

✓ Imports and Initialize API Key

```
!pip install google-generativeai PILLOW
```

```

⇒ Requirement already satisfied: google-generativeai in /usr/local/lib/python3.11/
Requirement already satisfied: PILLOW in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: google-ai-generativelanguage==0.6.15 in /usr/loca
Requirement already satisfied: google-api-core in /usr/local/lib/python3.11/dist
Requirement already satisfied: google-api-python-client in /usr/local/lib/python
Requirement already satisfied: google-auth>=2.15.0 in /usr/local/lib/python3.11/
Requirement already satisfied: protobuf in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: pydantic in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: tqdm in /usr/local/lib/python3.11/dist-packages (
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.11/di
Requirement already satisfied: proto-plus<2.0.0dev,>=1.22.3 in /usr/local/lib/py
Requirement already satisfied: googleapis-common-protos<2.0.dev0,>=1.56.2 in /us
Requirement already satisfied: requests<3.0.0.dev0,>=2.18.0 in /usr/local/lib/py
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.1
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: httplib2<1.dev0,>=0.19.0 in /usr/local/lib/python
Requirement already satisfied: google-auth-httplib2<1.0.0,>=0.2.0 in /usr/local/
Requirement already satisfied: uritemplate<5,>=3.0.1 in /usr/local/lib/python3.1
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.
Requirement already satisfied: pydantic-core==2.27.2 in /usr/local/lib/python3.1
Requirement already satisfied: grpcio<2.0dev,>=1.33.2 in /usr/local/lib/python3.
Requirement already satisfied: grpcio-status<2.0.dev0,>=1.33.2 in /usr/local/lib
Requirement already satisfied: pyparsing!=3.0.0,!3.0.1,!3.0.2,!3.0.3,<4,>=2.4
Requirement already satisfied: pyasn1<0.7.0,>=0.4.6 in /usr/local/lib/python3.11
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/d
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.11/d

```

```
!pip install -U kaleido
```

```

⇒ Collecting kaleido
  Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl.metadata (15 kB)
  Downloading kaleido-0.2.1-py2.py3-none-manylinux1_x86_64.whl (79.9 MB)
    ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 79.9/79.9 MB 10.4 MB/s eta 0:00:00
  Installing collected packages: kaleido
  Successfully installed kaleido-0.2.1

```

```
pip install PyMuPDF python-pptx pandas openpyxl
```

```

⇒ Collecting PyMuPDF
  Downloading pymupdf-1.25.3-cp39-abi3-manylinux2014_x86_64.manylinux_2_17_x86_6
Collecting python-pptx

```

```

Downloading python_pptx-1.0.2-py3-none-any.whl.metadata (2.5 kB)
Requirement already satisfied: pandas in /usr/local/lib/python3.11/dist-packages
Requirement already satisfied: openpyxl in /usr/local/lib/python3.11/dist-packag
Requirement already satisfied: Pillow>=3.3.2 in /usr/local/lib/python3.11/dist-p
Collecting XlsxWriter>=0.5.7 (from python-pptx)
  Downloading XlsxWriter-3.2.2-py3-none-any.whl.metadata (2.8 kB)
Requirement already satisfied: lxml>=3.1.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: typing-extensions>=4.9.0 in /usr/local/lib/python
Requirement already satisfied: numpy>=1.23.2 in /usr/local/lib/python3.11/dist-p
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-pa
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-
Requirement already satisfied: et-xmlfile in /usr/local/lib/python3.11/dist-pack
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packag
Downloading pymupdf-1.25.3-cp39-abi3-manylinux2014_x86_64.manylinux_2_17_x86_64.
  20.0/20.0 MB 47.4 MB/s eta 0:00:00
Downloading python_pptx-1.0.2-py3-none-any.whl (472 kB)
  472.8/472.8 kB 28.4 MB/s eta 0:00:00
Downloading XlsxWriter-3.2.2-py3-none-any.whl (165 kB)
  165.1/165.1 kB 12.7 MB/s eta 0:00:00
Installing collected packages: XlsxWriter, PyMuPDF, python-pptx
Successfully installed PyMuPDF-1.25.3 XlsxWriter-3.2.2 python-pptx-1.0.2

```

```
!pip install python-docx
```



```

Collecting python-docx
  Downloading python_docx-1.1.2-py3-none-any.whl.metadata (2.0 kB)
Requirement already satisfied: lxml>=3.1.0 in /usr/local/lib/python3.11/dist-pac
Requirement already satisfied: typing-extensions>=4.9.0 in /usr/local/lib/python
Downloading python_docx-1.1.2-py3-none-any.whl (244 kB)
  244.3/244.3 kB 14.4 MB/s eta 0:00:00
Installing collected packages: python-docx
Successfully installed python-docx-1.1.2

```

```

import os
import fitz # PyMuPDF for PDFs
import pandas as pd
from pptx import Presentation

import plotly.graph_objects as go
from IPython.display import display
from PIL import Image
from docx import Document


import pandas as pd
from google.colab import drive
drive.mount('/content/drive')
import plotly.graph_objects as go
import matplotlib.pyplot as plt
import PIL.Image

```

```
PIL.Image.init()
```

```
import numpy as np
import seaborn as sns
from IPython.display import display
from google.colab import drive
import os
```

```
# Enable inline plotting for Jupyter Notebook
%matplotlib inline
```

 Mounted at /content/drive

Start coding or [generate](#) with AI.


```
import requests
import base64
import json
```

```
import google.generativeai as genai
```

```
# Set your API key
genai.configure(api_key="AIzaSyDbg_4DNQmAB7ZpcRdzU0q1cb_rY9TEfaI")
```

```
# Your Gemini API key (replace with your actual key)
#API_KEY = "AIzaSyDbg_4DNQmAB7ZpcRdzU0q1cb_rY9TEfaI"
```

```
!ls "/content/drive/My Drive/"
```

 'Colab Notebooks' 'demo sample.csv'
Correlation_Matrix.ipynb Insight_Generation_Consolidation.ipynb
corr.png Insight_Generation.ipynb
Datasets 'Insight Overview: Questions and Sample Responses.gsh'

```
## Correlation Functions
DATASET_PATH = '/content/drive/MyDrive/Datasets'
%ls $DATASET_PATH
```

 IAR_ELA.csv IAR_MATH.csv

```
## Correlation Functions
DATASET_PATH = '/content/drive/MyDrive/Datasets'
%ls $DATASET_PATH
```

```
# Function to load dataset from a file located in Google Drive
def load_and_merge_datasets(file1='IAR_ELA.csv', file2='IAR_MATH.csv', folder_path='
    ....
```

Load two datasets from Google Drive and merge them on 'Student_ID'.

```
"""
```

```
try:
```

```
    # Load the first dataset
```

```
    file_path1 = os.path.join(folder_path, file1)
```

```
    if file_path1.endswith('.csv'):
```

```
        df1 = pd.read_csv(file_path1)
```

```
    elif file_path1.endswith(('xls', 'xlsx')):
```

```
        df1 = pd.read_excel(file_path1)
```

```
    else:
```

```
        raise ValueError(f"Unsupported file format for {file1}.")
```

```
    # Load the second dataset
```

```
    file_path2 = os.path.join(folder_path, file2)
```

```
    if file_path2.endswith('.csv'):
```

```
        df2 = pd.read_csv(file_path2)
```

```
    elif file_path2.endswith(('xls', 'xlsx')):
```

```
        df2 = pd.read_excel(file_path2)
```

```
    else:
```

```
        raise ValueError(f"Unsupported file format for {file2}.")
```

```
    # Merge the datasets on 'Student_ID'
```

```
    merged_df = pd.merge(df1, df2, on=['Student_ID', 'Year'], how='inner')
```

```
    print("Datasets loaded and merged successfully.")
```

```
    print(merged_df.columns)
```

```
    return merged_df
```

```
except Exception as e:
```

```
    print(f"Error: {e}")
```

```
    return None
```

```
def basic_info(df):
```

```
    """Display basic information about the dataset."""
```

```
    print("Dataset Overview:")
```

```
    display(df.head())
```

```
    print("\nShape of dataset:", df.shape)
```

```
    print("\nColumn Information:")
```

```
    display(df.info())
```

```
    print("\nMissing values:")
```

```
    display(df.isnull().sum())
```

```
    print("\nDescriptive Statistics:")
```

```
    display(df.describe())
```

```
def handle_missing_values(df, strategy='mean'):
```

```
    """Handle missing values using a specified strategy (mean, median, mode, drop)."""
```

```
    df_cleaned = df.copy()
```

```
    for col in df_cleaned.select_dtypes(include=[np.number]).columns:
```

```
        if df_cleaned[col].isnull().sum() > 0:
```

```
            if strategy == 'mean':
```

```

        df_cleaned[col].fillna(df_cleaned[col].mean(), inplace=True)
    elif strategy == 'median':
        df_cleaned[col].fillna(df_cleaned[col].median(), inplace=True)
    elif strategy == 'mode':
        df_cleaned[col].fillna(df_cleaned[col].mode()[0], inplace=True)
    elif strategy == 'drop':
        df_cleaned.dropna(inplace=True)
    else:
        raise ValueError("Invalid strategy. Choose from 'mean', 'median', 'mode', 'drop'")

