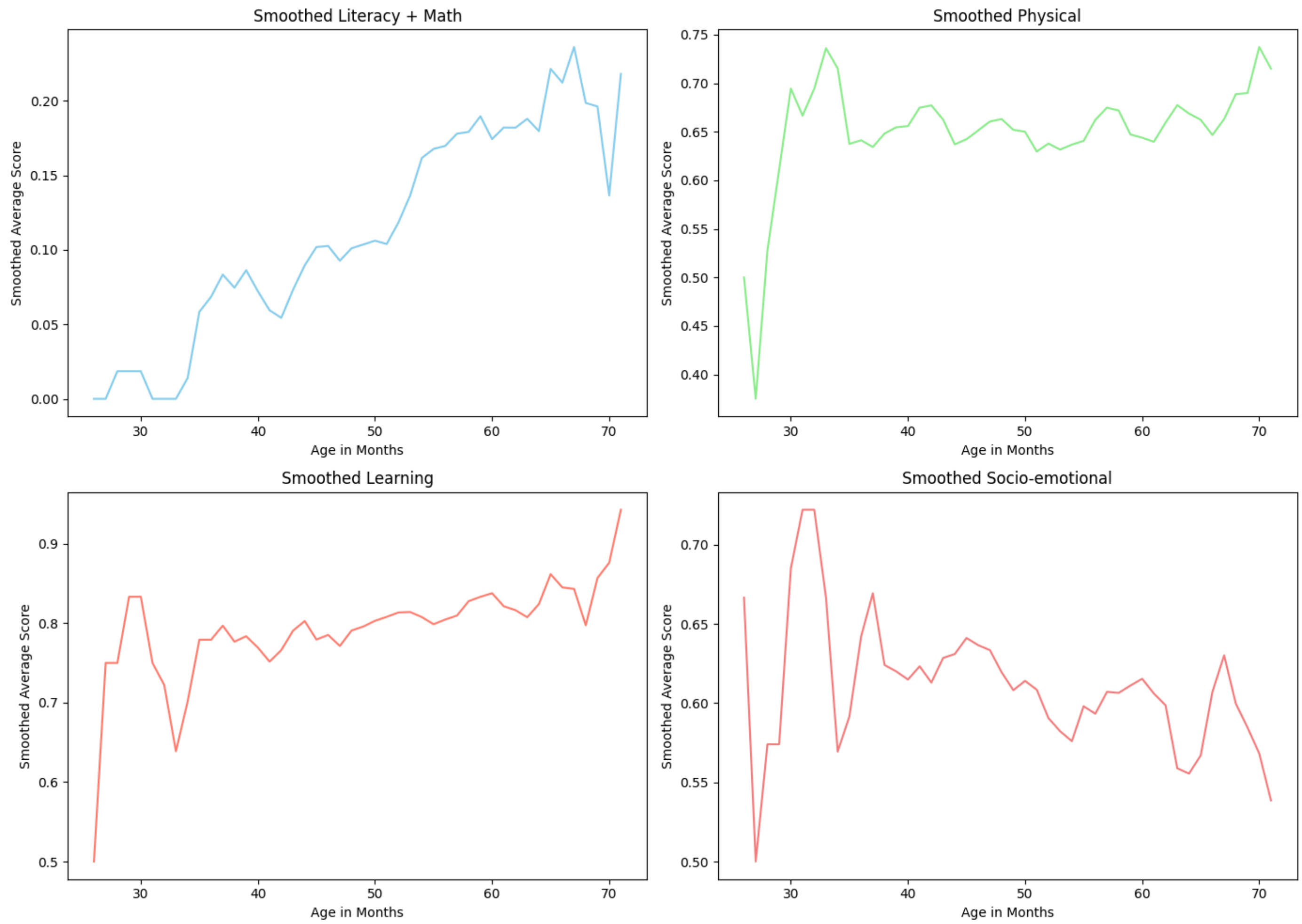
Summary:

- **Literacy_Math** shows moderate reliability, suggesting that these questions can be used together to form an index.
- **Physical** has a negative Alpha, indicating that the items may not be related or might need to be reconsidered.
- **Learning** has lower reliability, implying that the questions are somewhat related but may need refinement.
- **Socio-emotional** shows poor consistency, suggesting these items are not measuring a cohesive concept.

