Tugas Besar IF2220 Probabilitas dan Statistika 2023/2024

Topik : Health

Dibuat oleh:

K01 - Kelompok 24

NIM	Nama
13522071	Bagas Sambega Rosyada
13522091	Raden Francisco Trianto Bratadiningrat

Inisialisasi Dependencies Python dan Pembacaan Data

In [ ]:
```
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np
from scipy import stats
from IPython.display import Markdown, display
import statsmodels
import math

# Read data from CSV
data = pd.read_csv("health.csv", index_col=0)
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 2000 entries, 0 to 1999
Data columns (total 11 columns):
 #   Column                  Non-Null Count  Dtype  
---  ------                  --------------  -----  
 0   Age                     2000 non-null   int64  
 1   Income                  2000 non-null   float64
 2   Gender                  2000 non-null   object 
 3   Education               2000 non-null   object 
 4   Stress_Level            2000 non-null   float64
 5   Exercise_Hours_Per_Week 2000 non-null   float64
 6   Cholesterol_Level       2000 non-null   float64
 7   Weight                  2000 non-null   float64
 8   Height                  2000 non-null   float64
 9   Blood_Pressure          2000 non-null   float64
 10  Health_Status           2000 non-null   object 
dtypes: float64(7), int64(1), object(3)
memory usage: 187.5+ KB
```

Soal General

Nomor 1

Menulis deskripsi statistika (Descriptive Statistics) dari semua kolom pada data. Data yang bersifat numerik dapat diberikan nilai mean, median, modus, standar deviasi, variansi, range, nilai minimum, maksimum, kuartil, IQR, skewness dan kurtosis. Data dalam bentuk string dapat dicari unique values, dan proporsi nya.

Jawab: Menulis Deskripsi Statistika dari semua kolom pada data

Kolom-Kolom dibagi menjadi bentuk data numerik dan string:

- String: Gender, Education, Health_Status
- Numerik: Age, Income, Stress_level, Exercise_Hours_Per_Week, Cholesterol_Level, Weight, Height, Blood_Pressure

Data String

Untuk data string akan dicari:

- Unique values
- Proporsi

In [ ]:
```
# Get columns with string type
string_columns = data.select_dtypes(include=['object']).columns

# Using built-in function
display(Markdown("### Using Built-in Functions"))

# For all colum with type string
for column in string_columns:
    stat = pd.DataFrame()

    # Get unique values and their counts
    unique_values = data[column].value_counts()

    # Calculate proportions
    proportions = unique_values*100 / len(data)

    # Construct frame
    stat["Unique Value"] = unique_values.index
    stat["Frekuensi"] = unique_values.values
    stat["Proporsi (%)"] = proportions.values

    # Print result
    print("Data for " + column)
```

```
    print(stat)
    print("\n")

# Using built-in function
display(Markdown("### Without Built-in Functions"))

# For all columns with type string
for column in string_columns:
    stat = pd.DataFrame()

    # Dictionary to store the frequency of unique values
    value_counts = {}

    # Calculate frequencies of each unique value
    for value in data[column]:
        if value in value_counts:
            value_counts[value] += 1
        else:
            value_counts[value] = 1

    # Extract unique values and their counts
    unique_values = list(value_counts.keys())
    frequencies = list(value_counts.values())

    # Calculate proportions
    proportions = [(count * 100) / len(data) for count in frequencies]

    # Construct frame
    stat["Unique Value"] = unique_values
    stat["Frekuensi"] = frequencies
    stat["Proporsi (%)"] = proportions

    # Print result
    print("Data for " + column)
    print(stat)
    print("\n")
```

Using Built-in Functions

```
Data for Gender
  Unique Value  Frekuensi  Proporsi (%)
0       Female       1011         50.55
1         Male        989         49.45


Data for Education
        Unique Value  Frekuensi  Proporsi (%)
0  Bachelor's Degree        940         47.00
1    Master's Degree        482         24.10
2        High School        271         13.55
3                PhD        247         12.35
4              Other         57          2.85
5          undefined          3          0.15


Data for Health_Status
  Unique Value  Frekuensi  Proporsi (%)
0         Fair        519         25.95
1         Poor        504         25.20
2    Excellent        503         25.15
3         Good        474         23.70
```

Without Built-in Functions

```
Data for Gender
  Unique Value  Frekuensi  Proporsi (%)
0         Male        989         49.45
1       Female       1011         50.55


Data for Education
        Unique Value  Frekuensi  Proporsi (%)
0  Bachelor's Degree        940         47.00
1    Master's Degree        482         24.10
2        High School        271         13.55
3                PhD        247         12.35
4              Other         57          2.85
5          undefined          3          0.15


Data for Health_Status
  Unique Value  Frekuensi  Proporsi (%)
0    Excellent        503         25.15
1         Fair        519         25.95
2         Poor        504         25.20
3         Good        474         23.70
```

Data Numerik

Untuk data numerik akan dicari:

- mean
- median
- modus
- standar deviasi
- variansi
- range
- minimum
- maksimum
- kuartil
- IQR
- skewness
- kurtosis

In [ ]:
```python
# Get numeric data
numerik_data = data.select_dtypes(include=['number'])

# Calculate the mode for each column
def get_mode_built_in(x: pd.DataFrame) -> pd.Series | None:
    try:
        _ = x.mode()[1]
        mode = 'multivalue'
    except KeyError:
        mode = x.mode()[0]
    return mode

# Construct dataframe
numerik_stat = pd.DataFrame()

numerik_stat["Mean"] = numerik_data.mean().round(3)
numerik_stat["Median"] = numerik_data.median().round(3)
numerik_stat["Modus"] = numerik_data.apply(get_mode_built_in)
numerik_stat["Deviasi"] = numerik_data.std().round(3)
numerik_stat["Variansi"] = numerik_data.var().round(3)
numerik_stat["Min"] = numerik_data.min().round(3)
numerik_stat["Max"] = numerik_data.max().round(3)
numerik_stat["Range"] = numerik_stat["Max"] - numerik_stat["Min"]
numerik_stat["Q1"] = numerik_data.quantile(0.25).round(3)
numerik_stat["Q2"] = numerik_data.quantile(0.5).round(3)
numerik_stat["Q3"] = numerik_data.quantile(0.75).round(3)
numerik_stat["IQR"] = (numerik_stat["Q3"] - numerik_stat["Q1"]).round(3)
numerik_stat["Skewness"] = numerik_data.skew().round(3)
numerik_stat["Kurtosis"] = numerik_data.kurtosis().round(3)

display(Markdown("### Using Built-in Functions"))

# Print result
numerik_stat
```

Using Built-in Functions

Out[ ]:

	Mean	Median	Modus	Deviasi	Variansi
Age	39.418	39.000	53	11.561	1.336510e+02
Income	4889928.319	4898900.675	multivalue	2010795.025	4.043297e+12
Stress_Level	5.357	5.352	multivalue	0.917	8.410000e-01
Exercise_Hours_Per_Week	9.952	9.927	multivalue	4.910	2.410400e+01
Cholesterol_Level	200.223	200.222	multivalue	0.973	9.460000e-01
Weight	70.234	70.392	multivalue	10.174	1.035170e+02
Height	150.731	150.664	multivalue	0.680	4.630000e-01
Blood_Pressure	119.682	119.937	0.0	11.201	1.254720e+02

In [ ]:

```python
# calculate mode
def get_mode(x):
    values_counts = {}
    for value in x:
        if value in values_counts:
            values_counts[value] += 1
        else:
            values_counts[value] = 1
    max_count = max(values_counts.values())
    modes = [key for key, value in values_counts.items() if value == max_count]
    if len(modes) > 1:
        return 'multivalue'
    else:
        return modes[0]

# Calculate the mean
def calculate_mean(column):
    return sum(column) / len(column)

# Calculate the median
def calculate_median(column):
    sorted_column = sorted(column)
    n = len(sorted_column)
    mid = n // 2
    if n % 2 == 0:
        return (sorted_column[mid - 1] + sorted_column[mid]) / 2
    else:
        return sorted_column[mid]

# Calculate the standard deviation
def calculate_std(column, mean):
    variance = sum((x - mean) ** 2 for x in column) / len(column)
    return variance ** 0.5

# Calculate the variance
def calculate_variance(column, mean):
    return sum((x - mean) ** 2 for x in column) / len(column)
```

```python
# Calculate quantiles
def calculate_quantile(column, q):
    sorted_column = sorted(column)
    n = len(sorted_column)
    idx = q * (n - 1)
    if idx.is_integer():
        return sorted_column[int(idx)]
    else:
        lower_idx = int(np.floor(idx))
        upper_idx = int(np.ceil(idx))
        return (sorted_column[lower_idx] + sorted_column[upper_idx]) / 2

# Calculate skewness
def calculate_skewness(column, mean, std):
    n = len(column)
    return (n / ((n - 1) * (n - 2))) * sum(((x - mean) / std) ** 3 for x in column)

# Calculate kurtosis
def calculate_kurtosis(column, mean, std):
    n = len(column)
    return ((n * (n + 1)) / ((n - 1) * (n - 2) * (n - 3))) * sum(((x - mean) / std)

# Construct dataframe
numerik_stat_without_builtin = pd.DataFrame()

for column in numerik_data.columns:
    col_data = numerik_data[column]
    mean = calculate_mean(col_data)
    median = calculate_median(col_data)
    mode = get_mode(col_data)
    std_dev = calculate_std(col_data, mean)
    variance = calculate_variance(col_data, mean)
    min_val = min(col_data)
    max_val = max(col_data)
    q1 = calculate_quantile(col_data, 0.25)
    q2 = calculate_quantile(col_data, 0.5)
    q3 = calculate_quantile(col_data, 0.75)
    iqr = q3 - q1
    skewness = calculate_skewness(col_data, mean, std_dev)
    kurtosis = calculate_kurtosis(col_data, mean, std_dev)

    numerik_stat_without_builtin.loc[column, "Mean"] = round(mean, 3)
    numerik_stat_without_builtin.loc[column, "Median"] = round(median, 3)
    numerik_stat_without_builtin.loc[column, "Modus"] = str(mode)
    numerik_stat_without_builtin.loc[column, "Deviasi"] = round(std_dev, 3)
    numerik_stat_without_builtin.loc[column, "Variansi"] = round(variance, 3)
    numerik_stat_without_builtin.loc[column, "Min"] = round(min_val, 3)
    numerik_stat_without_builtin.loc[column, "Max"] = round(max_val, 3)
    numerik_stat_without_builtin.loc[column, "Range"] = round(max_val - min_val, 3)
    numerik_stat_without_builtin.loc[column, "Q1"] = round(q1, 3)
    numerik_stat_without_builtin.loc[column, "Q2"] = round(q2, 3)
    numerik_stat_without_builtin.loc[column, "Q3"] = round(q3, 3)
    numerik_stat_without_builtin.loc[column, "IQR"] = round(iqr, 3)
    numerik_stat_without_builtin.loc[column, "Skewness"] = round(skewness, 3)
    numerik_stat_without_builtin.loc[column, "Kurtosis"] = round(kurtosis, 3)
```

```
display(Markdown("### Without Built-in Functions"))

# Print result
numerik_stat_without_builtin
```

Without Built-in Functions

Out[ ]:

	Mean	Median	Modus	Deviasi	Variansi
Age	39.417	39.000	53	11.558	1.335840e+02
Income	4889928.319	4898900.675	multivalue	2010292.263	4.041275e+12
Stress_Level	5.357	5.352	multivalue	0.917	8.410000e-01
Exercise_Hours_Per_Week	9.952	9.927	multivalue	4.908	2.409200e+01
Cholesterol_Level	200.223	200.222	multivalue	0.972	9.450000e-01
Weight	70.234	70.392	multivalue	10.172	1.034650e+02
Height	150.731	150.664	multivalue	0.680	4.630000e-01
Blood_Pressure	119.682	119.937	0.0	11.199	1.254090e+02

Nomor 2

Apakah pada data tersebut terdapat outlier? Jika ya, dapatkah anda menanganinya? Jelaskan apa yang umumnya dilakukan untuk menangani outlier.

Jawab

Outlier adalah data yang memiliki nilai jauh dari kumpulan data-data lainnya yang terjadi misalkan karena kegagalan pengukuran, salah masukan, dll.

Berdasarkan proses untuk mengatasi dan menghapus data outlier, kami menemukan data memiliki outlier. Hal ini dapat terlihat karena jumlah data pada hasil proses penghapusan outlier, jumlah data mengurang. Namun hal tersebut juga menunjukkan bahwa kami dapat menangani data-data outlier dengan melakukan penghapusa data outlier sebelum digunakan untuk analisis yang lebih lanjut.

Langkah Mengatasi Outlier

1. Menghilangkan data kosong seperti null dan undefined

Data yang kosong atau null, akan menggangu serta merusak analis. data dengan nilai kosong akan dihapus.

In [ ]:
```
# get initial size
original_size = data.shape[0]

# 1. Remove Null or Undefinded Values
no_null_data = data.dropna()
no_null_data = no_null_data[no_null_data['Education'] != 'undefined']

# get new size
no_null_size = no_null_data.shape[0]

# display the result
display(Markdown("### Using Built-in Functions"))
display(Markdown("#### data removed = %d" % (original_size - no_null_size)))

filtered_data = []

for index, row in data.iterrows():
    if not any(pd.isnull(row)) and row['Education'] != 'undefined':
        filtered_data.append(row)

no_null_data_without_builtin = pd.DataFrame(filtered_data)

# Get new size
no_null_size_without_builtin = no_null_data_without_builtin.shape[0]

# Display the result
display(Markdown("### Without Built-in Functions"))
display(Markdown("#### data removed = %d" % (original_size - no_null_size_without_b
```

Using Built-in Functions

data removed = 3

Without Built-in Functions

data removed = 3

2. Menghilangkan data yang tidak masuk akal

Batas-batas yang digunakan:

- Age: Harus Positif
- Stress_Level: Boleh
- Exercise_Hours_Per_Week: Tidak boleh negatif
- Cholesterol_Level: Harus Positif
- Blood_Pressure: Tidak boleh 0, jika 0 maka orang tersebut mati
- Income: Tidak boleh negatif

In [ ]:
```
# Already not using built-in function

# 2. Remove non logical data
result_data = no_null_data.copy()
```

```
result_data = result_data[result_data['Age'] > 0]
result_data = result_data[result_data['Stress_Level'] >= 0]
result_data = result_data[result_data['Exercise_Hours_Per_Week'] >= 0]
result_data = result_data[result_data['Cholesterol_Level'] > 0]
result_data = result_data[result_data['Blood_Pressure'] > 0]
result_data = result_data[result_data['Income'] > 0]

# get new size
logical_size = result_data.shape[0]

display(Markdown("### data removed = %d" % (no_null_size - logical_size)))
display(Markdown("### current size = %d" % logical_size))
```

data removed = 60

current size = 1937

3. Menggunakan Interquartile Range (IQR) untuk Mengatasi Outlier

Metode Interquartile Range (IQR) merupakan salah satu pendekatan yang efektif untuk mengilangkan data-data outlier. IQR mengukur sebaran tengah dari data dengan menghitung rentang antara kuartil pertama (Q1) dan kuartil ketiga (Q3). Data yang berada di luar rentang ini akan dianggap sebagai outlier dan dihapus.

