Problem Set - 3 CASE STUDY

In [1]:

```
pip install pandas numpy matplotlib seaborn skimpy
```

```
Requirement already satisfied: pandas in c:\users\panda\appdata\roaming\python
\python39\site-packages (2.0.3)
Requirement already satisfied: numpy in c:\users\panda\anaconda3\lib\site-pack
ages (1.26.4)
Requirement already satisfied: matplotlib in c:\users\panda\appdata\roaming\py
thon\python39\site-packages (3.7.3)
Requirement already satisfied: seaborn in c:\users\panda\appdata\roaming\pytho
n\python39\site-packages (0.12.2)
Requirement already satisfied: skimpy in c:\users\panda\anaconda3\lib\site-pac
kages (0.0.15)
Requirement already satisfied: pytz>=2020.1 in c:\users\panda\anaconda3\lib\si
te-packages (from pandas) (2021.3)
Requirement already satisfied: tzdata>=2022.1 in c:\users\panda\anaconda3\lib
\site-packages (from pandas) (2023.3)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\panda\anacon
da3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\users\panda\anaconda3\l
ib\site-packages (from matplotlib) (1.3.2)
Requirement already satisfied: cycler>=0.10 in c:\users\panda\anaconda3\lib\si
te-packages (from matplotlib) (0.11.0)
Requirement already satisfied: pyparsing>=2.3.1 in c:\users\panda\anaconda3\li
b\site-packages (from matplotlib) (3.0.4)
Requirement already satisfied: fonttools>=4.22.0 in c:\users\panda\anaconda3\l
ib\site-packages (from matplotlib) (4.25.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\users\panda\anaconda3\li
b\site-packages (from matplotlib) (1.1.1)
Requirement already satisfied: importlib-resources>=3.2.0 in c:\users\panda\an
aconda3\lib\site-packages (from matplotlib) (6.1.0)
Requirement already satisfied: pillow>=6.2.0 in c:\users\panda\anaconda3\lib\s
ite-packages (from matplotlib) (10.4.0)
Requirement already satisfied: packaging>=20.0 in c:\users\panda\anaconda3\lib
\site-packages (from matplotlib) (23.2)
Requirement already satisfied: ipykernel<7.0.0,>=6.7.0 in c:\users\panda\anaco
nda3\lib\site-packages (from skimpy) (6.9.1)
Requirement already satisfied: polars<0.21,>=0.19 in c:\users\panda\anaconda3
\lib\site-packages (from skimpy) (0.20.31)
Requirement already satisfied: pyarrow<17,>=13 in c:\users\panda\anaconda3\lib
\site-packages (from skimpy) (14.0.2)
Requirement already satisfied: rich<14.0,>=10.9 in c:\users\panda\anaconda3\li
b\site-packages (from skimpy) (13.6.0)
Requirement already satisfied: typeguard==4.2.1 in c:\users\panda\anaconda3\li
b\site-packages (from skimpy) (4.2.1)
Requirement already satisfied: click<9.0.0,>=8.1.6 in c:\users\panda\anaconda3
\lib\site-packages (from skimpy) (8.1.8)
Requirement already satisfied: Pygments<3.0.0,>=2.10.0 in c:\users\panda\anaco
nda3\lib\site-packages (from skimpy) (2.16.1)
Requirement already satisfied: importlib-metadata>=3.6 in c:\users\panda\appda
ta\roaming\python\python39\site-packages (from typeguard==4.2.1->skimpy) (6.8.
0)
Requirement already satisfied: typing-extensions>=4.10.0 in c:\users\panda\ana
conda3\lib\site-packages (from typeguard==4.2.1->skimpy) (4.12.2)
Requirement already satisfied: colorama in c:\users\panda\anaconda3\lib\site-p
ackages (from click<9.0.0,>=8.1.6->skimpy) (0.4.4)
Requirement already satisfied: zipp>=0.5 in c:\users\panda\anaconda3\lib\site-
packages (from importlib-metadata>=3.6->typeguard==4.2.1->skimpy) (3.7.0)
```

```
Requirement already satisfied: debugpy<2.0,>=1.0.0 in c:\users\panda\anaconda3
\lib\site-packages (from ipykernel<7.0.0,>=6.7.0->skimpy) (1.5.1)
Requirement already satisfied: traitlets<6.0,>=5.1.0 in c:\users\panda\anacond
a3\lib\site-packages (from ipykernel<7.0.0,>=6.7.0->skimpy) (5.1.1)
Requirement already satisfied: tornado<7.0,>=4.2 in c:\users\panda\anaconda3\l
