"Data for Heart Disease Analysis"

1. id → Unique identifier for each patient

2. gender → Sex of the patient (male/female).

3. Age → Patient's age in years.

4. age distribution

5. height → Patient's height in centimeters.

6. weight → Patient's weight in kilograms.

7. BMI

8. BMI_Class → BMI level (Normal weight, Obese , Over weight, Under weight).

9. ap_hi → Systolic blood pressure (upper reading).

10. ap_lo → Diastolic blood pressure (lower reading).

11. smoke → Smoking status (smoker or non-smoker).

12. glucose → glucose level (normal , pre diabetic , diabetic).

13. active → Physical activity status (active/non-active).

14. cholesterol → cholesterol level (normal, Elevated , Hypertension Stage 1, Hypertension Stage 2).

15. Alcohol→ Alcohol status (Alcohol or non- Alcohol).

16. Blood pressure → Blood pressure level (1 = normal, 2 =type 1 , 3 = type2).

17- BP_Class → Blood pressure Class (normal, Elevated , Hypertension Stage 1, Hypertension Stage 2).

18. CVD → Cardiovascular disease status (CVD = has disease, NON-CVD = does not have).

"Data Cleaning for Heart Disease Analysis"

-Importing the Library

In this step, I imported the Pandas library, which is essential for data analysis. It allows me to read, clean, and manipulate data efficiently in tabular form.

-Loading the Dataset

In this step, I loaded the dataset from an Excel file into a Pandas DataFrame. This allows me to view and work with the data easily for cleaning and preprocessing tasks.

-Displaying the First Few Rows

Here, I displayed the first few rows of the dataset to get an initial look at the data — including column names, structure, and sample values — before starting the cleaning and preprocessing process.

-Removing the Unnecessary Column

In this step, I removed the first column from the dataset because it wasn't useful for the analysis. This helps keep the data clean and focused only on the relevant features.

-Checking the Updated Data

Here, I displayed the first few rows again to confirm that the column name was successfully changed and that the dataset structure looked correct after the modification.

-**Replacing Numeric Values with Descriptive Labels**

In this step, I replaced numeric codes with meaningful text labels to make the dataset easier to understand. For example, gender values were changed from numbers to "male" and "female," and other categorical variables like cholesterol, glucose level, and activity were also updated with descriptive terms for better readability and interpretation.

-**Converting Age from Days to Years**

Here, I converted the age column from days to years by dividing each value by 365 and changing the result to an integer. This makes the age data more understandable and practical for analysis.

-**Displaying the Age Column**

In this step, I displayed the values of the **age** column to verify that the conversion from days to years was done correctly.

-**Checking Dataset Information**

In this step, I displayed the dataset's general information to review the number of entries, column names, data types, and any missing values. This helps ensure the data is clean and ready for further preprocessing or analysis.

-**Checking for Missing Values**

Here, I checked how many missing (null) values exist in each column. This step helps identify if any data needs to be filled, corrected, or removed before continuing with the analysis.

-**Checking for Duplicate Rows**

In this step, I checked how many duplicate rows exist in the dataset. Detecting duplicates is important to ensure that each record is unique and to maintain data accuracy before analysis.

-**Removing Outliers and Invalid Data**

In this step, I filtered the dataset to keep only realistic values for key columns like **age**, **height**, **weight**, and **blood pressure (ap_hi, ap_lo)**.
This helps remove outliers and incorrect entries, ensuring that the data used for analysis is accurate and reliable.

-**Calculating BMI**

In this step, I created a new column called **BMI (Body Mass Index)** by dividing the person's weight (in kilograms) by the square of their height (in meters).
This metric helps assess whether a person is underweight, healthy, overweight, or obese — which is important for analyzing health patterns.

-**Rounding BMI Values**

Here, I rounded the **BMI** values to **two decimal places** to make the data cleaner and easier to read, ensuring consistent formatting for analysis and visualization.

-**Creating a BMI Classification Column**

In this step, I categorized each person's **BMI** value into classes — Under weight, Normal weight, Over weight, or Obese — based on standard BMI ranges.
This helps make the data more interpretable by grouping continuous BMI values into meaningful health categories.