In [ ]:
```
# 3. using IQR to remove outliers
numeric_columns = data.select_dtypes(include=['number']).columns

# For each numeric column, calculate Q1, Q3, IQR, and remove outliers
for column in numeric_columns:
    Q1 = result_data[column].quantile(0.25)
    Q3 = result_data[column].quantile(0.75)
    IQR = Q3 - Q1

    # Define outliers and remove them
    outliers = (result_data[column] < (Q1 - 1.5 * IQR)) | (result_data[column] > (Q
    result_data = result_data[~outliers]

# Reset data frame index
result_data.reset_index(drop=True, inplace=True)

# get new size
no_outlier_size = result_data.shape[0]

display(Markdown("### Using Built-in Functions"))
display(Markdown("#### data removed = %d" % (logical_size - no_outlier_size)))
display(Markdown("<br>"))

# Assume data after null and 'undefined' removal
result_data_without_builtin = result_data

# For each numeric column, calculate Q1, Q3, IQR, and remove outliers
filtered_rows_without_builtin = result_data_without_builtin.index.tolist()
```

```
for column in numeric_columns:
    column_data_without_builtin = result_data_without_builtin[column]

    # Calculate Q1, Q3, and IQR manually
    Q1_without_builtin = calculate_quantile(column_data_without_builtin, 0.25)
    Q3_without_builtin = calculate_quantile(column_data_without_builtin, 0.75)
    IQR_without_builtin = Q3_without_builtin - Q1_without_builtin

    # Define outliers
    for idx in filtered_rows_without_builtin[:]:  # Create a copy of the list for s
        value = column_data_without_builtin[idx]
        if value < (Q1_without_builtin - 1.5 * IQR_without_builtin) or value > (Q3_
            filtered_rows_without_builtin.remove(idx)

# Create new DataFrame without outliers
no_outlier_data_without_builtin = result_data_without_builtin.loc[filtered_rows_wit

# Reset data frame index
no_outlier_data_without_builtin.reset_index(drop=True, inplace=True)

# Get new size
no_outlier_size_without_builtin = no_outlier_data_without_builtin.shape[0]

# Display the number of rows removed due to outliers
display(Markdown("### Without Built-in Functions"))
display(Markdown("#### data removed = %d" % (logical_size - no_outlier_size_without
display(Markdown("<br>"))
```

Using Built-in Functions

data removed = 115

Without Built-in Functions

data removed = 132

In [ ]:
```
total_removed = original_size - no_outlier_size

display(Markdown("## Resulting data without outliers using built-in functions: "))
display(Markdown("### total data removed = %d" % total_removed))
display(Markdown("#### outlier percentage = %d%%" % (total_removed/original_size*10
display(Markdown("#### new size = %d" % no_outlier_size))

result_data
```

Resulting data without outliers using built-in functions:

total data removed = 178

outlier percentage = 8%

new size = 1822

Out[ ]:

	Age	Income	Gender	Education	Stress_Level	Exercise_Hours_Per_Week	Chole
0	50	3.093457e+06	Male	Bachelor's Degree	4.967887	16.632494	
1	44	5.545445e+06	Male	Bachelor's Degree	5.833649	13.255988	
2	25	4.401808e+06	Male	Master's Degree	5.587946	11.144370	
3	41	4.606865e+06	Female	High School	6.271119	9.140268	
4	24	6.956049e+06	Male	Bachelor's Degree	5.895226	9.775610	
...	...	...	...	...	...	...	
1817	22	2.072546e+06	Male	Master's Degree	5.356903	14.428793	
1818	25	7.456909e+06	Female	Master's Degree	5.499054	5.349180	
1819	44	5.560391e+06	Male	PhD	7.034253	3.746812	
1820	20	3.436759e+06	Male	Bachelor's Degree	5.690408	5.720289	
1821	51	3.343102e+06	Male	Bachelor's Degree	5.572117	11.885388	

1822 rows × 11 columns

Nomor 3

Membuat Visualisasi plot distribusi. Berikan uraian penjelasan kondisi setiap kolom berdasarkan kedua plot tersebut. Jika numerik dapat dibuat dalam bentuk histogram dan box plot, dan jika string dengan histogram.

In [ ]:
```
data_column_numeric = result_data.select_dtypes(include=['number']).columns
data_column_string = result_data.select_dtypes(exclude=['number']).columns

# Box plot settings
plt.rcParams['boxplot.boxprops.linewidth'] = 2
plt.rcParams['boxplot.whiskerprops.linewidth'] = 2
plt.rcParams['boxplot.capprops.linewidth'] = 2
plt.rcParams['boxplot.medianprops.linewidth'] = 2
```

```
plt.rcParams['boxplot.flierprops.marker'] = 'o'
plt.rcParams['boxplot.flierprops.markersize'] = 8
plt.rcParams['boxplot.flierprops.linewidth'] = 2
plt.rcParams['boxplot.flierprops.markeredgecolor'] = 'mediumorchid'

def distribution_plot_numeric(column_name: str, bar_label: bool = False, bin_count:
    """
    Create a distribution plot for numeric type
    """

    # Create Plot
    fig_hist, ax_hist = plt.subplots(figsize=(8, 4))
    fig_box, ax_box = plt.subplots(figsize=(8, 3))

    # Plot the histogram
    column_data = result_data[column_name]
    ax_hist.set_title(f"Distribusi data {column_name}", fontsize=16)
    ax_hist.set_xlabel(column_name, fontsize=14)
    ax_hist.set_ylabel("Frekuensi", fontsize=14)
    hist, bins, _ = ax_hist.hist(column_data, bins=bin_count, color='mediumorchid',
    ax_hist.grid(True, linestyle='--', alpha=0.5)
    ax_hist.tick_params(axis='both', which='major', labelsize=12)

    # Add text label to each bar
    if bar_label :
        bin_centers = 0.5 * (bins[:-1] + bins[1:])
        for count, (x, y) in enumerate(zip(bin_centers, hist)):
            ax_hist.text(x, y, f'{int(bins[count]):,} - {int(bins[count+1]):,}', ha

    # Plot the box plot
    ax_box.set_title(f"Distribusi data {column_name}", fontsize=16)
    ax_box.boxplot(column_data, vert=False)
    ax_box.set_xlabel(column_name, fontsize=14)
    ax_box.set_ylabel("Box Plot", fontsize=14)
    ax_box.boxplot(column_data, vert=False, patch_artist=True, boxprops=dict(faceco
    ax_box.grid(True, linestyle='--', alpha=0.5)
    ax_box.tick_params(axis='both', which='major', labelsize=12)

    plt.tight_layout()
    plt.show()

def distribution_plot_string(column_name: str) -> None:
    """
    Create a distribution plot for string type
    """

    # Extract column data
    column_data = result_data[column_name]

    # Count the frequency of each category
    category_counts = column_data.value_counts()

    # Create plot
    plt.figure(figsize=(8, 4))
    plt.bar(category_counts.index, category_counts.values, color='mediumorchid', al
    plt.title(f"Distribusi data {column_name}", fontsize=16)
```

```
    plt.ylabel("Frekuensi", fontsize=14)
    plt.grid(True, linestyle='--', alpha=0.5)
    plt.xticks(rotation=45, ha='right', fontsize=12)
    plt.yticks(fontsize=12)
    plt.tight_layout()
    plt.show()

# Show new data without outliers
numerik_stat = pd.DataFrame()
numerik_data = result_data.select_dtypes(include=['number'])
numerik_stat["Mean"] = numerik_data.mean().round(3)
numerik_stat["Median"] = numerik_data.median().round(3)
numerik_stat["Modus"] = numerik_data.apply(get_mode)
numerik_stat["Deviasi"] = numerik_data.std().round(3)
numerik_stat["Variansi"] = numerik_data.var().round(3)
numerik_stat["Min"] = numerik_data.min().round(3)
numerik_stat["Max"] = numerik_data.max().round(3)
numerik_stat["Range"] = numerik_stat["Max"] - numerik_stat["Min"]
numerik_stat["Q1"] = numerik_data.quantile(0.25).round(3)
numerik_stat["Q2"] = numerik_data.quantile(0.5).round(3)
numerik_stat["Q3"] = numerik_data.quantile(0.75).round(3)
numerik_stat["IQR"] = (numerik_stat["Q3"] - numerik_stat["Q1"]).round(3)
numerik_stat["Skewness"] = numerik_data.skew().round(3)
numerik_stat["Kurtosis"] = numerik_data.kurtosis().round(3)

# Print result
numerik_stat
```

Out[ ]:

	Mean	Median	Modus	Deviasi	Variansi
Age	39.429	39.000	20	11.547	1.333410e+02
Income	4904525.748	4881380.193	multivalue	1924955.096	3.705452e+12
Stress_Level	5.358	5.355	multivalue	0.893	7.980000e-01
Exercise_Hours_Per_Week	10.109	9.981	multivalue	4.552	2.071900e+01
Cholesterol_Level	200.225	200.217	multivalue	0.945	8.940000e-01
Weight	70.146	70.317	multivalue	9.741	9.487800e+01
Height	150.699	150.646	multivalue	0.633	4.000000e-01
Blood_Pressure	119.949	120.113	multivalue	9.617	9.248000e+01

In [ ]:

```
# Get columns with string type
string_columns = result_data.select_dtypes(include=['object']).columns

# For all colum with type string
for column in string_columns:
    stat = pd.DataFrame()

    # Get unique values and their counts
    unique_values = result_data[column].value_counts()
```

```
    # Calculate proportions
    proportions = unique_values*100 / len(result_data)

    # Construct frame
    stat["Unique Value"] = unique_values.index
    stat["Frekuensi"] = unique_values.values
    stat["Proporsi (%)"] = proportions.values

    # Print result
    print("Data for " + column)
    print(stat)
    print("\n")
```

```
Data for Gender
  Unique Value  Frekuensi  Proporsi (%)
0       Female        918     50.384193
1         Male        904     49.615807


Data for Education
        Unique Value  Frekuensi  Proporsi (%)
0  Bachelor's Degree        846     46.432492
1    Master's Degree        449     24.643249
2        High School        247     13.556531
3                PhD        231     12.678375
4              Other         49      2.689352


Data for Health_Status
  Unique Value  Frekuensi  Proporsi (%)
0         Fair        473     25.960483
1         Poor        463     25.411636
2    Excellent        461     25.301866
3         Good        425     23.326015
```

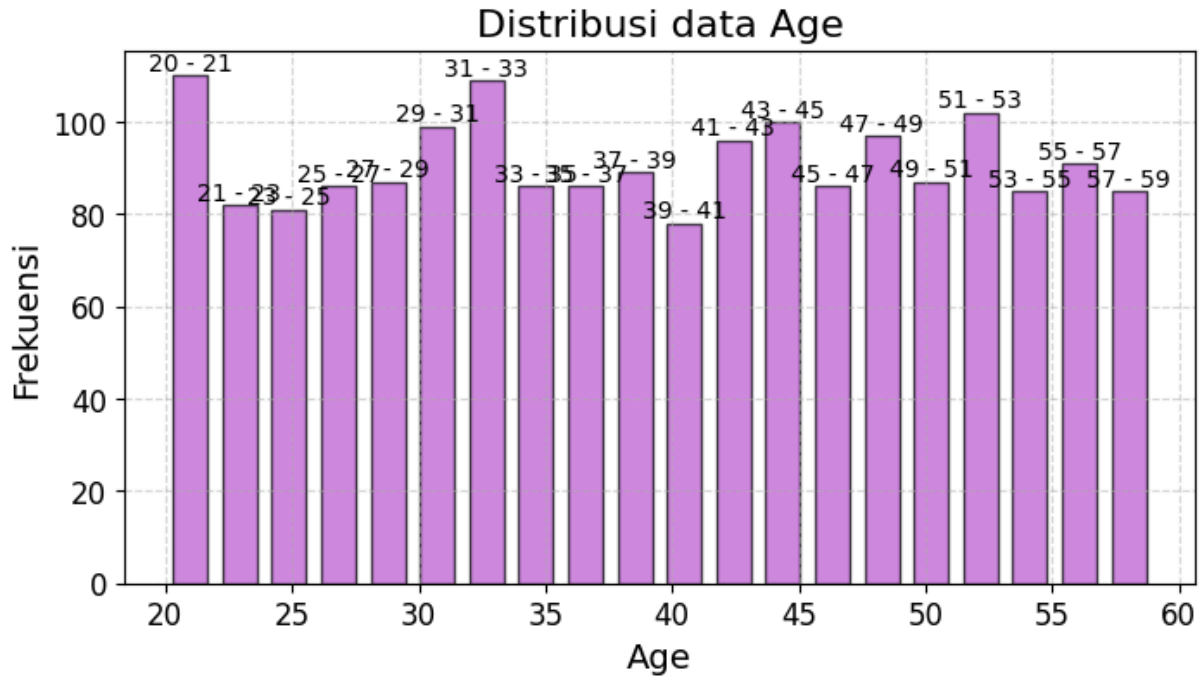
Data Age

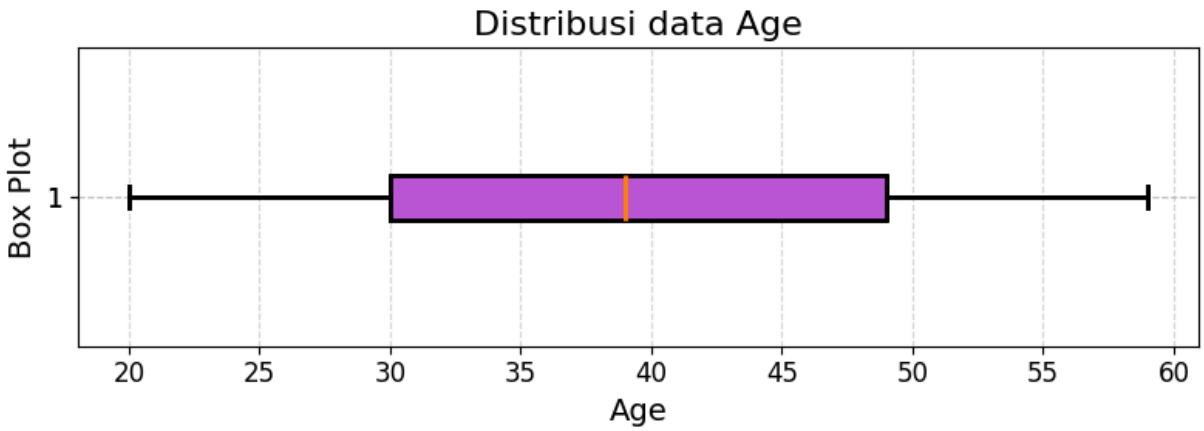
Kolom Age memiliki data umur berupa bilangan bulat yang dibagi menjadi 20 bagian dengan lebar yang sama. Kolom Age memiliki range 39, dengan nilai minimum 20 dan maksismum 59.

Hasil histogram, menunjukan distribusi Age yang cukup rata (uniform) dengan frekuensi tertingi berada pada range umur 20-21 dan terkecil pada range umur 39-41.

Hasil box plot, menunjukan adanya negatif skewness(-0.012). Hal ini disebabkan posisi nilai median(39) yang lebih dekat dengan nilai Kuartil Q1(30) dibandingkan dengan nilai Kuartil Q3(49).

In [ ]:
```
distribution_plot_numeric(data_column_numeric[0], bar_label=True)
```





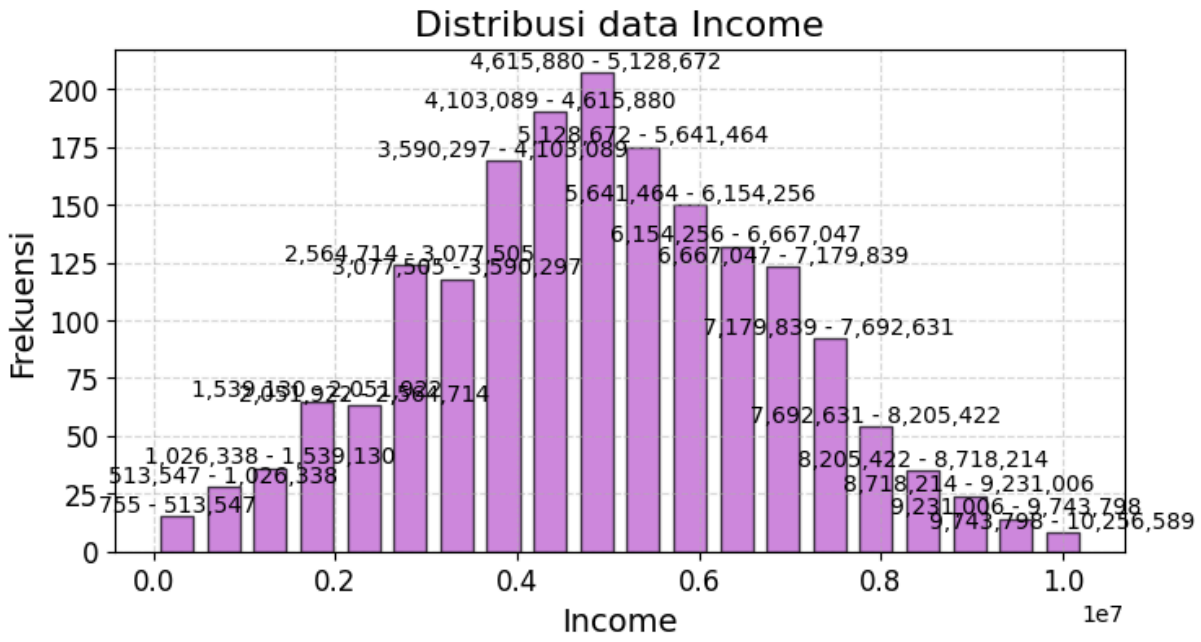
Data Income

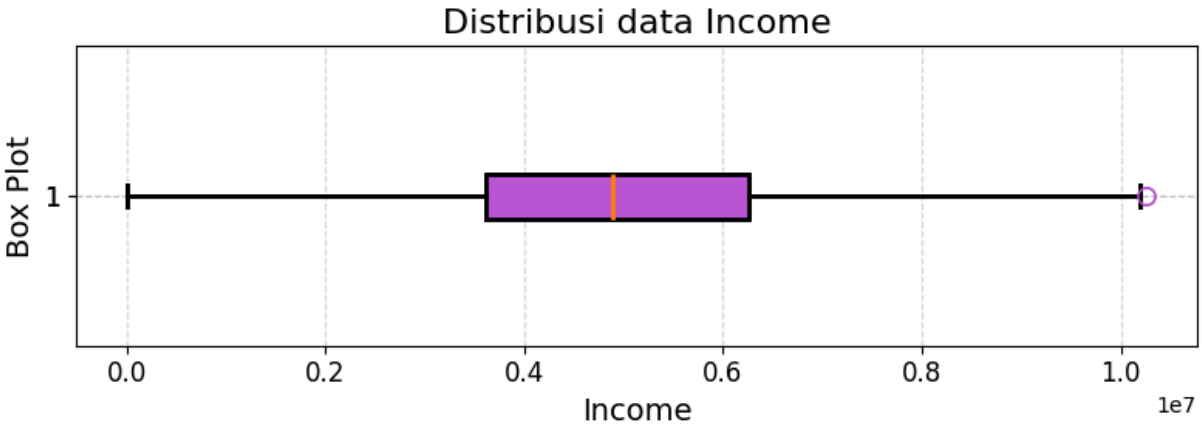
Kolom Income memiliki data penghasilan yang berupa bilangan real yang dibagi menjadi 20 bagian dengan lebar yang sama. Kolom Income memiliki nilai minimum 755.409 dan nilai maksismum 10.256.589.

Hasil histogram, menunjukan distribusi Income yang simetris dengan frekuensi tertingi berada pada range penghasilan 4.615.880 - 5.128.672 dan terkecil pada range penghasilan 9.743.798 hingga 10.256.589.

Hasil box plot, menunjukan adanya positif skewness (0.028). Hal ini disebabkan posisi nilai median(4.904.525,748) yang lebih dekat dengan nilai Kuartil Q3(6.265.044,813) dibandingkan dengan nilai Kuartil Q1(3.612.627,851). Terlihat juga terdapat nilai-nilai yang berada di luar range antara kuartil pertama dan ketiga.

In [ ]:
```
distribution_plot_numeric(data_column_numeric[1], True)
```





Data Stress_Level

Kolom Stress_Level memiliki data tingkat stress yang berupa bilangan real yang dibagi menjadi 10 bagian dengan lebar yang sama. Kolom Stress_Level memiliki range 4.999 dengan nilai minimum 2.874 dan nilai maksismum 7,873.

Hasil histogram, menunjukan distribusi Stress_Level yang simetris dengan frekuensi tertingi berada pada range 4 - 5 (desimal tidak terlihat) .

Hasil box plot, menunjukan adanya negatif skewness (-0,024). Hal ini disebabkan posisi nilai median(5,358) yang lebih dekat dengan nilai Kuartil Q1(4,747) dibandingkan dengan nilai Kuartil Q3(5,997). Terlihat juga terdapat nilai-nilai yang berada di luar range antara kuartil pertama dan ketiga.

In [ ]:
```
distribution_plot_numeric(data_column_numeric[2], True, bin_count=10)
```

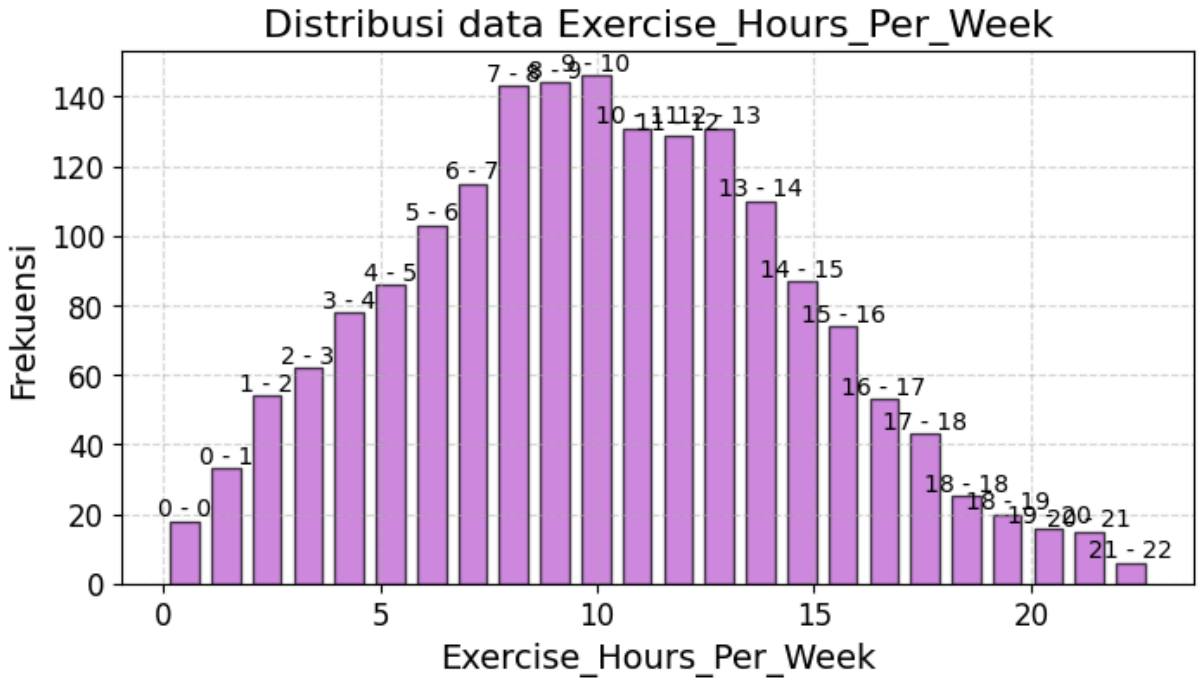
Data Exercise_Hours_Per_Week

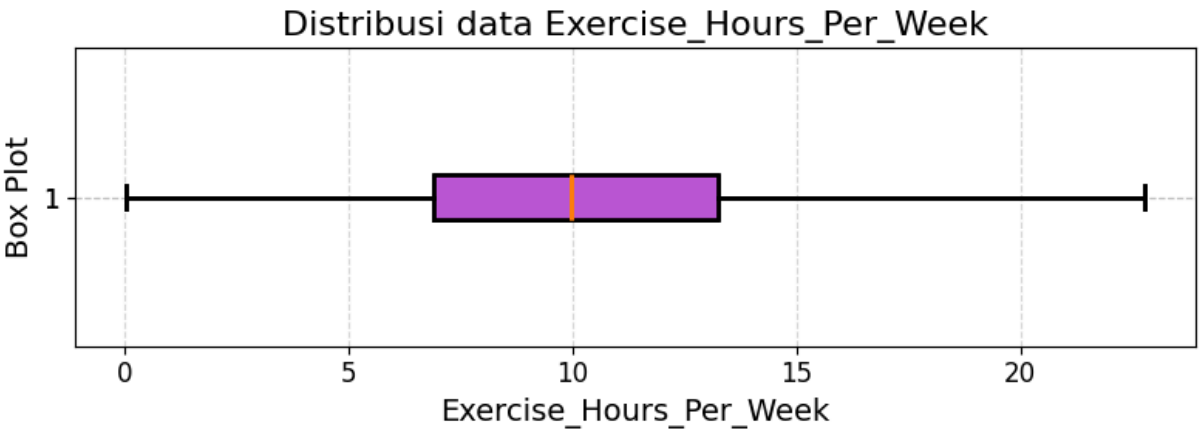
Kolom Exercise_Hours_Per_Week merupakan data jumlah waktu olahraga dalam jam per minggu yang berupa bilangan real yang dibagi menjadi 24 bagian dengan lebar yang sama. Kolom Exercise_Hours_Per_Week memiliki range 22,543 dengan nilai minimum 0,023 dan nilai maksismum 22,773.

Hasil histogram, menunjukan distribusi Exercise_Hours_Per_Week yang mengalami skewnes dengan frekuensi tertingi berada pada range 9-10 jam per minggu dan terkecil pada range 21-22 jam per minggu.

Hasil box plot, menunjukan adanya positif skewness (0,155). Hal ini disebabkan posisi nilai median(9,981) yang lebih dekat dengan nilai Kuartil Q3(13,250) dibandingkan dengan nilai Kuartil Q1(6,901).

In [ ]:
```
distribution_plot_numeric(data_column_numeric[3], bin_count=24, bar_label=True)
```





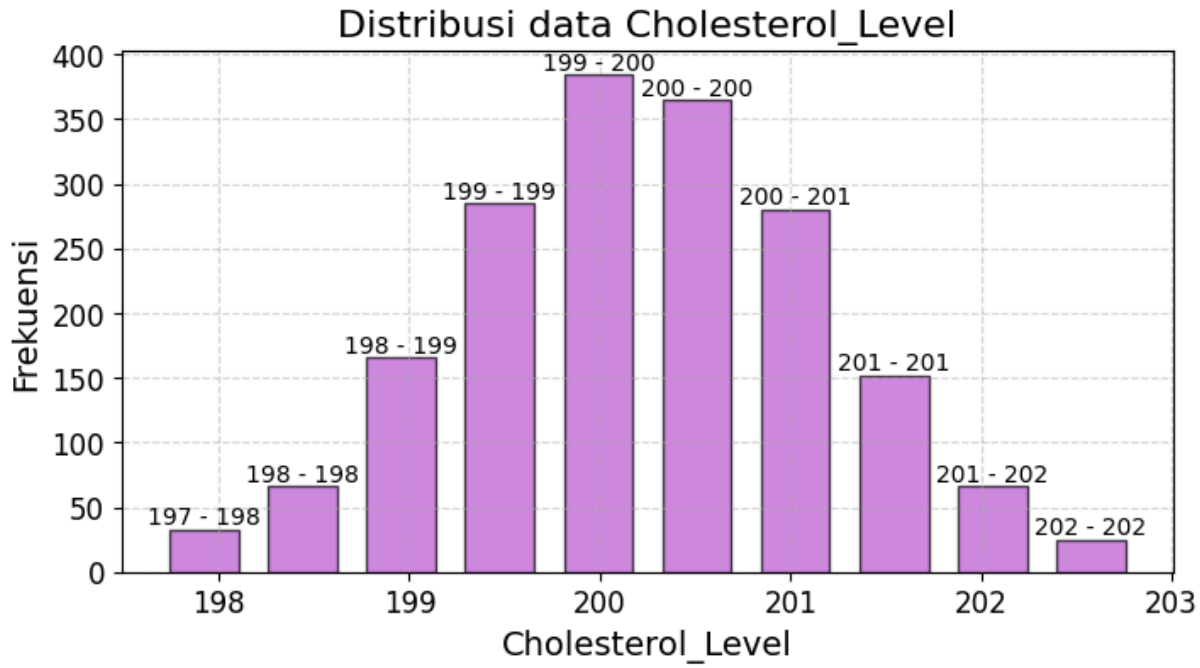
Data Cholesterol_Level

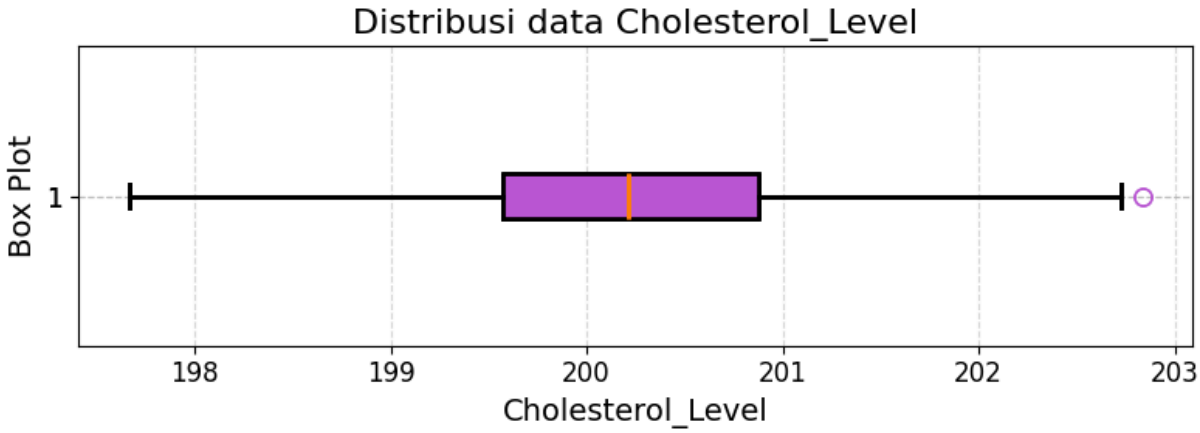
Kolom Cholesterol_Level merupakan data tingkat kolesterol yang berupa bilangan real yang dibagi menjadi 10 bagian dengan lebar yang sama. Kolom Cholesterol_Level memiliki range 5,169 dengan nilai minimum 197,665 dan nilai maksismum 202,834.

Hasil histogram, menunjukan distribusi Cholesterol_Level yang simetris dengan frekuensi tertingi berada pada range 199-200(desimal tidak terlihat) dan terkecil pada range 202-202(desimal tidak terlihat).

Hasil box plot, menunjukan sedikitnya skewness (0,001). Hal ini disebabkan posisi nilai median(200,224) yang terhadap nilai Kuartil Q1(199,577) dibandingkan dengan nilai Kuartil Q3(200,877) yang hampir sama. Terlihat juga terdapat nilai-nilai yang berada di luar range antara kuartil pertama dan ketiga.

In [ ]:
```
distribution_plot_numeric(data_column_numeric[4], True, bin_count=10)
```





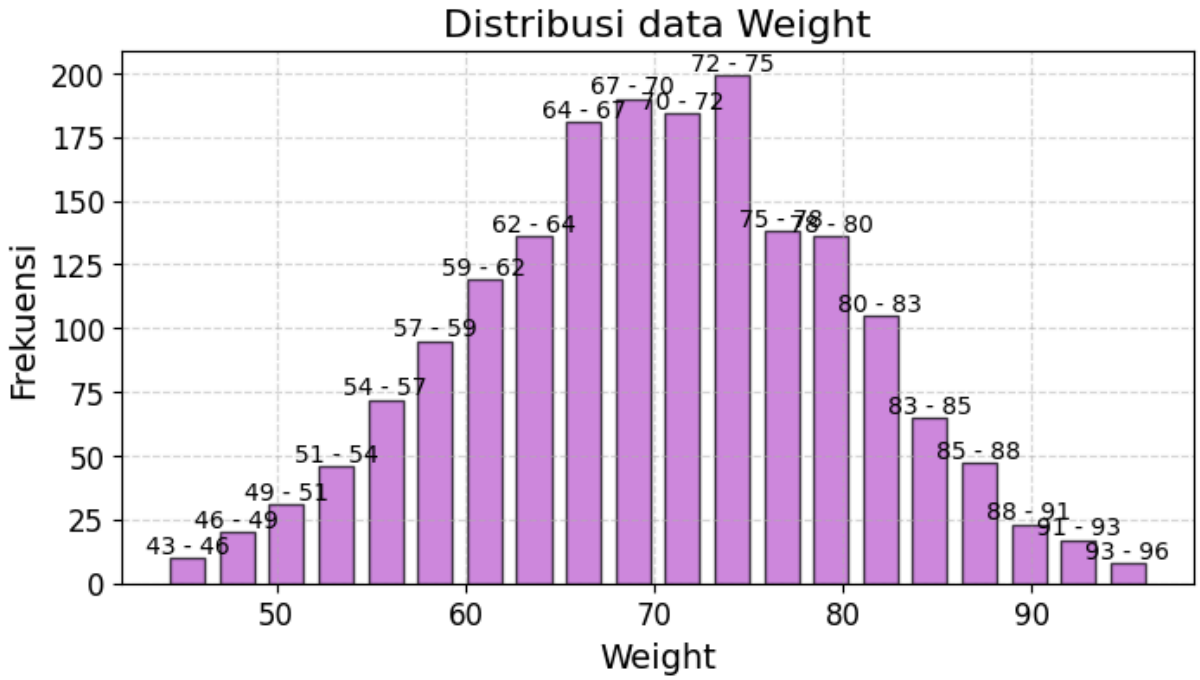
Data Weight

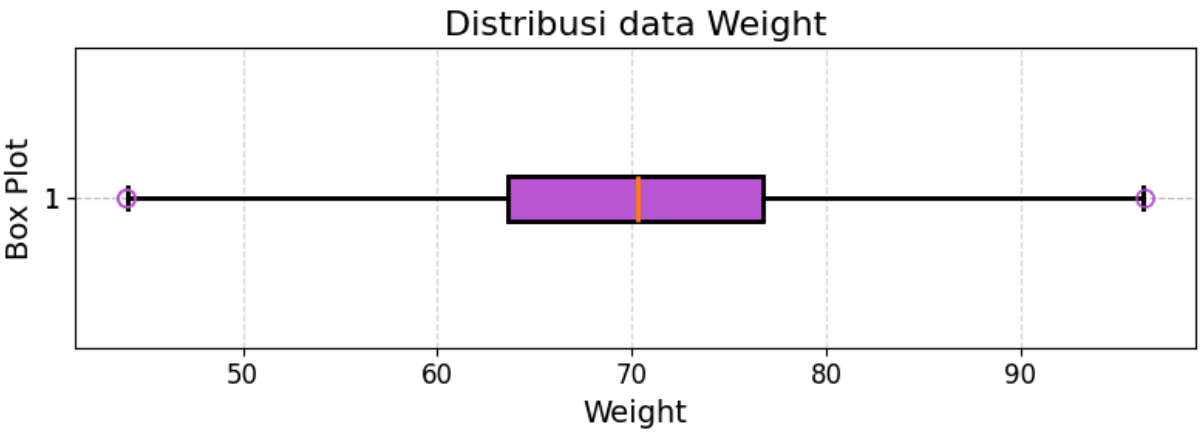
Kolom Weight merupakan data berat badan yang berupa bilangan real yang dibagi menjadi 20 bagian dengan lebar yang sama. Kolom Weight memiliki range 52,408 dengan nilai minimum 43,974 dan nilai maksismum 96,382.

Hasil histogram, menunjukan distribusi Weight yang simetris dengan frekuensi tertingi berada pada range 72-75 dan terkecil pada range 93-96.

Hasil box plot, menunjukan adanya negatif skewness (-0,071). Hal ini disebabkan posisi nilai median(70,317) yang lebih dekat dengan nilai Kuartil Q1(63,634) dibandingkan dengan nilai Kuartil Q3(76,708). Terlihat juga terdapat nilai-nilai yang berada di luar range antara kuartil pertama dan ketiga.

In [ ]:
```
distribution_plot_numeric(data_column_numeric[5], True)
```





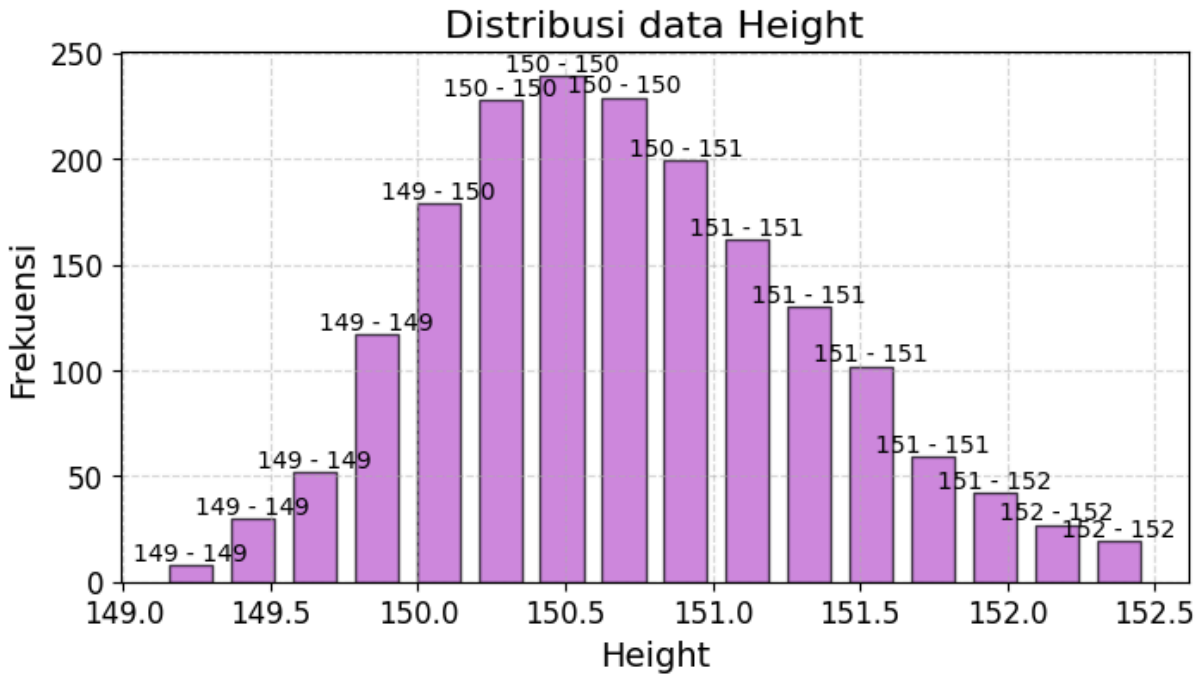
Data Height

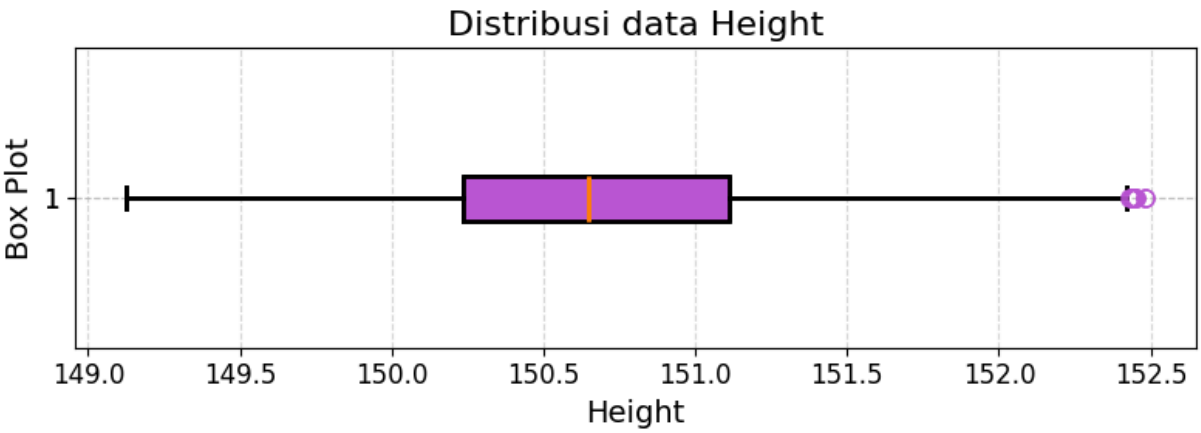
Kolom Height merupakan data tinggi badan yang berupa bilangan real yang dibagi menjadi 16 bagian dengan lebar yang sama. Kolom Height memiliki range 3,357 dengan nilai minimum 149.125 dan nilai maksismum 152,482.

Hasil histogram, menunjukan distribusi Height yang mengalami skewness positif dengan frekuensi tertingi berada pada range 159-150 (Desimal tidak terlihat) dan terkecil pada range 149-149 (Desimal tidak terlihat).

Hasil box plot, menunjukan adanya positif skewness (0,347). Hal ini disebabkan posisi nilai median(150,657) yang lebih dekat dengan nilai Kuartil Q3(151,114) dibandingkan dengan nilai Kuartil Q1(150,236). Terlihat juga terdapat nilai-nilai yang berada di luar range antara kuartil pertama dan ketiga.

In [ ]:
```
distribution_plot_numeric(data_column_numeric[6], bin_count=16, bar_label=True)
```





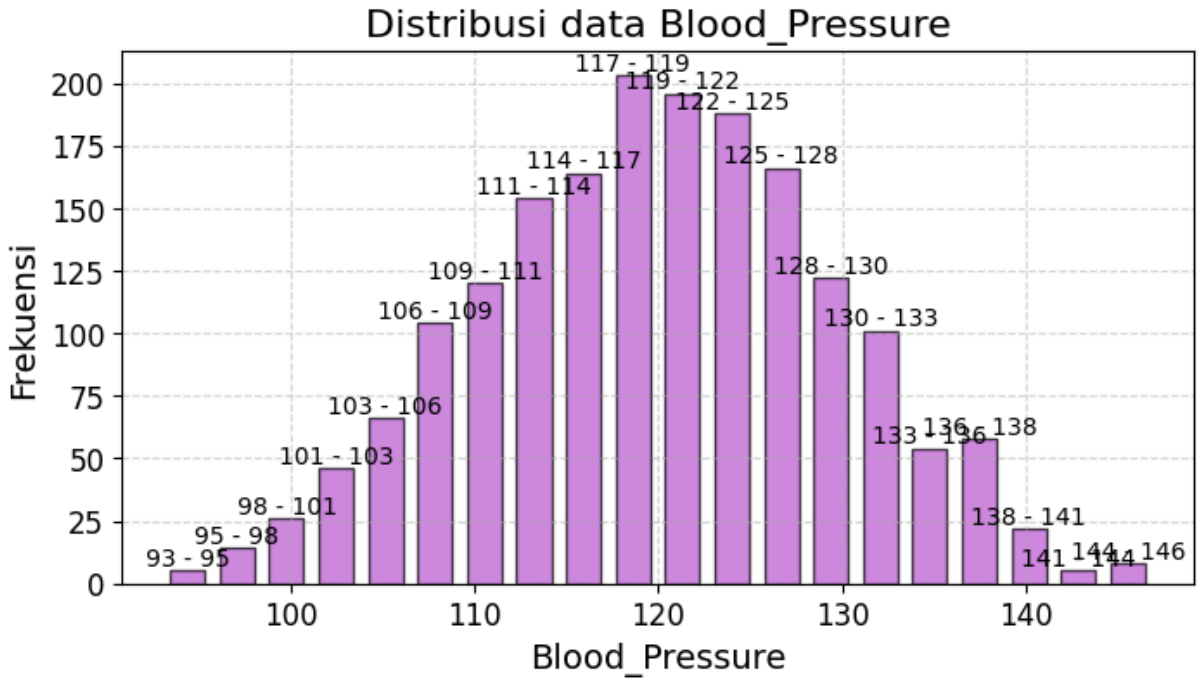
Data Blood_Pressure

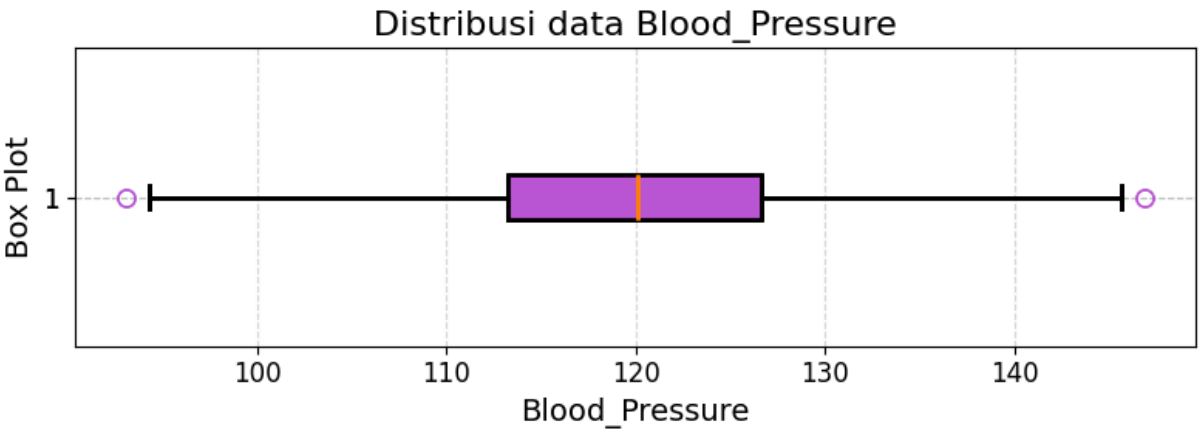
Kolom Blood_Pressure merupakan data ukuran tekanan darah yang dibagi menjadi 20 bagian dengan lebar yang sama. Kolom Blood_Pressure memiliki range 53,828 dengan nilai minimum 93,055 dan nilai maksismum 146,883.

Hasil histogram, menunjukan distribusi Blood_Pressure yang mengalami skewness positif dengan frekuensi tertingi berada pada range 1179-119 dan terkecil pada range 93-95.

Hasil box plot, menunjukan adanya negatif skewness (-0,034). Hal ini disebabkan posisi nilai median(120,113) yang lebih dekat dengan nilai Kuartil Q1(113,246) dibandingkan dengan nilai Kuartil Q3(126,614). Terlihat juga terdapat nilai-nilai yang berada di luar range antara kuartil pertama dan ketiga.

In [ ]:
```
distribution_plot_numeric(data_column_numeric[7], True)
```





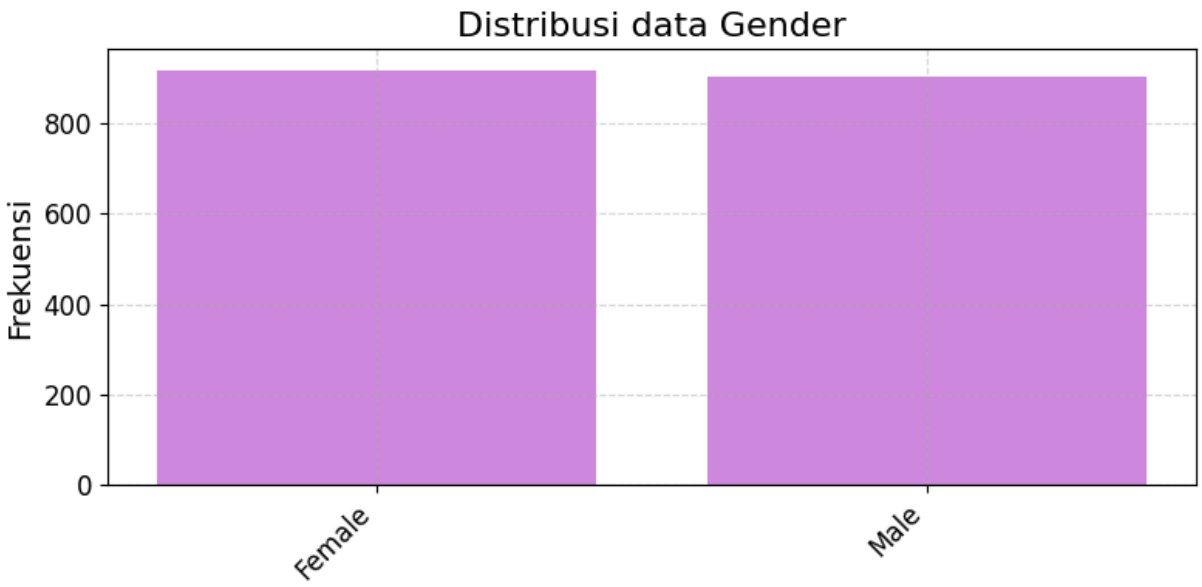
Data Gender

Kolom Gender merupakan data kelamin yang terbagi menjadi 2, yaitu Male dan Female

Distribusi data kolom Gender adalah rata(uniform). Hal tersebut dikarenakan proporsi Female bernilai 50,575% dan Male bernilai 49,425%.

Unique Value	Frekuensi	Proporsi (%)
Female	968	50.384193
Male	946	49.615807

In [ ]:
```
distribution_plot_string(data_column_string[0])
```



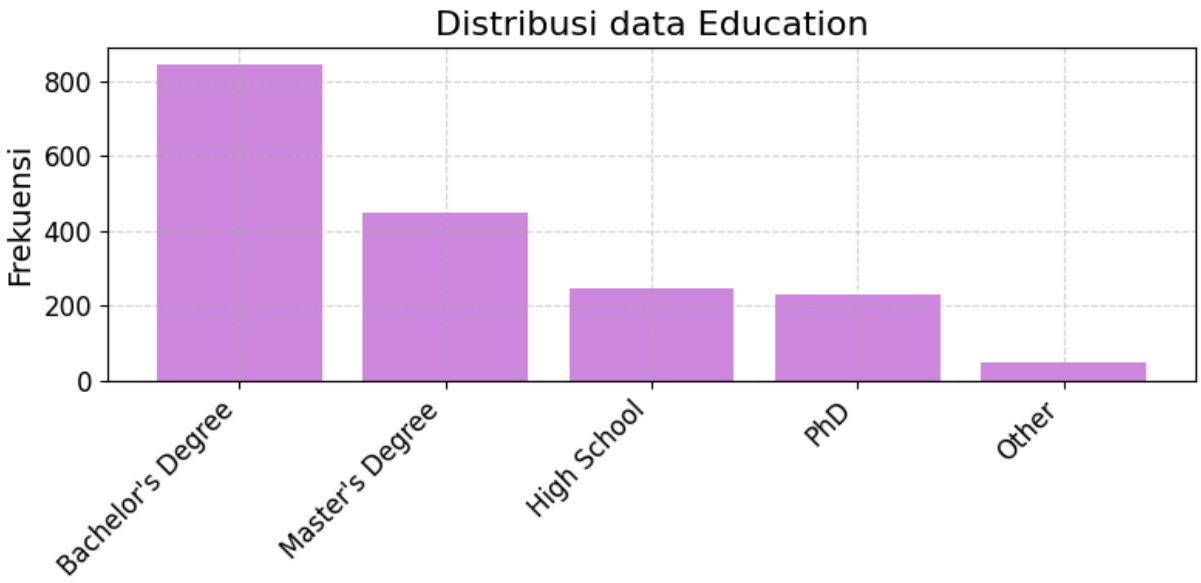
Data Education

Kolom Education merupakan data tingkat edukasi yang terbagi menjadi 5, yaitu Banchelor's Degree, Master's Degree, High School, Phd dan Other

Distribusi data kolom Education adalah tidak rata dengan 46,76% adalah Banchelor's Degree

Unique Value	Frekuensi	Proporsi (%)
Bachelor's Degree	895	46.432492
Master's Degree	469	24.643249
High School	258	13.556531
PhD	240	12.678375
Other	52	2.689352

In [ ]:
```
distribution_plot_string(data_column_string[1])
```



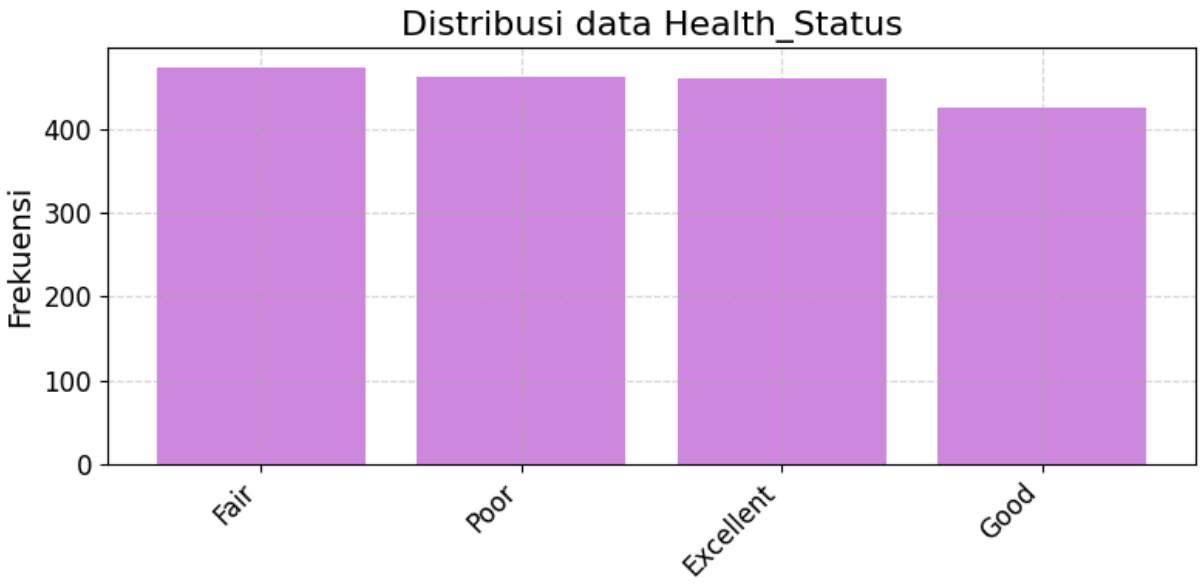
Data Health_Status

Kolom Health_Status merupakan data status kesehatan yang terbagi menjadi 4, yaitu Fair, Poor, Excellent, dan Good

Distribusi data kolom Health_Status adalah rata(uniform) dengan rata-rata proporsi adalah 24,986% atau dibulatkan menjadi 25%

Unique Value	Frekuensi	Proporsi (%)
Fair	498	25.960483
Poor	486	25.411636
Excellent	481	25.301866
Good	449	23.326015

In [ ]:
```
distribution_plot_string(data_column_string[2])
```



Nomor 4

Menentukan distribusi setiap kolom numerik menggunakan hasil visualisasi histogram. Apakah kolom tersebut berdistribusi normal? Jika bukan, terdistribusi seperti apa kolom tersebut?

Jawab

Untuk menentukan jenis dari distribusi kami melakukan test-test berikut:

- D'Agostino's K-squared Test -> distribusi normal
- Kolmogorov-Smirnov Test -> distribusi uniform
- Chi-Square Test -> distribusi poisson

Tidak semua test untuk semua jenis distribusi digunakan. Namun dengan menggunakan test-test tersebut, sudah cukup untuk menentukan semua distribusi dari data numerik.

In [ ]:
```python
def get_chi_square_p(column_values):
    """
    Calculates the p-value of the Chi-squared test for a given column's distributio
    """

    # Fit a normal distribution to the data for expected counts
    mu = column_values.mean()
    sigma = column_values.std()

    # Define number of bins and bin edges
    num_bins = 10
    bin_edges = np.linspace(column_values.min(), column_values.max(), num_bins + 1)

    # Calculate expected frequencies for each bin using the normal distribution
    expected_counts = []
    for i in range(num_bins):
```

```
        start = bin_edges[i]
        end = bin_edges[i + 1]
        expected_prob = stats.norm.cdf(end, mu, sigma) - stats.norm.cdf(start, mu,
        expected_counts.append(expected_prob * len(column_values))
    expected_counts = np.array(expected_counts)

    # Calculate observed frequencies
    observed_counts, _ = np.histogram(column_values, bins=bin_edges)

    # Perform the chi-squared test
    try:
        _, p_value = stats.chisquare(observed_counts, expected_counts)
        return p_value
    except Exception as e:
        return 0


def test_distribution(column_name: str) -> None:
    # Check if the data contain string or non-calculated data
    if result_data[column_name].dtype == 'object' :
        display(Markdown("#### Kolom %s Memiliki Tipe Data String" % column_name))
        display(Markdown("#### Sehingga tidak berdistribusi"))
        return

    column_values = result_data[column_name]

    # D'Agostino's K-squared Test
    # Checks if the distribution is normal distribution
    stat_normal, p_normal = stats.normaltest(column_values, nan_policy='omit')
    print(f"Nilai p-Value: {p_normal}\n")
    print(f"Nilai skewness: {result_data[column_name].skew()}")
    print(f"Nilai kurt: {result_data[column_name].kurt()}")

    if p_normal >= 0.05:
        # Normal distribution

        # Display p value and conclusion
        display(Markdown("#### D'Agostino's K-squared Test"))
        display(Markdown("#### nilai $p-normal$ = $%.20f$" % p_normal))
        display(Markdown("#### Kolom %s Berdistribusi Normal" % column_name))

    else:
        # Non-normal distribution

        # Kolmogorov-Smirnov Test
        # Checks if the distribution is uniform distribution
        stat_uniform, p_uniform = stats.kstest(column_values, 'uniform')

        if p_uniform > 0.05:

            # Distribution is similar to uniform distribution
            display(Markdown("#### Kolmogorov-Smirnov Test"))
            display(Markdown("#### nilai $p-uniform$ = $%.20f$" % p_uniform))
            display(Markdown("#### Kolom %s Mirip Dengan Distribusi Uniform" % colu

        else:
```

```
            # Non-normal distribution
            # Non-uniform distribution

            # Chi-Squared Test
            # Checks if the distribution is poisson distribution
            if get_chi_square_p(column_values) >= 0.5:
                display(Markdown("#### Kolom %s Mirip Dengan Distribusi Uniform" %

    plt.figure(figsize=(8, 4))
    column_values.plot(kind="hist", color="mediumorchid", linewidth=1.9, edgecolor=
    column_values.plot(kind="kde", color="#22ff22", linewidth=3, figsize=(10, 6))
    plt.title(column_name)
    plt.show()
```

In [ ]:
```
display(Markdown(f"#### Jenis distribusi kolom Age adalah distribusi Uniform"))
test_distribution(data_column_numeric[0])
```

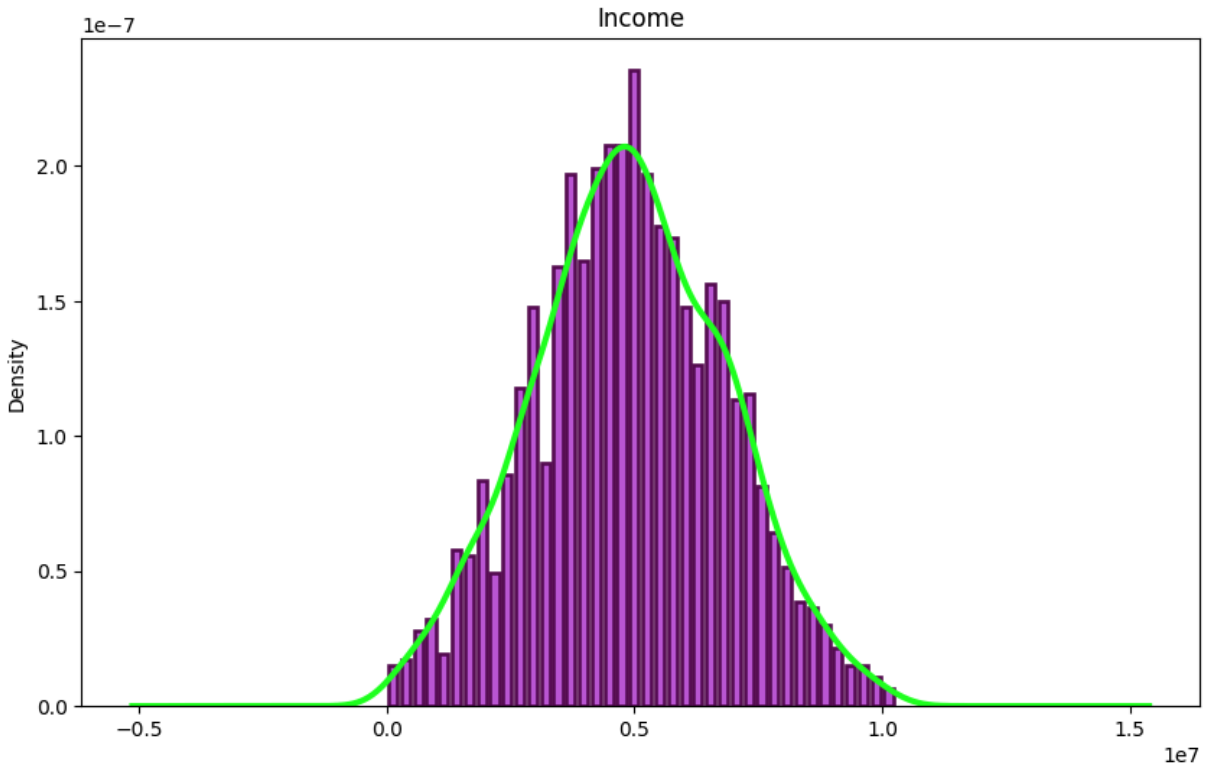
Jenis distribusi kolom Age adalah distribusi Uniform

```
Nilai p-Value: 8.846688786343492e-301

Nilai skewness: -0.012077819511978245
Nilai kurt: -1.1942642466245585
```

In [ ]:
```
display(Markdown("#### Jenis distribusi tidak dapat ditentukan, namun mendekati dis
test_distribution(data_column_numeric[1])
```

Jenis distribusi tidak dapat ditentukan, namun mendekati distribusi normal

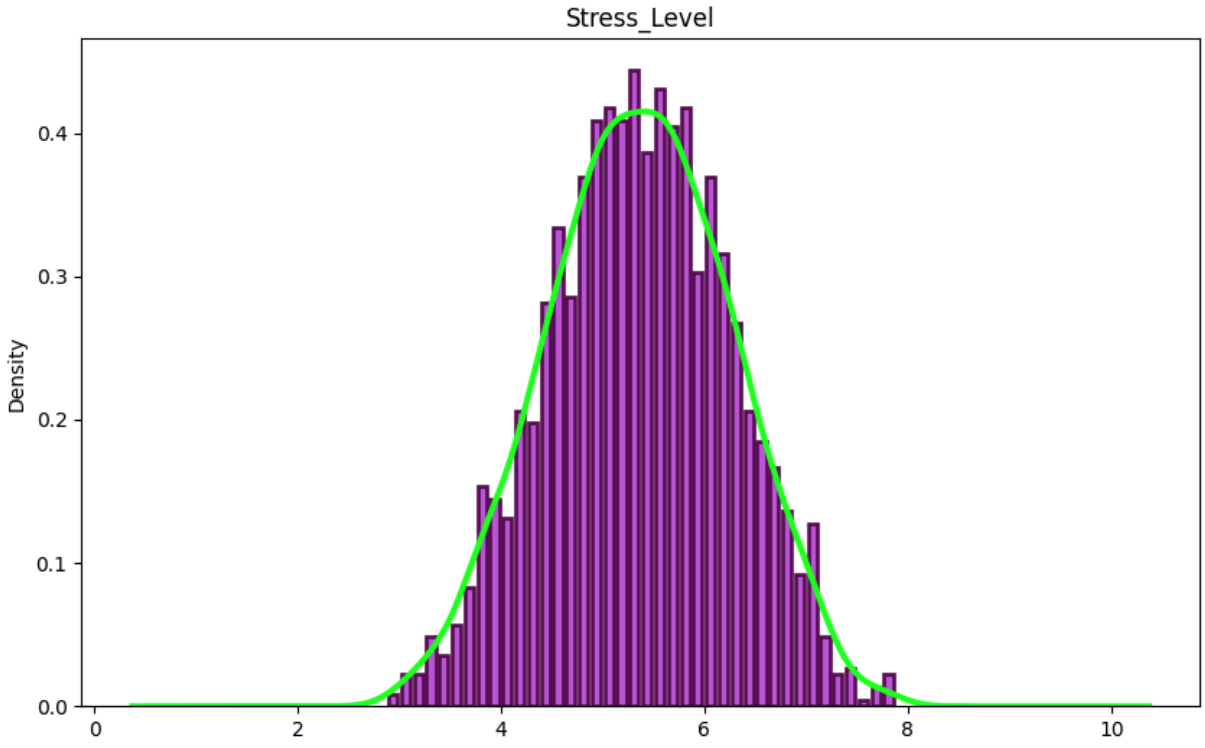
```
Nilai p-Value: 0.008976711670815726

Nilai skewness: 0.027614306893063753
Nilai kurt: -0.29891462620300935
```



In [ ]:
```
display(Markdown("### Distribusi Normal"))
test_distribution(data_column_numeric[2])
```

Distribusi Normal

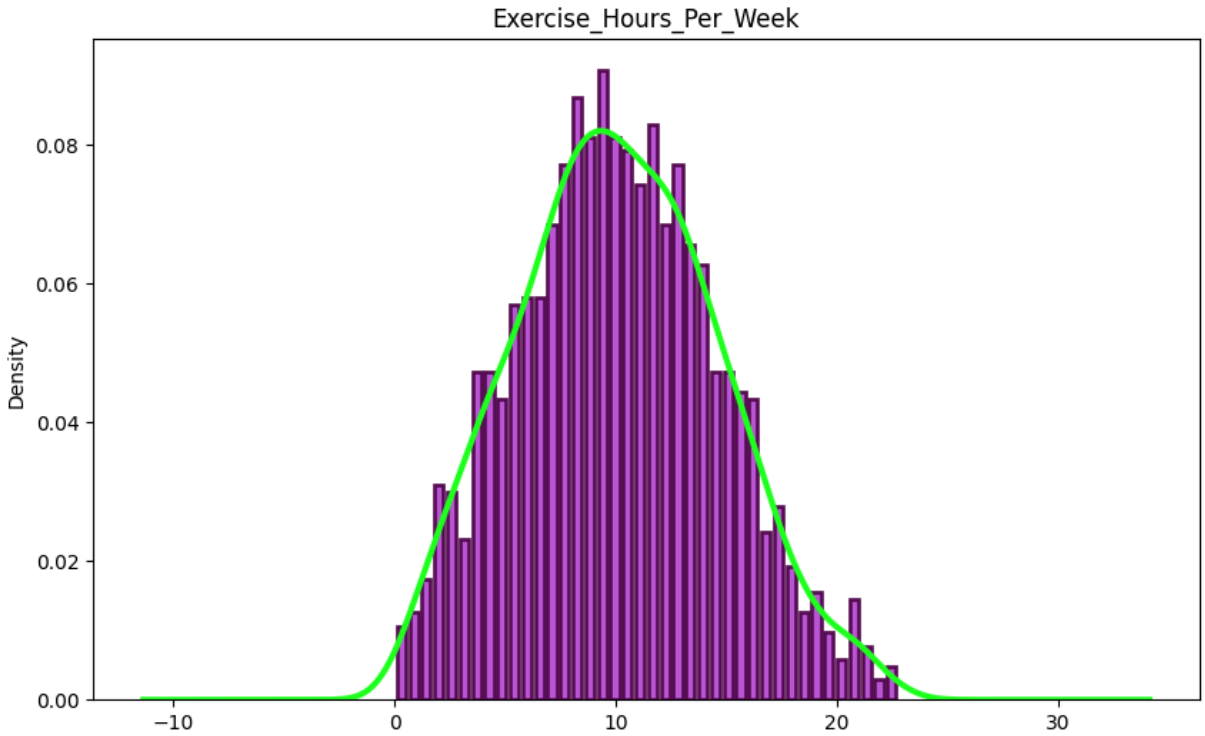
```
Nilai p-Value: 0.0012895750520694694

Nilai skewness: -0.023891293873349626
Nilai kurt: -0.34584961093575517
```



In [ ]:
```
display(Markdown("### Distribusi Normal"))
test_distribution(data_column_numeric[3])
```

Distribusi Normal

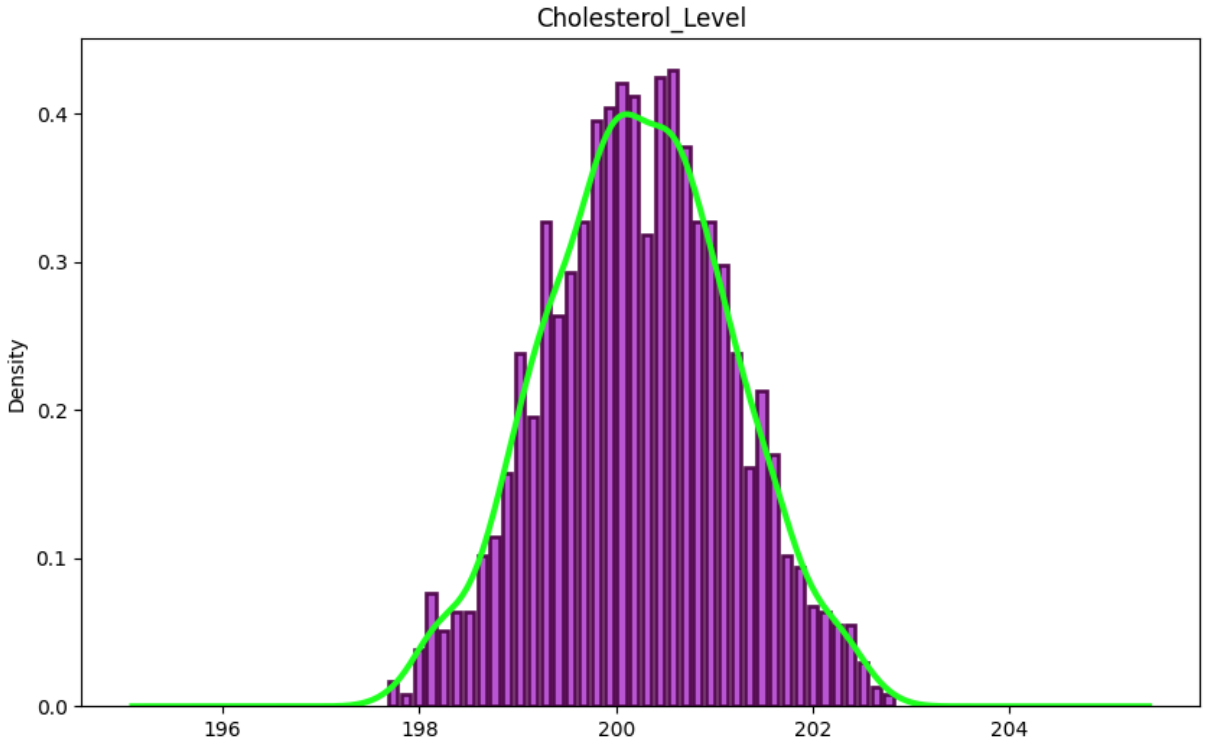
```
Nilai p-Value: 1.5516221633294157e-06

Nilai skewness: 0.15486310998406536
Nilai kurt: -0.4044533392599705
```



In [ ]:
```
display(Markdown("#### Distribusi Normal"))
test_distribution(data_column_numeric[4])
```

Distribusi Normal

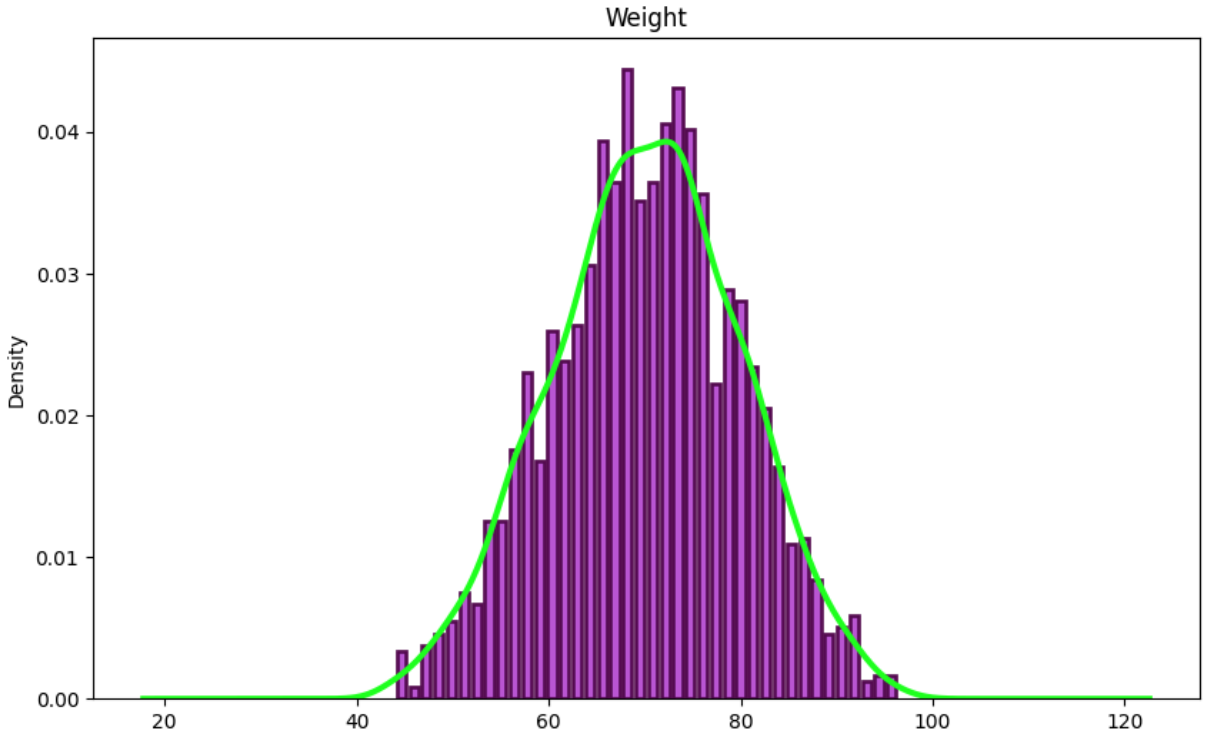
```
Nilai p-Value: 0.010052802581676538

Nilai skewness: -0.0014241325160359258
Nilai kurt: -0.2989944824022248
```



In [ ]:
```
display(Markdown("#### Jenis distribusi tidak dapat ditentukan namun mendekati dist
test_distribution(data_column_numeric[5])
```

Jenis distribusi tidak dapat ditentukan namun mendekati distribusi normal secara visual

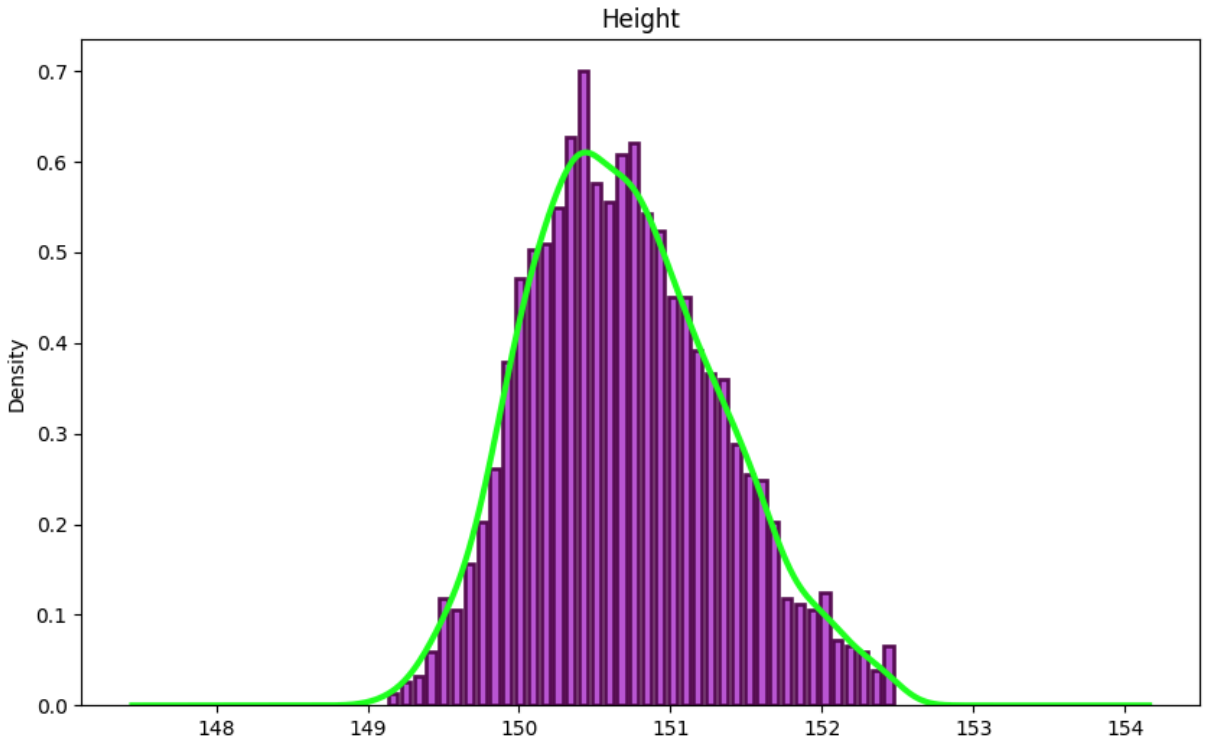
```
Nilai p-Value: 0.0024892107338937335

Nilai skewness: -0.07096847696081976
Nilai kurt: -0.31521889642576584
```



In [ ]:
```
display(Markdown("#### Jenis distribusi tidak dapat ditentukan"))
test_distribution(data_column_numeric[6])
```

Jenis distribusi tidak dapat ditentukan

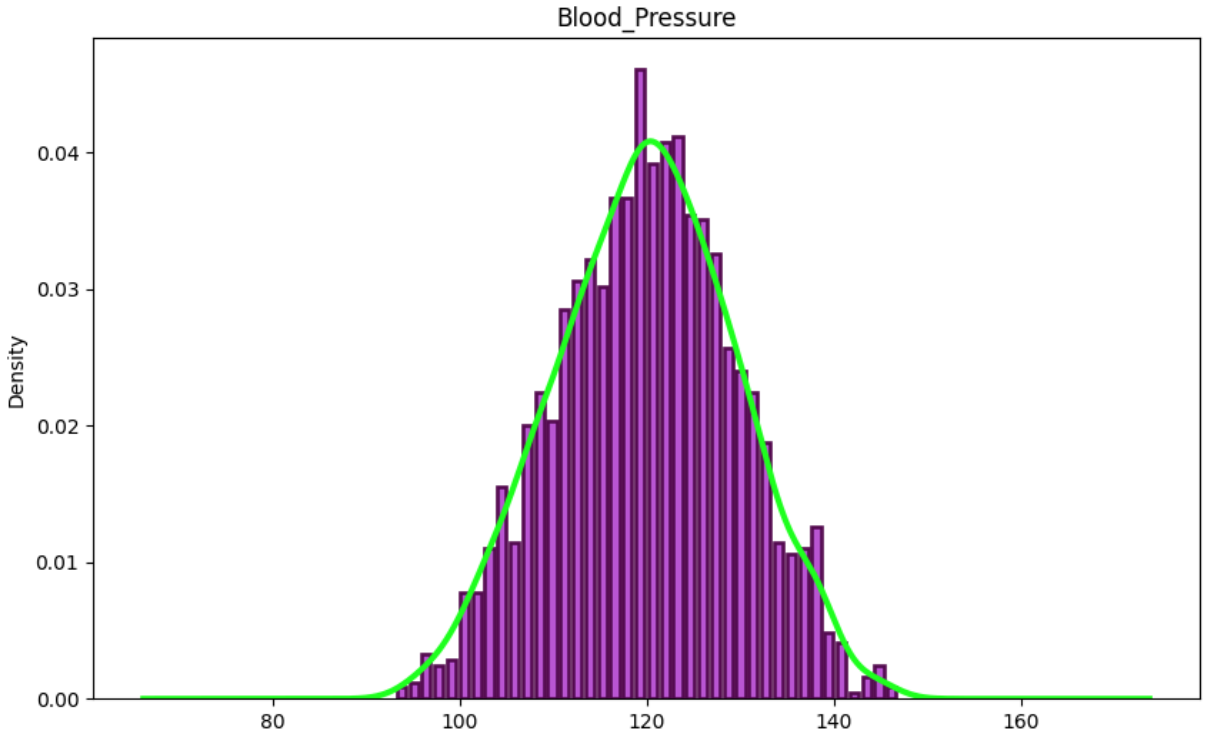
```
Nilai p-Value: 1.078574664970807e-09

Nilai skewness: 0.3469503974648352
Nilai kurt: -0.2584809097530343
```



In [ ]:
```
display(Markdown("### Distribusi Normal"))
test_distribution(data_column_numeric[7])
```

Distribusi Normal

```
Nilai p-Value: 0.002190937754804233

Nilai skewness: -0.03354246196371709
Nilai kurt: -0.3323625783265225
```



In [ ]:
```
test_distribution(data_column_string[0])
```

Kolom Gender Memiliki Tipe Data String

Sehingga tidak berdistribusi

In [ ]:
```
test_distribution(data_column_string[1])
```

Kolom Education Memiliki Tipe Data String

Sehingga tidak berdistribusi

In [ ]:
```
test_distribution(data_column_string[2])
```

Kolom Health_Status Memiliki Tipe Data String

Sehingga tidak berdistribusi

Soal khusus dataset

Mira adalah seorang peneliti kesehatan yang bekerja di sebuah lembaga riset medis yang terkemuka. Sebagai bagian dari tugasnya, Mira memiliki akses ke sebuah dataset yang berisi informasi tentang profil kesehatan dan gaya hidup dari sejumlah individu. Mira bertanggung jawab untuk melakukan analisis statistika terhadap dataset ini guna mendapatkan pemahaman yang lebih baik tentang faktor-faktor yang mempengaruhi kesehatan dan kualitas hidup individu. Selain itu, Mira juga diminta untuk mengidentifikasi pola dan

hubungan yang signifikan antara variabel-variabel tersebut, serta untuk menjawab berbagai pertanyaan penelitian yang diajukan oleh lembaga riset.

Atribut: Age, Income, Gender, Education, Stress_Level, Exercise_Hours_Per_Week, Cholesterol_Level, Weight,Height, Blood_Pressure, Health_Status

Gunakan alpha = 0.05

Nomor 5: Hipotesis 1 sampel

- Lembaga riset saat ini sedang mempertanyakan data berat badan individu yang disimpan untuk kepentingan riset lanjutan. Identifikasilah apakah rata-rata berat badan pasien diatas 65 kg?
- Tekanan darah sistole yang normal berada pada rentang 120 mmHg. Lembaga riset perlu untuk memastikan apakah data individu yang diukur cukup normal. Periksalah apakah rata-rata tekanan darah sistole bernilai 120 mmHg?
- Periksalah apakah data 200 individu pertama pengujian (baris teratas) memiliki rata-rata waktu olahraga per minggu tidak sama dengan 15 jam?
- Apakah penduduk dengan pendapatan yang lebih besar dari Rp 7.500.000,00 tidak sama dengan 30% dari data keseluruhan individu?

Inisialisasi Fungsi Single Value

In [ ]:
```python
# Definisi uji dengan t-test
# t-test digunakan saat tabel berdistribusi normal dan hanya diketahui standar devi
def calculateT(xbar: float, mu: float, n: int, deviation: float) -> float:
    """
    Menghitung nilai-t dari data
    Parameters
    ----------
        xbar: mean of sample with size n
        mu: real mean of population
        n: size of the sample
        deviation: population standard deviation
    Return:
        t-value of data with population size n and xbar
    """
    return (xbar - mu) / (deviation / math.sqrt(n))


def findTValue(value: float, freedomDegree: int) -> float:
    """
    Given the P-value, find the T-value
    """
    return stats.t.ppf(value, freedomDegree)

def findZValue(value: float) -> float:
    """
    Given the P-value, find the Z-value
```

```
    """
    return stats.norm.ppf(value)

def calculatePValueZLeft(zValue: float) -> float:
    """
    Given Z-Value, calculate the P-value to the left
    """
    return stats.norm.cdf(zValue)

def calculatePValueZRight(zValue: float) -> float:
    """
    Given z-value, calculate P-value to the right
    """
    return stats.norm.sf(zValue)

def calculatePValueTLeft(tValue: float, freedomDegree: int) -> float:
    """
    Given T-value, calculate the P-value to the left

    Args:
        tValue (float): T-value
        freedomDegree (int): degrees of freedom (length - 1)
    """
    return stats.t.cdf(abs(tValue), freedomDegree)

def calculatePValueTRight(tValue: float, freedomDegree: int) -> float:
    return stats.t.sf(tValue, freedomDegree)
```

Pertanyaan 5.1

Lembaga riset saat ini sedang mempertanyakan data berat badan individu yang disimpan untuk kepentingan riset lanjutan. Identifikasilah apakah rata-rata berat badan pasien diatas 65 kg?