```

```

print("Missing values handled successfully.")
return df_cleaned

```

```

def encode_categorical(df, max_categories=10):
    """Encode categorical variables selectively to prevent excessive column expansion"""
    df_encoded = df.copy()

    # Identify categorical columns excluding known identifiers and homeroom
    id_columns = ['Student_Name', 'Student_ID', 'Current_School', 'Tested_School', 'Homeroom']
    categorical_cols = [col for col in df_encoded.select_dtypes(include=['object']) if col not in id_columns]

    # Apply one-hot encoding only to categorical columns with a reasonable number of categories
    for col in categorical_cols:
        if df_encoded[col].nunique() <= max_categories:
            df_encoded = pd.get_dummies(df_encoded, columns=[col], drop_first=True)
        else:
            df_encoded.drop(columns=[col], inplace=True) # Drop high-cardinality categorical variables

    print("Categorical variables encoded successfully, excluding identifiers, homeroom")
    return df_encoded

```

```

def rename_merged_columns(df):
    """Rename duplicate columns with _x and _y suffixes to English and Math respectively"""
    df = df.rename(columns=lambda x: x.replace('_', '_English').replace('_', '_Math'))
    return df

```

```

def correlation_matrix2(df, threshold=0.5):
    """Generate and display a readable correlation matrix with strong correlations only"""
    corr_matrix = df.corr().dropna(how='all').dropna(axis=1, how='all')
    strong_corrs = corr_matrix[(corr_matrix >= threshold) | (corr_matrix <= -threshold)]
    strong_corrs = strong_corrs.dropna(how='all').dropna(axis=1, how='all')

    if not strong_corrs.empty:
        plt.figure(figsize=(24, 21)) # Increased figure size for better readability
        sns.heatmap(strong_corrs, annot=True, cmap='coolwarm', fmt='.2f', linewidths=1,
                    cbar_kws={'label': 'Correlation Coefficient'})
        plt.xticks(rotation=90, ha='right', fontsize=9)
        plt.yticks(rotation=0, fontsize=9)
        plt.title("Filtered Correlation Matrix (Strong Correlations Only)", fontsize=12)
        plt.tight_layout()
        plt.show()

```

```


else:
    print("No strong correlations found above the threshold.")

return strong_corrs

def full_analysis(file_name, folder_path=DATASET_PATH):
    """Run the full analysis pipeline on a dataset in Google Drive."""
    df = load_and_merge_datasets('IAR_ELA.csv', 'IAR_MATH.csv', '/content/gdrive/MyD
if df is not None:
    df = rename_merged_columns(df)
    basic_info(df)
    df = handle_missing_values(df, strategy='mean')
    print(df.columns)
    df = encode_categorical(df)
    print(df.columns)
    strong_corrs = correlation_matrix2(df)
    return strong_corrs
else:
    print("Analysis could not be performed due to data loading issues.")

strong_corrs = full_analysis('your_dataset.csv')

```

 ls: cannot access '/content/drive/MyDrive/Datasets': No such file or directory
 Error: name 'os' is not defined
 Analysis could not be performed due to data loading issues.

✓ Read saved image and generate insights

use mtss documents saved in drive

```

# Define folder path
folder_path = "/content/drive/My Drive/Datasets/MTSS Documentation"

# Function to extract text from PDF
def extract_text_from_pdf(pdf_path):
    text = ""
    try:
        doc = fitz.open(pdf_path)
        for page in doc:
            text += page.get_text("text") + "\n"
    except Exception as e:
        text = f"Error reading PDF: {e}"
    return text

# Function to extract text from PowerPoint
def extract_text_from_pptx(pptx_path):
    text = ""

```

```

try:
    prs = Presentation(pptx_path)
    for slide in prs.slides:
        for shape in slide.shapes:
            if hasattr(shape, "text"):
                text += shape.text + "\n"
except Exception as e:
    text = f"Error reading PPTX: {e}"
return text

# Function to extract text from Excel
def extract_text_from_excel(excel_path):
    text = ""
    try:
        df = pd.read_excel(excel_path, sheet_name=None) # Read all sheets
        for sheet_name, sheet_data in df.items():
            text += f"\n### Sheet: {sheet_name} ###\n"
            text += sheet_data.to_string(index=False) + "\n"
    except Exception as e:
        text = f"Error reading Excel: {e}"
    return text

# Iterate through folder and extract file contents
file_contents = []
for file in os.listdir(folder_path):
    file_path = os.path.join(folder_path, file)

    if file.endswith(".pdf"):
        file_contents.append(f"### {file} ###\n" + extract_text_from_pdf(file_path))
    elif file.endswith(".pptx"):
        file_contents.append(f"### {file} ###\n" + extract_text_from_pptx(file_path))
    elif file.endswith(".xlsx") or file.endswith(".xls"):
        file_contents.append(f"### {file} ###\n" + extract_text_from_excel(file_path))

# Combine all extracted content
context = "\n\n".join(file_contents)