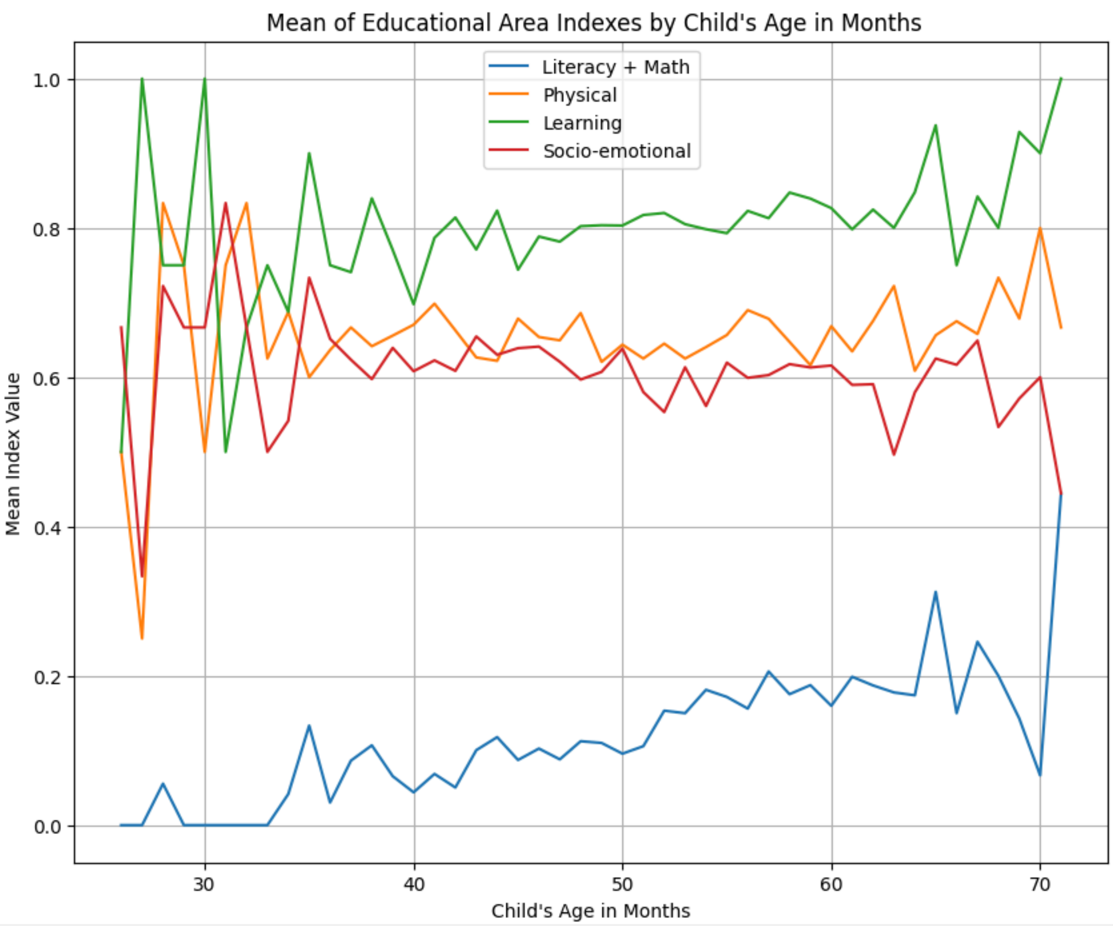
Predictive Analytics: What Could Happen?