In [ ]:
```
# Inisialisasi data berat badan
weight_data = result_data[data_column_numeric[5]]

# 1. Definisi hipotesis 0 H0 saat rata-rata = 65, menjadi nilai Mu
mu = 65
n = len(weight_data)
display(Markdown("#### 1. Definisi Hipotesis 0"))
display(Markdown(f"Hipotesis 0 (H0) saat rata-rata (μ) = {mu}"))

# 2. Definisi hipotesis alternatif H1 saat rata-rata > 65
display(Markdown("#### 2. Definisi Hipotesis 1"))
display(Markdown(f"Hipotesis 1 (H1) saat rata-rata (μ) > {mu}"))

# 3. Tingkat signifikan α yang digunakan adalah 0.05.
alpha = 0.05
display(Markdown("#### 3. Tingkat Signifikan α"))
display(Markdown(f"Digunakan tingkat signifikan α = {alpha}"))

# 4. Tentukan uji statistik yang sesuai dan tentukan daerah kritis.
# Tes uji statistik yang digunakan adalah menggunakan t-test, karena yang digunakan
```

```
t_critical = findTValue(1 - alpha, len(weight_data) - 1)
display(Markdown("#### 4. Penentuan Uji Statistik dan Daerah Kritis"))
display(Markdown("Uji statistik yang digunakan adalah 1-tailed t-test karena nilai
display(Markdown(f"Critical region untuk α > 0.05 berada di nilai t > {t_critical}"


# 5. Hitung nilai uji statistik dari data sample. Hitung p-value sesuai dengan uji
# Jika berdasarkan uji tes nilai peluang masuk ke daerah kritis, maka tolak H0, jik
#
t_value = calculateT(np.mean(weight_data), mu, len(weight_data), np.std(weight_data
p_value = calculatePValueTRight(t_value, n-1)
display(Markdown("#### 5. Uji Statistik"))
display(Markdown(f"Didapatkan nilai t = {t_value}"))
display(Markdown(f"Nilai P untuk t = {t_value} = {p_value}"))


"""
6. Ambil keputusan dengan TOLAK H0 jika nilai uji terletak di daerah kritis atau de
signifikan, TOLAK H0 jika p-value lebih kecil dibanding tingkat signifikansi α yang
diinginkan
"""
display(Markdown("#### 6. Pengambilan Keputusan"))
if t_value > t_critical:
    display(Markdown(f"Tolak H0: rata-rata berat badan pasien lebih dari 65 kg ({np
else:
    display(Markdown(f"Gagal tolak H0: rata-rata berat badan pasien tidak lebih dar

if p_value > alpha:
    display(Markdown(f"Tolak keputusan untuk menolak H0"))
else:
    display(Markdown(f"Dukung keputusan untuk menolak H0 karena {p_value} < {0.05},

# Perbandingan dengan fungsi bawaan
display(Markdown("#### Nilai Fungsi Bawaan"))
builtin_t, builtin_p = stats.ttest_1samp(weight_data, mu)
display(Markdown(f"Dengan menggunakan fungsi bawaan, didapatkan t-value = {builtin_
```

1. Definisi Hipotesis 0

Hipotesis 0 (H0) saat rata-rata (μ) = 65

2. Definisi Hipotesis 1

Hipotesis 1 (H1) saat rata-rata (μ) > 65

3. Tingkat Signifikan α

Digunakan tingkat signifikan α = 0.05

4. Penentuan Uji Statistik dan Daerah Kritis

Uji statistik yang digunakan adalah 1-tailed t-test karena nilai σ tidak terdefinisi

Critical region untuk α > 0.05 berada di nilai t > 1.645690831425319

5. Uji Statistik

Didapatkan nilai t = 22.558561651964325

Nilai P untuk t = 22.558561651964325 = 7.140546130005205e-100

6. Pengambilan Keputusan

Tolak H0: rata-rata berat badan pasien lebih dari 65 kg (70.14635405301912), dengan pengujian tes didapatkan t_critical > t_value = 22.558561651964325 > 1.645690831425319

Dukung keputusan untuk menolak H0 karena 7.140546130005205e-100 < 0.05, sehingga rata-rata berat badan adalah benar di atas 65

Nilai Fungsi Bawaan

Dengan menggunakan fungsi bawaan, didapatkan t-value = 22.55237019812136 dan p-value = 1.593161239989148e-99

Pertanyaan 5.2

Tekanan darah sistole yang normal berada pada rentang 120 mmHg. Lembaga riset perlu untuk memastikan apakah data individu yang diukur cukup normal. Periksalah apakah rata-rata tekanan darah sistole bernilai 120 mmHg?

In [ ]:
```
# H0 : mu = 120
# H1 : mu != 120
sistole = result_data[data_column_numeric[7]]

# 1. Definisi hipotesis 0
mu = 120
display(Markdown("#### 1. Definisi Hipotesis 0"))
display(Markdown(f"Hipotesis 0 (H0) saat rata-rata (μ) = {mu}"))

# 2. Definisi hipotesis 1
xbar = np.mean(sistole)
deviation = np.std(sistole)
n = len(sistole)
display(Markdown("#### 2. Definisi Hipotesis 1"))
display(Markdown(f"Hipotesis 1 (H1) saat rata-rata (μ) ≠ {mu}"))

# 3. Tingkat signifikansi
alpha = 0.05
display(Markdown("#### 3. Tingkat Signifikan α"))
display(Markdown(f"Digunakan tingkat signifikan α = {alpha}"))

# 4. Penentuan jenis tes
t_critical_up = findTValue(1 - alpha / 2, n - 1)
t_critical_down = findTValue(alpha / 2, n - 1)
display(Markdown("#### 4. Penentuan Uji Statistik dan Daerah Kritis"))
display(Markdown("Uji statistik yang digunakan adalah 2-tailed t-test karena nilai
display(Markdown(f"Critical region untuk α > 0.05 berada di nilai t > {t_critical_u
display(Markdown(f"Didapat nilai rata-rata data: {xbar}, s: {deviation}, panjang da

# 5. Uji statistik
t_value = calculateT(xbar, mu, n, deviation)
```

```
p_value = 2 * calculatePValueTRight(abs(t_value), n - 1)
display(Markdown("#### 5. Uji Statistik"))
display(Markdown(f"Didapat nilai t = {t_value}"))
display(Markdown(f"Nilai P = {p_value}"))


# 6. Pengambilan keputusan
display(Markdown("#### 6. Pengambilan Keputusan"))
if t_value < (t_critical_down) or t_value > t_critical_up or p_value < 0.05:
    display(Markdown("Tolak H0"))
else:
    display(Markdown(f"Gagal tolak H0: nilai t berada di antara {t_critical_down} <

# 7. Perbandingan dengan Fungsi Bawaan
display(Markdown("#### Nilai Fungsi Bawaan"))
builtin_t, builtin_p = stats.ttest_1samp(sistole, mu)
display(Markdown(f"Nilai fungsi bawaan: t = {builtin_t} dan p = {builtin_p}"))
```

1. Definisi Hipotesis 0

Hipotesis 0 (H0) saat rata-rata (μ) = 120

2. Definisi Hipotesis 1

Hipotesis 1 (H1) saat rata-rata (μ) ≠ 120

3. Tingkat Signifikan α

Digunakan tingkat signifikan α = 0.05

4. Penentuan Uji Statistik dan Daerah Kritis

Uji statistik yang digunakan adalah 2-tailed t-test karena nilai σ tidak terdefinisi

Critical region untuk α > 0.05 berada di nilai t > 1.96126756607388 atau t < -1.9612675660738805

Didapat nilai rata-rata data: 119.94907242143402, s: 9.61402822866432, panjang data: 1822

5. Uji Statistik

Didapat nilai t = -0.2261110677224003

Nilai P = 0.8211404121554984

6. Pengambilan Keputusan

Gagal tolak H0: nilai t berada di antara -1.9612675660738805 < -0.2261110677224003 < 1.96126756607388 dan p-value = 0.8211404121554984 >= 0.05 sehingga rata-rata sistole berada di rentang 120 mmHg, yaitu 119.94907242143402

Nilai Fungsi Bawaan

Nilai fungsi bawaan: t = -0.22604900896791114 dan p = 0.8211886711961978

Pertanyaan 5.3

Periksalah apakah data 200 individu pertama pengujian (baris teratas) memiliki rata-rata waktu olahraga per minggu tidak sama dengan 15 jam?

In [ ]:
```
n = 200
sport_200 = result_data[numeric_columns[3]].head(n)
mu = 15
xbar = np.mean(sport_200)
deviation = np.std(sport_200)
# 1. H0 : mu = 15
display(Markdown("#### 1. Definisi Hipotesis 0"))
display(Markdown(f"Hipotesis 0 (H0) saat rata-rata (μ) = {mu}"))

# 2. H1 : mu != 15
display(Markdown("#### 2. Definisi Hipotesis 1"))
display(Markdown(f"Hipotesis 1 (H1) saat rata-rata (μ) ≠ {mu}"))

# 3. Signifikansi
alpha = 0.05
display(Markdown("#### 3. Tingkat Signifikan α"))
display(Markdown(f"Digunakan tingkat signifikan α = {alpha}"))

# 4. Penentuan Pengujian
t_critical_up = findTValue(1 - alpha / 2, n - 1)
t_critical_down = findTValue(alpha / 2, n - 1)
t_value = calculateT(xbar, mu, n, deviation)
display(Markdown("#### 4. Penentuan Uji Statistik dan Daerah Kritis"))
display(Markdown("Uji statistik yang digunakan adalah 2-tailed t-test karena tidak
display(Markdown(f"Critical region untuk α > 0.05 berada di nilai t > {t_critical_u
display(Markdown(f"Didapat nilai rata-rata data: {xbar}, s: {deviation}, panjang da

# 5. Uji statistik
p_value = 2 * calculatePValueTRight(abs(t_value), n - 1)
display(Markdown("#### 5. Uji Statistik"))
display(Markdown(f"Didapat nilai t = {t_value}"))
display(Markdown(f"Nilai P = {p_value}"))
builtin_t, builtin_p = stats.ttest_1samp(sport_200, mu)

# 6. Pengambilan Keputusan
display(Markdown("#### 6. Ambil Keputusan: Tolak H0"))
if t_value < t_critical_down or t_value > t_critical_up or p_value < alpha:
    display(Markdown(f"Tolak H0 karena {p_value} < {alpha}: Rata-rata tiap orang ti
else:
    display(Markdown("Gagal tolak H0: Rata-rata tiap orang berolahraga per minggu s

display(Markdown("#### Nilai Fungsi Bawaan"))
display(Markdown(f"Nilai fungsi bawaan: t = {builtin_t} dan p = {builtin_p}"))
```

1. Definisi Hipotesis 0

Hipotesis 0 (H0) saat rata-rata (μ) = 15

2. Definisi Hipotesis 1

Hipotesis 1 (H1) saat rata-rata (μ) ≠ 15

3. Tingkat Signifikan α

Digunakan tingkat signifikan α = 0.05

4. Penentuan Uji Statistik dan Daerah Kritis

Uji statistik yang digunakan adalah 2-tailed t-test karena tidak ada standar deviasi sampel

Critical region untuk α > 0.05 berada di nilai t > 1.971956544249395 atau t < -1.9719565442493954

Didapat nilai rata-rata data: 9.893257018339058, s: 4.468064149936544, panjang data: 200

5. Uji Statistik

Didapat nilai t = -16.163655985827976

Nilai P = 4.3652198224188466e-38

6. Ambil Keputusan: Tolak H0

Tolak H0 karena 4.3652198224188466e-38 < 0.05: Rata-rata tiap orang tidak berolahraga per minggu selama 15 jam (9.893257018339058 jam), dengan nilai t < critical = -16.163655985827976 < -1.9719565442493954

Nilai Fungsi Bawaan

Nilai fungsi bawaan: t = -16.123196207763883 dan p = 5.798139207494498e-38

Pertanyaan 5.4

Apakah penduduk dengan pendapatan yang lebih besar dari Rp 7.500.000,00 tidak sama dengan 30% dari data keseluruhan individu?

In [ ]:
```
from statsmodels.stats.proportion import proportions_ztest
# H0 : p = 0.3
# H1 : p != 0.3

# 1. Definisi H0
p0 = 0.3
display(Markdown("#### 1. Definisi Hipotesis 0"))
display(Markdown(f"Hipotesis 0 (H0) saat proporsi p = {p0}"))

# Inisialisasi data
n = len(result_data[numeric_columns[1]])
income_7500 = result_data[result_data[numeric_columns[1]] > 7500000]

# 2. Definisi H1
display(Markdown("#### 2. Definisi Hipotesis 1"))
display(Markdown(f"Hipotesis 1 (H1) saat proporsi p ≠ {p0}"))

# 3. Definisi siginifikansi
display(Markdown("#### 3. Definisi Signifikansi"))
display(Markdown(f"Digunakan tingkat signifikan α = {alpha}"))
```

```
# 4. Penentuan Pengujian
z_critical_up = findZValue(1 - alpha / 2)
z_critical_down = findZValue(alpha / 2)
display(Markdown("#### 4. Penentuan Uji Statistik dan Daerah Kritis"))
display(Markdown("Uji statistik yang digunakan adalah 2-tailed z-test dengan menggu
display(Markdown(f"Critical region untuk α > 0.05 berada di nilai z > {z_critical_u

# 5. Uji Statistik
p = len(income_7500) / n
z_value = (p - p0) / math.sqrt(p * (1 - p0) / math.sqrt(n))
p_value = 2 * calculatePValueZLeft(z_value)
display(Markdown("#### 5. Uji Statistik"))
display(Markdown(f"Didapat nilai z = {z_value}"))
display(Markdown(f"Didapat nilai p = {p_value}"))
builtin_t, builtin_p = proportions_ztest(count=len(income_7500), nobs=n, value=0.3,

# 6. Kesimpulan
display(Markdown("#### 6. Pengambilan Kesimpulan"))
if z_value < z_critical_down or z_value > z_critical_up:
    display(Markdown(f"H0 ditolak: nilai z < {z_critical_down} dan p = {p_value} <
else:
    display(Markdown("H0 gagal ditolak"))

display(Markdown("#### Nilai Fungsi Bawaan"))
display(Markdown(f"Nilai fungsi bawaan: z = {builtin_t} dan p = {builtin_p}"))
```

1. Definisi Hipotesis 0

Hipotesis 0 (H0) saat proporsi p = 0.3

2. Definisi Hipotesis 1

Hipotesis 1 (H1) saat proporsi p ≠ 0.3

3. Definisi Signifikansi

Digunakan tingkat signifikan α = 0.05

4. Penentuan Uji Statistik dan Daerah Kritis

Uji statistik yang digunakan adalah 2-tailed z-test dengan menggunakan uji proporsi

Critical region untuk α > 0.05 berada di nilai z > 1.959963984540054 atau z < -1.9599639845400545

5. Uji Statistik

Didapat nilai z = -5.496657581876371

Didapat nilai p = 3.870572257687624e-08

6. Pengambilan Kesimpulan

H0 ditolak: nilai z < -1.9599639845400545 dan p = 3.870572257687624e-08 < 0.05, sehingga penduduk dengan pendapatan di atas 7.500.000 tidak sama dengan 30% populasi

Nilai Fungsi Bawaan

Nilai fungsi bawaan: z = -31.487326653662826 dan p = 1.2952616255233202e-217

Nomor 6: Hipotesis 2 sampel

Lembaga riset membagi data individu menjadi dua bagian, yaitu data individu yang lebih awal masuk data penelitian (bagian atas) dan yang baru saja (bagian bawah).

- Periksa apakah rata-rata berat badan individu yang lebih awal masuk data penelitian sama dengan rata-rata berat badan individu yang masuk baru saja?
- Bagaimana dengan pendapatan individu, apakah pendapatan sistole individu yang lebih awal masuk data penelitian lebih besar Rp 1.250.000,00 dari yang baru saja masuk?
- Lembaga riset ingin membandingkan kondisi kesehatan individu dari dua bagian data. Apakah variansi tekanan darah individu yang lebih awal masuk data penelitian sama dengan yang baru saja masuk?
- Apakah proporsi waktu olahraga yang lebih dari 8 jam per minggu pada data individu awal lebih besar daripada kuantitas proporsi pada data individu akhir dengan waktu olahraga yang sama?

Inisialisasi Fungsi 2 Sampel

In [ ]:
```
# Inisialisasi
n = len(result_data)

data_top = result_data.head(n // 2)
n_top = len(data_top)
data_bot = result_data.iloc[n // 2:]
n_bot = len(data_bot)

def calculateF(var1: float, var2: float) -> float:
    """
    Given two variances, find the F-value
    f = s1^2 / s2^2
    """
    return var1/var2

def findFValue(p_value: float, numerator_degree: int, denom_degree: int) -> float:
    """
    Find F-value, given P-value and degrees of freedom
    Args:
        p_value (float)
        numerator_degree (int)
        denom_degree (int)

    Returns:
        F-value
    """
    return stats.f.ppf(p_value, numerator_degree, denom_degree)

def calculatePValueF(f_value: float, numerator_degree: int, denom_degree: int) -> f
```

```
    """Calculate P-value from given F-value

    Args:
        f_value (float): _description_
        numerator_degree (int): _description_
        denom_degree (int): _description_

    Returns:
        P-value
    """
    return stats.f.cdf(f_value, numerator_degree, denom_degree)
```

Pertanyaan 6.1

Periksa apakah rata-rata berat badan individu yang lebih awal masuk data penelitian sama dengan rata-rata berat badan individu yang masuk baru saja?

In [ ]:
```
from statsmodels.stats.weightstats import ztest
# mu1 - mu2 = d0, d0 = 0
mu1 = np.mean(data_top[numeric_columns[5]])
mu2 = np.mean(data_bot[numeric_columns[5]])

# 1. Definisi hipotesis 0
# H0 : mu1 - mu2 = d0
display(Markdown("#### 1. Definisi Hipotesis 0"))
display(Markdown(f"Hipotesis 0 (H0) saat μ1 - μ2 = d0, dengan d0 = 0"))

# 2. Definisi hipotesis 1
# H1 : mu1 - mu2 != d0
display(Markdown("#### 2. Definisi Hipotesis 1"))
display(Markdown(f"Hipotesis 0 (H0) saat μ1 - μ2 ≠ d0, dengan d0 = 0"))

# 3. Signifikansi
alpha = 0.05
display(Markdown("#### 3. Tingkat Signifikan α"))
display(Markdown(f"Digunakan tingkat signifikan α = {alpha}"))
z_critical_down = findZValue(alpha / 2)
z_critical_up = findZValue(1 - alpha / 2)
display(Markdown(f"Daerah kritis: z < {z_critical_down} dan z > {z_critical_up}"))

# 4. Penentuan Pengujian
n1 = len(data_top[numeric_columns[5]])
n2 = len(data_bot[numeric_columns[5]])
std1 = np.std(data_top[numeric_columns[5]])
std2 = np.std(data_bot[numeric_columns[5]])
display(Markdown("#### 4. Penentuan Uji Statistik dan Daerah Kritis"))
display(Markdown(f"Pengujian menggunakan 2-tailed z-test untuk 2 sampel karena stan

# 5. Uji Statistik
display(Markdown("#### 5. Uji Statistik"))
d = {'Top': pd.Series([mu1, std1, n1], index=['μ', 'σ', 'n']),
     'Bot': pd.Series([mu2, std2, n2], index=['μ', 'σ', 'n'])}
print(pd.DataFrame(d))
z_value = (mu1 - mu2 - 0) / math.sqrt((std1 * std1 / n1) + (std2 * std2 / n2))
```

```
display(Markdown(f"Didapat nilai z = {z_value}"))
p_value = 2 * calculatePValueZLeft(z_value)
builtin_z, builtin_p = ztest(data_top[numeric_columns[5]], data_bot[numeric_columns

# 6. Pengambilan Keputusan
display(Markdown("#### 6. Pengambilan Keputusan"))
if z_value < z_critical_down or z_value > z_critical_up or p_value < alpha:
    print("Tolak H0")
else:
    display(Markdown(f"Gagal tolak H0: Nilai z tidak masuk critical value ({z_criti

display(Markdown("#### Nilai Fungsi Bawaan"))
display(Markdown(f"Fungsi bawaan: z = {builtin_z} dan p = {builtin_p}"))
```

1. Definisi Hipotesis 0

Hipotesis 0 (H0) saat μ1 - μ2 = d0, dengan d0 = 0

2. Definisi Hipotesis 1

Hipotesis 0 (H0) saat μ1 - μ2 ≠ d0, dengan d0 = 0

3. Tingkat Signifikan α

Digunakan tingkat signifikan α = 0.05

Daerah kritis: z < -1.9599639845400545 dan z > 1.959963984540054

4. Penentuan Uji Statistik dan Daerah Kritis

Pengujian menggunakan 2-tailed z-test untuk 2 sampel karena standar deviasi ditemukan

5. Uji Statistik

	Top	Bot
μ	70.280845	70.011863
σ	9.539709	9.930189
n	911.000000	911.000000

Didapat nilai z = 0.5895853167853029

6. Pengambilan Keputusan

Gagal tolak H0: Nilai z tidak masuk critical value (-1.9599639845400545 < 0.5895853167853029 < 1.959963984540054) sehingga rata-rata awal dekat dengan rata-rata akhir, dan p = 1.444531301913968 >= 0.05

Nilai Fungsi Bawaan

Fungsi bawaan: z = 0.5892616355548761 dan p = 0.5556857763174801

Pertanyaan 6.2

Bagaimana dengan pendapatan individu, apakah pendapatan individu yang lebih awal masuk data penelitian lebih besar Rp 1.250.000,00 dari yang baru saja masuk?

In [ ]:
```
# mu1 - mu2 = d0, d0 = 1250000
# H0 : mu1 - mu2 = d0
# H1 : mu1 - mu2 ≠ d0

# 1. Penentuan hipotesis 0
d0 = 1250000
display(Markdown("#### 1. Definisi Hipotesis 0"))
display(Markdown(f"H0 saat μ1 - μ2 = d0, dengan d0 = {d0}"))

# 2. Definisi hipotesis 1
# H1 : mu1 - mu2 != d0
display(Markdown("#### 2. Definisi Hipotesis 1"))
display(Markdown(f"Hipotesis 0 (H0) saat μ1 - μ2 ≠ d0, dengan d0 = {d0}"))

# 3. Signifikansi
alpha = 0.05
display(Markdown("#### 3. Tingkat Signifikan α"))
display(Markdown(f"Digunakan tingkat signifikan α = {alpha}"))

# 4. Penentuan Pengujian
mu1 = np.mean(data_top[numeric_columns[1]])
mu2 = np.mean(data_bot[numeric_columns[1]])
std1 = np.std(data_top[numeric_columns[1]])
std2 = np.std(data_bot[numeric_columns[1]])
n1 = len(data_top[numeric_columns[1]])
n2 = len(data_bot[numeric_columns[1]])
display(Markdown("#### 4. Penentuan Uji Statistik dan Daerah Kritis"))
display(Markdown(f"Pengujian menggunakan 2-tailed z-test untuk 2 sampel karena stan
z_critical_down = findZValue(alpha / 2)
z_critical_up = findZValue(1 - alpha / 2)

# 5. Uji Statistik
display(Markdown("#### 5. Pengambilan Keputusan"))
d = {'Top': pd.Series([mu1, std1, n1], index=['μ', 'σ', 'n']),
     'Bot': pd.Series([mu2, std2, n2], index=['μ', 'σ', 'n'])}
print(pd.DataFrame(d))
z_value = (mu1 - mu2 - d0) / math.sqrt((std1 * std1 / n1) + (std2 * std2 / n2))
display(Markdown(f"Didapatkan nilai z = {z_value}"))
p_value = calculatePValueZLeft(z_value) * 2
display(Markdown(f"Nilai p-value = {p_value}"))
builtin_z, builtin_p = ztest(data_top[numeric_columns[1]], data_bot[numeric_columns

# 6. Pengambilan Keputusan
display(Markdown("#### 6. Pengambilan Keputusan"))
if z_value > z_critical_up or z_value < z_critical_down or p_value < alpha:
    display(Markdown(f"Tolak H0: Nilai z = {z_value} < {z_critical_down}, sehingga
else:
    print("Gagal tolak H0")

display(Markdown("#### Nilai Fungsi Bawaan"))
display(Markdown(f"Fungsi bawaan: z-value = {builtin_z} dan p-value = {builtin_p}")
```

1. Definisi Hipotesis 0

H0 saat μ1 - μ2 = d0, dengan d0 = 1250000

2. Definisi Hipotesis 1

Hipotesis 0 (H0) saat μ1 - μ2 ≠ d0, dengan d0 = 1250000

3. Tingkat Signifikan α

Digunakan tingkat signifikan α = 0.05

4. Penentuan Uji Statistik dan Daerah Kritis

Pengujian menggunakan 2-tailed z-test untuk 2 sampel karena standar deviasi ditemukan

5. Pengambilan Keputusan

	Top	Bot
μ	4.915064e+06	4.893987e+06
σ	1.898291e+06	1.950155e+06
n	9.110000e+02	9.110000e+02

Didapatkan nilai z = -13.62931917341134

Nilai p-value = 2.6806946343730382e-42

6. Pengambilan Keputusan

Tolak H0: Nilai z = -13.62931917341134 < -1.9599639845400545, sehingga pendapatan individu yang lebih awal (4915063.997196051) tidak lebih besar dari 1.250.000 dibandingkan individu data bawah (4893987.497927514), dibuktikan dengan p-value = 2.6806946343730382e-42 < 0.05

Nilai Fungsi Bawaan

Fungsi bawaan: z-value = 0.23361961190588593 dan p-value = 0.8152803009213743

Pertanyaan 6.3

Lembaga riset ingin membandingkan kondisi kesehatan individu dari dua bagian data. Apakah variansi tekanan darah individu yang lebih awal masuk data penelitian sama dengan yang baru saja masuk?

In [ ]:
```
# var1 - var2 = d0, d0 = 0

# 1. Definisi hipotesis 0
# H0 : var1 - var2 = 0
display(Markdown("#### 1. Definisi Hipotesis 0"))
display(Markdown(f"Hipotesis 0 (H0) saat Var1 - Var2 = d0, dengan d0 = 0"))
var1 = np.var(data_top[numeric_columns[7]])
var2 = np.var(data_bot[numeric_columns[7]])

# 2. Definisi hipotesis 1
# H1 : var1 - var2 != 0
display(Markdown("#### 2. Definisi Hipotesis 1"))
display(Markdown(f"Hipotesis 0 (H0) saat Var1 - Var2 ≠ d0, dengan d0 = 0"))

# 3. Definisi signifikansi
```

```
alpha = 0.05
display(Markdown("#### 3. Tingkat Signifikan α"))
display(Markdown(f"Digunakan tingkat signifikan α = {alpha}"))

# 4. Penentuan pengujian
# Digunakan f-distribution test
display(Markdown("#### 4. Penentuan Pengujian"))
display(Markdown("Digunakan pengujian dengan f-distribution test karena membandingk
f_critical_low = findFValue(alpha / 2, n_top - 1, n_bot - 1)
f_critical_top = findFValue(1 - alpha / 2, n_top - 1, n_bot - 1)
display(Markdown(f"Critical region: f < {f_critical_low} atau f > {f_critical_top}"

# 5. Uji statistik
display(Markdown("#### 5. Pengujian Statistik"))
f = calculateF(float(var1), float(var2))
display(Markdown(f"Dengan varians atas = {var1} dan varians bawah = {var2}, didapat
p_value = calculatePValueF(f, len(data_top)-1, len(data_bot)-1)
display(Markdown(f"Nilai p-value = {p_value}"))
bart = stats.bartlett(data_top[numeric_columns[7]], data_bot[numeric_columns[7]])
builtin_t = bart.statistic
builtin_z = bart.pvalue

# 6. Pengambilan keputusan
display(Markdown("#### 6. Pengambilan Keputusan"))
if f < f_critical_low or f > f_critical_top or p_value < alpha:
    print("H0 ditolak")
else:
    display(Markdown(f"H0 gagal ditolak: karena nilai f ada pada range {f_critical_

display(Markdown("#### Nilai Fungsi Bawaan"))
display(Markdown(f"Nilai fungsi bawaan: z-value = {builtin_z} dan p-value = {builti
```

1. Definisi Hipotesis 0

Hipotesis 0 (H0) saat Var1 - Var2 = d0, dengan d0 = 0

2. Definisi Hipotesis 1

Hipotesis 0 (H0) saat Var1 - Var2 ≠ d0, dengan d0 = 0

3. Tingkat Signifikan α

Digunakan tingkat signifikan α = 0.05

4. Penentuan Pengujian

Digunakan pengujian dengan f-distribution test karena membandingkan varians dengan varians lainnya

Critical region: f < 0.8780727827211063 atau f > 1.1388577572135274

5. Pengujian Statistik

Dengan varians atas = 95.94057672442094 dan varians bawah = 88.6856436739666, didapatkan nilai f = 1.0818050447615346

Nilai p-value = 0.8820942075865073

6. Pengambilan Keputusan

H0 gagal ditolak: karena nilai f ada pada range 0.8780727827211063 < 1.0818050447615346 < 1.1388577572135274 dan p-value = 0.8820942075865073 >= 0.05, sehingga variansi tekanan darah antara data awal (95.94057672442094) dan data akhir (88.6856436739666) tidak jauh berbeda

Nilai Fungsi Bawaan

Nilai fungsi bawaan: z-value = 0.23581156609245668 dan p-value = 0.23581156609245668

Pertanyaan 6.4

Apakah proporsi waktu olahraga yang lebih dari 8 jam per minggu pada data individu awal lebih besar daripada kuantitas proporsi pada data individu akhir dengan waktu olahraga yang sama?

In [ ]:
```
# p1 - p2 = 8
# H1 : p1 - p2 != 0

# 1. Definisi hipotesis 0
# H0 : p1 - p2 = 0
display(Markdown("#### 1. Definisi Hipotesis 0"))
display(Markdown(f"H0 saat p1 - p2 = d0, dengan d0 = 0"))

# 2. Definisi hipotesis 1
display(Markdown("#### 2. Definisi Hipotesis 1"))
display(Markdown(f"Hipotesis 1 (H1) saat p1 - p2 ≠ 0"))

# 3. Definisi signifikansi
alpha = 0.05
display(Markdown("#### 3. Tingkat Signifikan α"))
display(Markdown(f"Digunakan tingkat signifikan α = {alpha}"))

# 4. Penentuan Pengujian
data_top_exercise = data_top[data_top[numeric_columns[3]] > 8]
data_bot_exercise = data_bot[data_bot[numeric_columns[3]] > 8]
display(Markdown("#### 4. Penentuan Pengujian dan Daerah Kritis"))
display(Markdown(f"Pengujian menggunakan tes proporsi untuk 2 sampel dengan menggun
z_critical_up = findZValue(1 - alpha / 2)
z_critical_down = findZValue(alpha / 2)
display(Markdown(f"Daerah kritis: z < {z_critical_down} atau z > {z_critical_up}"))

# 5. Uji Statistik
display(Markdown("#### 5. Uji Statistik"))
n1 = len(data_top)
n2 = len(data_bot)
x1 = len(data_top_exercise)
x2 = len(data_bot_exercise)
p1 = x1 / n1
p2 = x2 / n2
d = {'Top': pd.Series([p1, x1, n1], index=['p', 'x', 'n']),
     'Bot': pd.Series([p2, x2, n2], index=['p', 'x', 'n'])}
```

```
print(pd.DataFrame(d))

p = (x1 + x2) / (n1 + n2)
z_value = (p1 - p2) / math.sqrt(p * (1 - p) * (1/n1 + 1/n2))
display(Markdown(f"Didapatkan z-value = {z_value}"))
p_value = calculatePValueZLeft(z_value)
display(Markdown(f"Didapatkan p-value = {p_value}"))
builtin_z, builtin_p = proportions_ztest(count=[x1, x2], nobs=[n1, n2], alternative

# 6. Penentuan Keputusan
display(Markdown("#### 6. Pengambilan Keputusan"))
if z_value < z_critical_down or z_value > z_critical_up or p_value < alpha:
    display(Markdown(f"H0 ditolak "))
else:
    display(Markdown(f"H0 gagal ditolak: nilai z ada dalam range {z_critical_down}

display(Markdown("#### Nilai Fungsi Bawaan"))
display(Markdown(f"Fungsi builtin: p-value = {builtin_p} dan z-value = {builtin_z}"
```

1. Definisi Hipotesis 0

H0 saat p1 - p2 = d0, dengan d0 = 0

2. Definisi Hipotesis 1

Hipotesis 1 (H1) saat p1 - p2 ≠ 0

3. Tingkat Signifikan α

Digunakan tingkat signifikan α = 0.05

4. Penentuan Pengujian dan Daerah Kritis

Pengujian menggunakan tes proporsi untuk 2 sampel dengan menggunakan z-test

Daerah kritis: z < -1.9599639845400545 atau z > 1.959963984540054

5. Uji Statistik

	Top	Bot
p	0.673985	0.660812
x	614.000000	602.000000
n	911.000000	911.000000

Didapatkan z-value = 0.5966949094870971

Didapatkan p-value = 0.724644454056119

6. Pengambilan Keputusan

H0 gagal ditolak: nilai z ada dalam range -1.9599639845400545 < 0.5966949094870971 < 1.959963984540054 dan p-value = 0.724644454056119 >= 0.05, sehingga proporsi untuk yang olahraga lebih dari 8 jam per minggu antara data atas (0.6739846322722283) dengan data bawah (0.6608122941822173) cukup dekat

Nilai Fungsi Bawaan

Fungsi builtin: p-value = 0.5507110918877621 dan z-value = 0.5966949094870971