1.Write a Python program to collect, load, and perform initial exploration of the Diabetes dataset using Pandas. Display the first few records of the dataset and summarize its structure and contents.

In [8]:

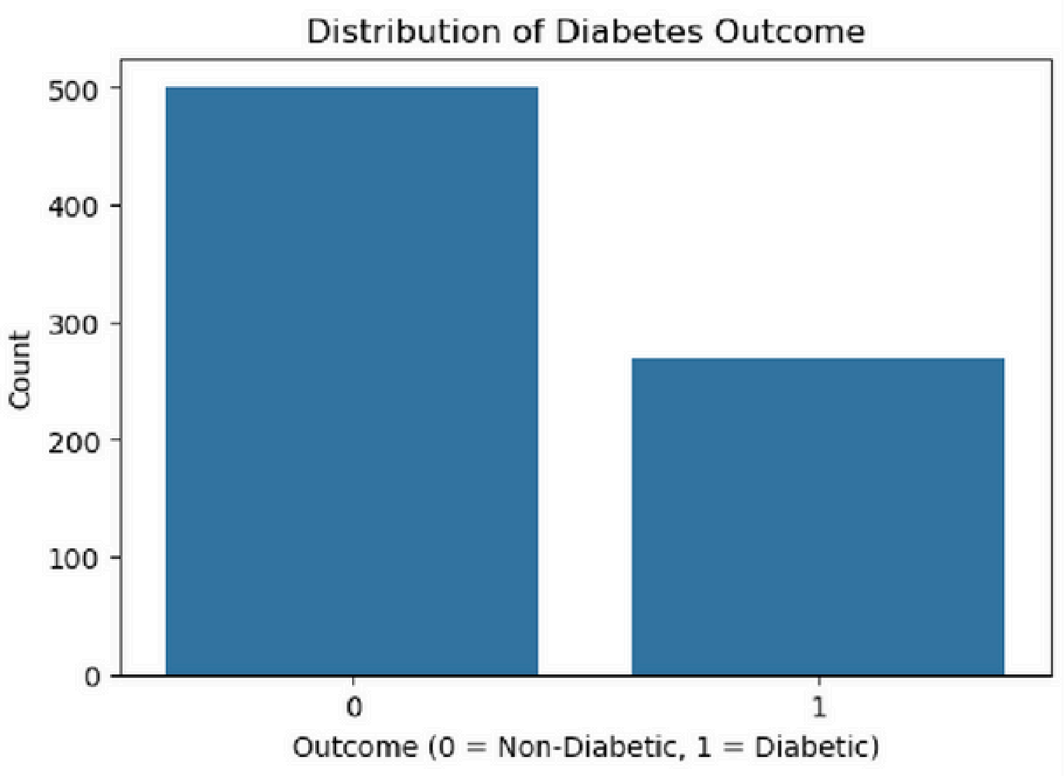
```
import pandas as pd import numpy as np import
matplotlib.pyplot as plt import seaborn as sns data =
pd.read_csv("diabetes.csv") print(data.head())
print(data.info()) print(data.describe())
print(data.isnull().sum()) plt.figure(figsize=(6,4))
sns.countplot(x='Outcome', data=data)
plt.title('Distribution of Diabetes Outcome')
  plt.xlabel('Outcome (0 = Non-Diabetic, 1 = Diabetic)')
plt.ylabel('Count') plt.show() plt.figure(figsize=
(10,8)) sns.heatmap(data.corr(), annot=True,
cmap='coolwarm') plt.title('Correlation Heatmap of
Features') plt.show()
```

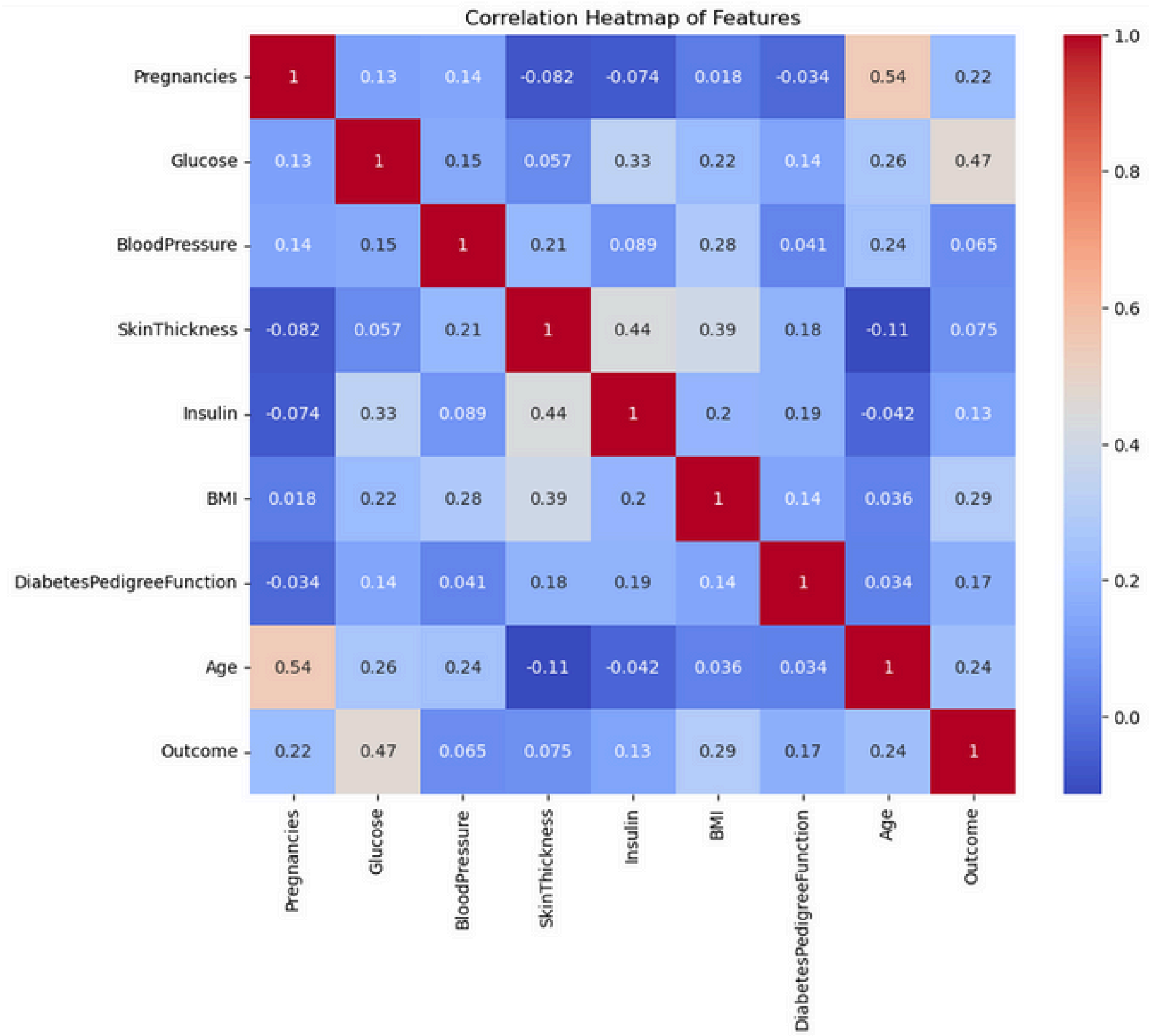
```
   Pregnancies Glucose BloodPressure SkinThickness Insulin       BMI \
0            6       148            72             35          0 33.6
1            1        85            66             29          0 26.6
2            8       183            64              0          0 23.3
3            1        89            66             23         94 28.1
4            0      137            40             35        168 43.1

        DiabetesPedigreeFunction Age Outcome
0                     0.627   50          1
1                     0.351   31          0
2                     0.672   32          1
3                     0.167   21          0
4                     2.288   33         1
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count Dtype -
--- ------                    ------------- -----
 0   Pregnancies    GlucosB7l6o8SkinThickness ulrl64
 1   DiabetesPedigreeFunctionsul7687 6n8o-nuBlMilt64
 2   onu-lnlu  ll  768  non-null  768  non-null  int64
 3   768 non-null                              int64
 4                                             int64
 5                                             float64
 6                                             float64
 7   Age                       768   non-null   int64
 8   Outcome                   768 non-null    int64
dtypes: float64(2), int64(7)
memory usage: 54.1 KB None

count mPeraeng nasntcdi emsi n 25G%l u5c0o%se BloodPressure SkinThicknessulin \
75% max    768.000000 768.000000     768.000000     768.000000 768.000000
             3.845052 120.894531      69.105469     20.536458    79.799479
         3.369578 10.907020601080      19.355807      15.952218 115.244002
         1.000000      03.000000       0.000000       0.000000    0.000000
         117.000000  99.60.00000 0    62.000000       0.000000    0.000000
         140.250000    17.000000      72.000000     23.000000    30.500000
         199.000000                   80.000000      32.000000 127.250000
                                    122.000000       99.000000 846.000000

              BMI DiabetesPedigreeFunction          Age        Outcome
count 768.000000                         768.000000 768.000000 768.000000
mean std 3m1i.n9 925%7 850% 75%    0.471876   33.240885       0.348958
max  Pregn7a.n8c8i4e1s6 0Glucose   0.331329   11.760232       0.476951
BloodPressu0r.e000000              0.078000   21.000000       0.000000
SkinThickn2e7s.s3 00I0n0s0ulin BMI 0.243750   24.000000       0.000000
DiabetesPe3d2i.g0r0e0e0F0u0nction  0.372500   29.000000       0.000000
Age Outcom3e6 .d6t0y0p0e0:0 int64  0.626250   41.000000       1.000000
        67.100000                  2.420000   81.000000      1.000000
                           0
                           0
                           0
                           0
                           0
                           0
                           0
                           0
                           0
```

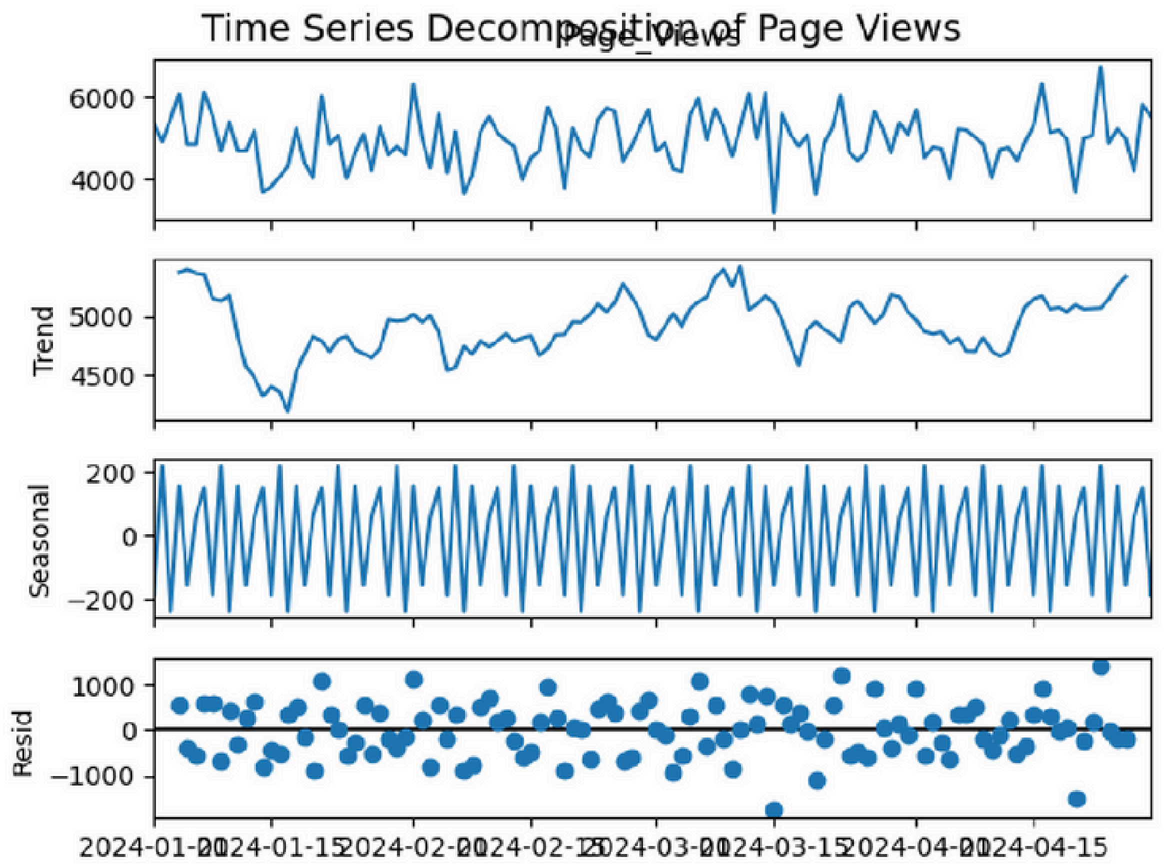


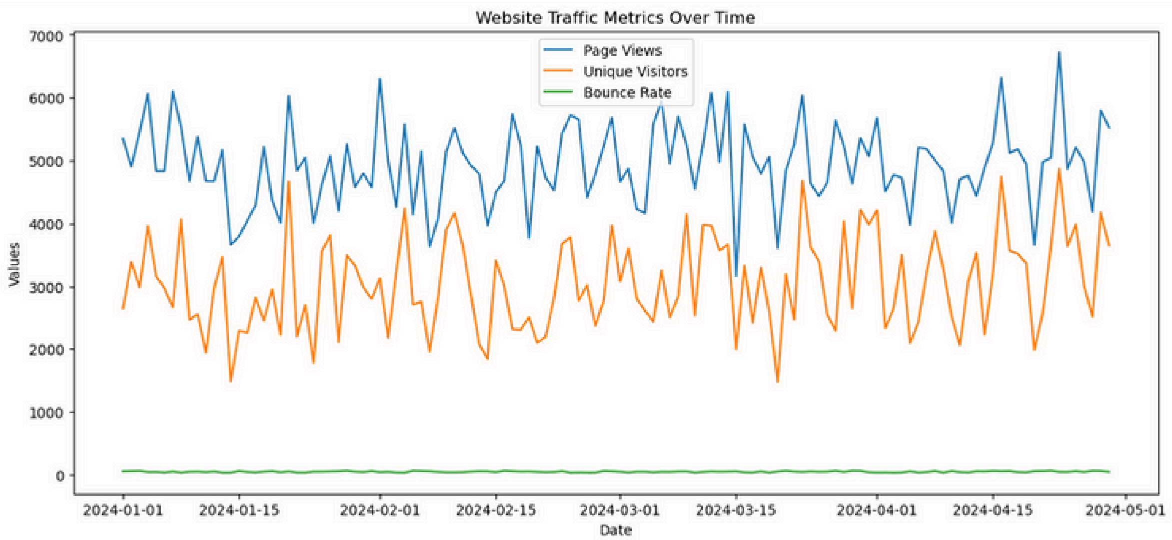


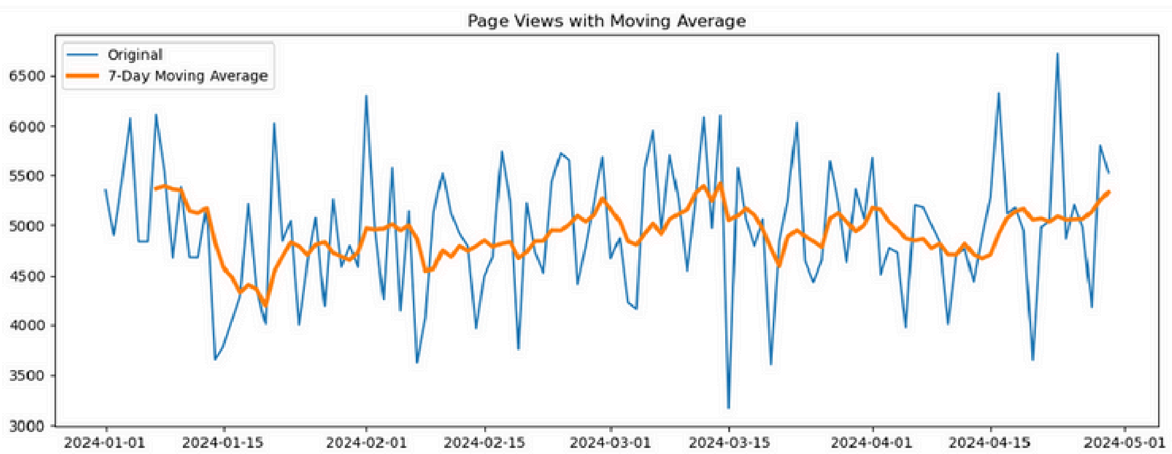
2.Write a Python program to perform Time Series Analysis on a website traffic dataset. The program should load and clean the dataset, decompose the time series to identify trends and seasonality, visualize key metrics using moving averages and seasonal plots, and detect anomalies using statistical methods.

In [7]:
```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from statsmodels.tsa.seasonal import seasonal_decompose
from scipy import stats
df = pd.read_csv('website_traffic.csv', parse_dates=['Date'])
df = df.sort_values('Date')
df.set_index('Date', inplace=True)
df = df.interpolate()
decomposition = seasonal_decompose(df['Page_Views'], model='additive', period=7)
plt.figure(figsize=(12, 8))
decomposition.plot()
plt.suptitle('Time Series Decomposition of Page Views', fontsize=14)
plt.show()
plt.figure(figsize=(14, 6))
plt.plot(df.index, df['Page_Views'], label='Page Views')
plt.plot(df.index, df['Unique_Visitors'], label='Unique Visitors')
plt.plot(df.index, df['Bounce_Rate'], label='Bounce Rate')
plt.title('Website Traffic Metrics Over Time')
plt.xlabel('Date')
plt.ylabel('Values')
plt.legend()
plt.show()
df['PageViews_MA7'] = df['Page_Views'].rolling(window=7).mean()
plt.figure(figsize=(14, 5))
plt.plot(df.index, df['Page_Views'], label='Original')
plt.plot(df.index, df['PageViews_MA7'], label='7-Day Moving Average', linewidth=
plt.title('Page Views with Moving Average')
plt.legend()
plt.show()
```

```
<Figure size 1200x800 with 0 Axes>
```







3. Random Sampling and Sampling Distribution To explore random sampling from a population and understand the concept of sampling distribution using Python in

Jupyter Notebook. Steps:

1. Generate a Population: ◯ Create a population of data with a specified distribution (e.g., normal distribution).
2. Random Sampling: ◯ Perform random sampling from the population to create multiple samples of different sizes. ◯ Compute sample statistics (mean, standard deviation, etc.) for each sample.
3. Sampling Distribution: ◯ Plot histograms or density plots of sample statistics (e.g., sample means). ◯ Compare the sampling distribution of the sample statistic (mean) with the population distribution.
4. Central Limit Theorem (Optional): ◯ Demonstrate the Central Limit Theorem by showing that as sample size increases, the sampling distribution of the sample mean approaches a normal distribution regardless of the population distribution.

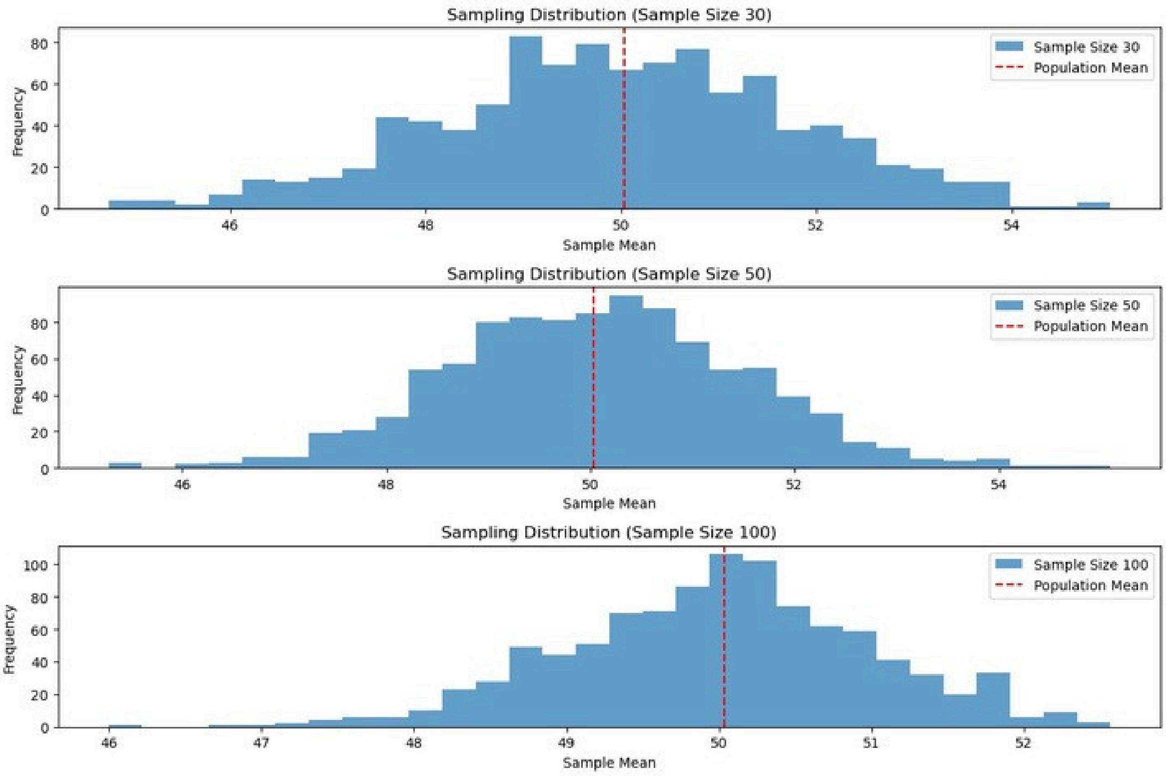
In [6]:

```
import numpy as np
import matplotlib.pyplot as plt
population_mean = 50
population_std = 10
population_size = 100000
population = np.random.normal(population_mean, population_std, population_size)
sample_sizes = [30, 50, 100]
num_samples = 1000
sample_means = {}
for size in sample_sizes:

    sample_means[size] = []
    for _ in range(num_samples):
        sample = np.random.choice(population, size=size, replace=False)
        mean = np.mean(sample)
        sample_means[size].append(mean)
plt.figure(figsize=(12, 8))
for i, size in enumerate(sample_sizes):
    plt.subplot(len(sample_sizes),   1,   i   +   1)   plt.hist(sample_means[size],
    bins=30, alpha=0.7, label=f'Sample Size {size}')  plt.axvline(np.mean(population), color='red', linestyle='dashed',
    linewidth=   plt.title(f'Sampling   Distribution   (Sample   Size   {size})')
    plt.xlabel('Sample Mean') plt.ylabel('Frequency') plt.legend()



plt.tight_layout()
plt.show()
```


Sampling Distribution (Sample Size 30)
Sample Size 30
Population Mean
Frequency
Sample Mean
Sampling Distribution (Sample Size 50)
Sample Size 50
Population Mean
Frequency
Sample Mean
Sampling Distribution (Sample Size 100)
Sample Size 100
Population Mean
Frequency
Sample Mean

In [ ]: