In [1]:
```
# import the needed libraries
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from statsmodels.tsa.seasonal import seasonal_decompose
from statsmodels.tsa.stattools import adfuller
import statsmodels.api as sm
```

```
C:\Users\HELLO\anaconda3\Lib\site-packages\pandas\core\arrays\masked.py:60: UserWarning: Pandas requires
version '1.3.6' or newer of 'bottleneck' (version '1.3.5' currently installed).
  from pandas.core import (
```

In [2]:
```
# Load the dataset
data = pd.read_csv("drug-resistant-tb-cases.csv")
data.columns = ['Year', 'Diagnosed', 'Enrolled']
data['Year'] = pd.to_datetime(data['Year'], format='%Y')
data.set_index('Year', inplace=True)
```

In [3]:
```
# Show the first 13 rows of the dataset
data.head(13)
```

Out [3]:

Year	Diagnosed	Enrolled
2010-01-01	25	23
2011-01-01	39	27
2012-01-01	185	154
2013-01-01	665	345
2014-01-01	781	412
2015-01-01	1241	656
2016-01-01	1686	1251
2017-01-01	2286	1786
2018-01-01	2275	1895
2019-01-01	2384	1975
2020-01-01	2061	1492
2021-01-01	2975	2197
2022-01-01	3932	3185

DATA CLEANING

In [4]:
```
# # To know the dataset columns
data.columns
```

Out [4]:
```
Index(['Diagnosed', 'Enrolled'], dtype='object')
```

In [5]:
```
# To know the shape or size of the dataset
data.shape
```

Out [5]:
```
(13, 2)
```

In [6]:
```
# To know the data types and get familiarise with the data to know how to work with them.
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 13 entries, 2010-01-01 to 2022-01-01
Data columns (total 2 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   Diagnosed   13 non-null     int64
 1   Enrolled    13 non-null     int64
dtypes: int64(2)
memory usage: 312.0 bytes
```

In [7]:
```
# To check for misssing value
data.isnull().sum()
```

Out [7]:
```
Diagnosed    0
Enrolled     0
dtype: int64
```

EXPLORATORY DATA ANALYSIS (EDA)

In [8]:
```
# To get the summary statistics of the data
data.describe()
```

Out [8]:

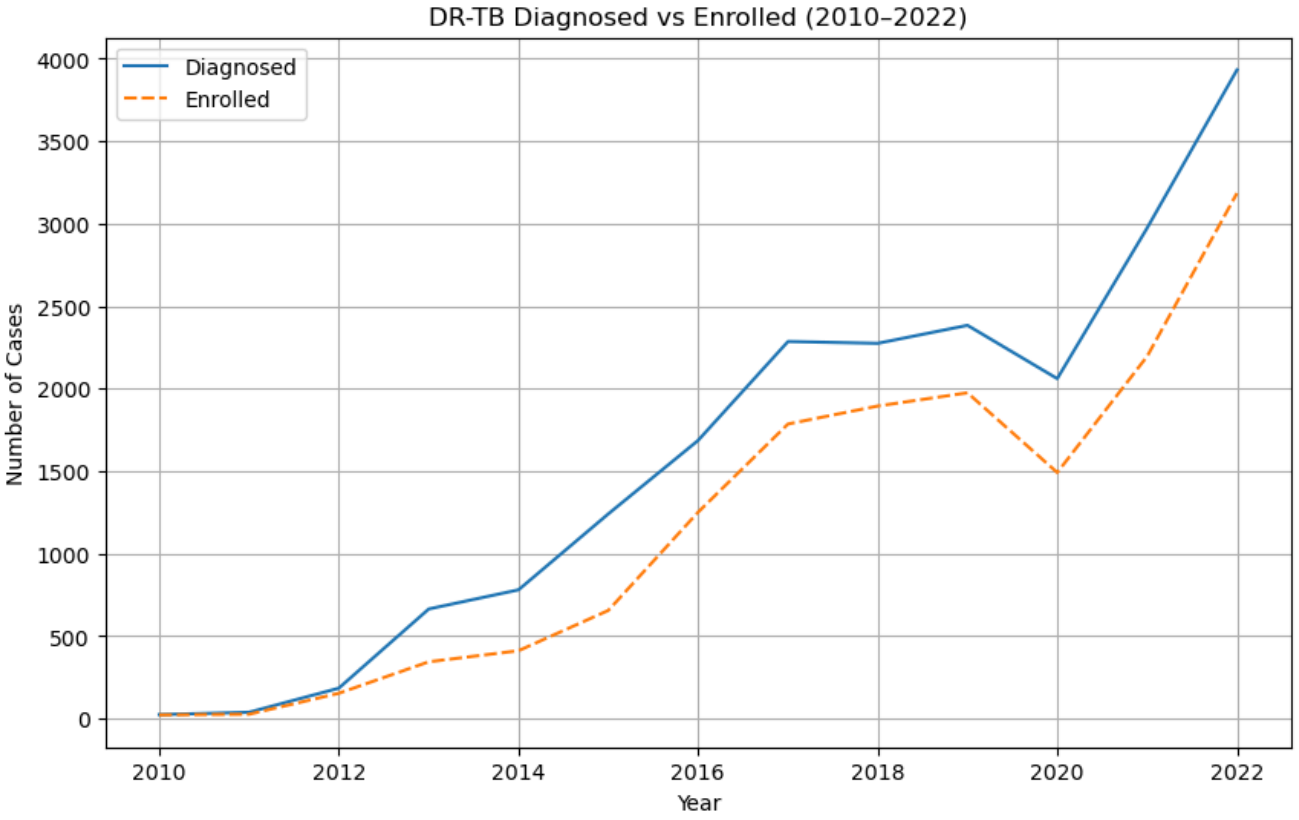
	Diagnosed	Enrolled
count	13.000000	13.000000
mean	1579.615385	1184.461538
std	1213.729757	997.884647
min	25.000000	23.000000
25%	665.000000	345.000000
50%	1686.000000	1251.000000
75%	2286.000000	1895.000000
max	3932.000000	3185.000000