-**Classifying Blood Pressure Levels**

Here, I created a new column that classifies each person's **blood pressure** based on their systolic (*ap_hi*) and diastolic (*ap_lo*) readings.
Each record is labeled as *Normal*, *Elevated*, *Hypertension Stage 1*, or *Hypertension Stage 2*, following standard medical guidelines.
This makes it easier to analyze how blood pressure relates to other health indicators in the dataset.

-**Creating Age Distribution Groups**

In this step, I divided the **age column** into specific **age ranges** (from 29 to 64) using bins.
Each person is assigned to a labeled group such as *29-34*, *35-39*, *40-44*, etc.
This helps in understanding how different age groups are distributed in the dataset and allows for easier comparison across categories in later analysis.

-**Displaying Age and Age Distribution**

Here, I displayed only the **age** and **age_distribution** columns to verify that the age values were correctly grouped into their respective ranges.
This quick check helps confirm that the binning process worked as expected and that each record falls into the right age category..

-**Final Verification**

At this final stage, I displayed the first 10 rows of the cleaned and enhanced dataset to review the results.
This helped ensure that all transformations — including renaming columns, replacing values, handling outliers, and adding new calculated features like **BMI**, **BMI_Class**, **BP_Class**, and **age_distribution** — were successfully applied and that the data is now well-prepared for analysis.

-**Reordering Columns for Better Organization**

In this step, I rearranged the columns in a more logical and organized order to make the dataset easier to read and analyze.

-**Displaying the Final Data Preview**

In this final step, I displayed the first few rows of the cleaned and well-structured dataset to confirm that all transformations — including renaming, cleaning, feature creation, and reordering — were applied correctly and the data is ready for analysis.

-Exporting the Cleaned Dataset

In the last step, I exported the fully cleaned and processed dataset to a new Excel file named **"heart_data1_cleaned1.xlsx"**.
This makes it easy to share or use the refined data for further analysis, visualization, or reporting.

"discover data with python"

-EDA

As part of the Exploratory Data Analysis , I checked the structure of the dataset using df.info(). This command provides an overview of the dataset, including the number of rows, columns, data types, and non-null values. It helps me quickly identify missing data and understand how each variable is stored.

```
df.info()
```

The output shows the dataset contains 300000 rows and 17 columns. Each column is listed with its data type and the count of non-null.

```
df.describe()
```

To continue the Exploratory Data Analysis, I generated summary statistics for the dataset to understand the distribution of numerical features, The output shows averages, ranges, and quartiles, helping spot anomalies or extreme values.

```
print(df.isna().sum())
```

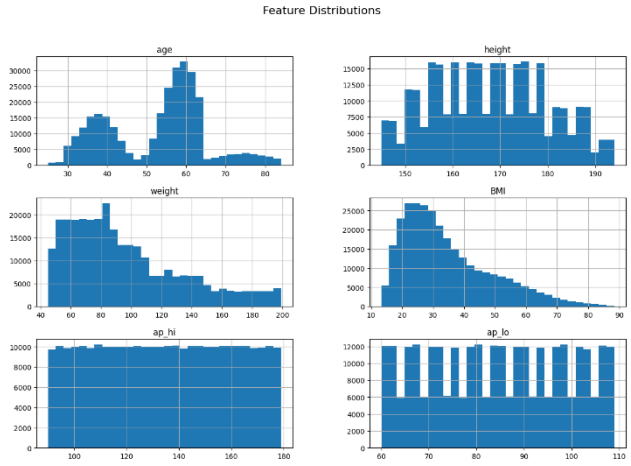
I checked for missing values in each column to ensure data completeness, The output shows how many null values exist per column. This helps identify features that may need cleaning before analysis.

```
df = df.drop(columns=['id'])
```

I removed the id column since it is just an identifier and does not contribute to the analysis.

```
import matplotlib.pyplot as plt
df.hist(figsize=(15, 10), bins=30)
plt.suptitle("Feature Distributions", fontsize=16)
plt.show()
```

I plotted histograms for all numerical features to examine their distributions and detecskewness or outliers.



The histograms show how values are spread across each feature. This helps identify normal vs. skewed distributions and highlights potential outliers that may need handling.