Objective: To use historical data to forecast future trends and outcomes.

```
import pandas as pd
import matplotlib.pyplot as plt

# Define the window size for the moving average
window_size = 3

# Calculate moving averages for each educational area
smoothed_data = indexes_df[['age_in_months', 'Literacy + Math', 'Physical', 'Learning', 'Socio-emotional']].groupby('age_in_months').mean()

# Plotting the smoothed data
plt.figure(figsize=(14, 10))

# Plot for Literacy + Math
plt.subplot(2, 2, 1)
plt.plot(smoothed_data.index, smoothed_data['Literacy + Math'], color='skyblue')
plt.title('Smoothed Literacy + Math')
plt.xlabel('Age in Months')
plt.ylabel('Smoothed Average Score')

# Plot for Physical
plt.subplot(2, 2, 2)
plt.plot(smoothed_data.index, smoothed_data['Physical'], color='lightgreen')
plt.title('Smoothed Physical')
plt.xlabel('Age in Months')
plt.ylabel('Smoothed Average Score')

# Plot for Learning
plt.subplot(2, 2, 3)
plt.plot(smoothed_data.index, smoothed_data['Learning'], color='salmon')
plt.title('Smoothed Learning')
plt.xlabel('Age in Months')
plt.ylabel('Smoothed Average Score')

# Plot for Socio-emotional
plt.subplot(2, 2, 4)
plt.plot(smoothed_data.index, smoothed_data['Socio-emotional'], color='lightcoral')
plt.title('Smoothed Socio-emotional')
plt.xlabel('Age in Months')
plt.ylabel('Smoothed Average Score')

plt.tight_layout()
plt.show()
```


Smoothed Literacy + Math
Smoothed Physical
Smoothed Learning
Smoothed Socio-emotional
Smoothed Average Score
Age in Months

```
import matplotlib.pyplot as plt
# Calculate the child's age in months at the time of the interview
indexes_df['age_in_months'] = (indexes_df['interview_date'] - indexes_df['child_birthday']).dt.days // 30

# Convert all relevant columns to numeric and coerce any errors to NaN
indexes_df['age_in_months'] = pd.to_numeric(indexes_df['age_in_months'], errors='coerce')
indexes_df['child_age_years'] = pd.to_numeric(indexes_df['child_age_years'], errors='coerce')
indexes_df[['Literacy + Math', 'Physical', 'Learning', 'Socio-emotional']] = indexes_df[[
    'Literacy + Math', 'Physical', 'Learning', 'Socio-emotional']].apply(pd.to_numeric, errors='coerce')

# Drop rows with any NaN values that might interfere with the groupby operation
indexes_df_cleaned = indexes_df.dropna(subset=['age_in_months', 'Literacy + Math', 'Physical', 'Learning', 'Socio-emotional'])

# Calculate the conditional mean
conditional_means = indexes_df_cleaned.groupby('age_in_months').mean()[['Literacy + Math', 'Physical', 'Learning', 'Socio-emotional']]

# Plot the conditional means
plt.figure(figsize=(10, 8))
for column in conditional_means.columns:
    plt.plot(conditional_means.index, conditional_means[column], label=column)

plt.xlabel('Child\'s Age in Months')
plt.ylabel('Mean Index Value')
plt.title(' Mean of Educational Area Indexes by Child\'s Age in Months')
plt.legend()
plt.grid(True)
plt.show()
```



The plot you see represents how the average values of the educational area indexes change as children grow older, specifically by their age in months. Here's a detailed explanation:

X-Axis: Child's Age in Months

- The horizontal axis (X-axis) shows the child's age in months. This is calculated as the difference between the interview date and the child's birthdate.
- The ages are presented in months to provide a more granular view of the child's development over time, rather than just grouping by years.

Y-Axis: Mean Index Value

- The vertical axis (Y-axis) shows the mean index value for each educational area (Literacy + Math, Physical, Learning, Socio-emotional).
- The index values range between 0 and 1 because they represent the proportion of correct responses to the relevant questions within each category.

Lines Representing Each Educational Area

- **Literacy + Math**: This line shows the average performance in areas related to literacy and math (e.g., recognizing letters, reading simple words).
- **Physical**: This line reflects physical abilities (e.g., fine motor skills, whether the child is sometimes too sick to play).
- **Learning**: This represents cognitive learning abilities (e.g., following directions, completing tasks independently).
- **Socio-emotional**: This line indicates social and emotional skills (e.g., getting along with others, not getting easily distracted).

Interpretation of the Plot:

Trend Analysis: The plot shows how the average scores in each area evolve as children get older.

