**HEALTH CARE DATAEXPLORATION**

**INTRODUCTION**

This report analyzes a small healthcare dataset comprising patient records with five parameters: PatientID, Age, BloodPressure, SugarLevel, and Weight. The goals of this analysis are as follows:

* **Identify Trends:** Evaluate blood pressure, sugar levels, and weight patterns across patients.
* **Screen for High Risk:** Flag high-risk patients based on thresholds in blood pressure, sugar levels, and weight.
* **Visualize Data:** Produce meaningful graphs to illustrate trends, distributions, and relationships in the data for better insight.

This analysis offers a snapshot for healthcare professionals to monitor patient health trends and potentially focus on patients requiring more careful management.

METHODOLGY

**1. Data Acquisition**

* **Objective:** To load the healthcare dataset stored in a CSV file into a structured format for further analysis.
* **Approach:** We use Python's pandas library, a powerful tool for data manipulation and analysis, to read the dataset.
* **Function Name:** fetch\_by\_patient\_id
* **Parameters:**
  + data: A pandas DataFrame containing the healthcare dataset.
  + patient\_id: The unique identifier corresponding to the patient whose details are needed.
* **Return Value:**
  + The function returns a new DataFrame that contains only those rows where the PatientID matches the provided patient\_id.
* # Input age range to fetch patients
* def fetch\_by\_age(data, min\_age, max\_age):
* return data[(data['Age'] >= min\_age) & (data['Age'] <= max\_age)]
* # Example usage
* age\_filtered\_data = fetch\_by\_age(data, 30, 50)
* print(age\_filtered\_data)

 **Define the Function:**

* Create a function named fetch\_by\_age that takes three arguments:
  + data: the DataFrame containing healthcare records.
  + min\_age: the lower boundary of the age range.
  + max\_age: the upper boundary of the age range.

 **Apply Boolean Indexing:**

* Evaluate the condition data['Age'] >= min\_age which returns a Boolean Series (True when Age is at least min\_age).
* Evaluate the condition data['Age'] <= max\_age which returns a Boolean Series (True when Age is at most max\_age).

 **Combine Conditions:**

* Use the bitwise AND operator (&) to merge the two Boolean Series:
  + This creates a single mask that is True only when both conditions are met.

 **Filter the DataFrame:**

* Apply the combined Boolean mask to the original DataFrame:
  + data[(data['Age'] >= min\_age) & (data['Age'] <= max\_age)]
* Return the filtered DataFrame which includes only the records for patients whose age falls between min\_age and max\_age.
* # Input Blood Pressure range to fetch patients
* def fetch\_by\_blood\_pressure(data, min\_bp, max\_bp):
* return data[(data['BloodPressure'] >= min\_bp) & (data['BloodPressure'] <= max\_bp)]
* # Example usage
* bp\_filtered\_data = fetch\_by\_blood\_pressure(data, 90, 120)
* print(bp\_filtered\_data)

Define a function named fetch\_by\_blood\_pressure that accepts three parameters:

* + data: The DataFrame containing the healthcare records.
  + min\_bp: The minimum blood pressure value.
  + max\_bp: The maximum blood pressure value.

 **Setting Up Boolean Conditions:**

* Create a condition to check if the blood pressure is **greater than or equal to** min\_bp:
  + data['BloodPressure'] >= min\_bp
* Create a condition to check if the blood pressure is **less than or equal to** max\_bp:
  + data['BloodPressure'] <= max\_bp

 **Combining the Conditions:**

* Use the bitwise AND operator (&) to combine the two conditions into a single Boolean mask:
  + (data['BloodPressure'] >= min\_bp) & (data['BloodPressure'] <= max\_bp)

 **Filtering the DataFrame:**

* Apply the combined Boolean mask to the DataFrame to extract only those records that meet both conditions:
  + data[(data['BloodPressure'] >= min\_bp) & (data['BloodPressure'] <= max\_bp)]
* The function returns this filtered DataFrame

.

* import matplotlib.pyplot as plt
* # Line graph for Blood Pressure
* plt.figure(figsize=(8, 5))
* plt.plot(data["PatientID"], data["BloodPressure"], marker='o', label="Blood Pressure")
* plt.title("Blood Pressure Trend Across Patients")
* plt.xlabel("PatientID")
* plt.ylabel("Blood Pressure")
* plt.grid(True)
* plt.legend()
* plt.show()

 **Import the Library:**

* Import matplotlib.pyplot as plt for plotting capabilities.

 **Set Up the Figure:**

* Define the figure size with plt.figure(figsize=(8, 5)) to establish an 8x5 inches plotting area.

 **Plot the Data:**

* Use plt.plot() to create a line graph:
  + **X-axis Data:** data["PatientID"] to represent each patient.
  + **Y-axis Data:** data["BloodPressure"] showing corresponding blood pressure values.
  + **Marker:** Set to 'o' to mark data points.
  + **Label:** "Blood Pressure" for the plot legend.

 **Customize the Graph:**

* **Title:** Add a descriptive title using plt.title("Blood Pressure Trend Across Patients").
* **X-Axis Label:** Use plt.xlabel("PatientID") to label the horizontal axis.
* **Y-Axis Label:** Use plt.ylabel("Blood Pressure") to label the vertical axis.
* **Grid:** Enable grid lines with plt.grid(True) to enhance readability.
* **Legend:** Activate the legend with plt.legend() to show labels.

 **Display the Graph:**

* Use plt.show() to render and display the graph.

SCREEN SHOOT-