ib\site-packages (from ipykernel<7.0.0,>=6.7.0->skimpy) (6.1)
Requirement already satisfied: matplotlib-inline<0.2.0,>=0.1.0 in c:\users\pan
da\anaconda3\lib\site-packages (from ipykernel<7.0.0,>=6.7.0->skimpy) (0.1.2)
Requirement already satisfied: ipython>=7.23.1 in c:\users\panda\anaconda3\lib
\site-packages (from ipykernel<7.0.0,>=6.7.0->skimpy) (7.31.1)
Requirement already satisfied: nest-asyncio in c:\users\panda\anaconda3\lib\si
te-packages (from ipykernel<7.0.0,>=6.7.0->skimpy) (1.5.5)
Requirement already satisfied: jupyter-client<8.0 in c:\users\panda\anaconda3
\lib\site-packages (from ipykernel<7.0.0,>=6.7.0->skimpy) (7.2.2)
Requirement already satisfied: setuptools>=18.5 in c:\users\panda\anaconda3\li
b\site-packages (from ipython>=7.23.1->ipykernel<7.0.0,>=6.7.0->skimpy) (61.2.
0)
Requirement already satisfied: jedi>=0.16 in c:\users\panda\anaconda3\lib\site
-packages (from ipython>=7.23.1->ipykernel<7.0.0,>=6.7.0->skimpy) (0.18.1)
Requirement already satisfied: pickleshare in c:\users\panda\anaconda3\lib\sit
e-packages (from ipython>=7.23.1->ipykernel<7.0.0,>=6.7.0->skimpy) (0.7.5)
Requirement already satisfied: backcall in c:\users\panda\anaconda3\lib\site-p
ackages (from ipython>=7.23.1->ipykernel<7.0.0,>=6.7.0->skimpy) (0.2.0)
Requirement already satisfied: prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0 in
c:\users\panda\anaconda3\lib\site-packages (from ipython>=7.23.1->ipykernel<7.
0.0,>=6.7.0->skimpy) (3.0.20)
Requirement already satisfied: decorator in c:\users\panda\anaconda3\lib\site-
packages (from ipython>=7.23.1->ipykernel<7.0.0,>=6.7.0->skimpy) (5.1.1)
Requirement already satisfied: parso<0.9.0,>=0.8.0 in c:\users\panda\anaconda3
\lib\site-packages (from jedi>=0.16->ipython>=7.23.1->ipykernel<7.0.0,>=6.7.0-
>skimpy) (0.8.3)
Requirement already satisfied: entrypoints in c:\users\panda\anaconda3\lib\sit
e-packages (from jupyter-client<8.0->ipykernel<7.0.0,>=6.7.0->skimpy) (0.4)
Requirement already satisfied: pyzmq>=22.3 in c:\users\panda\anaconda3\lib\sit
e-packages (from jupyter-client<8.0->ipykernel<7.0.0,>=6.7.0->skimpy) (22.3.0)
Requirement already satisfied: jupyter-core>=4.9.2 in c:\users\panda\anaconda3
\lib\site-packages (from jupyter-client<8.0->ipykernel<7.0.0,>=6.7.0->skimpy)
(4.9.2)
Requirement already satisfied: pywin32>=1.0 in c:\users\panda\anaconda3\lib\si
te-packages (from jupyter-core>=4.9.2->jupyter-client<8.0->ipykernel<7.0.0,>=
6.7.0->skimpy) (302)
Requirement already satisfied: wcwidth in c:\users\panda\anaconda3\lib\site-pa
ckages (from prompt-toolkit!=3.0.0,!=3.0.1,<3.1.0,>=2.0.0->ipython>=7.23.1->ip
ykernel<7.0.0,>=6.7.0->skimpy) (0.2.5)
Requirement already satisfied: six>=1.5 in c:\users\panda\anaconda3\lib\site-p
ackages (from python-dateutil>=2.8.2->pandas) (1.16.0)
Requirement already satisfied: markdown-it-py>=2.2.0 in c:\users\panda\anacond
a3\lib\site-packages (from rich<14.0,>=10.9->skimpy) (3.0.0)
Requirement already satisfied: mdurl~=0.1 in c:\users\panda\anaconda3\lib\site
-packages (from markdown-it-py>=2.2.0->rich<14.0,>=10.9->skimpy) (0.1.2)
Note: you may need to restart the kernel to use updated packages.
```

```
WARNING: Ignoring invalid distribution -rotobuf (c:\users\panda\anaconda3\lib
\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\panda\anaconda3\lib
\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\panda\anaconda3\lib
\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\panda\anaconda3\lib
\site-packages)
WARNING: Ignoring invalid distribution -rotobuf (c:\users\panda\anaconda3\lib
\site-packages)
```

```
WARNING: Ignoring invalid distribution -rotobuf (c:\users\panda\anaconda3\lib
\site-packages)
```

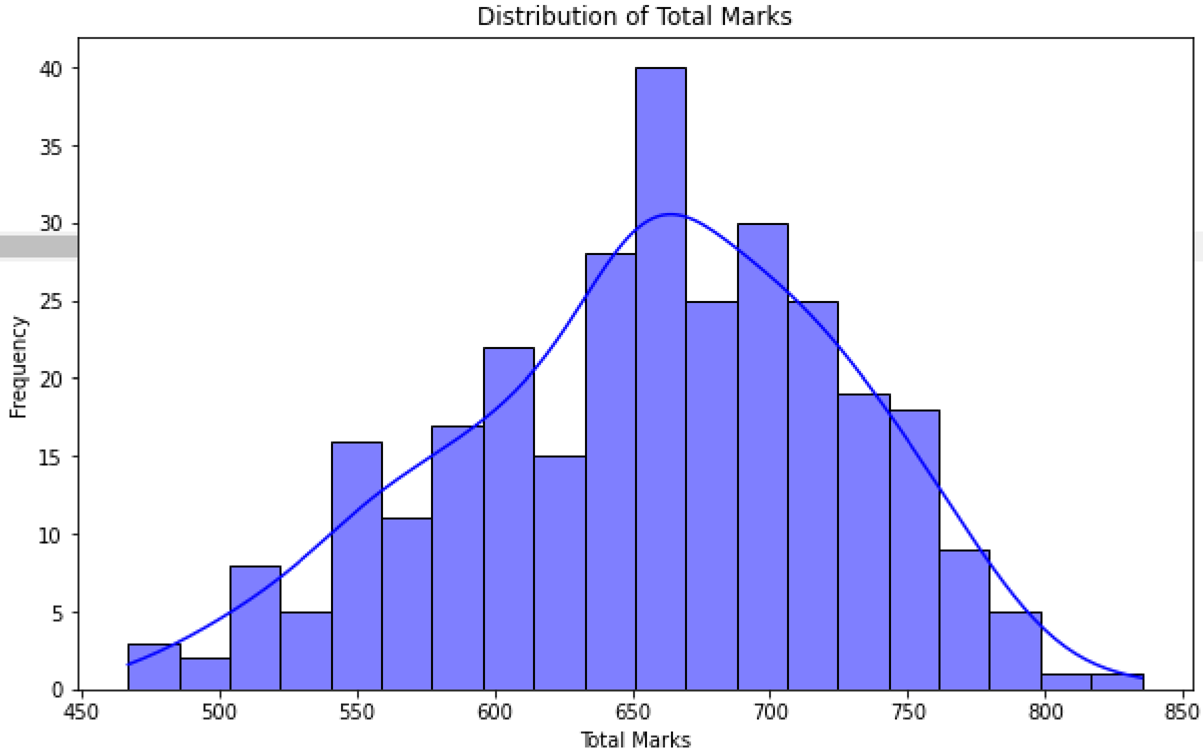
In [2]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from skimpy import skim

# File paths for the uploaded Excel files
file_2019 = "C:\\Users\\Panda\\Desktop\\DataAnalystEvaluation\\BangaloreSchool
file_2020 = "C:\\Users\\Panda\\Desktop\\DataAnalystEvaluation\\BangaloreSchool
file_2021 = "C:\\Users\\Panda\\Desktop\\DataAnalystEvaluation\\BangaloreSchool
```

In [3]:
```python
# Load dataframes
data_2019 = pd.ExcelFile(file_2019)
data_2020 = pd.ExcelFile(file_2020)
data_2021 = pd.ExcelFile(file_2021)