- **Increasing Trends**: If you see an upward trend, it suggests that as children age, they tend to perform better in that area. For instance, Literacy + Math might show an upward trend, indicating that older children are better at recognizing letters and numbers.
- **Flat or Decreasing Trends**: If a line is flat or declining, it might indicate that performance in that area doesn't change much with age or could decrease for various reasons.

Comparison Between Areas: By comparing the different lines, you can see how development in one area compares to another. For example, if the Socio-emotional line is consistently higher or lower than the Physical line, it suggests differences in how children develop skills in these areas. **Why This Analysis Matters:**

- **Developmental Insights**: Understanding how these indexes change with age helps identify critical periods in a child's development. For example, if Literacy + Math skills rapidly improve between certain months, it might indicate an ideal window for targeted educational interventions.
- **Program Evaluation**: For educators or program designers, this analysis can inform the effectiveness of programs aimed at improving certain skills. If a program targets physical skills but the Physical index doesn't improve as expected, adjustments might be needed.

Summary: The plot helps visualize the developmental trajectory of children across different educational domains. By analyzing how these indexes change with age, stakeholders can better understand the timing and effectiveness of educational interventions, ultimately supporting better outcomes for children.

OLS regression results

```
import pandas as pd
import statsmodels.api as sm
import numpy as np

# Assuming indexes_df is already prepared as per previous steps

# Remove rows with missing values
cleaned_data = indexes_df.dropna(subset=['age_in_months', 'Literacy + Math', 'Physical', 'Learning', 'Socio-emotional'])

# Prepare the data for OLS regression
X_cleaned = cleaned_data['age_in_months']
X_cleaned = sm.add_constant(X_cleaned)  # Adds a constant term to the predictor

# Run OLS regression for each educational area index
results_cleaned = {}

for area in ['Literacy + Math', 'Physical', 'Learning', 'Socio-emotional']:
    y_cleaned = cleaned_data[area]
    model_cleaned = sm.OLS(y_cleaned, X_cleaned).fit()
    results_cleaned[area] = {
        'Coefficient on Age': model_cleaned.params['age_in_months'],
        'Standard Error': model_cleaned.bse['age_in_months'],
        'R-squared': model_cleaned.rsquared,
        'Number of Observations': int(model_cleaned.nobs)
    }

# Convert the results to a DataFrame for display
ols_results_cleaned_df = pd.DataFrame(results_cleaned).transpose()

# Display the OLS regression results
from IPython.display import display

display(ols_results_cleaned_df)
```

	Coefficient on Age	Standard Error	R-squared	Number of Observations
Literacy + Math	0.005526	0.000609	0.032057	2490.0
Physical	0.000206	0.000668	0.000038	2490.0
Learning	0.002697	0.000792	0.004641	2490.0
Socio-emotional	-0.001596	0.000607	0.002769	2490.0

The Exponential Smoothing forecasts

Interpretation:

- **Coefficient on Age**: This indicates the average change in the index value per month of age. For example, in the Literacy + Math category, the index increases by approximately 0.0055 for each additional month of age.
- **Standard Error**: This measures the precision of the estimated coefficient. Smaller values indicate more precise estimates.
- **R-squared**: This indicates the proportion of the variance in the index that can be explained by the child's age. The values are relatively low, suggesting that while age does have an impact, there are other factors also contributing to the index scores.
- **Number of Observations**: This shows the number of data points used in each regression. After cleaning, 2,490 observations were used.

