```
import pandas as pd

# ===== Sample Student Dataset =====
data_students = {
    "Student_ID": [201, 202, 203, 204, 205],
    "Name": ["Ravi", "Sneha", "Arjun", "Meena", "Kiran"],
    "Math": [92, None, 68, 81, None],
    "Science": [85, 77, None, 90, 72],
    "English": [88, 79, 91, None, 80]
}

df_students = pd.DataFrame(data_students)
print("Original Student Data:")
print(df_students)

# Fill missing values with column averages
df_students.fillna(df_students.mean(numeric_only=True), inplace=True)

print("\nAfter Filling Missing Values:")
print(df_students)

# Filter high achievers (score > 75 in all subjects)
high_achievers = df_students[
    (df_students["Math"] > 75) &
    (df_students["Science"] > 75) &
    (df_students["English"] > 75)
]

print("\n🎯 Students scoring >75 in all subjects:")
print(high_achievers)
```

```
Original Student Data:
   Student_ID   Name  Math  Science  English
0         201   Ravi  92.0     85.0     88.0
1         202  Sneha   NaN     77.0     79.0
2         203  Arjun  68.0      NaN     91.0
3         204  Meena  81.0     90.0      NaN
4         205  Kiran   NaN     72.0     80.0

After Filling Missing Values:
   Student_ID   Name       Math  Science  English
0         201   Ravi  92.000000     85.0     88.0
1         202  Sneha  80.333333     77.0     79.0
2         203  Arjun  68.000000     81.0     91.0
3         204  Meena  81.000000     90.0     84.5
4         205  Kiran  80.333333     72.0     80.0

🎯 Students scoring >75 in all subjects:
   Student_ID   Name       Math  Science  English
0         201   Ravi  92.000000     85.0     88.0
1         202  Sneha  80.333333     77.0     79.0
3         204  Meena  81.000000     90.0     84.5
```

```
# ===== Sample Patient Dataset =====
data_patients = {
    "Patient_ID": [301, 302, 303, 304, 305, 306],
    "Name": ["Anita", "Ramesh", "Geeta", "Suresh", "Latha", "Maya"],
    "Age": [58, None, 65, 45, None, 70],
    "Gender": ["Female", "Male", "Female", "Male", "Female", "Female"],
    "Diagnosis": ["Diabetes", "Flu", "Diabetes", "Cancer", "Diabetes", "Asthma"],
    "Admission_Date": ["2025-08-10", "2025-08-11", "2025-08-12", "2025-08-13", "2025-08-14", "2025-08-15"]
}

df_patients = pd.DataFrame(data_patients)
print("Original Patient Data:")
print(df_patients)

# Replace missing Age values with median
df_patients["Age"].fillna(df_patients["Age"].median(), inplace=True)

print("\nAfter Filling Missing Ages:")
print(df_patients)

# Filter elderly female patients (>50) diagnosed with Diabetes
elderly_female_diabetes = df_patients[
    (df_patients["Gender"].str.lower() == "female") &
```

```
    (df_patients["Age"] > 50) &
    (df_patients["Diagnosis"].str.lower() == "diabetes")
]

print("\n🎯 Elderly Female Patients with Diabetes:")
print(elderly_female_diabetes)
```

```
Original Patient Data:
   Patient_ID    Name   Age  Gender Diagnosis Admission_Date
0         301   Anita  58.0  Female  Diabetes     2025-08-10
1         302  Ramesh   NaN    Male       Flu     2025-08-11
2         303   Geeta  65.0  Female  Diabetes     2025-08-12
3         304  Suresh  45.0    Male    Cancer     2025-08-13
4         305   Latha   NaN  Female  Diabetes     2025-08-14
5         306    Maya  70.0  Female    Asthma     2025-08-15

After Filling Missing Ages:
   Patient_ID    Name   Age  Gender Diagnosis Admission_Date
0         301   Anita  58.0  Female  Diabetes     2025-08-10
1         302  Ramesh  61.5    Male       Flu     2025-08-11
2         303   Geeta  65.0  Female  Diabetes     2025-08-12
3         304  Suresh  45.0    Male    Cancer     2025-08-13
4         305   Latha  61.5  Female  Diabetes     2025-08-14
5         306    Maya  70.0  Female    Asthma     2025-08-15

🎯 Elderly Female Patients with Diabetes:
   Patient_ID   Name   Age  Gender Diagnosis Admission_Date
0         301  Anita  58.0  Female  Diabetes     2025-08-10
2         303  Geeta  65.0  Female  Diabetes     2025-08-12
4         305  Latha  61.5  Female  Diabetes     2025-08-14
/tmp/ipython-input-796431789.py:16: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assign
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting valu

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].me


  df_patients["Age"].fillna(df_patients["Age"].median(), inplace=True)
```