```

Use user stories

```
user_stories="/content/drive/My Drive/Datasets/User Stories"
```

```
def read_docx(file_path):
    doc = Document(file_path)
    text = "\n".join([para.text for para in doc.paragraphs])
    return text

#file_path = uploaded_filename if uploaded_filename else "/content/drive/My Drive/yo
user_stories_path="/content/drive/My Drive/Datasets/User Stories.docx"
user_stories_content = read_docx(user_stories_path)
```

```
prompt=f'''
```

```
Objective:
```

```
As an EdTech company focused on improving our learning platform for principals, teac
```

```
Key Areas of Focus:
```

- Identify significant patterns, correlations, or outliers.
- Highlight performance differences among various student groups.
- Assess how trends impact different groups and identify areas for improvement.
- **Base all observations strictly on the graph, without suggesting additional data
- **All recommendations must describe an immediate, direct action that can be implem
- **Provide recommendations based on the observed data in the graph, ensuring they a
- {context} (Only generate recommendations based on this defined MTSS context).**
- Recommendations may address school-wide strategies for equity, inclusivity, and de
- **Strictly prohibit any recommendation that involves reviewing past strategies, re
- **All recommendations must provide a tangible, implementable strategy that can be
- **DO NOT suggest reviewing, re-examining, or analyzing past programs, instructiona
- Ensure that MTSS recommendations are solely tied to the trends, disparities, and i
- Only suggest direct, actionable steps (e.g., "Implement peer tutoring sessions for
- Avoid any recommendations that require further data review, validation, program re
- Ensure that all recommendations can be executed without needing additional analysi
- Avoid suggesting that users reference additional reports, assessments, or school-w
- Avoid comparing insights to industry benchmarks, best practices, or general educat
- All recommendations must be directly supported by an observed trend in the graph.
- Ensure that recommendations focus on specific actions educators and administrators
- Wherever applicable, observations should be expressed in percentages rather than r
- Enhance mtss recommendations by also considering pain points and needs of users in

```
Output Format:
```

```
Your response should include:
```

- A concise blurb summarizing the key insights observed in the graph.
- A prioritized bullet-point list highlighting key observations in order of importan
- Only include observations explicitly supported by the graph.
- Suggestions on what other issues could be related, look at the correlation matrix
- Every recommendation must explicitly reference the supporting data trend from the
- Actionable recommendations that are specifically tailored to the insights derived
 - School-wide strategies to support all students.
 - Targeted interventions for students at moderate risk based on the observed data
 - Intensive supports for students requiring individualized assistance, as identifi

```
D0s:
```

- Focus on clear, data-driven insights.
- Every recommendation must be explicitly tied to a visible data trend in the graph.
- Ensure every recommendation is a direct action that can be implemented immediately
- Provide actionable steps that can be implemented without any additional research or
- Provide concise, actionable observations with justifications. Bullet points should
- **Ensure all recommendations are explicitly linked to the data trends shown in the
- Structure insights in a way that is easy for educators and administrators to under
- **Keep your response within 300 words for clarity.**
- **Keep each observation or recommendation within 30 words.**
- **Limit to max 4 observations and 3 recommendations.**
- Ensure some observations include a quantifiable metric if possible (e.g., "Student
- Ensure that percentages are used instead of raw numbers whenever applicable (e.g.,

DON'Ts:

- Do not provide recommendations for further data collection, further exploration, or
- Do not provide recommendations that require data review, assessment analysis, or v
- Do not suggest that users analyze past performance trends, review existing initiat
- **Do not include terms such as "review strategies," "reflect on past performance,"
- Do not generate recommendations unless they are directly supported by a trend visi
- Do not introduce hypothetical scenarios or potential factors that are not explicit
- Do not make assumptions without data to support them.
- Do not include unnecessary details that do not contribute to actionable insights.
- Do not reference external documents when providing MTSS recommendations. Ensure al
- Do not refer to MTSS tiers (Tier 1, Tier 2, Tier 3) when describing strategies. In
- Do not suggest professional development or training unless the data in the graph e
- DO NOT use vague terms like "school-wide strategies" or "targeted interventions."

Example Response Format:

Blurb: "The data reveals a 20% drop in student engagement in online courses compared

Key Observations (from the given graph):

- Online students show lower participation rates (-20%).
- Performance gaps are wider in low-income school districts.
- Engagement declines over time, dropping by 10 percentage points in the second half

Suggestions (From the correlation here: {strong_corrs})

- Test Grades (Math) is correlated with Test Grades (English), ensuring a student do
- Participation is correlated and linked to better grades
- Homerooms that do well in one subject are not guaranteed to do well in every subje

MTSS-Aligned Recommendations (specific to this graph):

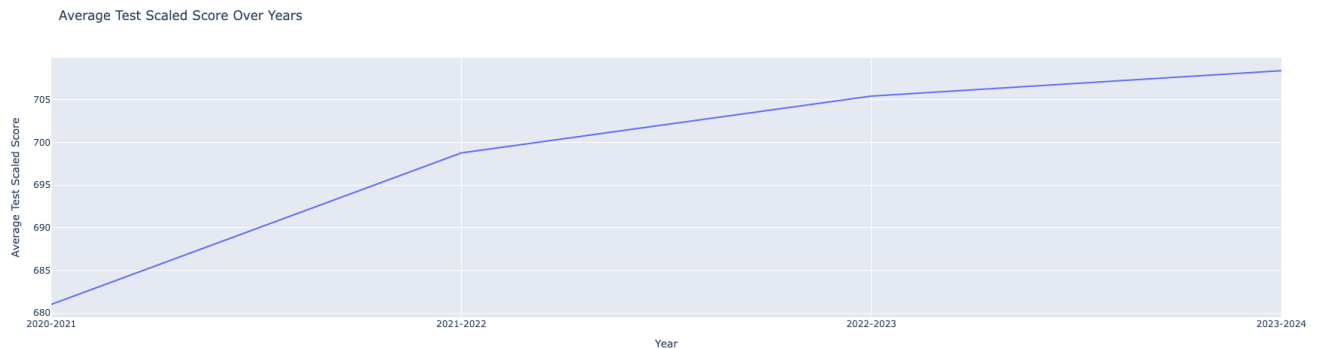
- Increase interactive learning opportunities, such as live Q&A sessions and peer di
- Provide targeted small-group interventions for students in low-income districts, a
- Offer one-on-one mentoring or personalized learning plans for students showing the

```
# Load Gemini 1.5 Pro model
model = genai.GenerativeModel("gemini-1.5-pro")
```

```
chart_folder = "/content/drive/My Drive/Datasets/images"
```

Double-click (or enter) to edit

```
chart_title='Average_Test_Scaled_Score' #example1, example2, example3
chart_path=os.path.join(chart_folder, f"{chart_title}.png")
image = Image.open(chart_path)
display(image)
```



```
# Test 1
```

```
# Generate insights
```

```
response = model.generate_content([prompt, image])
```

```
# Print response
print(response.text)
```

⇒ Blurb: Average scaled test scores show an upward trend from 2020–2021 (681) to 2

Key Observations:

- * Average test scores have increased each year.
- * Most significant improvement occurred between 2020–2021 and 2021–2022 (698), a
- * Growth has continued but has plateaued between 2022–2023 (705) and 2023–2024 (

Suggestions:

- Continue to monitor student performance over time.
- Ensure universal access to the core curriculum.
- Consider the needs of students at risk and those needing additional challenge

MTSS–Aligned Recommendations:

- Continue providing current school-wide supports for all students to maintain u
- Provide small group interventions focused on targeted skills for students whos
- Provide more intensive, individualized support for students whose scores indic

Test 2

```
# Generate insights
response = model.generate_content([prompt, image])
```

```
# Print response
print(response.text)
```

⇒ Blurb: Average test scaled scores show a steady increase from SY20–21 (681) to S

Key Observations:

- * Average test scaled score increased from 681 in SY20–21 to 698 in SY21–22 (17
- * Average test scaled scores show consistent growth, increasing by 10 points fro

Suggestions:

- The performance is linked to increased attendance
- If teachers continue their method of teaching, this may become a continuous tr
- Students with disabilities, and English learners, did significantly worse than

MTSS–Aligned Recommendations:

- * Continue current instructional strategies and supports, as demonstrated by the
- * Identify and share best practices contributing to the score increase.
- * Examine factors that helped to promote student achievement (attendance, leader

Try the prompt on more images

```
chart_title='annual_grade_by_grade' #example1, example2, example3
chart_path=os.path.join(chart_folder, f"{chart_title}.png")
image = Image.open(chart_path)
display(image)
```



```
# Generate insights
response = model.generate_content([prompt, image])
```

```
# Print response
print(response.text)
```



Blurb: Grade 3 students demonstrate the highest percentage count (37%) for the a

Key Observations:

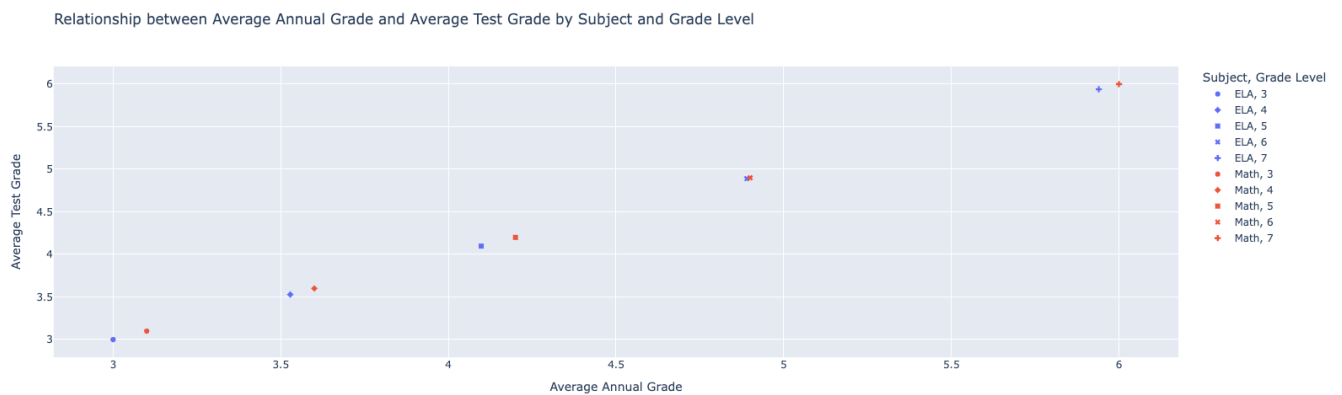
- * Grade 3 has the largest student count at 37% of all annual grades.

- * Grade 5 student count is greater than Grade 4 student count at 32% and 29% of

MTSS-Aligned Recommendations:

- * Implement peer tutoring or mentoring programs, pairing high-performing Grade 3
- * Provide targeted small group interventions focused on specific areas of need,
- * Offer individualized support plans and increased one-on-one time for strugglin

```
chart_title='relationship_grade' #example1, example2, example3
chart_path=os.path.join(chart_folder, f"{chart_title}.png")
image = Image.open(chart_path)
display(image)
```



```
# Generate insights
response = model.generate_content([prompt, image])
```

```
# Print response
print(response.text)
```

⇒ Blurb: Student performance on annual tests shows a strong positive correlation w

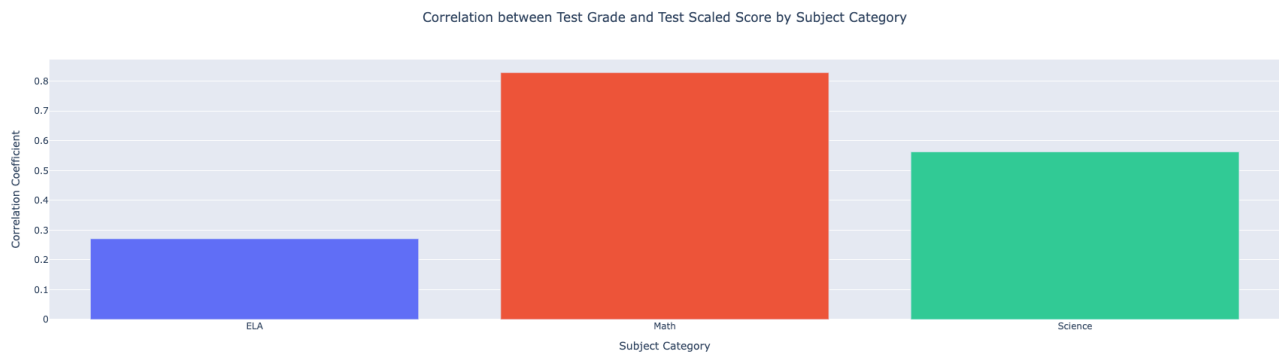
Key Observations:

- * There is a strong positive relationship between average annual grades and aver
- * Math and ELA show similar correlations with average annual grades.
- * The correlation appears consistent across the grade levels observed (3rd–7th g

MTSS–Aligned Recommendations:

- * Implement strategies to improve annual grades for all students. The graph ind
- * Examine current grading policies and practices with staff. For example, estab
- * Design a system to identify students at risk based on current grades. For exa

```
chart_title='correlation_subject' #example1, example2, example3
chart_path=os.path.join(chart_folder, f"{chart_title}.png")
image = Image.open(chart_path)
display(image)
```



```
# Generate insights
response = model.generate_content([prompt, image])
```

```
# Print response  
print(response.text)
```

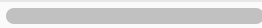
⇒ Blurb: Math exhibits the strongest correlation between test grade and scaled sco

Key Observations:

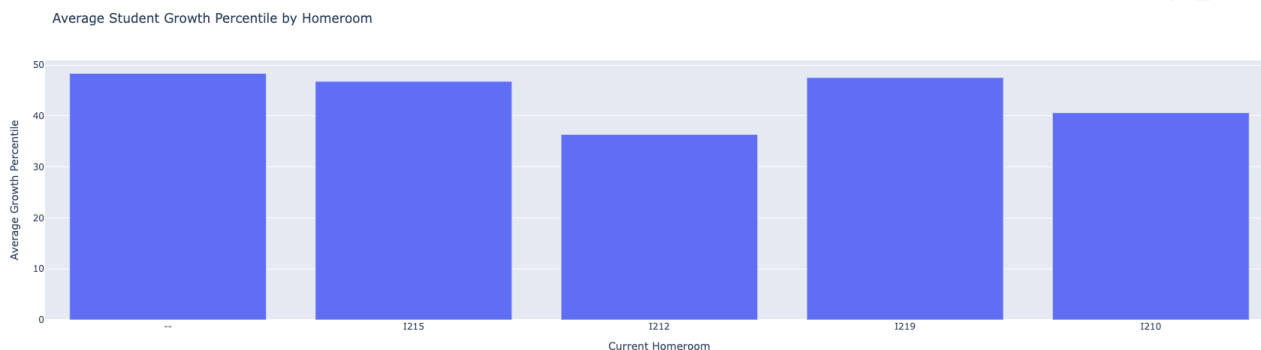
- * Math has the highest correlation coefficient (0.81), significantly higher than
- * Science shows a moderate correlation (0.55), positioned between Math and ELA.
- * ELA demonstrates the lowest correlation (0.25), considerably weaker than Math

MTSS-Aligned Recommendations:

- * Prioritize targeted interventions for ELA, given its low correlation (0.25) be
- * Implement peer tutoring or small-group discussions in Science, given its moder
- * Enhance math instruction by integrating real-world applications, problem-solvi



```
chart_title='homeroom_growth_percentile' #example1, example2, example3  
chart_path=os.path.join(chart_folder, f'{chart_title}.png')  
image = Image.open(chart_path)  
display(image)
```



```
# Generate insights
response = model.generate_content([prompt, image])
```

```
# Print response
print(response.text)
```

Blurb: Homeroom 1219 demonstrates the highest average student growth percentile,

Key Observations:

- * Homeroom 1219 shows the highest average student growth (47%).
- * Homeroom 1215 has the second highest average growth (46%).
- * Homeroom 1212 has the lowest average student growth (36%).
- * Homeroom 1210 has an average student growth of 40%.

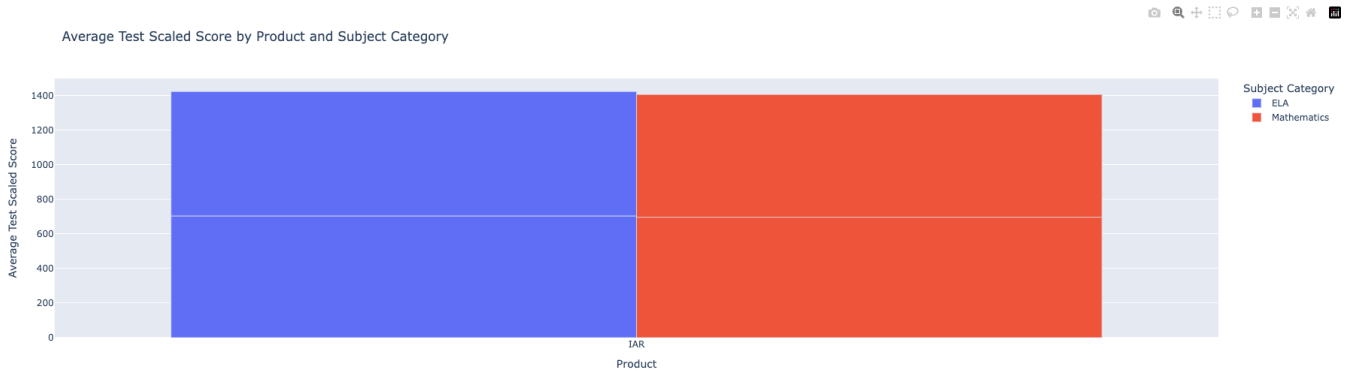
MTSS-Aligned Recommendations:

- * Implement peer tutoring or cross-age mentoring, pairing high-growth students f
- * Provide small-group enrichment activities or project-based learning opportunit
- * Offer individualized support, such as one-on-one tutoring or personalized lear

```

chart_title='Product Average Test Scaled Score Subject Category' #example1, example2
chart_path=os.path.join(chart_folder, f"{chart_title}.png")
image = Image.open(chart_path)
display(image)

```



```

# Generate insights
response = model.generate_content([prompt,image])

```

```

# Print response
print(response.text)

```



Blurb: ELA and Mathematics show similar average scaled scores on the IAR.

Key Observations:

- * Average scaled scores for ELA and Math are approximately equal.
- * No other data is available.

MTSS-Aligned Recommendations:

No recommendations can be made based on this graph alone. Additional data is ne

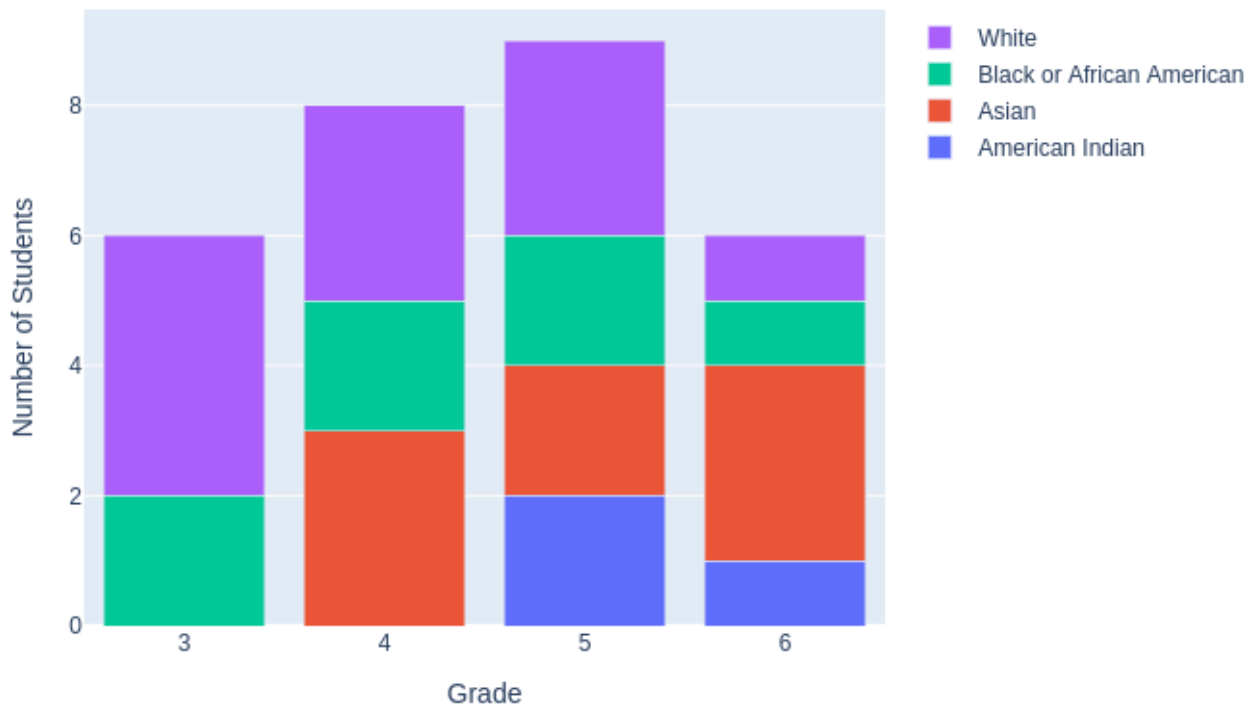
```

chart_title='Stacked Bar Chart of Demographics by Grade' #example1, example2, example3
chart_path=os.path.join(chart_folder, f'{chart_title}.png')
image = Image.open(chart_path)
display(image)

```



Stacked Bar Chart of Demographics by Grade



```

# Generate insights
response = model.generate_content([prompt, image])

```

```

# Print response
print(response.text)

```



****Blurb:**** Demographics across grades 3–6 show consistent representation across

****Key Observations:****

- * White students comprise the largest demographic group (35–45%).
- * Black or African American students represent the second largest group (15–25%).
- * Asian students comprise 15–20% of each grade.
- * American Indian student representation is lowest among all groups (8–12%).

****MTSS-Aligned Recommendations:****

- * Establish school-wide equity initiatives to analyze disparities in demograph
- * Implement diversity and inclusion training for staff to enhance cultural awa

```
chart_title='Subject_Category Grade_Level_Current Test_Primary_Result_Code'
chart_path=os.path.join(chart_folder, f'{chart_title}.png')
image = Image.open(chart_path)
display(image)
```