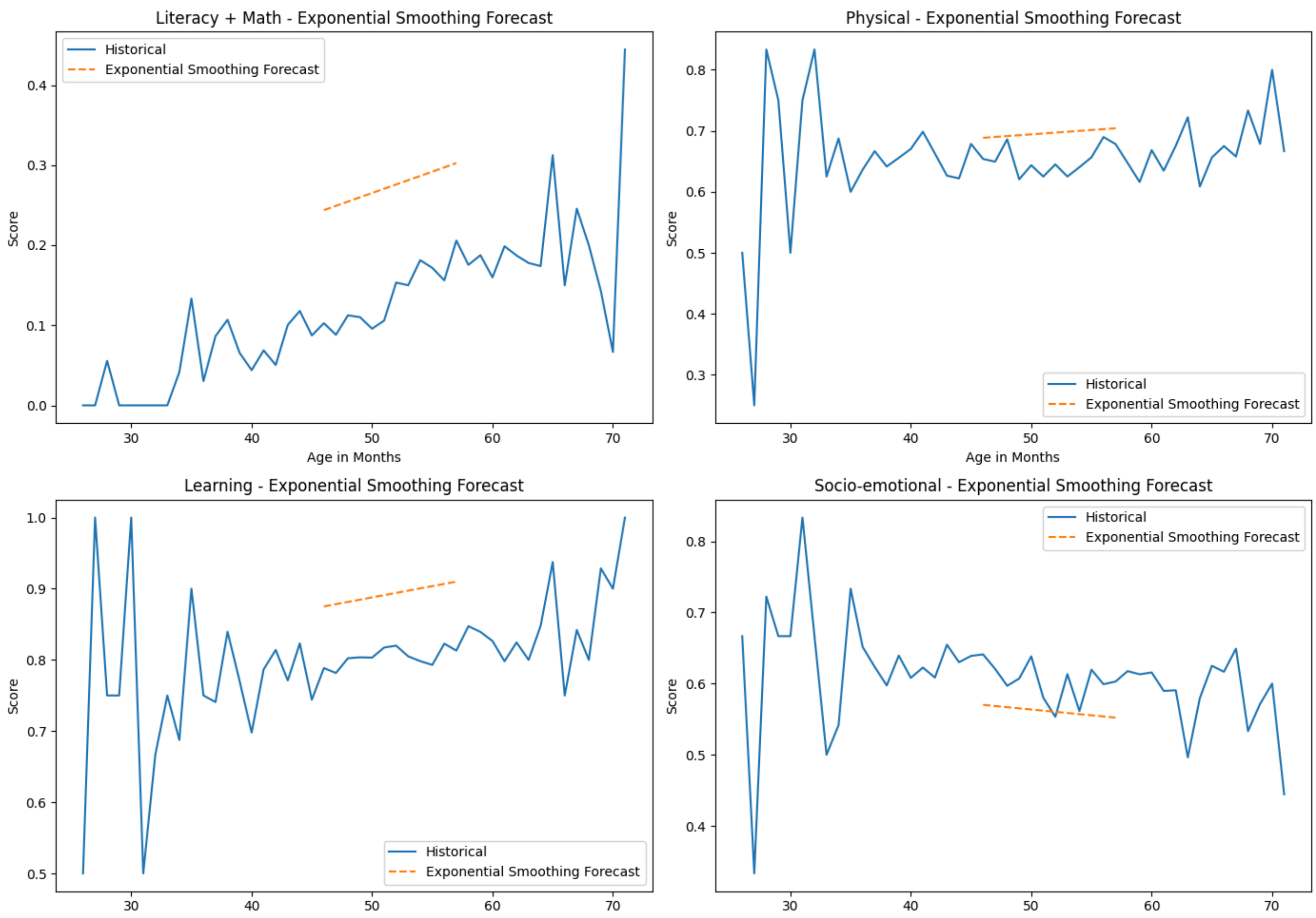
```
!pip install statsmodels
!pip install pmdarima
```

```
Requirement already satisfied: statsmodels in /usr/local/lib/python3.10/dist-packages (0.14.2)
Requirement already satisfied: numpy>=1.22.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (1.26.4)
Requirement already satisfied: scipy!=1.9.2,>=1.8 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (1.13.1)
Requirement already satisfied: pandas!=2.1.0,>=1.4 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (2.1.4)
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (0.5.6)
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.10/dist-packages (from statsmodels) (24.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas!=2.1.0,>=1.4->statsmodels
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas!=2.1.0,>=1.4->statsmodels) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas!=2.1.0,>=1.4->statsmodels) (2024.
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.6->statsmodels) (1.16.0)
Collecting pmdarima
  Downloading pmdarima-2.0.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl.metadata (7.8 kB)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.4.2)
```

```
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (3.0.11)
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.26.4)
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (2.1.4)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.3.2)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.13.1)
Requirement already satisfied: statsmodels>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (0.14.2)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (2.0.7)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (71.0.4)
Requirement already satisfied: packaging>=17.1 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (24.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2024.1)
Requirement already satisfied: tzdata>=2022.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2024.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22->pmdarima) (3.5
Requirement already satisfied: patsy>=0.5.6 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13.2->pmdarima) (0.5.6)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.6->statsmodels>=0.13.2->pmdarima) (1.16.
Downloading pmdarima-2.0.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl (2.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 31.0 MB/s eta 0:00:00
Installing collected packages: pmdarima
Successfully installed pmdarima-2.0.4
```