# Load all sheets for each year into a dictionary for consolidation
def load_school_data(data, year):
    school_data = {}
    for sheet in data.sheet_names:
        school_data[sheet] = data.parse(sheet)
        school_data[sheet]['School'] = sheet  # Added a column to identify the
        school_data[sheet]['Year'] = year  # Add a column to identify the year
    return pd.concat(school_data.values(), ignore_index=True)

# Consolidate data for all years
data_combined = pd.concat([
    load_school_data(data_2019, 2019),
    load_school_data(data_2020, 2020),
    load_school_data(data_2021, 2021)
], ignore_index=True)

# Data overview using skimpy
skim(data_combined)
```

```
┌──────────────────────────────────────────────────── skimpy summary ─────────
│          Data Summary                 Data Types
│ ┏━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━┓ ┏━━━━━━━━━━━━━┳━━━━━━━┓
│ ┃ dataframe          ┃ Values ┃ ┃ Column Type ┃ Count ┃
│ ┡━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━┩ ┡━━━━━━━━━━━━━╇━━━━━━━┩
│ │ Number of rows     │ 300    │ │ int32       │ 13    │
│ │ Number of columns  │ 15     │ │ string      │ 2     │
│ └────────────────────┴────────┘ └─────────────┴───────┘
│                                                      number
│ ┏━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━┳━━━━━━━┳━━━━━━━━┳━━━━━━━━━┳━━━━━━━┳━━━━
│ ┃ column_name             ┃ NA  ┃ NA %  ┃ mean   ┃ sd      ┃ p0    ┃ p25
│ ┡━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━╇━━━━━━━╇━━━━━━━━╇━━━━━━━━━╇━━━━━━━╇━━━━
│ │ Student Roll            │   0 │     0 │   3010 │    1417 │  1001 │   2
│ │ Hindi                   │   0 │     0 │  60.41 │   19.95 │    20 │
│ │ English                 │   0 │     0 │  60.78 │   19.31 │    20 │  46
│ │ Mathematics             │   0 │     0 │  62.47 │   20.39 │    20 │
│ │ Physics                 │   0 │     0 │  59.49 │   22.82 │    13 │
│ │ Chemistry               │   0 │     0 │  59.79 │    19.9 │    19 │  45
│ │ Biology                 │   0 │     0 │  59.48 │   20.51 │    18 │
│ │ History                 │   0 │     0 │  58.96 │   20.36 │    15 │
│ │ Geography               │   0 │     0 │  59.45 │   21.16 │    17 │
│ │ Civics                  │   0 │     0 │  56.44 │   21.18 │    10 │
│ │ Computer Science        │   0 │     0 │  60.31 │   21.86 │    11 │
│ │ Physical Education      │   0 │     0 │  59.14 │   20.34 │    11 │
│ Year                   │   0 │     0 │   2020 │  0.8179 │   2019 │  2
                                                        string
┃ column_name                ┃ NA      ┃ NA %         ┃ words per row
│ Student Name               │       0 │            0 │
│ School                     │       0 │            0 │
└──────────────────────────────────────────────────────── End ──────
```

In [4]:

```python
# Calculate cumulative marks
data_combined['Total Marks'] = data_combined[
    ['Hindi', 'English', 'Mathematics', 'Physics', 'Chemistry',
     'Biology', 'History', 'Geography', 'Civics',
     'Computer Science', 'Physical Education']
].sum(axis=1)

# Visualize distribution of total marks
plt.figure(figsize=(10, 6))
sns.histplot(data_combined['Total Marks'], bins=20, kde=True, color='blue')
plt.title('Distribution of Total Marks')
plt.xlabel('Total Marks')
plt.ylabel('Frequency')
plt.show()
```



1. Reward the top performer (student) of each school based on cumulative marks scored in last three years for all the subjects

In [5]:

```python
# Find the top performer of each school based on cumulative marks over three y
top_performers = data_combined.groupby(['School', 'Student Roll'])['Total Mark
top_performers = top_performers.loc[top_performers.groupby('School')['Total Ma

# Add student names to the result for clarity
top_performers = pd.merge(top_performers, data_combined[['Student Roll', 'Stud

# Display the top performers
top_performers
```

Out[5]:

	School	Student Roll	Total Marks	Student Name
0	Birla HS	4010	2209	Hashmukh Patel
3	DPS	3018	2043	Jivan Rao
6	International	5001	2166	Swetashi Aiyyar
9	St. Joseph	2007	2056	Agriya Marandi
12	Vidya Mandir	1020	2320	Nisha Saxena

2. Rank each student within their own school based on their total marks scored in the year 2020 and compare the marks of Rank 10 for each school by arranging them in descending order

In [6]:

```python
def rank_students_2020(data_combined):
    # Filter data for the year 2020
    data_2020 = data_combined[data_combined['Year'] == 2020].copy()

    # Calculate total marks for 2020
    data_2020['Total Marks 2020'] = data_2020[
        ['Hindi', 'English', 'Mathematics', 'Physics', 'Chemistry',
         'Biology', 'History', 'Geography', 'Civics',
         'Computer Science', 'Physical Education']
    ].sum(axis=1)

    # Rank students within their schools
    data_2020['Rank 2020'] = data_2020.groupby('School')['Total Marks 2020'].r

    # Extract Rank 10 students for each school and sort by marks in descending
    rank_10_students = data_2020[data_2020['Rank 2020'] == 10].sort_values(by=
    return rank_10_students[['School', 'Student Name', 'Total Marks 2020']]

# Call each function and store results
rank_10_results = rank_students_2020(data_combined)
# Display results
rank_10_results
```

Out[6]:

	School	Student Name	Total Marks 2020
111	Vidya Mandir	Ganesh Sekhar	705
176	Birla HS	Derek Pinto	673
191	International	Jashwant Bhide	660
139	St. Joseph	Deep Dasgupta	649

	School	Student Name	Total Marks 2020
153	DPS	Michel Dsuza	614

3. Find out students with the highest improvement for each subject from 2019-21 combining all the schools together

In [7]:

```
import pandas as pd

def highest_improvement(data_combined):
    # Define the subject columns
    subject_columns = [
        'Hindi', 'English', 'Mathematics', 'Physics', 'Chemistry',
        'Biology', 'History', 'Geography', 'Civics',
        'Computer Science', 'Physical Education'
    ]

    # Pivot the data to organize marks by year for each student
    improvement = data_combined.pivot_table(
        index=['Student Roll', 'Student Name'],
        columns='Year',
        values=subject_columns,
        aggfunc='mean'
    )

    # Ensure all required years are present
    for year in [2019, 2020, 2021]:
        if year not in improvement.columns.levels[1]:
            raise ValueError(f"Data for {year} is missing.")

    # Calculate improvement from 2019 to 2021
    improvement_2019_to_2021 = improvement.xs(2021, level=1, axis=1) - improve

    # Find the student with the highest improvement for each subject
    results = []
    for subject in subject_columns:
        if subject in improvement_2019_to_2021.columns:
            student_idx = improvement_2019_to_2021[subject].idxmax()  # Get in
            improvement_value = improvement_2019_to_2021[subject].max()  # Get
            roll, name = student_idx  # Decompose multi-index
            results.append({
                'Subject': subject,
                'Student Roll': roll,
                'Student Name': name,
                'Improvement (2019-2021)': improvement_value
            })

    # Convert results to DataFrame
    return pd.DataFrame(results)

# Assuming data_combined is a properly formatted DataFrame
# Example usage:
# data_combined = pd.read_csv('path_to_your_data.csv')
highest_improvement_results = highest_improvement(data_combined)
```

```
# Display results
highest_improvement_results
```

Out[7]:

	Subject	Student Roll	Student Name	Improvement (2019-2021)
0	Hindi	1011	Sonal Tripathi	71
1	English	3005	Besent Kumar	59
2	Mathematics	3008	Manyathi Shetty	67
3	Physics	1013	Praddep Meena	63
4	Chemistry	3020	Manshukh Bhayani	65
5	Biology	4019	Nitin Deewan	54
6	History	2004	Rahul Bansal	51
7	Geography	1010	Subhajeet Dutta	61
8	Civics	1018	Sanjana Venkatramana	65
9	Computer Science	1017	Rashmi Desai	82
10	Physical Education	1008	Anamika Kumari	58

4. Identify best school for Arts, Science and Commerce streams based on marks scored by students in respective subjects for those streams in last three years

In [8]:

```
def best_school_streamwise(data_combined):
    # Define subjects for each stream
    streams = {
        'Arts': ['History', 'Geography', 'Civics'],
        'Science': ['Mathematics', 'Physics', 'Chemistry', 'Biology'],
        'Commerce': ['Mathematics', 'Economics', 'Accounting'],
    }

    # Calculate total marks per stream for each school
    stream_scores = {}
    for stream, subjects in streams.items():
        available_subjects = [subject for subject in subjects if subject in da
        stream_scores[stream] = data_combined.groupby('School')[available_subj

    return stream_scores

best_streamwise = best_school_streamwise(data_combined)
best_streamwise
```

Out[8]:

```
{'Arts': 'Birla HS', 'Science': 'International', 'Commerce': 'Vidya Mandir'}
```

5. Calculate for each school how many students were in each category based on the avg. marks obtained each year

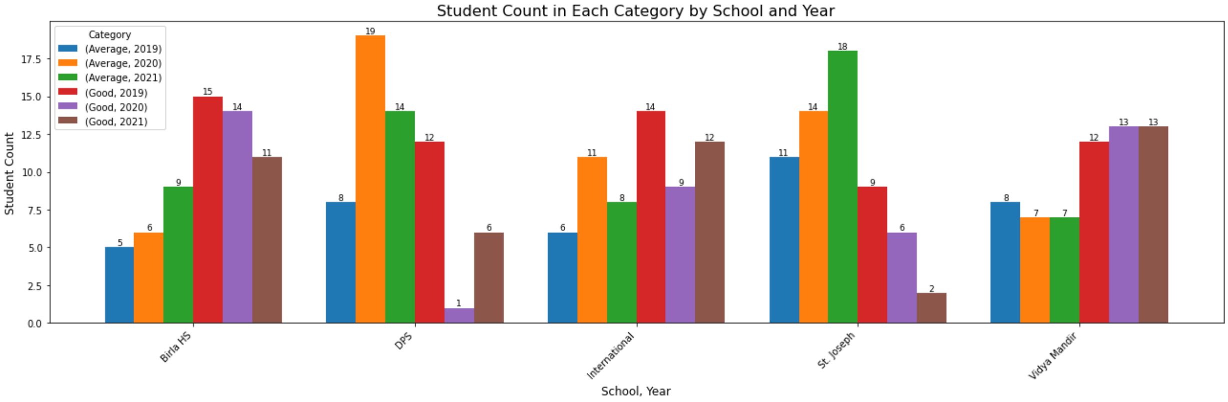
```
In [9]: # Category Analysis Visualization
        def category_analysis(data_combined):
            def categorize_mark(mark):
                if mark <= 20:
                    return 'Very Poor'
                elif mark <= 40:
                    return 'Poor'
                elif mark <= 60:
                    return 'Average'
                elif mark <= 80:
                    return 'Good'
                else:
                    return 'Very Good'

            subject_columns = ['Hindi', 'English', 'Mathematics', 'Physics', 'Chemistr
                               'Biology', 'History', 'Geography', 'Civics',
                               'Computer Science', 'Physical Education']
            data_combined['Average Marks'] = data_combined[subject_columns].mean(axis=
            data_combined['Category'] = data_combined['Average Marks'].apply(categoriz

            category_counts = data_combined.groupby(['School', 'Year', 'Category']).si
            return category_counts

        category_counts = category_analysis(data_combined)
        category_counts
```

Out[9]:

	Category	Average	Good
School	Year		
Birla HS	2019	5	15
	2020	6	14
	2021	9	11
DPS	2019	8	12
	2020	19	1
	2021	14	6
International	2019	6	14
	2020	11	9
	2021	8	12
St. Joseph	2019	11	9
	2020	14	6
	2021	18	2
Vidya Mandir	2019	8	12
	2020	7	13
	2021	7	13

```
In [10]: # Assuming `category_counts` is the DataFrame from your code
         # Unstacking the DataFrame to make it easier to plot
         category_counts_unstacked = category_counts.unstack()

         # Plotting the bar chart
         ax = category_counts_unstacked.plot(kind='bar', figsize=(18, 6), width=0.8)
```

```
# Customizing the chart
plt.title('Student Count in Each Category by School and Year', fontsize=16)
plt.xlabel('School, Year', fontsize=12)
plt.ylabel('Student Count', fontsize=12)
plt.xticks(rotation=45, ha='right')
plt.legend(title='Category', fontsize=10)
plt.tight_layout()

