

Exploratory Data Analysis (EDA) on Diabetes Dataset

This notebook performs a complete EDA on a diabetes dataset to understand key health indicators, detect missing values and outliers, visualize important trends, and prepare the data for further modeling or analysis.

Table of Contents

- Load and View Dataset
- Shape, Size, Head & Tail
- Descriptive Statistics
- Missing Value Detection & Imputation
- Outlier Detection (IQR & Z-Score)
- Visualizations (Histograms, Bar Graphs, Box Plots, Pie Charts)
- Correlation Analysis
- Outcome-Based Analysis
- Groupby analysis
- Conclusion and insights

1. Import Required Libraries

```
In [1]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

2. Load the Dataset

```
In [3]: diabetes_data = pd.read_csv('diabetes_data.csv')
diabetes_data.head()
```

```
Out [3]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
In [38]: diabetes_data.tail()
```

```
Out [38]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
763	10	101	76	48	180	32.9	0.171	63	0
764	2	122	70	27	0	36.8	0.340	27	0
765	5	121	72	23	112	26.2	0.245	30	0
766	1	126	60	0	0	30.1	0.349	47	1
767	1	93	70	31	0	30.4	0.315	23	0

3. Basic Information About the Dataset

```
In [5]: diabetes_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column              Non-Null Count  Dtype  
---  --
 0   Pregnancies         768 non-null    int64  
 1   Glucose             768 non-null    int64  
 2   BloodPressure       768 non-null    int64  
 3   SkinThickness       768 non-null    int64  
 4   Insulin             768 non-null    int64  
 5   BMI                 768 non-null    float64 
 6   DiabetesPedigreeFunction 768 non-null    float64 
 7   Age                 768 non-null    int64  
 8   Outcome             768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

4. Dataset Shape and Columns

```
In [7]: print('Shape:', diabetes_data.shape)
print('Columns:', diabetes_data.columns.tolist())

Shape: (768, 9)
Columns: ['Pregnancies', 'Glucose', 'BloodPressure', 'SkinThickness', 'Insulin', 'BMI', 'DiabetesPedigreeFunction', 'Age', 'Outcome']
```

5. Mean and Median of Age, Blood Pressure, and BMI

```
In [9]: print('Mean Values:')
print('Age:', diabetes_data['Age'].mean())
print('BloodPressure:', diabetes_data['BloodPressure'].mean())
print('BMI:', diabetes_data['BMI'].mean())

print('\nMedian Values:')
print('Age:', diabetes_data['Age'].median())
print('BloodPressure:', diabetes_data['BloodPressure'].median())
print('BMI:', diabetes_data['BMI'].median())

Mean Values:
Age: 33.24085416666664
BloodPressure: 69.10546875
BMI: 31.992578124999998

Median Values:
Age: 29.0
BloodPressure: 72.0
BMI: 32.0
```

6. Descriptive Statistics

```
In [11]: diabetes_data.describe()
```

```
Out [11]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
count	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000	768.000000
mean	3.845052	120.894531	69.105469	20.536458	79.799479	31.992578	0.471876	33.240885	0.348958
std	3.369578	31.972618	19.355807	15.952218	115.244002	7.684160	0.331329	11.760232	0.476951
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.078000	21.000000	0.000000
25%	1.000000	99.000000	62.000000	0.000000	0.000000	27.300000	0.245750	24.000000	0.000000
50%	3.000000	117.000000	72.000000	23.000000	30.500000	32.000000	0.372500	29.000000	0.000000
75%	6.000000	140.250000	80.000000	32.000000	127.250000	36.600000	0.626250	41.000000	1.000000
max	17.000000	199.000000	122.000000	99.000000	846.000000	67.100000	2.420000	81.000000	1.000000

7. Check for Missing Values

```
In [14]: diabetes_data.isnull().sum()
```

```
Out [14]: Pregnancies    0
Glucose              0
BloodPressure        0
SkinThickness        0
Insulin              0
BMI                  0
DiabetesPedigreeFunction 0
Age                  0
Outcome              0
dtype: int64
```

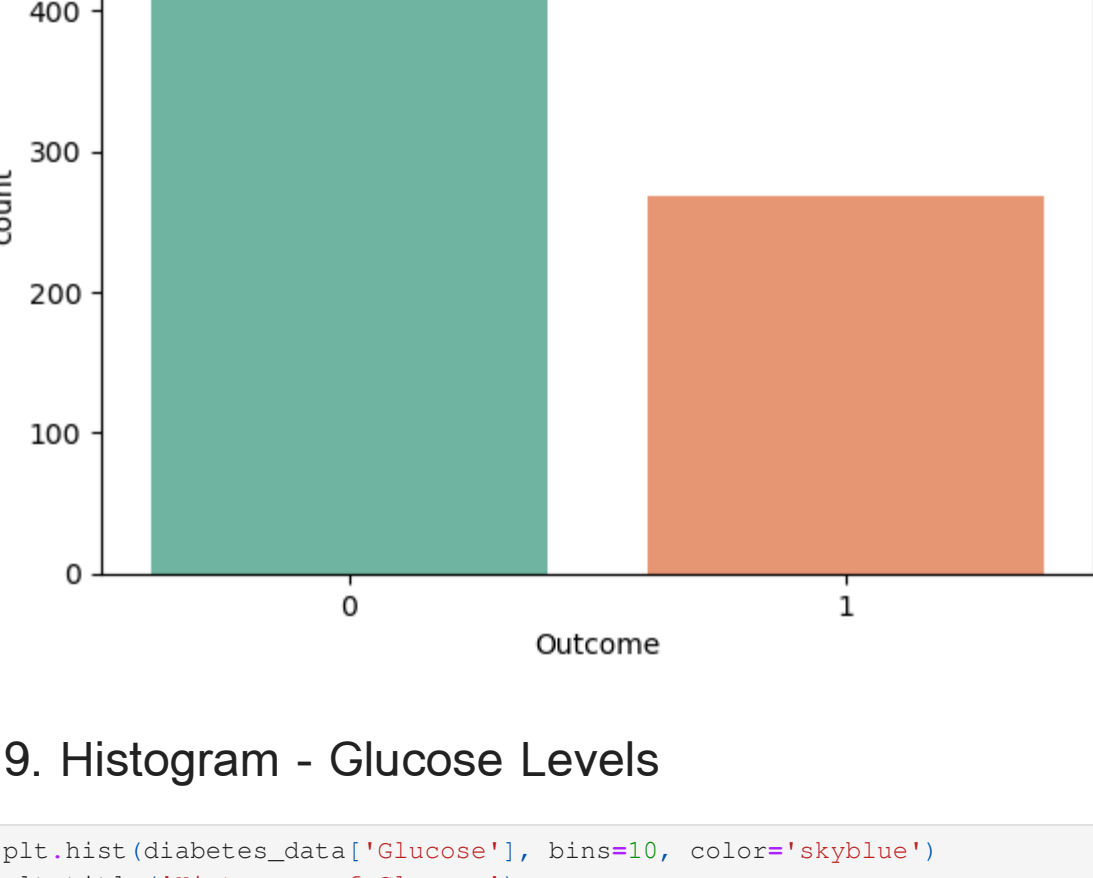
8. Bar Plot - Diabetes Outcome Count

```
In [56]: sns.countplot(x='Outcome', data=diabetes_data, palette='Set2')
plt.title('Distribution of Diabetes Outcome')
plt.show()
```

C:\Users\Saniya\AppData\Local\Temp\ipykernel_16968\2543927230.py:1: FutureWarning:

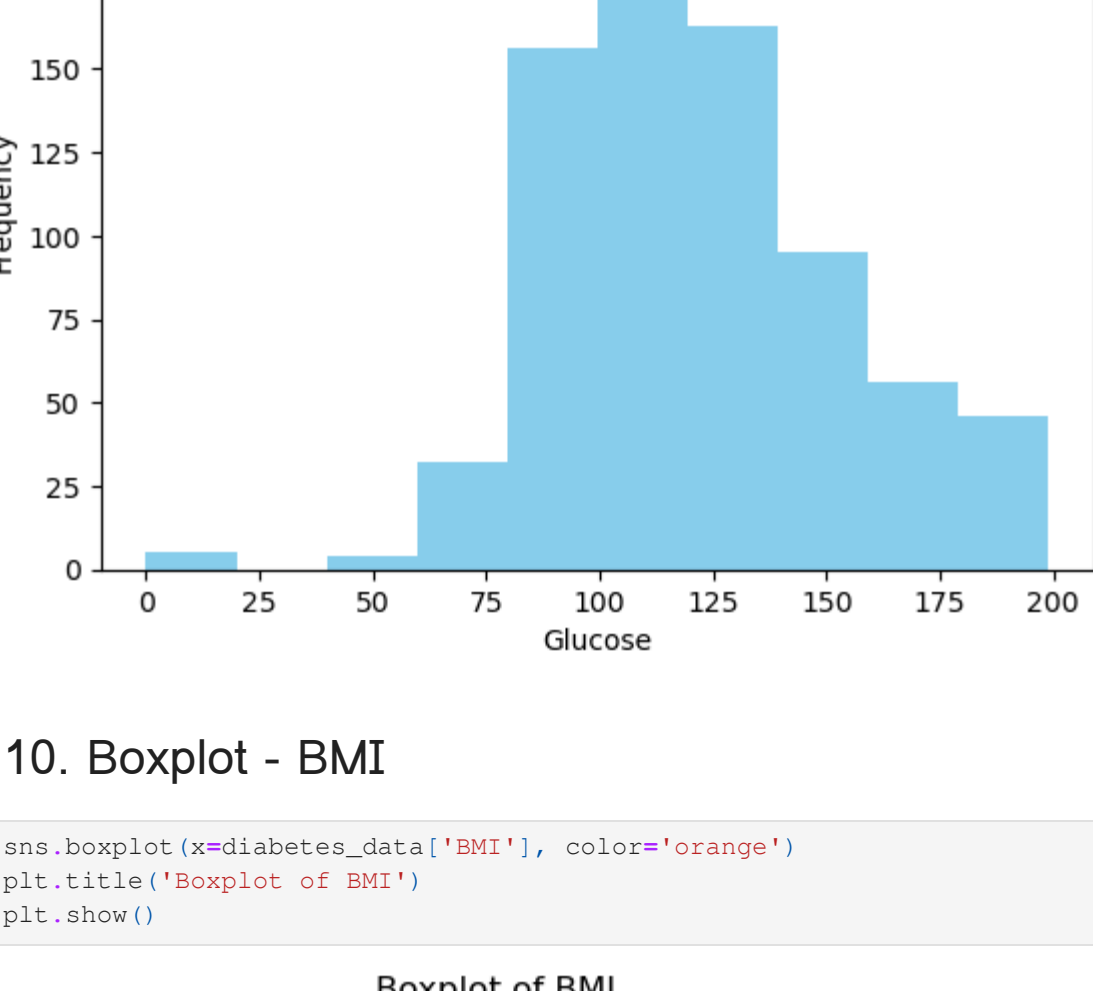
Passing 'palette' without assigning 'hue' is deprecated and will be removed in v0.14.0. Assign the 'x' variable to 'hue' and set 'legend=False' for the same effect.

```
sns.countplot(x='Outcome', data=diabetes_data, palette='Set2')
```



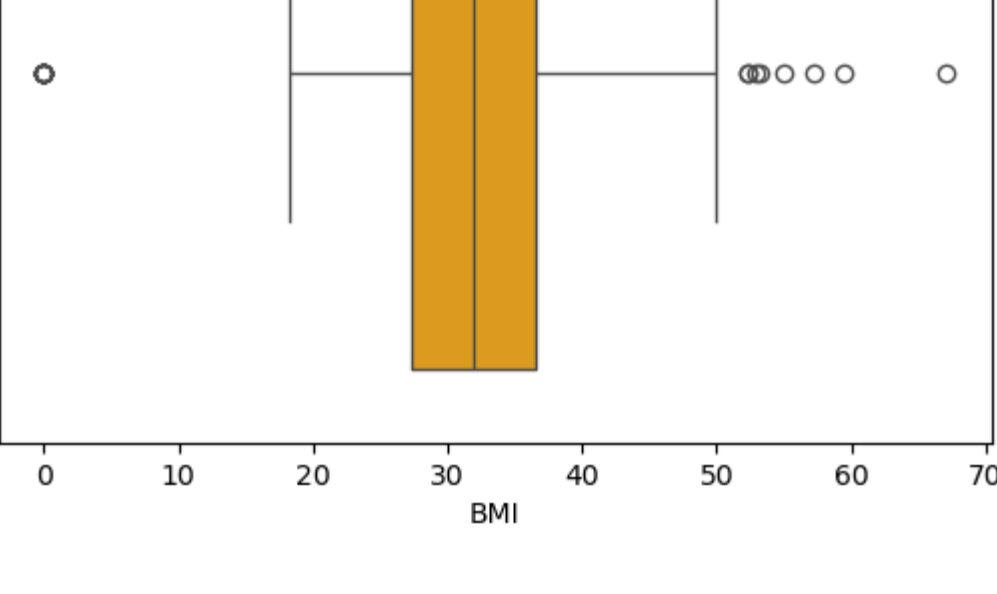
9. Histogram - Glucose Levels

```
In [20]: plt.hist(diabetes_data['Glucose'], bins=10, color='skyblue')
plt.title('Histogram of Glucose')
plt.xlabel('Glucose')
plt.ylabel('Frequency')
plt.show()
```



10. Boxplot - BMI

```
In [22]: sns.boxplot(x=diabetes_data['BMI'], color='orange')
plt.title('Boxplot of BMI')
plt.show()
```



11. Outlier Detection using IQR Method

```
In [24]: Q1 = diabetes_data['BMI'].quantile(0.25)
Q3 = diabetes_data['BMI'].quantile(0.75)
IQR = Q3 - Q1
outliers = diabetes_data[(diabetes_data['BMI'] < (Q1 - 1.5 * IQR)) | (diabetes_data['BMI'] > (Q3 + 1.5 * IQR))]
print('Outliers in BMI:')
print(outliers)
```

```
Outliers in BMI:
Pregnancies  Glucose  BloodPressure  SkinThickness  Insulin  BMI  \
49           7       105             0              0      0  0.0
60           2        84             0              0      0  0.0
81           2        74             0              0      0  0.0
120          0       162             76             56    100  53.2
125          1        88             30             42     99  55.0
145          0      102             75             23     0   0.0
177          0      129             110            46    130  67.1
193          1       135             64             23     89  60.0
247          0      145             90             33    680  52.3
303          5       115             98              0     52.9
371          0      118             64             23     89  60.0
426          0       94              0              0      0  0.0
445          0      180             78             63    14  59.4
494          3        80              0              0      0  0.0
522          6      114              0              0      0  0.0
673          3      123             100             35    240  57.3
684          5       136             82              0      0  0.0
706         10      115              0              0      0  0.0
```

```
DiabetesPedigreeFunction  Age  Outcome
9           0.232      54      1
49          0.305      24      0
60          0.304      21      0
81          0.102      22      0
120         0.759      25      1
125         0.496      26      1
145         0.572      21      0
177         0.319      26      1
193         0.578      40      1
247         0.427      23      0
303         0.209      28      1
371         1.731      21      0
426         0.256      25      0
445         2.420      25      1
494         0.174      22      0
522         0.189      26      0
673         0.880      22      0
684         0.640      69      0
706         0.261      30      1
```

12. Outlier Detection using Z-Score

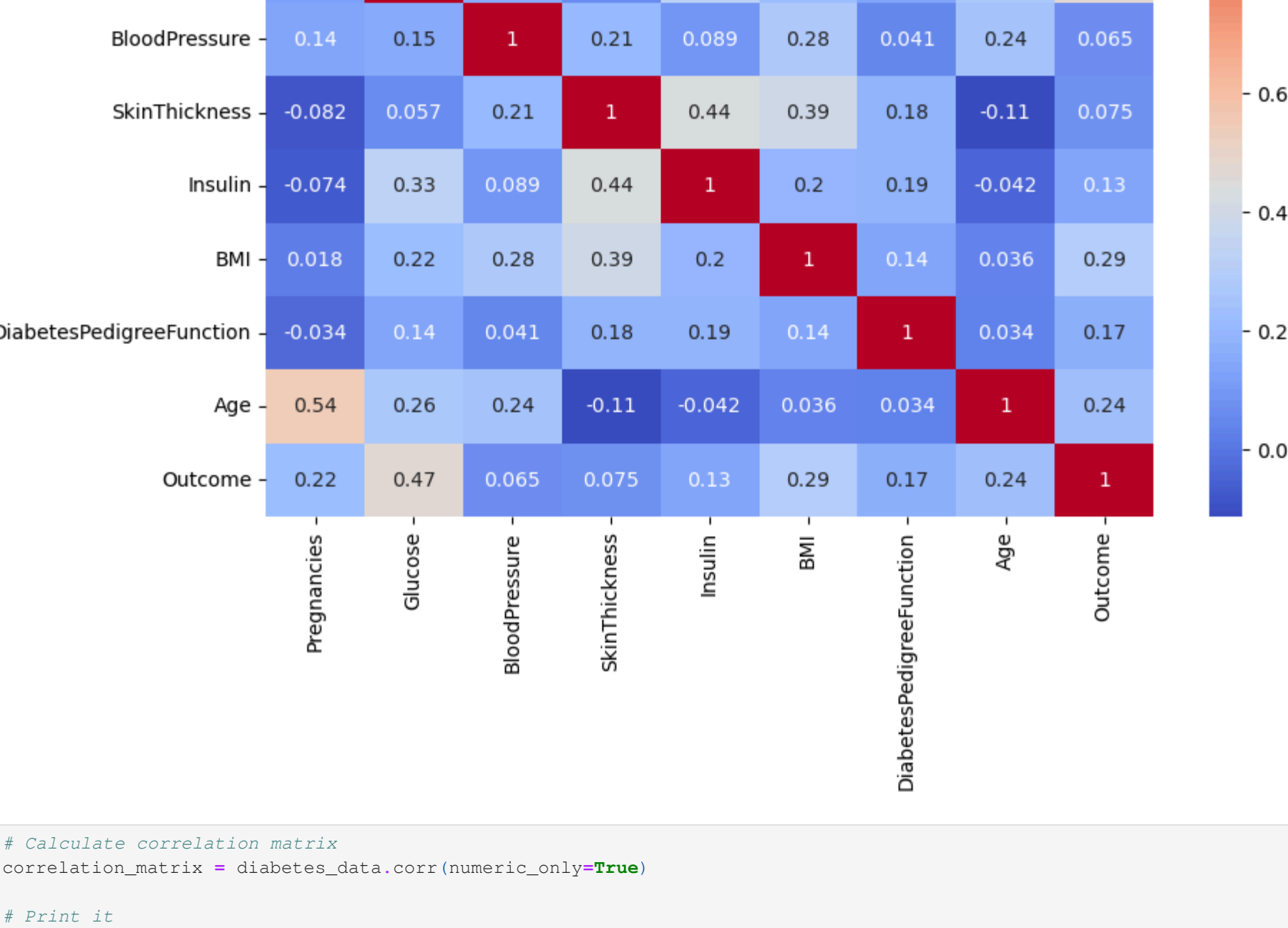
```
In [26]: from scipy.stats import zscore
z_scores = zscore(diabetes_data[['BMI']])
diabetes_data[np.abs(z_scores) > 3].any(axis=1)
```

```
Out [26]:
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
9	8	125	96	0	0	0.0	0.232	54	1
49	7	105	0	0	0	0.0	0.305	24	0
60	2	84	0	0	0	0.0	0.304	21	0
81	2	74	0	0	0	0.0	0.102	22	0
120	0	162	75	23	0	0.0	0.572	21	0
145	0	102	75	23	0	0.0	0.572	21	0
177	0	129	110	46	130	67.1	0.319	26	1
371	0	118	64	23	89	0.0	1.731	21	0
426	0	94	0	0	0	0.0	0.256	25	0
445	0	180	78	63	14	59.4	2.420	25	1
494	3	80	0	0	0	0.0	0.174	22	0
522	6	114	0	0	0	0.0	0.189	26	0
673	3	123	100	35	240	57.3	0.880	22	0
684	5	136	82	0	0	0.0	0.640	69	0
706	10	115	0	0	0	0.0	0.261	30	1

13. Correlation Heatmap

```
In [28]: plt.figure(figsize=(10,6))
sns.heatmap(diabetes_data.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```



```
In [66]: # Calculate correlation matrix
correlation_matrix = diabetes_data.corr(numeric_only=True)
```

```
# Print it
print (correlation_matrix)
```

```
Pregnancies    1.000000    0.129459    0.141282   -0.081672   -0.037337
Glucose         0.129459    1.000000    0.152590    0.057328    0.057328
BloodPressure   0.141282    0.152590    1.000000    0.207371    1.000000
SkinThickness   -0.081672    0.057328    0.207371    1.000000
Insulin         -0.073535    0.331357    0.088933    0.436783
BMI             0.017683    0.221071    0.281805    0.392573
DiabetesPedigreeFunction -0.033523    0.173737    0.641265    0.183928
Age             0.544341    0.263514    0.239528   -0.113970
Outcome         0.221898    0.465881    0.065068    0.074752
```

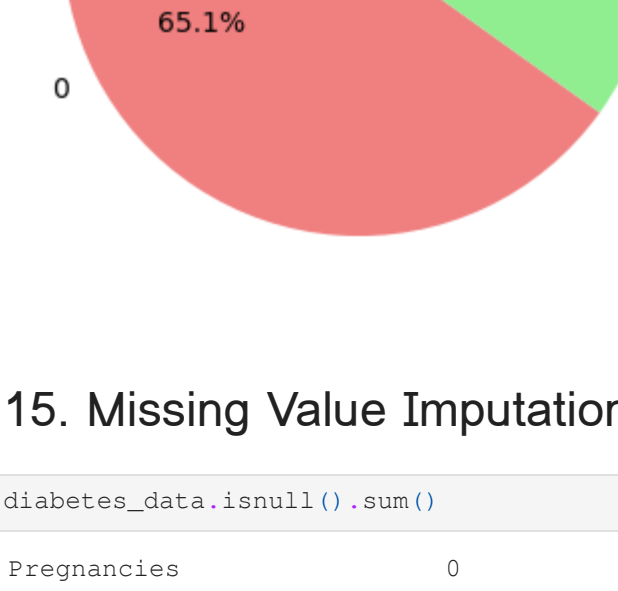
```
Insulin        BMI  DiabetesPedigreeFunction  \
Pregnancies   -0.073535    0.017683          -0.033523
Glucose        0.331357    0.221071          -0.173737
BloodPressure  0.088933    0.281805          0.041265
SkinThickness  0.436783    0.392573          0.183928
Insulin        1.000000    0.197859          0.185071
BMI            0.197859    1.000000          0.140647
DiabetesPedigreeFunction 0.185071    0.140647          1.000000
Age            0.263514    0.239528          0.033561
Outcome        0.130548    0.292695          0.173844
```

```
Age  Outcome
Pregnancies    0.544341    0.221898
Glucose        0.263514    0.465881
BloodPressure  0.239528    0.065068
SkinThickness  -0.113970    0.074752
Insulin        0.182163    0.130548
BMI            0.036242    0.292695
DiabetesPedigreeFunction 0.033561    0.173844
Age            1.000000    0.238556
Outcome        0.238556    1.000000
```

14. Pie Chart - Diabetes Outcome

```
In [30]: diabetes_data['Outcome'].value_counts().plot.pie(autopct='%1.1f%%', startangle=90, colors=['lightcoral', 'lightgreen'])
plt.title('Diabetes Outcome Distribution')
plt.ylabel('')
plt.show()
```

Diabetes Outcome Distribution



15. Missing Value Imputation (if Any)

```
In [36]: diabetes_data.isnull().sum()
```

```
Out [36]: Pregnancies    0
Glucose              0
BloodPressure        0
SkinThickness        0
Insulin              0
BMI                  0
DiabetesPedigreeFunction 0
Age                  0
Outcome              0
dtype: int64
```

```
In [ ]: ## Group by analysis
```

```
diabetes_data.groupby('Outcome')[['Glucose', 'BMI', 'Age']].mean()
```

```
Out [62]:
```

	Glucose	BMI	Age
Outcome			
0	109.980000	30.304200	31.190000
1	141.257463	35.142537	37.067164

Conclusion & Key Insights

- The dataset contains health-related attributes like Glucose, BloodPressure, Insulin, BMI, Age, etc., for predicting the likelihood of diabetes.

Data Quality:

- No missing values in terms of `NaN`, but several columns had biologically invalid zero values.
- These zero values were treated as missing and imputed using the mean/median to ensure clean and realistic data.

Statistical Summary:

- Mean Glucose level is significantly higher in diabetic patients.
- BMI and Age also show a higher average in people with diabetes.

Visual Insights:

- Boxplots revealed several outliers in `Insulin`, `BMI`, and `SkinThickness`.
- Distribution plots showed that features like `Glucose` and `BMI` are slightly right-skewed.

- Countplots and pie charts revealed:

- Around 35% of the people are diabetic (Outcome = 1).
- Pregnancies and Age show clear differences between diabetic and non-diabetic groups.

Correlation Analysis:

- `Glucose` and `BMI` show a positive correlation with the diabetes outcome.
- `Pregnancies` and `Age` are also moderately correlated with the outcome.

Final Note:

- The dataset is now clean and well-understood.
- It is ready for further feature engineering and machine learning modeling to build a diabetes prediction system.