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from statsmodels.tsa.holtwinters import ExponentialSmoothing

# Assuming indexes_df is already prepared

# Step 1: Ensure 'age_in_months' is numeric and all target columns are numeric
# Convert 'age_in_months' and the target columns to numeric types
indexes_df['age_in_months'] = pd.to_numeric(indexes_df['age_in_months'], errors='coerce')

# Ensure all target columns are numeric
for col in ['Literacy + Math', 'Physical', 'Learning', 'Socio-emotional']:
    indexes_df[col] = pd.to_numeric(indexes_df[col], errors='coerce')

# Step 2: Drop any rows with NaN values in 'age_in_months' or the target columns
indexes_df = indexes_df.dropna(subset=['age_in_months', 'Literacy + Math', 'Physical', 'Learning', 'Socio-emotional'])

# Step 3: Group by age in months and calculate the mean for each educational area
time_series_data = indexes_df.groupby('age_in_months')[['Literacy + Math', 'Physical', 'Learning', 'Socio-emotional']].mean()

# Step 4: Define a function to fit the Exponential Smoothing model and make predictions
def exp_smoothing_forecast(series, steps=12):
    # Fit the Exponential Smoothing model (using additive trend and no seasonal component)
    model = ExponentialSmoothing(series, trend='add', seasonal=None, seasonal_periods=None)
    model_fit = model.fit()

    # Forecast future values
    forecast = model_fit.forecast(steps=steps)
    return forecast

# Step 5: Apply Exponential Smoothing model to each educational area and forecast
exp_forecasts = {}
for area in time_series_data.columns:
    exp_forecasts[area] = exp_smoothing_forecast(time_series_data[area])

# Convert the forecasts to a DataFrame
exp_forecast_df = pd.DataFrame(exp_forecasts)

# Step 6: Plot the Forecasts along with Historical Data
plt.figure(figsize=(14, 10))

for i, area in enumerate(time_series_data.columns, 1):
    plt.subplot(2, 2, i)
    plt.plot(time_series_data[area], label='Historical')
    plt.plot(range(len(time_series_data), len(time_series_data) + len(exp_forecast_df)), exp_forecast_df[area], label='Exponential Smoothin
    plt.title(f'{area} - Exponential Smoothing Forecast')
    plt.xlabel('Age in Months')
    plt.ylabel('Score')
    plt.legend()

plt.tight_layout()
plt.show()
```

```
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: An unsupported index was provided and will
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:836: ValueWarning: No supported index is available. Predictio
  return get_prediction_index(
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:836: FutureWarning: No supported index is available. In the n
  return get_prediction_index(
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: An unsupported index was provided and will
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:836: ValueWarning: No supported index is available. Predictio
  return get_prediction_index(
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:836: FutureWarning: No supported index is available. In the n
  return get_prediction_index(
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: An unsupported index was provided and will
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:836: ValueWarning: No supported index is available. Predictio
  return get_prediction_index(
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:836: FutureWarning: No supported index is available. In the n
  return get_prediction_index(
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:473: ValueWarning: An unsupported index was provided and will
  self._init_dates(dates, freq)
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:836: ValueWarning: No supported index is available. Predictio
  return get_prediction_index(
/usr/local/lib/python3.10/dist-packages/statsmodels/tsa/base/tsa_model.py:836: FutureWarning: No supported index is available. In the n
  return get_prediction_index(
```



Interpretation and Actionable Insights: