```
In [3]: import pandas as pd
        import random
        from faker import Faker
        from datetime import date, timedelta
```

```
In [11]: import pandas as pd
         import random
         from faker import Faker
         from datetime import date, timedelta

         fake = Faker()

         data = []

         for _ in range(1000):  # Adjust the number of rows as needed
             age = random.randint(15, 80)
             disease = random.choice(['Eyesight Issues', 'Asthma', 'Diabetes', 'High
             data.append([
                 fake.unique.random_int(min=100000, max=999999),
                 age,
                 random.choice(['Male', 'Female', 'Non-binary']),
                 random.choice(['Caucasian', 'African American', 'Hispanic', 'Other']
                 fake.city(),
                 disease,
                 fake.date_between(start_date=date(age - 1, 1, 1), end_date='today'),
                 random.randint(0, 10),
                 random.randint(0, 5),
                 random.randint(0, 15),
                 random.choice(['Medication', 'Physical Therapy', 'Other']),
                 random.choice(['Recovery', 'Relapse', 'Ongoing Treatment']),
                 fake.date_between(start_date='-5y', end_date='today'),
                 round(random.uniform(1000, 50000), 2),
                 random.choice(['Yes', 'No'])
             ])

         columns = [
             'PatientID', 'Age', 'Gender', 'Ethnicity', 'Location',
             'DiseaseType', 'DiagnosisDate', 'HospitalVisits', 'ERVisits',
             'DoctorVisits', 'TreatmentType', 'TreatmentOutcome', 'TreatmentDate',
             'HealthcareCosts', 'InsuranceCoverage'
         ]
         df = pd.DataFrame(data, columns=columns)
         df.to_csv('HealthcareAnalyticsDatasetmain.csv', index=False)
         missing_percentage = 0.1  # 10% missing values
         mask = np.random.rand(*df.shape) < missing_percentage
         df[mask] = np.nan
         df = pd.DataFrame(data, columns=columns)
```

DATA CLEANING

Data Cleaning here is done for the purpose of handling missing values ,duplicate values.

In [12]:
```
import pandas as pd

# Load your healthcare dataset (replace 'your_dataset.csv' with your actual
df = pd.read_csv('HealthcareAnalyticsDatasetmain.csv')

# Handling missing values (impute with mean for numerical columns)
df['Age'].fillna(df['Age'].mean(), inplace=True)
df['HealthcareCosts'].fillna(df['HealthcareCosts'].mean(), inplace=True)

# Convert categorical columns to categorical data types
df['Gender'] = df['Gender'].astype('category')
df['Ethnicity'] = df['Ethnicity'].astype('category')
df['DiseaseType'] = df['DiseaseType'].astype('category')
df['InsuranceCoverage'] = df['InsuranceCoverage'].astype('category')

# Check for and remove duplicates
df.drop_duplicates(inplace=True)

# Data exploration and validation
print(df.info())  # Check data types and missing values
print(df.describe())  # Summary statistics
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000 entries, 0 to 999
Data columns (total 15 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   PatientID          1000 non-null   int64
 1   Age                1000 non-null   int64
 2   Gender             1000 non-null   category
 3   Ethnicity          1000 non-null   category
 4   Location           1000 non-null   object
 5   DiseaseType        1000 non-null   category
 6   DiagnosisDate      1000 non-null   object
 7   HospitalVisits     1000 non-null   int64
 8   ERVisits           1000 non-null   int64
 9   DoctorVisits       1000 non-null   int64
 10  TreatmentType      1000 non-null   object
 11  TreatmentOutcome   1000 non-null   object
 12  TreatmentDate      1000 non-null   object
 13  HealthcareCosts    1000 non-null   float64
 14  InsuranceCoverage  1000 non-null   category
dtypes: category(4), float64(1), int64(5), object(5)
memory usage: 90.8+ KB
None
           PatientID          Age  HospitalVisits     ERVisits  DoctorVisits  \
count    1000.000000  1000.000000     1000.000000  1000.000000   1000.000000
mean   547134.814000    47.562000        4.963000     2.545000      7.560000
std    258048.999811    19.095623        3.219005     1.710819      4.587532
min    100647.000000    15.000000        0.000000     0.000000      0.000000
25%    330752.000000    30.000000        2.000000     1.000000      4.000000
50%    542617.000000    48.000000        5.000000     3.000000      8.000000
75%    764732.000000    64.000000        8.000000     4.000000     11.000000
max    999769.000000    80.000000       10.000000     5.000000     15.000000

       HealthcareCosts
count      1000.000000
mean      26220.433230
std       13878.288683
min        1098.880000
25%       14479.732500
50%       26608.555000
75%       37954.207500
max       49892.120000
```

Disease Prevalence for the generated data.

What is Disease prevalance?

Disease prevalence is a measure that tells us how common a particular disease or health condition is within a specific population at a given point in time. Consider a town with 1,000 residents, and we want to know how common diabetes is in this town. You find that 100 people in the town have been diagnosed with diabetes. These

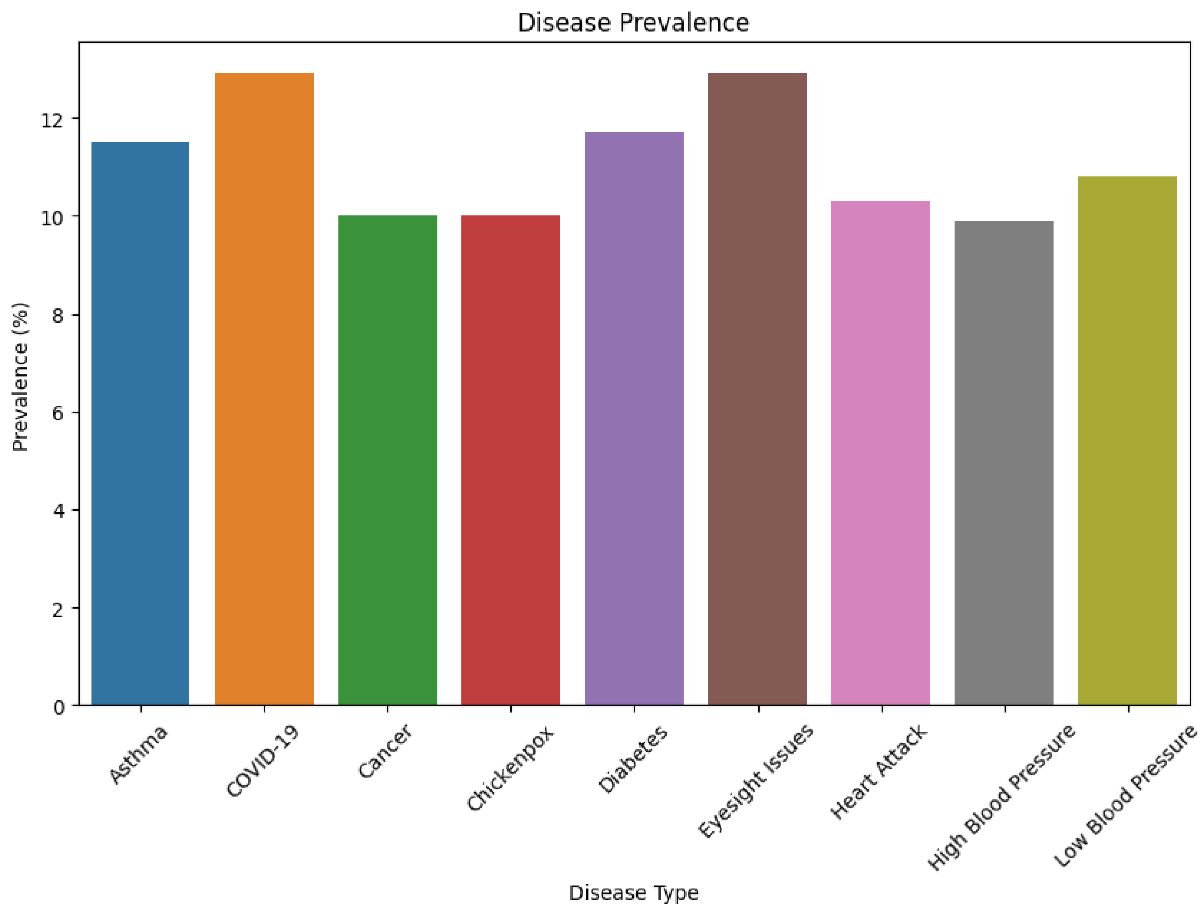
Loading [MathJax]/extensions/Safe.js **considered "cases" of diabetes. Disease Prevalence (%) = (Number of**

Cases / Total Population) x 100 Disease Prevalence (%) = (100 / 1,000) x 100 = 10% So, the disease prevalence of diabetes in the town is 10%. This means that 10% of the town's population has diabetes.

In [13]:
```
disease_prevalence = df.groupby('DiseaseType')['PatientID'].count() / len(df
```

In [14]:
```
import matplotlib.pyplot as plt
import seaborn as sns

# Create a bar chart for disease prevalence
plt.figure(figsize=(10, 6))
sns.barplot(x=disease_prevalence.index, y=disease_prevalence.values)
plt.title('Disease Prevalence')
plt.xlabel('Disease Type')
plt.ylabel('Prevalence (%)')
plt.xticks(rotation=45)
plt.show()
```



Relation between Age and HelathCare Costs.

To calculate how the age and healthcare costs are related we use a concept of data science and stats called CORREALATION. It statistical measure that expresses the

Loading [MathJax]/extensions/Safe.js

extent to which two variables are linearly related.

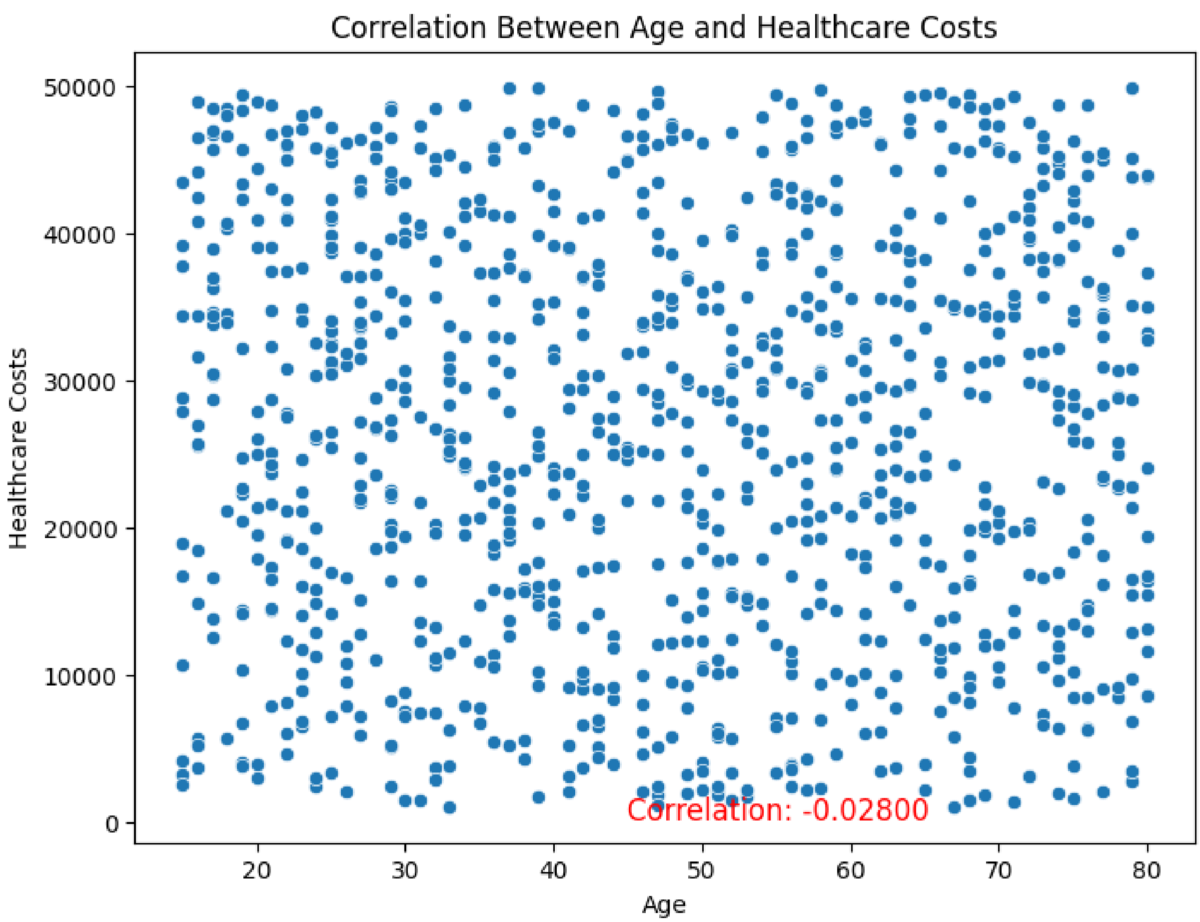
In [15]:

```
import numpy as np

# Calculate the correlation coefficient between age and healthcare costs
correlation_agecosts = np.corrcoef(df['Age'], df['HealthcareCosts'])[0, 1]

# Create a scatter plot to visualize the relationship
plt.figure(figsize=(8, 6))
sns.scatterplot(x='Age', y='HealthcareCosts', data=df)
plt.title('Correlation Between Age and Healthcare Costs')
plt.xlabel('Age')
plt.ylabel('Healthcare Costs')

# Add correlation coefficient as text on the plot
plt.text(45, 200, f'Correlation: {correlation_agecosts:.5f}', fontsize=12, c
plt.show()
```



Here we see that we get a negative correaltion. A negative correlation between age and healthcare costs in the analysis means that as individuals get older (their age increases), their healthcare costs tend to decrease or vice versa. In other words, there is an inverse relationship between age and healthcare costs in the dataset.

Analysis: Healthcare Costs by Disease Type

Now we will explore the distribution of healthcare costs across different disease types and visualize it using box plots. Additionally, we'll perform an analysis of variance (ANOVA) to test if there are significant differences in healthcare costs among disease types What is ANOVA and why are we using here?

ANOVA, or Analysis of Variance, is a statistical technique used to analyze the differences among group means in a dataset. It is primarily employed when you want to compare the means of three or more groups to determine if there are statistically significant differences between them.

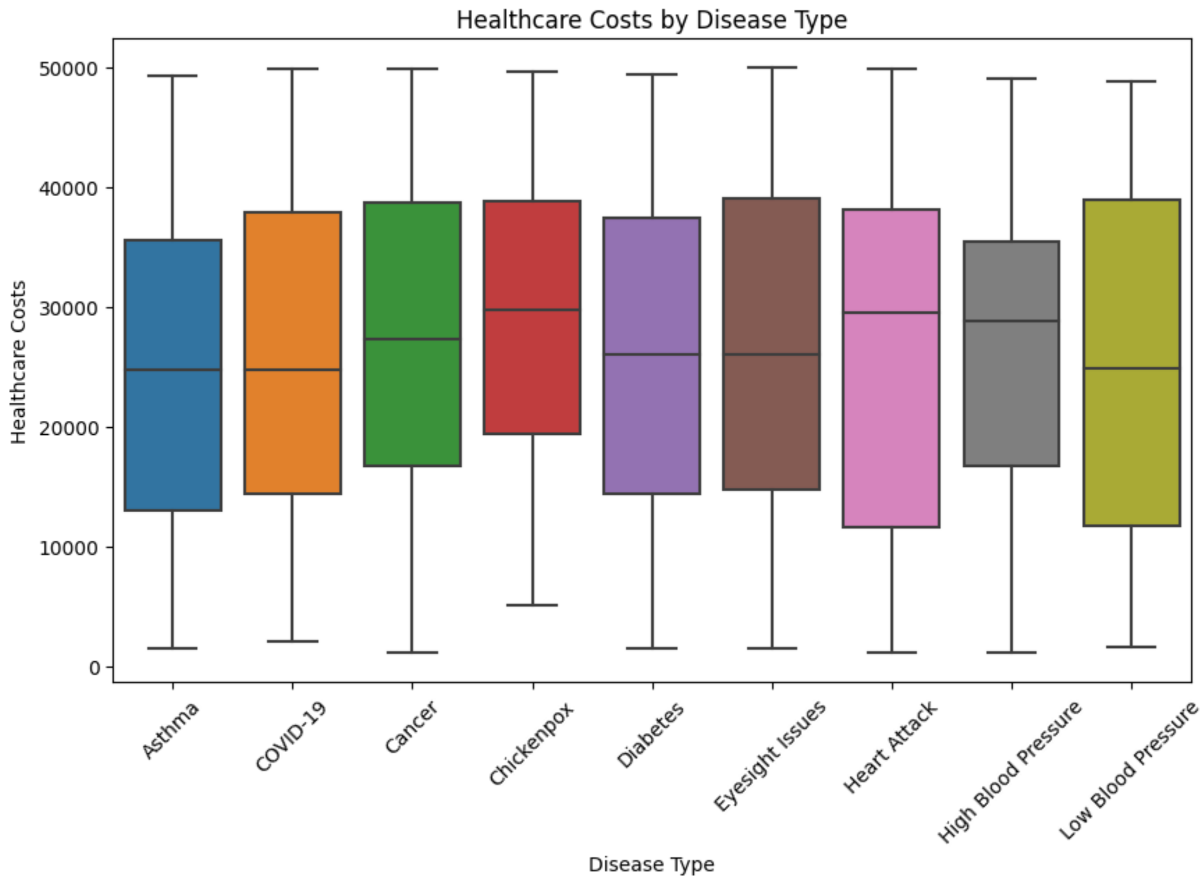
In [16]:

```
import scipy.stats as stats

# Create box plots to visualize healthcare costs by disease type
plt.figure(figsize=(10, 6))
sns.boxplot(x='DiseaseType', y='HealthcareCosts', data=df)
plt.title('Healthcare Costs by Disease Type')
plt.xlabel('Disease Type')
plt.ylabel('Healthcare Costs')
plt.xticks(rotation=45)
plt.show()

# Perform Analysis of Variance (ANOVA)
anova_result = stats.f_oneway(
    df['HealthcareCosts'][df['DiseaseType'] == 'Diabetes'],
    df['HealthcareCosts'][df['DiseaseType'] == 'Heart Disease'],
    df['HealthcareCosts'][df['DiseaseType'] == 'Cancer'],
    df['HealthcareCosts'][df['DiseaseType'] == 'COVID-19']
)

# Display ANOVA results
print("ANOVA p-value:", anova_result.pvalue)
```




```
ANOVA p-value: nan
```

```
C:\Users\ganes\AppData\Local\Programs\Python\Python311\Lib\site-packages\scip
y\stats\_stats_py.py:4133: DegenerateDataWarning: at least one input has leng
th 0
  warnings.warn(stats.DegenerateDataWarning('at least one input '
```

In [17]:
```
print(df['DiseaseType'].value_counts())
```

```
DiseaseType
COVID-19               129
Eyesight Issues        129
Diabetes               117
Asthma                 115
Low Blood Pressure     108
Heart Attack           103
Cancer                 100
Chickenpox             100
High Blood Pressure     99
Name: count, dtype: int64
```

What is output and how boxplot gives the required output?

A box plot, also known as a box-and-whisker plot, is a graphical representation that displays the distribution of a dataset, including its central tendency, spread, and the presence of outliers. In the context of healthcare costs for particular diseases, a box plot can show how the costs vary within each disease category. Here's how to interpret a box plot for healthcare costs by disease type

Loading [MathJax]/extensions/Safe.js

The box represents the interquartile range (IQR) for that specific disease type. The median line inside the box indicates the median healthcare cost for that disease. The whiskers show the data range for that disease, excluding outliers. Any individual data points (dots) beyond the whiskers are potential outliers.

Impact of Insurance Coverage on Healthcare costs

Now we'll compare healthcare costs for patients with insurance coverage versus those without coverage and assess whether there are significant differences. We create box plots to visualize the distribution of healthcare costs for patients with insurance coverage ("Yes") and those without coverage ("No"). This visualization helps compare the central tendency and spread of healthcare costs for the two groups.

We perform a two-sample t-test (stats.ttest_ind) to compare healthcare costs between insured and uninsured patients. The t-test evaluates whether there is a significant difference in means between the two groups.We calculate the test statistic (t) and the p-value associated with the t-test. Based on the p-value, we interpret the results. If the p-value is less than 0.05 (commonly chosen significance level), we conclude that there is a significant difference in healthcare costs between insured and uninsured patients. Otherwise, we conclude that there is no significant difference.

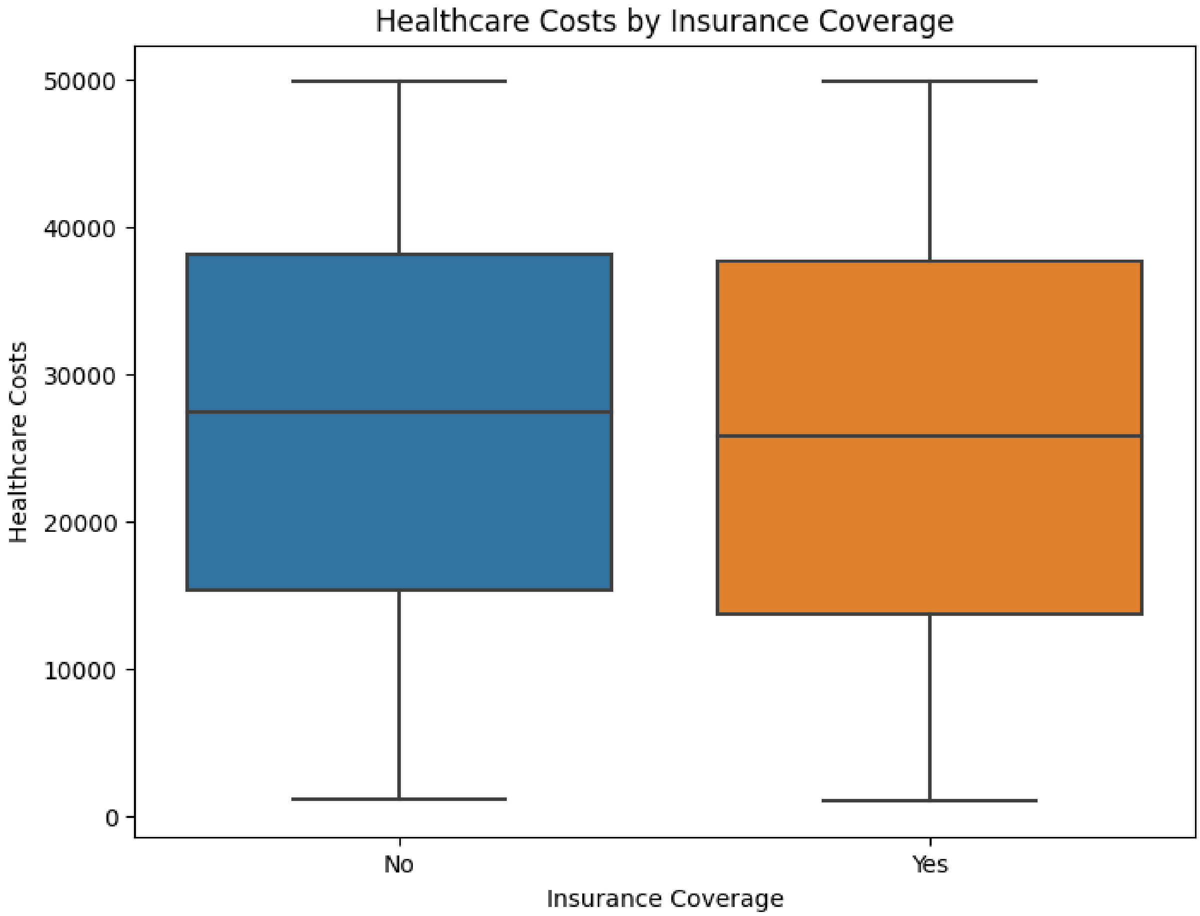
In [18]:
```
# box plots to visualize healthcare costs by insurance coverage
plt.figure(figsize=(8, 6))
sns.boxplot(x='InsuranceCoverage', y='HealthcareCosts', data=df)
plt.title('Healthcare Costs by Insurance Coverage')
plt.xlabel('Insurance Coverage')
plt.ylabel('Healthcare Costs')
plt.show()

# t-test to compare healthcare costs between insured and uninsured patients
insured_costs = df['HealthcareCosts'][df['InsuranceCoverage'] == 'Yes']
uninsured_costs = df['HealthcareCosts'][df['InsuranceCoverage'] == 'No']

t_statistic, p_value = stats.ttest_ind(insured_costs, uninsured_costs)

# t-test results
print(f'Test Statistic (t): {t_statistic:.2f}')
print(f'p-value: {p_value:.3f}')

# Results
if p_value < 0.05:
    print('There is a significant difference in healthcare costs between ins
else:
    print('There is no significant difference in healthcare costs between ir
```




```
Test Statistic (t): -0.83
p-value: 0.406
There is no significant difference in healthcare costs between insured and un
insured patients.
```

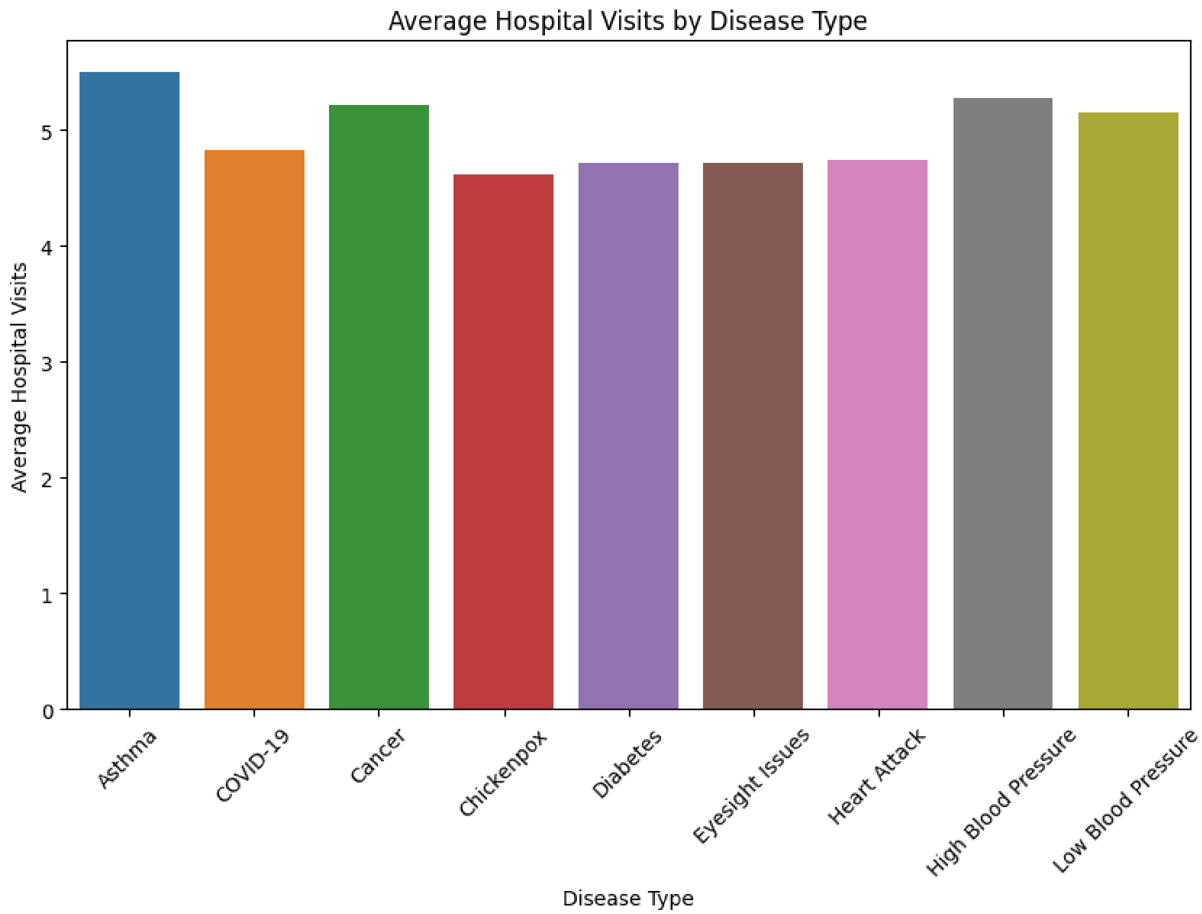
Average Hospital Visits by Disease types

In [19]:

```
average_hospital_visits = df.groupby('DiseaseType')['HospitalVisits'].mean()

plt.figure(figsize=(10, 6))
sns.barplot(x='DiseaseType', y='HospitalVisits', data=average_hospital_visit
plt.title('Average Hospital Visits by Disease Type')
plt.xlabel('Disease Type')
plt.ylabel('Average Hospital Visits')
plt.xticks(rotation=45)
plt.show()
```

Loading [MathJax]/extensions/Safe.js



Relationship Between Age and Doctor Visits for Specific Diseases (Diabetes and Heart Attack)

We create age groups (e.g., 20-29, 30-39, etc.) to categorize patients based on their age.

Now assign each patient to an appropriate age group using pd.cut.

Now filter the dataset to include only patients with "Diabetes" or "Heart Attack" as their disease type.

We can now calculate the average healthcare costs for each combination of age group and disease type.

Finally, we create a grouped bar plot to visualize the comparison of average healthcare costs between "Diabetes" and "Heart Attack" patients within different age groups.

This plot provides a clear comparison of healthcare costs between the two disease types for various age categories, helping you identify age-related patterns in

healthcare spending for these specific conditions.

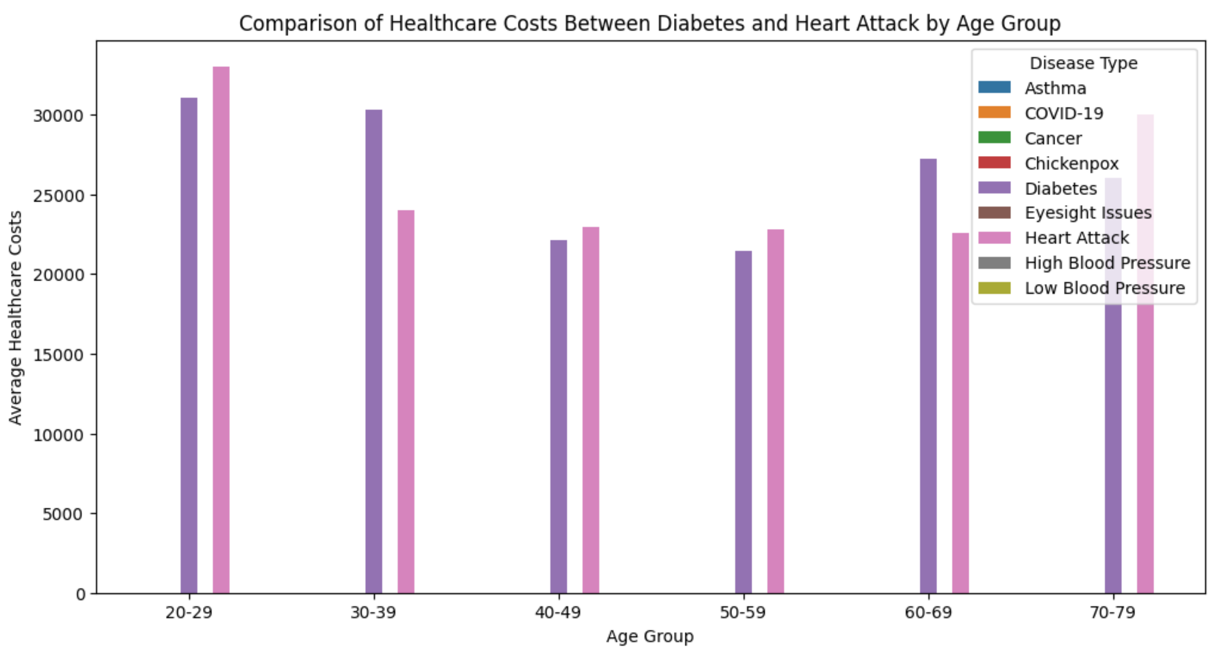
In [20]:
```
# Create age groups (e.g., 20-29, 30-39, etc.)
age_bins = [20, 30, 40, 50, 60, 70, 80]
age_labels = ['20-29', '30-39', '40-49', '50-59', '60-69', '70-79']

# Assign each patient to an age group
df['AgeGroup'] = pd.cut(df['Age'], bins=age_bins, labels=age_labels)

# Filter the dataset for patients with Diabetes or Heart Attack
selected_disease_types = ['Diabetes', 'Heart Attack']
filtered_df = df[df['DiseaseType'].isin(selected_disease_types)]

# Calculate the average healthcare costs for each combination of age group a
average_costs_by_age_group = filtered_df.groupby(['AgeGroup', 'DiseaseType']

# Create a grouped bar plot to visualize the comparison
plt.figure(figsize=(12, 6))
sns.barplot(x='AgeGroup', y='HealthcareCosts', hue='DiseaseType', data=avera
plt.title('Comparison of Healthcare Costs Between Diabetes and Heart Attack 
plt.xlabel('Age Group')
plt.ylabel('Average Healthcare Costs')
plt.legend(title='Disease Type')
plt.show()
```



In [ ]:

Loading [MathJax]/extensions/Safe.js