# Adding labels to each bar
for container in ax.containers:
    # Add a label above each bar
    ax.bar_label(container, fmt='%d', label_type='edge', fontsize=9)

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



6. Which is the best school for each year 2019, 2020 and 2021 based on highest no. of students in Good and Very Good category

In [11]:

```
def best_school_per_year(data_combined):
    # Filter Good and Very Good categories
    data_combined['Category'] = data_combined['Average Marks'].apply(
        lambda x: 'Good' if 60 < x <= 80 else ('Very Good' if x > 80 else 'Oth
    )

    # Filter for only Good and Very Good categories
    category_counts = data_combined[data_combined['Category'].isin(['Good', 'V

    # Group by Year and School and calculate the counts
    grouped = category_counts.groupby(['Year', 'School']).size()

    # Find the school with the highest count for each year
    best_schools = grouped.groupby(level=0).idxmax()

    return best_schools
best_school_yearly = best_school_per_year(data_combined)
best_school_yearly
```

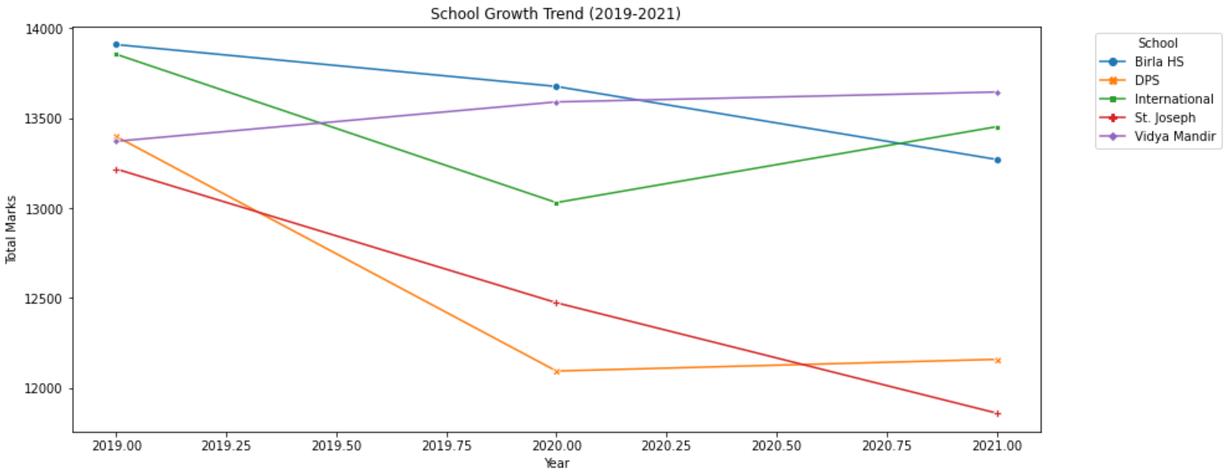
Out[11]:

```
Year
2019         (2019, Birla HS)
2020         (2020, Birla HS)
2021    (2021, Vidya Mandir)
dtype: object
```

7. Which is the fastest-growing School in Bangalore (Overall and Streamwise)?

In [12]:

```
# Growth Analysis Visualization
def fastest_growing_school(data_combined):
    school_growth = data_combined.groupby(['School', 'Year'])['Total Marks'].s
    school_growth['Growth'] = school_growth[2021] - school_growth[2019]
    return school_growth

school_growth = fastest_growing_school(data_combined)

# Line plot for growth
school_growth_plot = school_growth.drop(columns='Growth').T
plt.figure(figsize=(14, 6))
sns.lineplot(data=school_growth_plot, markers=True, dashes=False)
plt.title('School Growth Trend (2019-2021)')
plt.xlabel('Year')
plt.ylabel('Total Marks')
plt.legend(title='School', bbox_to_anchor=(1.05, 1), loc='upper left')
plt.show()
```



In [13]:

```
def fastest_growing_school(data_combined):
    # Calculate total marks per school per year
    school_growth = data_combined.groupby(['School', 'Year'])['Total Marks'].s
    school_growth['Growth'] = school_growth[2021] - school_growth[2019]

    fastest_growing_overall = school_growth['Growth'].idxmax()
    return fastest_growing_overall
fastest_growing = fastest_growing_school(data_combined)
fastest_growing
```

Out[13]:

```
'Vidya Mandir'
```