UNITED UNIVERSITY

SUBJECT NAME: DATA SCIENCE

PRACTICAL FILE

SESSION: 2025-26

SUBMITTED BY:

SAMEER SINGH

(UNIVERSITY ROLL NO.)

24201020070

SUBMITTED TO:

MR. SAMARTH AMRUTE

COURSE: BCA IBM

SEMESTER: 3ND

UNITED UNIVERSITY

RAWATPUR, PRAYAGRAJ, UTTAR PRADESH- 211012

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Make plots look better
sns.set(style="whitegrid")

diabetes = pd.read_csv("diabetes_dataset.csv")
```

```
diabetes.head()
```

	age	gender	ethnicity	education_level	income_level	employment_status	smoking_status	alcohol_consumption_per_week	physical
0	58	Male	Asian	Highschool	Lower-Middle	Employed	Never	0	
1	48	Female	White	Highschool	Middle	Employed	Former	1	
2	60	Male	Hispanic	Highschool	Middle	Unemployed	Never	1	
3	74	Female	Black	Highschool	Low	Retired	Never	0	
4	46	Male	White	Graduate	Middle	Retired	Never	1	

5 rows × 31 columns

-The dataset begins with demographic details like **age**, **gender**, and **ethnicity**.

It also includes lifestyle factors such as **smoking status**, **alcohol consumption**, and **physical activity**.

Several health measurements appear early, including **diet score**, **sleep hours**, and **screen time**.

Medical history indicators like **family diabetes**, **hypertension**, and **cardiovascular history** are included.

```
print("Shape of dataset:", diabetes.shape)
print("\nColumn names:", diabetes.columns.tolist())
```

```
Shape of dataset: (100000, 31)

Column names: ['age', 'gender', 'ethnicity', 'education_level', 'income_level', 'employment_status', 'smoking_status', 'alcohol_c
```

The dataset contains **31 columns** and a specific number of rows (shown in the shape).

It includes demographic fields like **age**, **gender**, and **ethnicity**.

Lifestyle attributes such as **diet score**, **sleep hours**, and **physical activity** are part of the dataset.

Clinical measurements like **BMI**, **blood pressure**, **cholesterol**, and **glucose levels** are also present.

```
diabetes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100000 entries, 0 to 99999
Data columns (total 31 columns):
 #   Column                            Non-Null Count   Dtype  
---  ------                            --------------   -----  
 0   age                               100000 non-null  int64  
 1   gender                            100000 non-null  object 
 2   ethnicity                         100000 non-null  object 
 3   education_level                   100000 non-null  object 
 4   income_level                      100000 non-null  object 
 5   employment_status                 100000 non-null  object 
 6   smoking_status                    100000 non-null  object 
 7   alcohol_consumption_per_week      100000 non-null  int64  
 8   physical_activity_minutes_per_week  100000 non-null  int64  
 9   diet_score                        100000 non-null  float64
 10  sleep_hours_per_day               100000 non-null  float64
 11  screen_time_hours_per_day         100000 non-null  float64
 12  family_history_diabetes           100000 non-null  int64  
 13  hypertension_history              100000 non-null  int64  
 14  cardiovascular_history            100000 non-null  int64  
 15  bmi                               100000 non-null  float64
 16  waist_to_hip_ratio                100000 non-null  float64
 17  systolic_bp                       100000 non-null  int64  
 18  diastolic_bp                      100000 non-null  int64  
 19  heart_rate                        100000 non-null  int64  
 20  cholesterol_total                 100000 non-null  int64  
 21  hdl_cholesterol                   100000 non-null  int64  
 22  ldl_cholesterol                   100000 non-null  int64  
 23  triglycerides                     100000 non-null  int64  
 24  glucose_fasting                   100000 non-null  int64  
 25  glucose_postprandial              100000 non-null  int64  
 26  insulin_level                     100000 non-null  float64
 27  hba1c                             100000 non-null  float64
 28  diabetes_risk_score               100000 non-null  float64
 29  diabetes_stage                    100000 non-null  object 
 30  diagnosed_diabetes                100000 non-null  int64  
dtypes: float64(8), int64(16), object(7)
memory usage: 23.7+ MB
```

The dataset contains **31 columns** with various data types (int, float, object).

There are **no missing values** in any column—everything is fully complete.

Columns include demographics, lifestyle factors, medical history, lab results, and diabetes outcomes.

```
diabetes.describe()
```

	age	alcohol_consumption_per_week	physical_activity_minutes_per_week	diet_score	sleep_hours_per_day	screen_tim
count	100000.00000	100000.000000	100000.000000	100000.000000	100000.000000	
mean	50.12041	2.003670	118.911640	5.994787	6.997818	
std	15.60460	1.417779	84.409662	1.780954	1.094622	
min	18.00000	0.000000	0.000000	0.000000	3.000000	
25%	39.00000	1.000000	57.000000	4.800000	6.300000	
50%	50.00000	2.000000	100.000000	6.000000	7.000000	
75%	61.00000	3.000000	160.000000	7.200000	7.700000	
max	90.00000	10.000000	833.000000	10.000000	10.000000	

8 rows × 24 columns

The dataset's numeric columns show a wide range of health and lifestyle measurements.

Age, **BMI**, **blood pressure**, and **cholesterol** values fall within realistic medical ranges.

Glucose and HbA1c distributions indicate clear variation between non-diabetic and diabetic individuals.

Risk-related metrics (e.g., *diabetes_risk_score*) show meaningful spread useful for prediction.

[9]
✓ 0s

```
diabetes.isnull().sum()
```

	0
age	0
gender	0
ethnicity	0
education_level	0
income_level	0
employment_status	0
bmi	0
waist_to_hip_ratio	0
systolic_bp	0
diastolic_bp	0
heart_rate	0
cholesterol_total	0
hdl_cholesterol	0
ldl_cholesterol	0
triglycerides	0
glucose_fasting	0
glucose_postprandial	0
insulin_level	0
hba1c	0
diabetes_risk_score	0
diabetes_stage	0
diagnosed_diabetes	0

dtype: int64

+ Code

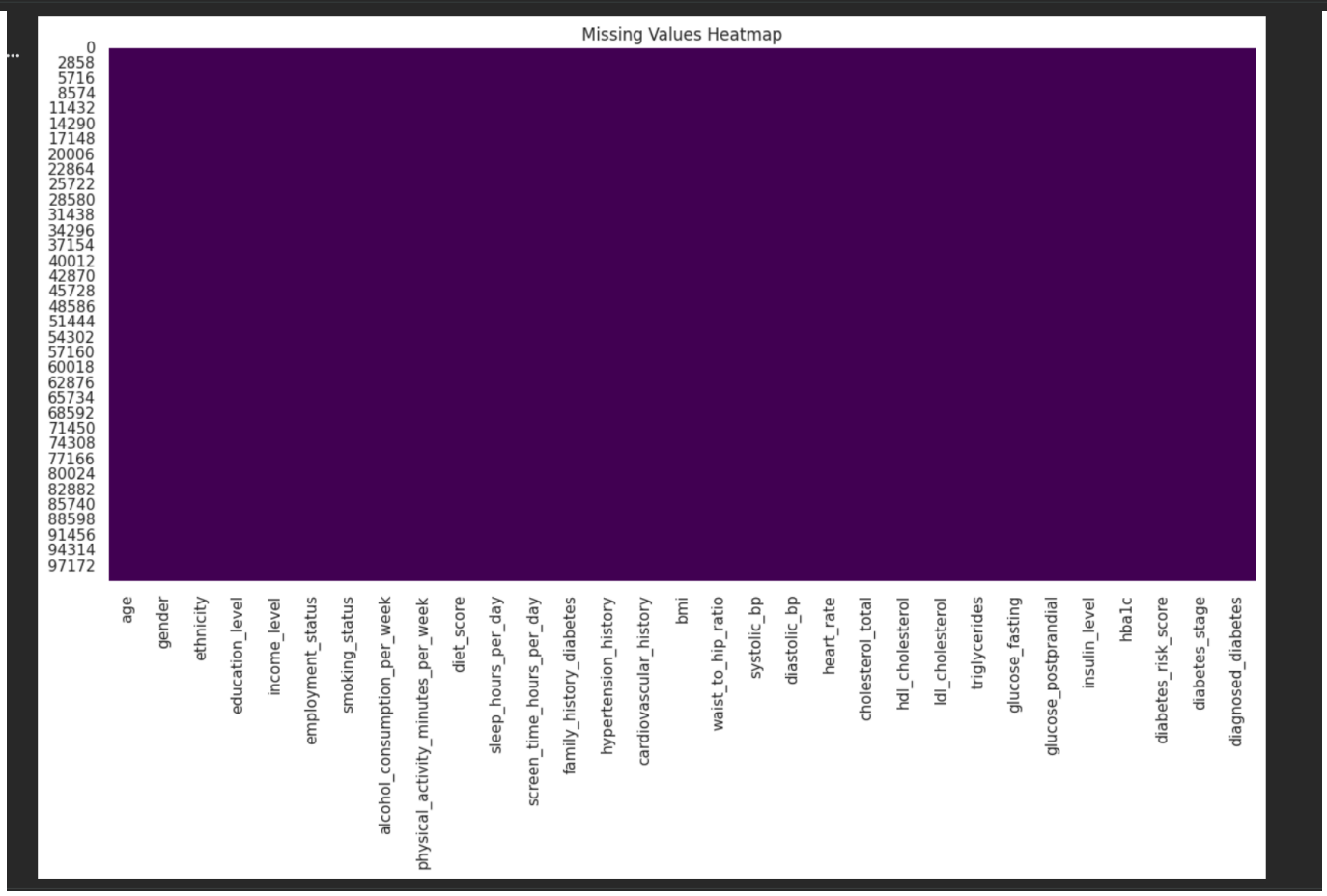
All **31 columns** in the diabetes dataset contain **0 missing values**.

Every variable—demographic, lifestyle, medical history, and lab results—is fully complete.

No imputation or cleaning for missing data is required.

This makes the dataset **ready for analysis** without preprocessing gaps.

```
[19]
✓ 3s
# Visualize missing data
plt.figure(figsize=(15,7))
sns.heatmap(diabetes.isnull(), cbar=False, cmap='viridis')
plt.title("Missing Values Heatmap")
plt.show()
```



The heatmap shows that **all columns have no missing values**.

The entire plot appears uniformly filled, indicating a **complete and clean dataset**.

There are **no gaps, blank spaces, or highlighted areas** that would suggest missing entries.

```
[26]
✓ 0s
diabetes['age'].fillna(diabetes['age'].mean(), inplace=True)

/tmp/ipython-input-456026655.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original

  diabetes['age'].fillna(diabetes['age'].mean(), inplace=True)

[27]
✓ 0s
diabetes['ethnicity'].fillna(diabetes['ethnicity'].mode()[0], inplace=True)

/tmp/ipython-input-1627986379.py:1: FutureWarning: A value is trying to be set on a copy of a DataFrame or Series through chained assignment using an inplace method.
The behavior will change in pandas 3.0. This inplace method will never work because the intermediate object on which we are setting values always behaves as a copy.

For example, when doing 'df[col].method(value, inplace=True)', try using 'df.method({col: value}, inplace=True)' or df[col] = df[col].method(value) instead, to perform the operation inplace on the original

  diabetes['ethnicity'].fillna(diabetes['ethnicity'].mode()[0], inplace=True)
```

```
diabetes = diabetes.drop(columns=['diabetes_risk_score', 'diabetes_stage'])
display(diabetes.head())
```

	age	gender	ethnicity	education_level	income_level	employment_status	smoking_status	alcohol_consumption_per_week	physical_activity_minutes_per_week	diet_score	...	heart_rate	cholesterol_total	hd
0	58	Male	Asian	Highschool	Lower-Middle	Employed	Never	0	215	5.7	...	68	239	
1	48	Female	White	Highschool	Middle	Employed	Former	1	143	6.7	...	67	116	
2	60	Male	Hispanic	Highschool	Middle	Unemployed	Never	1	57	6.4	...	74	213	
3	74	Female	Black	Highschool	Low	Retired	Never	0	49	3.4	...	68	171	
4	46	Male	White	Graduate	Middle	Retired	Never	1	109	7.2	...	67	210	

5 rows × 29 columns

The code removes the columns **`diabetes_risk_score`** and **`diabetes_stage`** from the dataset.

These columns are likely dropped to focus on other features or avoid data leakage in modeling.

The remaining dataset now contains **29 columns** instead of 31.

`display(diabetes.head())` shows the **first 5 rows** of the updated dataset.

The dataset remains fully clean and ready for further EDA or modeling steps.

```
categorical_cols = diabetes.select_dtypes(include=['object']).columns
diabetes_encoded = pd.get_dummies(diabetes, columns=categorical_cols, drop_first=True)
display(diabetes_encoded.head())
```

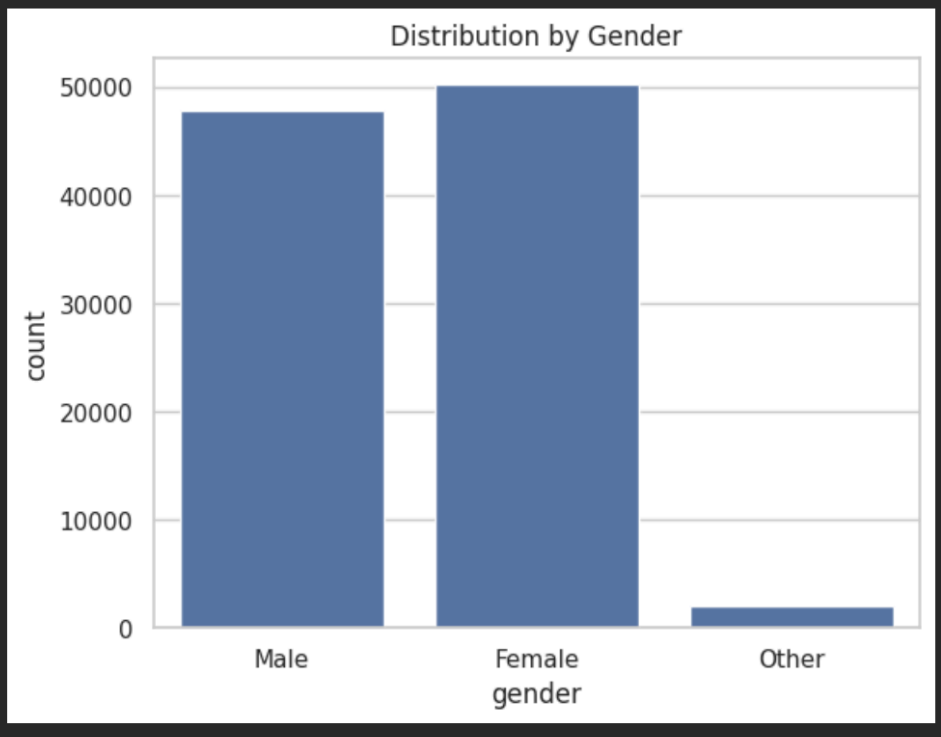
	age	alcohol_consumption_per_week	physical_activity_minutes_per_week	diet_score	sleep_hours_per_day	screen_time_hours_per_day	family_history_diabetes	hypertension_history	cardiovascular_history	bm
0	58	0	215	5.7	7.9	7.9	0	0	0	30.
1	48	1	143	6.7	6.5	8.7	0	0	0	23.
2	60	1	57	6.4	10.0	8.1	1	0	0	22.
3	74	0	49	3.4	6.6	5.2	0	0	0	26.
4	46	1	109	7.2	7.4	5.0	0	0	0	21.

5 rows × 41 columns

The code identifies all **categorical (object-type) columns** in the diabetes dataset.

It converts these categorical variables into **dummy/one-hot encoded numeric columns**.

```
sns.countplot(x='gender', data=diabetes)
plt.title("Distribution by Gender")
plt.show()
```



The countplot shows how many individuals in the dataset are **Male vs Female**.

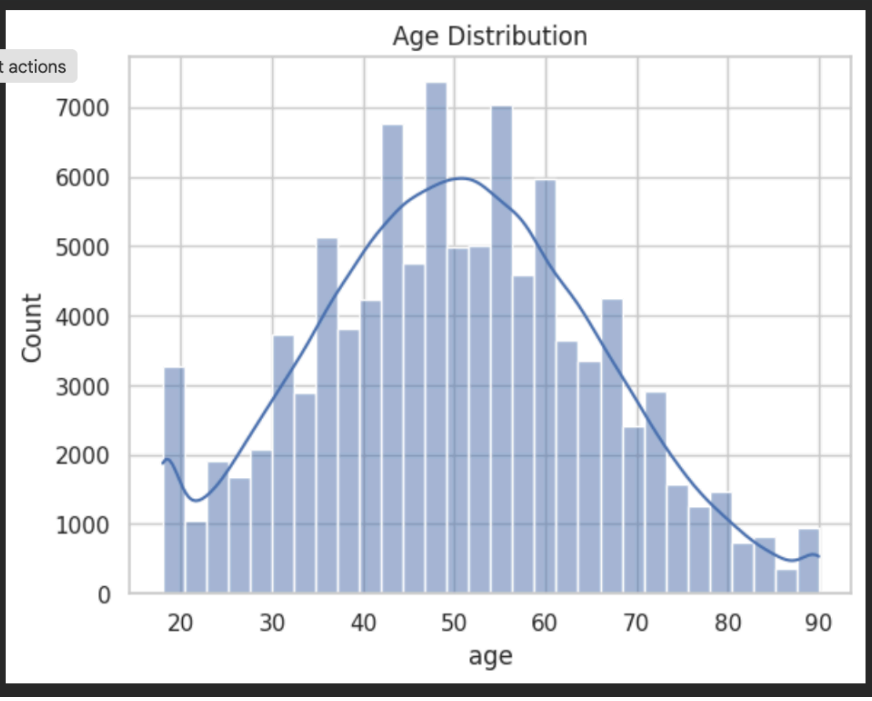
It provides a quick visual comparison of gender representation.

The bars reveal whether the dataset is **balanced or skewed** toward one gender.

This is useful for understanding demographic composition before modeling.

The plot helps identify if gender-related patterns (e.g., diabetes rates) may be influenced by sample size differences.

```
# Age distribution
sns.histplot(diabetes['age'], bins=30, kde=True)
plt.title("Age Distribution")
plt.show()
```



The histogram shows how ages are spread across all individuals in the dataset.

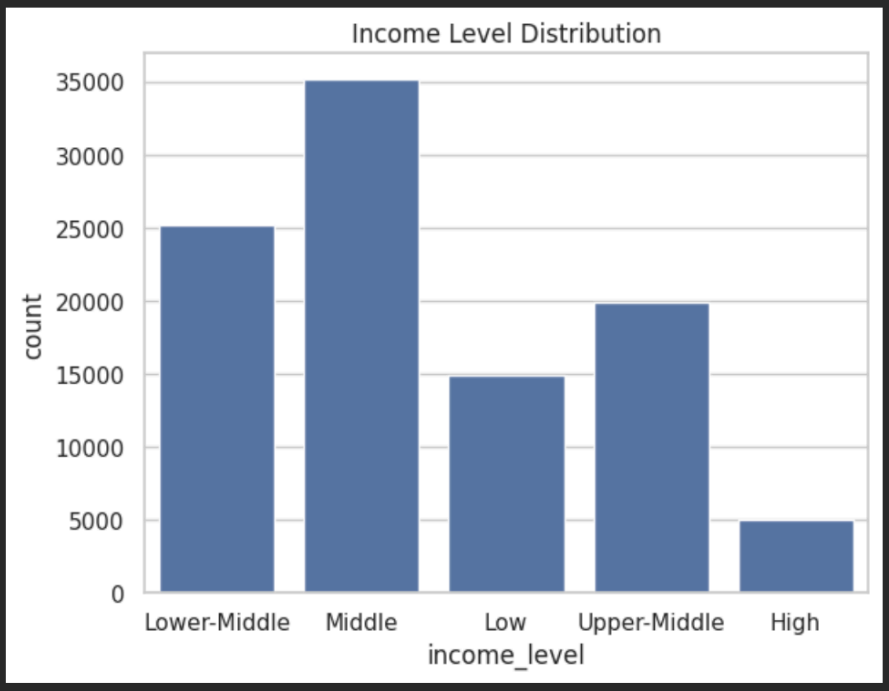
The distribution helps identify whether most people are younger, middle-aged, or older.

The KDE curve highlights the **overall shape** (e.g., skewness or peaks).

This plot helps understand age patterns that may influence diabetes risk.

It also reveals whether the dataset is age-balanced or concentrated in certain age groups.

```
sns.countplot(x='income_level', data=diabetes)
plt.title("Income Level Distribution")
plt.show()
```



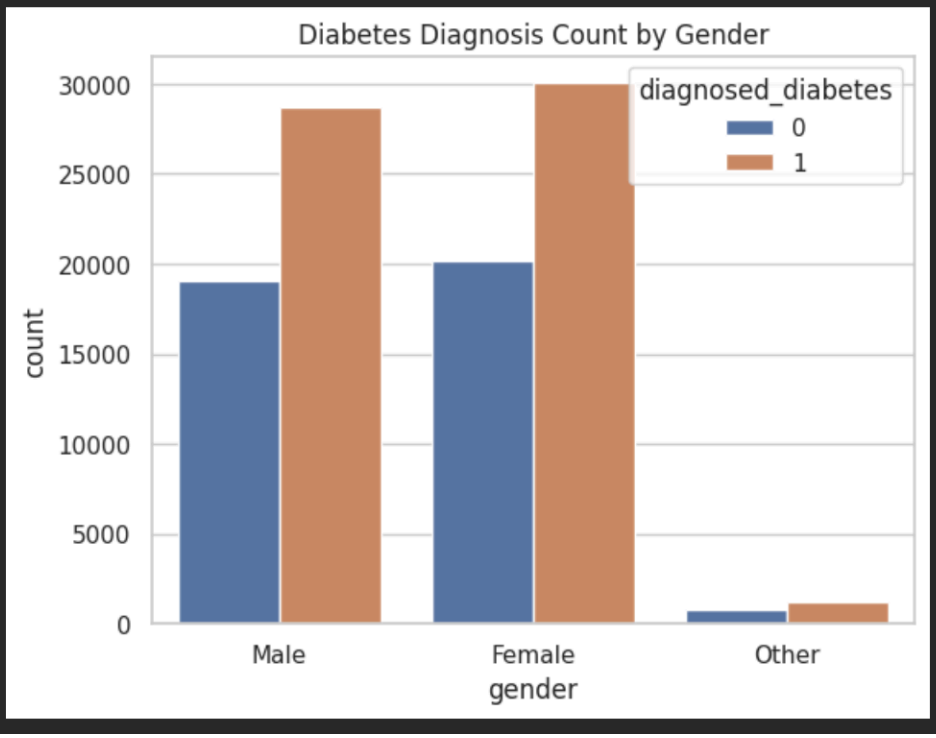
The countplot displays how many individuals fall into each **income level category**.

It helps you see whether the dataset has more people from low, middle, or higher income groups.

The distribution provides insight into the **socioeconomic diversity** of the sample.

This is useful because income level may influence lifestyle, health behaviors, and diabetes risk..

```
sns.countplot(x='gender', hue='diagnosed_diabetes', data=diabetes)
plt.title("Diabetes Diagnosis Count by Gender")
plt.show()
```



The plot compares diabetes diagnosis counts between **males and females**.

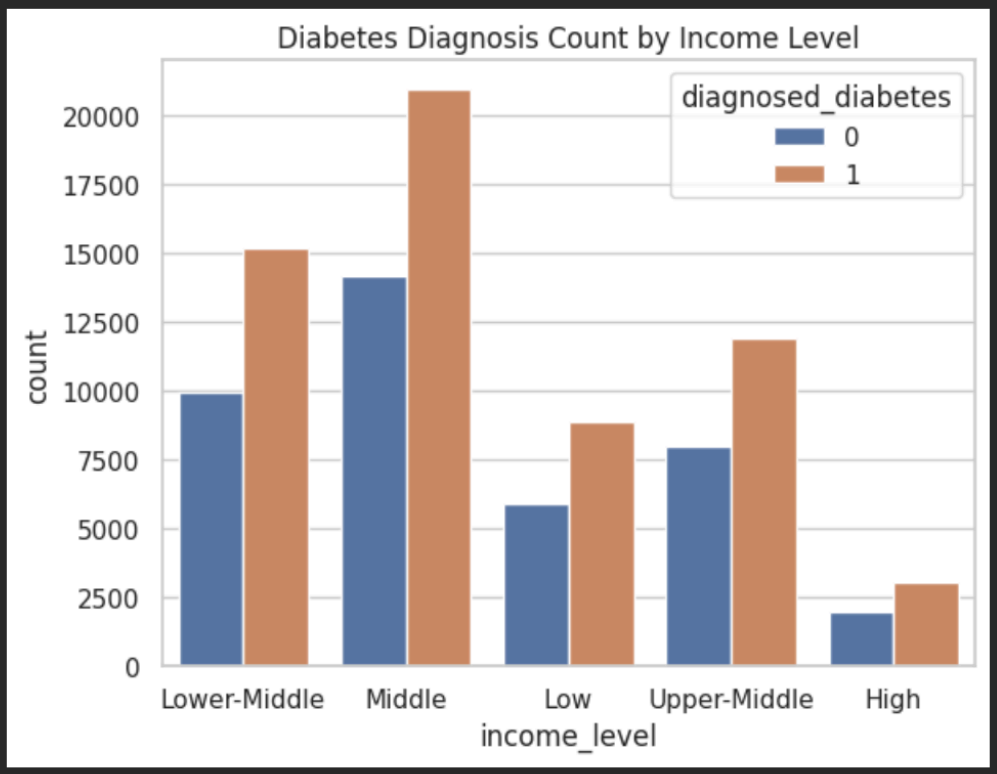
Each gender category shows two bars: **0 = no diabetes**, **1 = diagnosed diabetes**.

This helps identify whether one gender has a **higher prevalence of diabetes**.

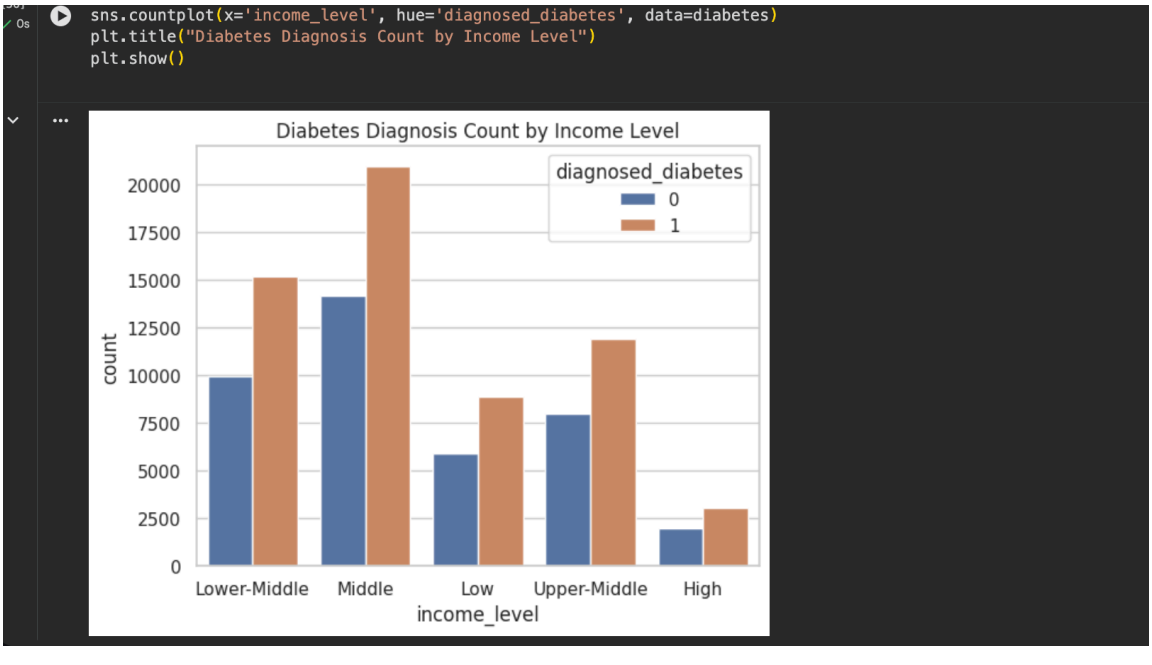
The visualization also shows if the sample sizes differ across genders.

Overall, it provides a clear picture of **gender-based dif**

```
sns.countplot(x='income_level', hue='diagnosed_diabetes', data=diabetes)
plt.title("Diabetes Diagnosis Count by Income Level")
plt.show()
```




```
sns.countplot(x='income_level', hue='diagnosed_diabetes', data=diabetes)
plt.title("Diabetes Diagnosis Count by Income Level")
plt.show()
```

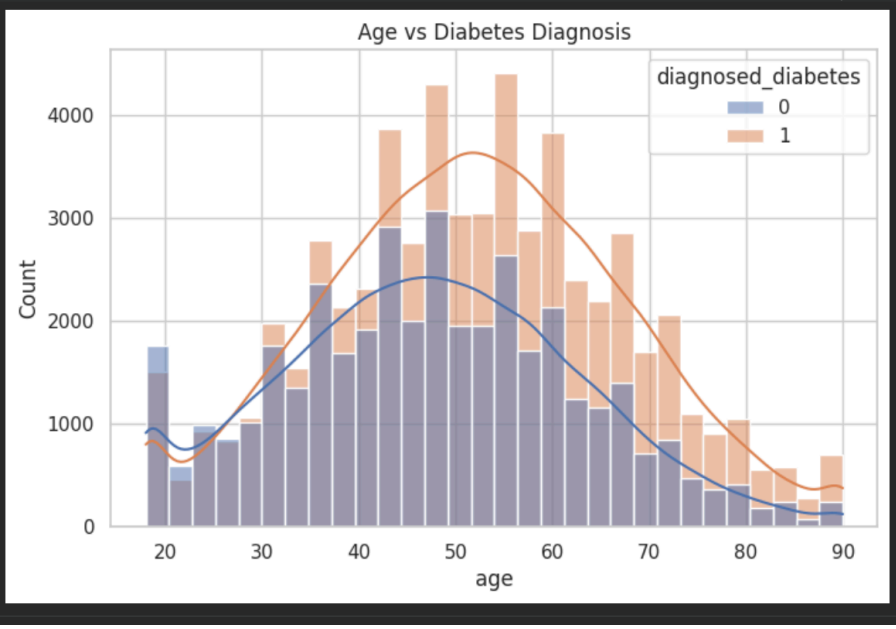
The plot shows diabetes diagnosis counts across different **income levels**.

Each income group displays two bars: **non-diabetic (0)** and **diabetic (1)** individuals.

It helps identify whether diabetes is more common in **low**, **middle**, or **high** income groups.

The visualization reveals any socioeconomic patterns related to diabetes prevalence.

```
# Age vs Diabetes Diagnosis
plt.figure(figsize=(8,5))
sns.histplot(data=diabetes, x='age', hue='diagnosed_diabetes', bins=30, kde=True)
plt.title("Age vs Diabetes Diagnosis")
plt.show()
```



The plot shows how age is distributed among people **with and without diabetes**.

Two overlapping histograms help compare age patterns across both groups.

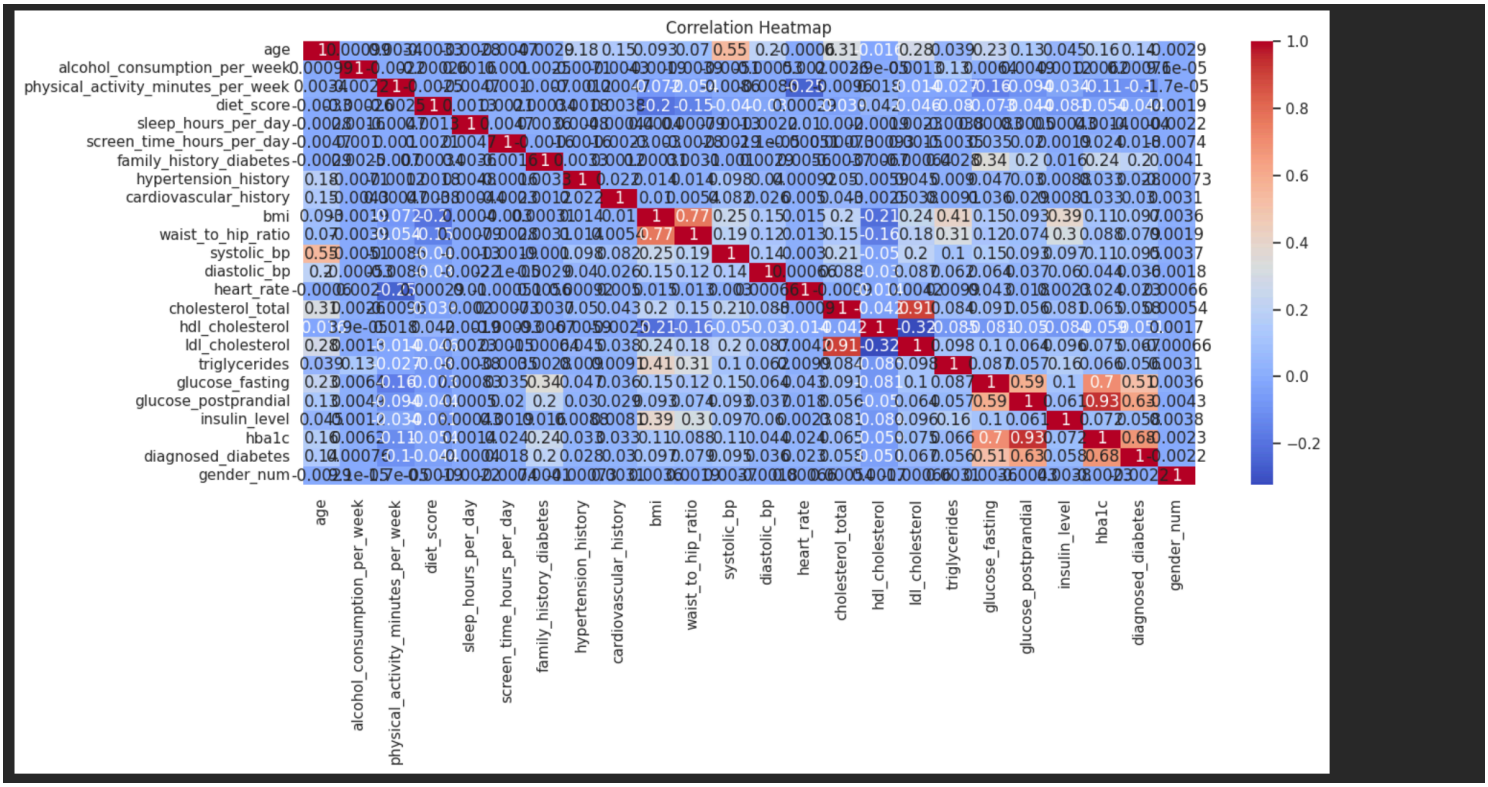
It reveals whether diabetes is more common in **older age groups**.

The KDE curves offer a smooth comparison of the age trends for each diagnosis category.

```
diabetes['gender_num'] = diabetes['gender'].map({'Male': 0, 'Female': 1})
```

```
corr = diabetes.corr(numeric_only=True)
```

```
# Plot heatmap
plt.figure(figsize=(25,15))
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title("Correlation Heatmap")
plt.show()
```



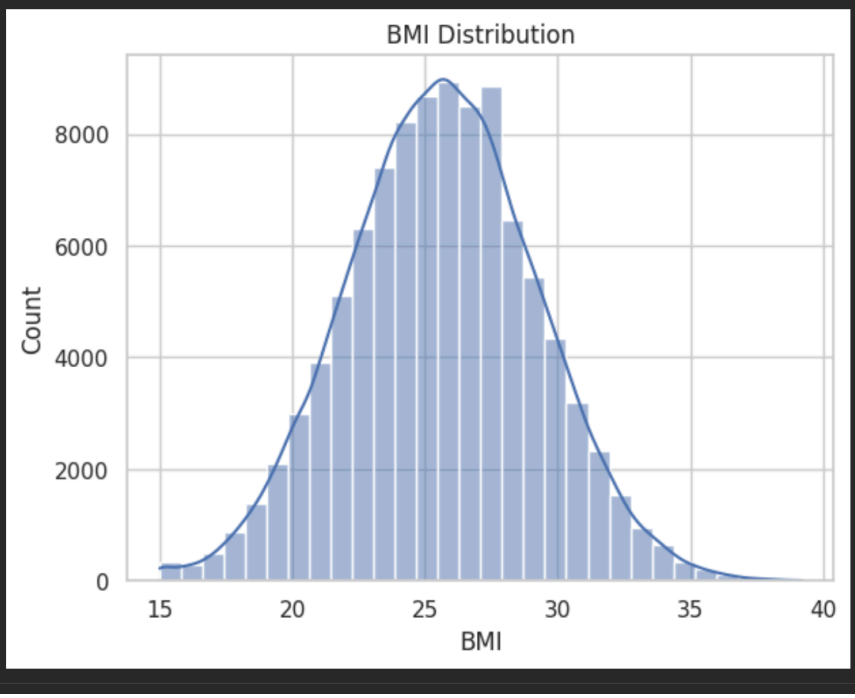
The heatmap displays the **correlation strength** between all numerical variables in the dataset.

Darker red or blue colors indicate **strong positive or negative relationships**.

It helps identify which features are most strongly related to **diabetes diagnosis, glucose levels, and HbA1c**.

Highly correlated variables may signal multicollinearity, important for model building.

```
sns.histplot(diabetes['bmi'], bins=30, kde=True)
plt.title("BMI Distribution")
plt.xlabel("BMI")
plt.show()
```



The histogram shows how BMI values are distributed across all individuals in the dataset.

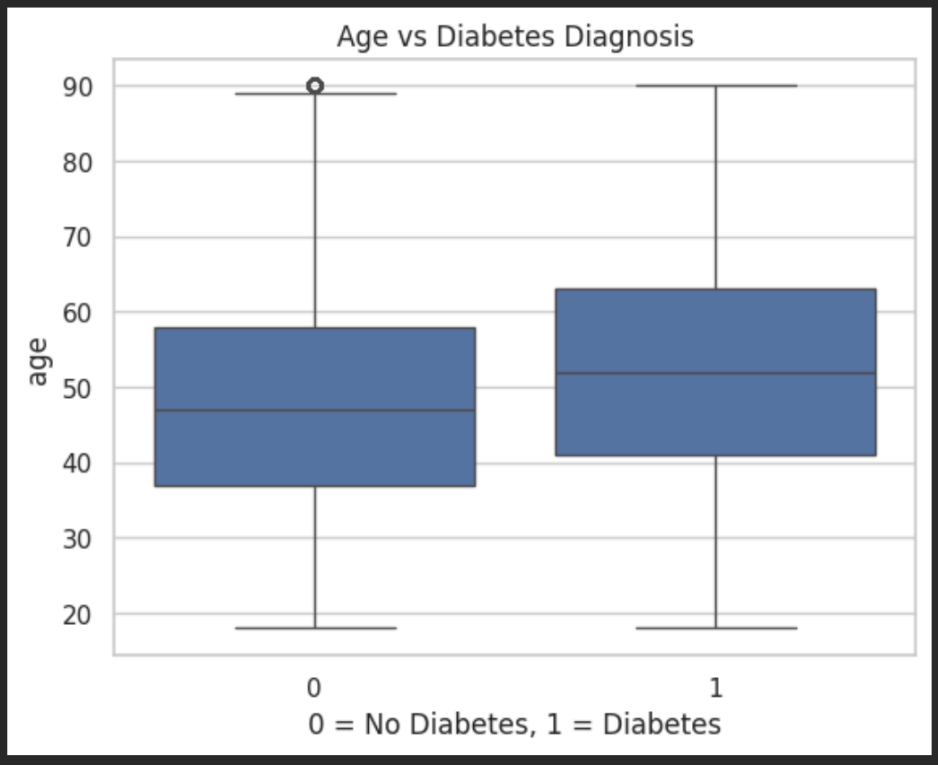
The KDE curve highlights the overall **shape and central tendency** of BMI.

It helps identify whether the population is mostly in the normal, overweight, or obese BMI range.

Any skewness or peaks in the BMI distribution indicate common health patterns in the dataset.

This plot is useful for understanding how **body weight characteristics** relate to diabetes risk.

```
sns.boxplot(x='diagnosed_diabetes', y='age', data=diabetes)
plt.title("Age vs Diabetes Diagnosis")
plt.xlabel("0 = No Diabetes, 1 = Diabetes")
plt.show()
```



The boxplot compares the age distribution between people **with** and **without** diabetes.

It shows whether diabetic individuals tend to be **older** than non-diabetics.

The median line helps highlight the typical age in each group.

The spread and whiskers reveal **age variability** within each diagnosis category.