```
PEP install faker
from faker import Faker



import random
import pandas as pd
# Initialize Faker
fake = Faker()

# Define subjects
subjects = ["Electronics", "Mathematics", "DSA", "Programming", "Database", "Data Science"]

# Generate student data
students = []
for student_id in range(1, 10001):  # 10,000 students
    student_name = fake.name()  # Generate a fake name
    department = random.choice(["CSE", "ECE", "IT", "EEE"])  # Choose a random department
    year = random.randint(1, 4)  # Choose a random year (1st - 4th)

    # Generate marks for each subject (0 to 100)
    marks = {subject: random.randint(30, 100) for subject in subjects}

    # Create student dictionary
    student = {
        "Student_ID": student_id,
        "Name": student_name,
        "Department": department,
        "Year": year,
        **marks  # Unpacking marks dictionary
    }
    students.append(student)
```

```
# Convert to Pandas DataFrame
df = pd.DataFrame(students)

# Save to CSV
df.to_csv("students_data.csv", index=False)

print("Student data generated successfully!")



import seaborn as sns

import matplotlib.pyplot as plt
plt.figure(figsize=(10, 6))
sns.heatmap(df[["Electronics", "Mathematics", "DSA", "Programming", "Database", "Data Science"]].corr(), annot=True, cmap="coolwarm", fmt=".2f")
plt.title("Subject Marks Correlation Heatmap")
plt.show()



from multiprocessing import Pool

# Define subjects
subjects = ["Electronics", "Mathematics", "DSA", "Programming", "Database", "Data Science"]

def mapper(row):
    """
    Simulates the Mapper function by extracting subject marks
    """
    student_id = row["Student_ID"]
    output = []
```

```
    for subject in subjects:
        output.append((subject, row[subject]))  # Key-Value pairs (subject, marks)

    return output

# Apply the mapper function to all rows
with Pool() as pool:
    mapped_data = pool.map(mapper, [row for _, row in df.iterrows()])

# Flatten the mapped output
mapped_data = [item for sublist in mapped_data for item in sublist]

# Display sample output from the mapper
mapped_data[:10]  # First 10 key-value pairs

# Define passing marks (e.g., 40)
import pandas as pd
passing_marks = 40



# Calculate pass and fail counts per subject
pass_fail_stats = pd.DataFrame({
    "Pass Count": (df[subjects] >= passing_marks).sum(),
    "Fail Count": (df[subjects] < passing_marks).sum()
})

# Calculate Pass Percentage
pass_fail_stats["Pass Percentage"] = (pass_fail_stats["Pass Count"] / len(df)) * 100

print(pass_fail_stats)
```

```
# List of subjects
subjects = ["Electronics", "Mathematics", "DSA", "Programming", "Database", "Data Science"]

# Compute the average marks per student
df["Average Marks"] = df[subjects].mean(axis=1)

# Display the first few rows to verify
print(df.head())

from sklearn.linear_model import LinearRegression
# Define features (subject marks) and target (overall average)
X = df[["Electronics", "Mathematics", "DSA", "Programming", "Database", "Data Science"]]
y = df["Average Marks"]

# Fit the linear regression model
model = LinearRegression()
model.fit(X, y)

# Get regression coefficients
coefficients = pd.DataFrame({
    "Subject": X.columns,
    "Coefficient": model.coef_
})
print(coefficients)

# Top 10 Students
top_students = df.nlargest(10, "Average Marks")
# Bottom 10 Students
bottom_students = df.nsmallest(10, "Average Marks")
```

```
print("Top 10 Students:\n", top_students[["Student_ID", "Average Marks"]])
print("\nBottom 10 Students:\n", bottom_students[["Student_ID", "Average Marks"]])

# Convert marks (out of 100) to CGPA (out of 10)
df["CGPA"] = df[subjects].mean(axis=1) / 10

# Display summary statistics
print(df["CGPA"].describe())

# Visualizing CGPA Distribution
plt.figure(figsize=(8, 5))
sns.histplot(df["CGPA"], bins=20, kde=True, color="green")
plt.xlabel("CGPA")
plt.ylabel("Number of Students")
plt.title("CGPA Distribution of Students")
plt.show()
```