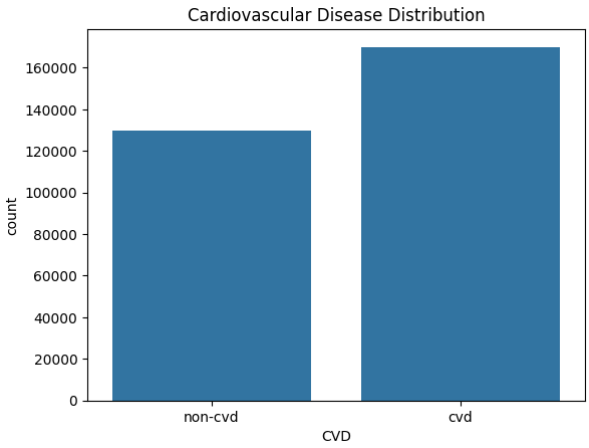
```
df['gender'].value_counts(dropna=False)
```

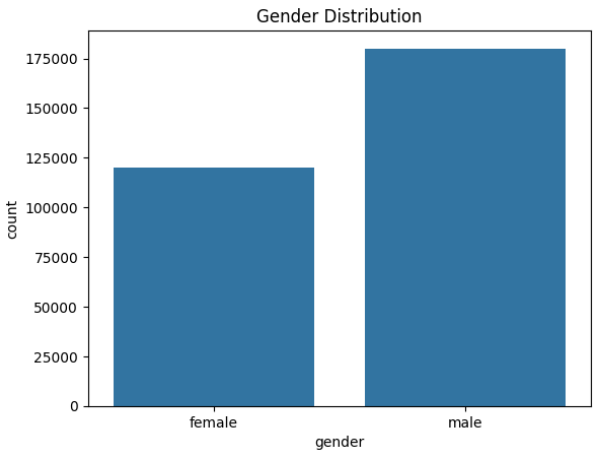
gender
male 180000
female 120000
Name: count, dtype: int64

I checked the distribution of the gender column to see how many records belong to each category, The dataset contains 180,000 male entries and 120,000 female entries, showing a balanced but slightly male-dominant distribution.

```
import seaborn as sns
sns.countplot(x='CVD', data=df)
plt.title("Cardiovascular Disease Distribution")
plt.show()
sns.countplot(x='gender', data=df)
plt.title("Gender Distribution")
```

```
plt.show()
```

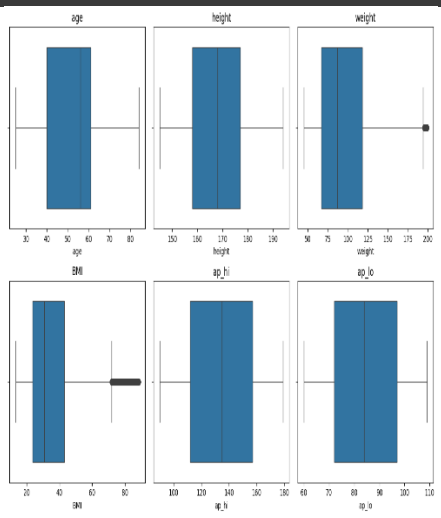




I used count plots to visualize the distribution of cardiovascular disease cases and gender in the dataset, The first plot shows how many individuals are labeled with or without cardiovascular disease. The second plot confirms the gender distribution

Outlier Detection

```
import matplotlib.pyplot as plt
import seaborn as sns
num_cols = ["age", "height", "weight", "BMI", "ap_hi", "ap_lo"]
plt.figure(figsize=(12, 8))
for i, col in enumerate(num_cols, 1):
   plt.subplot(2, 3, i)
   sns.boxplot(x=df[col])
   plt.title(col)
plt.tight_layout()
plt.show()
```



I plotted boxplots for key numerical features to detect outliers and assess value ranges, The boxplots highlight the spread of each variable and reveal potential outliers, especially in BMI values.

```
def remove_outliers_iqr(data, col):

   Q1 = data[col].quantile(0.25)
```

```
  Q3 = data[col].quantile(0.75)
  IQR = Q3 - Q1
  lower = Q1 - 1.5 * IQR
  upper = Q3 + 1.5 * IQR

  mask_outliers = (data[col] < lower) | (data[col] > upper)
  num_removed = mask_outliers.sum()
  cleaned = data[mask_outliers].copy()

  print(f"{num_removed} outliers removed from column '{col}'.")
  return cleaned, num_removed
```

I applied the Interquartile Range (IQR) method to detect and remove outliers from the age, weight, and BMI columns, The function calculates the IQR and removes values outside the acceptable range.

In [ ]:

```
df_clean_age, removed_age = remove_outliers_iqr(df, 'age')
df_clean_weight, removed_weight = remove_outliers_iqr(df, 'weight')
df_clean_bmi, removed_bmi = remove_outliers_iqr(df, 'BMI')

print("Removed counts:", removed_age, removed_weight, removed_bmi)
```

```
0 outliers removed from column 'age'.
3379 outliers removed from column 'weight'.
5605 outliers removed from column 'BMI'.
```
Removed counts: 0 3379 5605

The printed counts show how many outliers were removed from each column, ensuring cleaner data for analysis.

"Data Calculation"

Total Records — Calculates the total number of records in the dataset

Total Records = COUNTROWS(Heart)

Total CVD — Calculates how many people are diagnosed with CVD

Total CVD = CALCULATE([Total Records], Heart[CVD] = "cvd")

Total Non CVD — Calculates how many people do NOT have CVD

Total Non CVD = CALCULATE([Total Records], Heart[CVD] = "non-cvd")

CVD % — Shows the percentage of people who have CVD

CVD % = DIVIDE([Total CVD], [Total Records], 0)

Non CVD % — Shows the percentage of people who do NOT have CVD

```
Non CVD % = DIVIDE([Total Non CVD], [Total Records], 0)
```

---

smoke cvd — Calculates how many smokers have CVD

```
smoke cvd = CALCULATE(COUNT(Heart[CVD]), Heart[smoke] = "smoke", Heart[CVD] = "cvd")
```

---

alco cvd — Calculates how many alcohol users have CVD

```
alco cvd = CALCULATE(COUNT(Heart[CVD]), Heart[alco] = "alco", Heart[CVD] = "cvd")
```

Non active cvd — Calculates how many non-active people have CVD

```
Non active cvd = CALCULATE(COUNT(Heart[CVD]), Heart[active] = "non-active", Heart[CVD] = "cvd")
```

Total_Matched_Records — Calculates the number of records after applying the current filters

```
Total_Matched_Records = CALCULATE( COUNTROWS('Heart'), ALLSELECTED('Heart') )
```

Expected_CVD_Risk_Percentage — Calculates the expected CVD risk percentage based on filtered data

```
Expected_CVD_Risk_Percentage =
VAR CVD_Cases_Filtered = CALCULATE( COUNTROWS('Heart'), 'Heart'[CVD] = "cvd", ALLSELECTED('Heart') )
VAR Total_Population_Filtered = [Total_Matched_Records]
RETURN
 IF( Total_Population_Filtered > 0,     DIVIDE(CVD_Cases_Filtered, Total_Population_Filtered), -1 )
```

Risk_Guidance_Message — Generates a text message explaining the risk category

```
Risk_Guidance_Message =
VAR Risk = [Expected_CVD_Risk_Percentage]
VAR High_Risk_Threshold = 0.63
VAR Medium_Risk_Threshold = 0.30
RETURN
  IF( [Total_Matched_Records] = 0,
    "--- Please make your selection above to see guidance. ---",
```

```
SWITCH(
    TRUE(),
    Risk >= High_Risk_Threshold, "⚠️ HIGH RISK: Cardiology review needed",
    Risk >= Medium_Risk_Threshold, "🔶 MODERATE RISK: Keep an eye on it",
    Risk >= 0, "Low risk: maintain your healthy habits" ) )
```

Gauge_Risk_Value — Returns a clean value for gauge visuals (0 if invalid)

```
Gauge_Risk_Value =
VAR Risk = [Expected_CVD_Risk_Percentage]
RETURN IF( Risk < 0, 0, Risk )
```

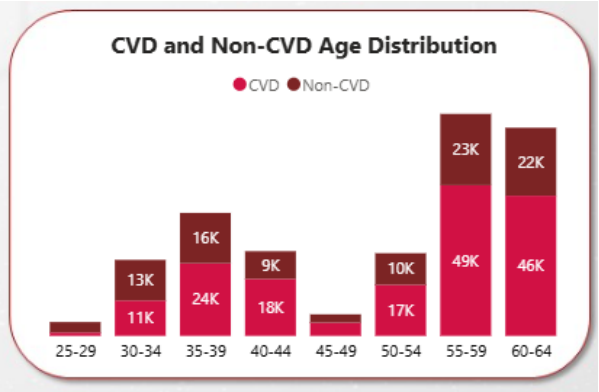
Baseline_CVD_Risk — Calculates the overall CVD risk in the full population (no filters)

```
Baseline_CVD_Risk =
VAR TotalCVD = CALCULATE( COUNTROWS('Heart'), 'Heart'[CVD] = "cvd", ALL('Heart') )
VAR TotalPopulation = CALCULATE( COUNTROWS('Heart'), ALL('Heart') )
RETURN DIVIDE(TotalCVD, TotalPopulation
```

"Data Visualization "

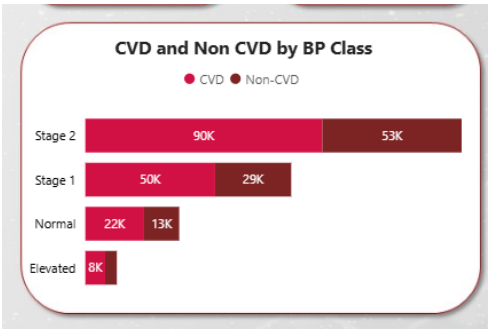
CVD and Non-CVD Age Distribution

Purpose: Shows how CVD and non-CVD cases are distributed across different age groups. Helps identify which age ranges have higher CVD prevalence.
Chart Type: Stacked Column Chart



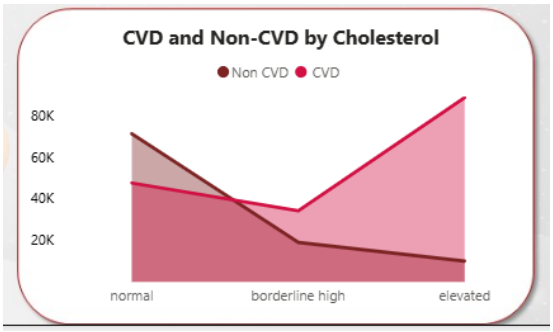
CVD and Non-CVD by BP Class

Purpose: Displays the relationship between blood pressure categories and CVD status. Helps detect which BP classes are associated with higher CVD risk.
Chart Type: Stacked Bar Chart



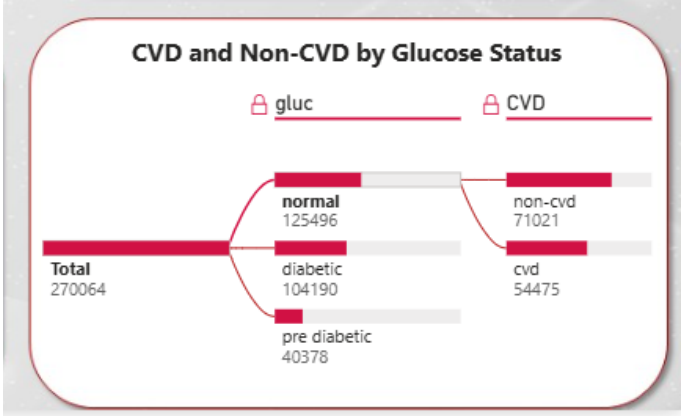
CVD and Non-CVD by Cholesterol

Purpose: Shows how cholesterol levels relate to CVD and non-CVD cases. Helps understand the impact of cholesterol on heart disease risk.
Chart Type: Line + Area Chart



CVD and Non-CVD by Glucose Status

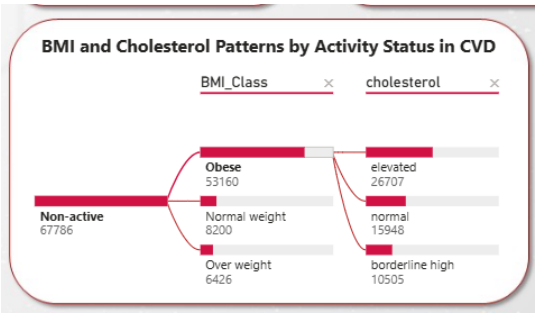
Purpose: Visualizes how glucose level categories (normal, pre-diabetic, diabetic) flow into CVD vs. non-CVD outcomes.
Chart Type: Decomposition Tree



BMI and Cholesterol Patterns by Activity Status in CVD

Purpose:
Shows how non-active CVD patients are distributed across BMI categories (obese, normal, overweight) and how each BMI group maps to different cholesterol levels. Helps identify combined lifestyle–clinical risk patterns.

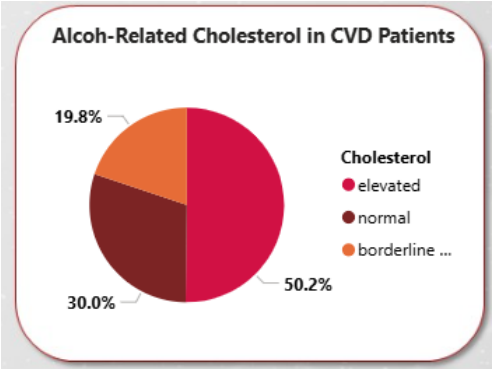
Chart Type: Decomposition Tree



Alcohol-Related Cholesterol in CVD Patients

Purpose:
Visualizes the distribution of cholesterol levels (normal, elevated, borderline) among CVD patients who consume alcohol. Helps identify whether alcohol users show higher cholesterol concentrations.

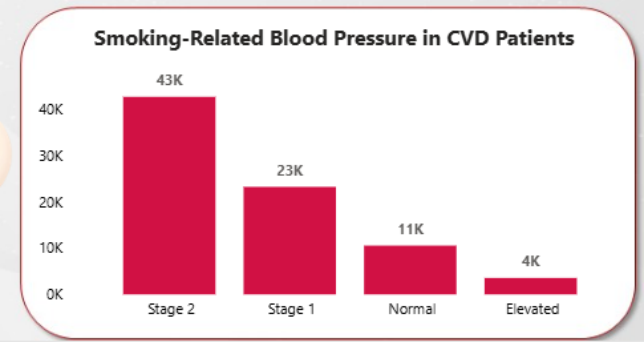
Chart Type:
Pie Chart



Smoking-Related Blood Pressure in CVD Patients

Purpose:
Shows how blood pressure levels (Normal, Stage 1, Stage 2, Elevated) are distributed among smokers with CVD. Helps understand the relationship between smoking intensity and blood pressure severity.

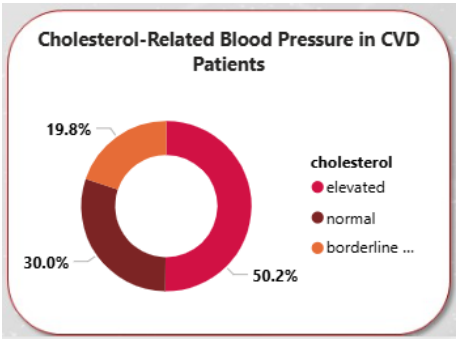
Chart Type:
Column Chart



Cholesterol-Related Blood Pressure in CVD Patients

Purpose:
Shows how cholesterol categories map to blood pressure categories in CVD patients. Helps identify whether certain cholesterol levels are associated with higher BP stages.

Chart Type:
Donut Chart



This page is designed to evaluate the user's potential risk of developing cardiovascular disease (CVD) based on the selected health characteristics.
It functions as an interactive assessment tool where the user selects their health profile through multiple filters, and the system instantly calculates and displays the personalized risk level.

How the Page Works

The user selects their health attributes through several filters, such as:

- Age distribution
- Blood pressure class
- Cholesterol level
- Smoking status
- Alcohol consumption
- Physical activity

Once the selections are made, all visual elements update dynamically to reflect the user's health condition.

Key Visual Components

a. Risk Gauge Indicator

Displays the calculated CVD risk on a scale from 0 to 1, giving a fast visual interpretation of whether the risk is low, moderate, or high.

b. Total Matched Records

Shows the number of individuals in the dataset who share the same health characteristics as the user, helping estimate how common the user's profile is within the population.

c. Expected CVD Risk Percentage

Presents the risk as a percentage, offering a clearer and more intuitive understanding of the calculated risk.

d. Guidance Message

Displays a dynamic recommendation message based on the calculated risk, such as:

- *Low risk – maintain your healthy habits*
- *Moderate risk – consider improving specific lifestyle areas*
- *High risk – medical attention or lifestyle changes highly recommended*