In [9]:
```
# To Compute treatment gap and coverage
data['Gap'] = data['Diagnosed'] - data['Enrolled']
data['Treatment Coverage (%)'] = (data['Enrolled'] / data['Diagnosed'] * 100).round(2)
print(data)
```

```
            Diagnosed  Enrolled  Gap  Treatment Coverage (%)
Year
2010-01-01         25        23    2                   92.00
2011-01-01         39        27   12                   69.23
2012-01-01        185       154   31                   83.24
2013-01-01        665       345  320                   51.88
2014-01-01        781       412  369                   52.75
2015-01-01       1241       656  585                   52.86
2016-01-01       1686      1251  435                   74.20
2017-01-01       2286      1786  500                   78.13
2018-01-01       2275      1895  380                   83.30
2019-01-01       2384      1975  409                   82.84
2020-01-01       2061      1492  569                   72.39
2021-01-01       2975      2197  778                   73.85
2022-01-01       3932      3185  747                   81.00
```

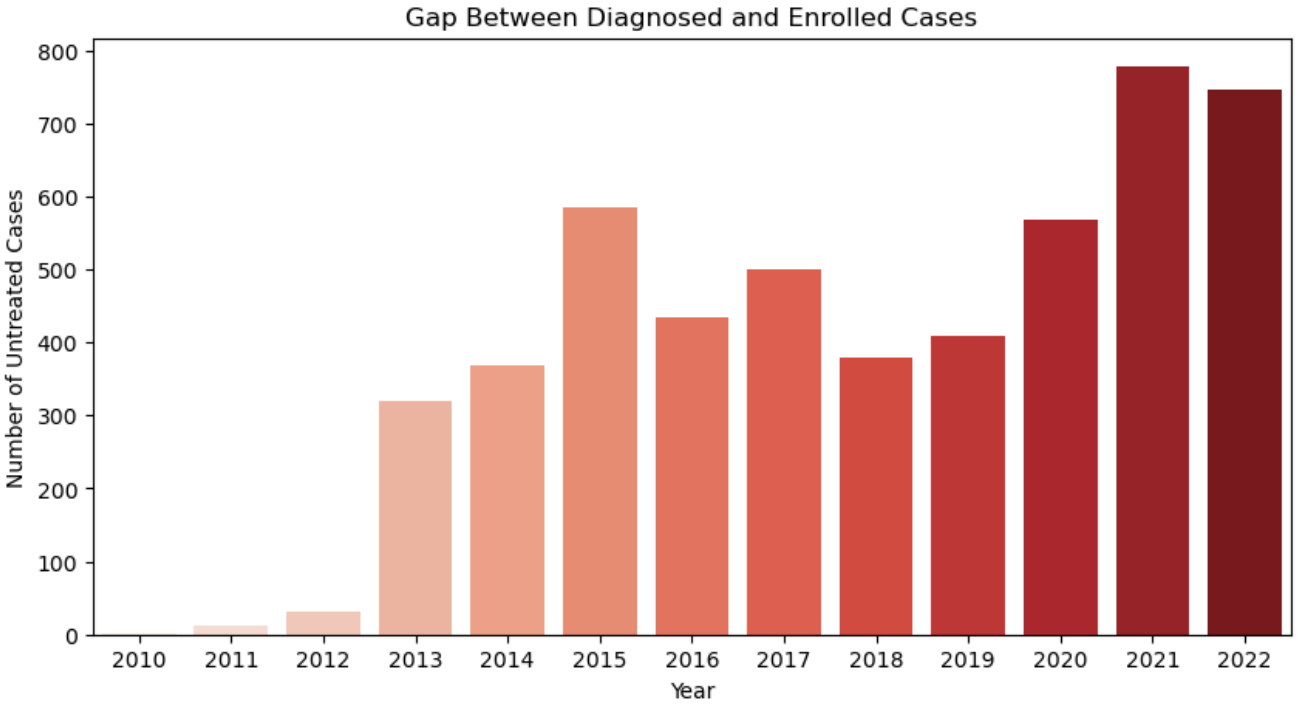
In [10]:
```
# Line plot for Diagnosed vs Enrolled
plt.figure(figsize=(10, 6))
sns.lineplot(data=data[['Diagnosed', 'Enrolled']])
plt.title("DR-TB Diagnosed vs Enrolled (2010–2022)")
plt.ylabel("Number of Cases")
plt.grid(True)
plt.show()
```

```
C:\Users\HELLO\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
C:\Users\HELLO\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119: FutureWarning: use_inf_as_na option is deprecated and will be removed in a future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



In [11]:

```
# Bar chart of treatment gap
plt.figure(figsize=(10, 5))
sns.barplot(x=data.index.year, y=data['Gap'], palette='Reds')
plt.title("Gap Between Diagnosed and Enrolled Cases")
plt.ylabel("Number of Untreated Cases")
plt.xlabel("Year")
plt.show()
```

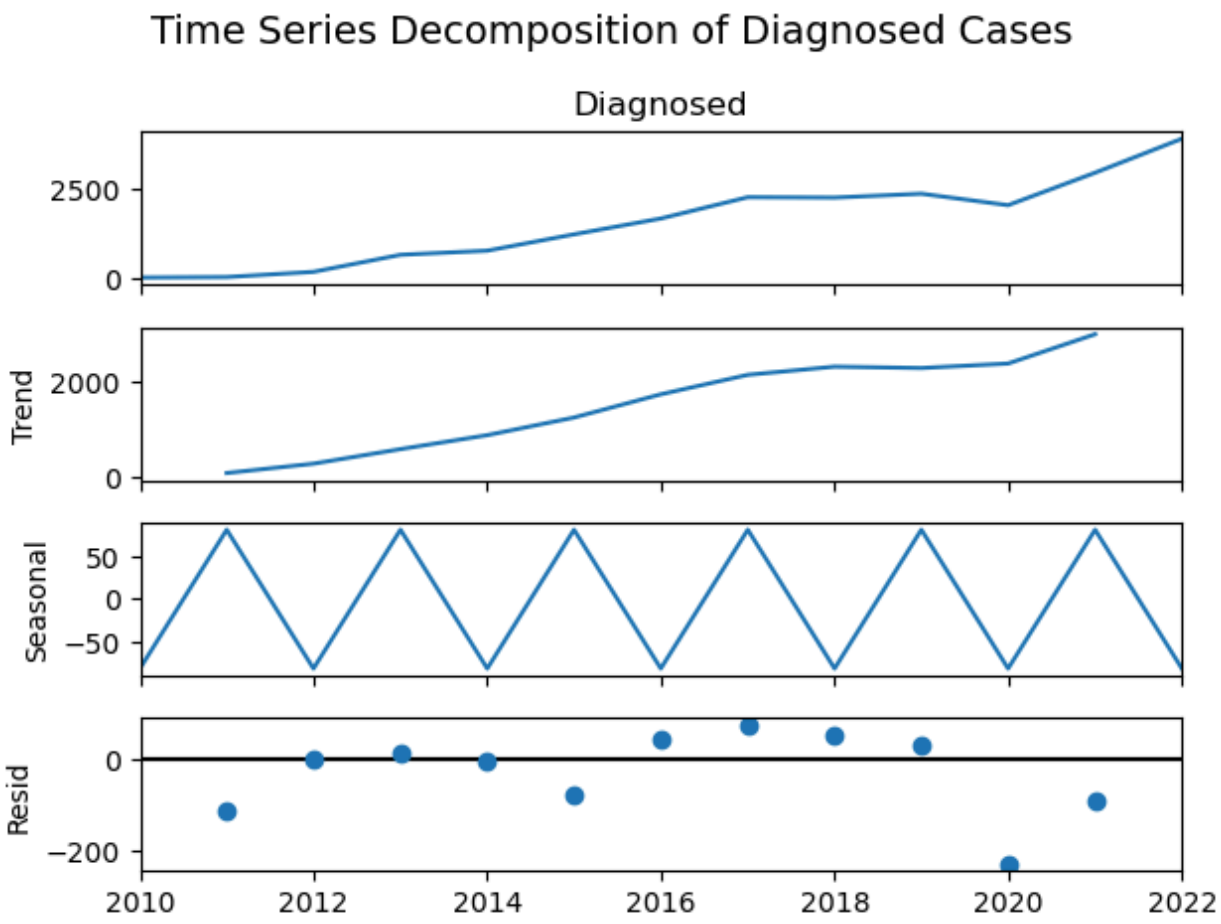


In [12]:

```
# Time Series Decomposition
decomposition = seasonal_decompose(data['Diagnosed'], model='additive', period=2)
decomposition.plot()
```

```
plt.suptitle("Time Series Decomposition of Diagnosed Cases", fontsize=14)
plt.tight_layout()
plt.show()
```

```
C:\Users\HELLO\AppData\Local\Temp\ipykernel_8116\55108308.py:5: UserWarning: The figure layout has changed to tight
  plt.tight_layout()
```



In [13]:
```
# To compute the Augmented Dickey-Fuller Test
adf_result = adfuller(data['Diagnosed'])
print("\nADF Test Results for Diagnosed Cases:")
print(f"ADF Statistic: {adf_result[0]:.4f}")
print(f"p-value: {adf_result[1]:.4f}")
for key, value in adf_result[4].items():
    print(f"Critical Value ({key}): {value:.4f}")
```

```
ADF Test Results for Diagnosed Cases:
ADF Statistic: 0.7879
p-value: 0.9914
Critical Value (1%): -4.4731
Critical Value (5%): -3.2899
Critical Value (10%): -2.7724
```

In [14]:
```
# To compute the First-order Differencing
data['Diagnosed_diff'] = data['Diagnosed'].diff()
data['Diagnosed_diff']
```

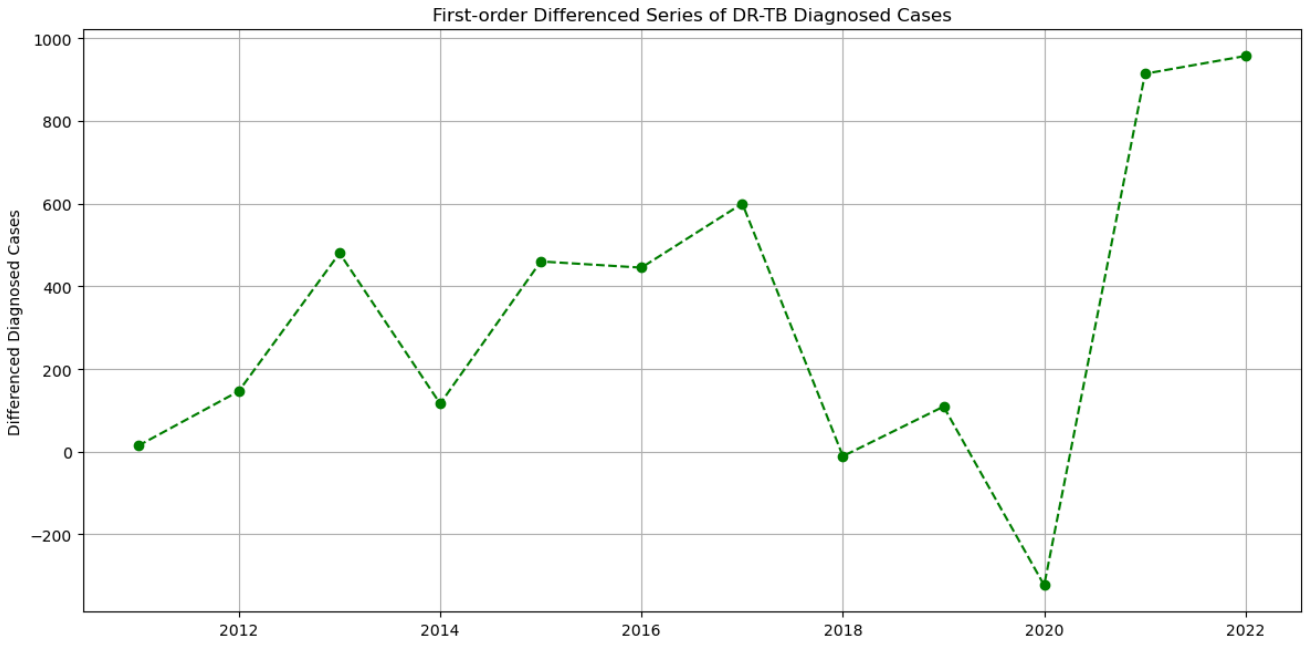
Out [14]:
```
Year
2010-01-01      NaN
2011-01-01     14.0
2012-01-01    146.0
2013-01-01    480.0
2014-01-01    116.0
2015-01-01    460.0
2016-01-01    445.0
2017-01-01    600.0
2018-01-01    -11.0
2019-01-01    109.0
2020-01-01   -323.0
2021-01-01    914.0
```

```
2022-01-01    957.0
Name: Diagnosed_diff, dtype: float64
```

In [15]:
```python
# To compute the ADF Test (Differenced Series)
print("\nADF Test (First Differenced Diagnosed Series):")
adf_result_diff = adfuller(data['Diagnosed_diff'].dropna())
print(f"ADF Statistic: {adf_result_diff[0]:.4f}")
print(f"p-value: {adf_result_diff[1]:.4f}")
for key, value in adf_result_diff[4].items():
    print(f"Critical Value ({key}): {value:.4f}")
```

```
ADF Test (First Differenced Diagnosed Series):
ADF Statistic: -24.6434
p-value: 0.0000
Critical Value (1%): -4.9387
Critical Value (5%): -3.4776
Critical Value (10%): -2.8439
```

In [16]:
```python
# To Plot Differenced Series
plt.figure(figsize=(12, 6))
plt.plot(data['Diagnosed_diff'], marker='o', linestyle='--', color='green')
plt.title("First-order Differenced Series of DR-TB Diagnosed Cases")
plt.ylabel("Differenced Diagnosed Cases")
plt.grid(True)
plt.tight_layout()
plt.show()
```



In [17]:
```python
# ARIMA Model (ARIMA(1,1,1))
model = sm.tsa.ARIMA(data['Diagnosed'], order=(1, 1, 1))
model_fit = model.fit()
print("\nARIMA Model Summary:")
print(model_fit.summary())
```

```
ARIMA Model Summary:
                               SARIMAX Results
==============================================================================
Dep. Variable:              Diagnosed   No. Observations:                   13
Model:                 ARIMA(1, 1, 1)   Log Likelihood                -89.535
Date:                Mon, 26 May 2025   AIC                            185.070
Time:                        12:22:48   BIC                            186.525
Sample:                    01-01-2010   HQIC                           184.531
                         - 01-01-2022
Covariance Type:                  opg
==============================================================================
                 coef    std err          z      P>|z|      [0.025      0.975]
------------------------------------------------------------------------------
ar.L1          0.9999      0.008    129.441      0.000       0.985       1.015
ma.L1         -0.9916      0.633     -1.565      0.118      -2.233       0.250
sigma2      1.494e+05   4.27e-06    3.5e+10      0.000    1.49e+05    1.49e+05
===================================================================================
Ljung-Box (L1) (Q):                   0.07   Jarque-Bera (JB):                 0.24
```

```
Prob(Q):                              0.79   Prob(JB):                         0.89
Heteroskedasticity (H):               7.84   Skew:                            -0.14
Prob(H) (two-sided):                  0.07   Kurtosis:                         2.36
================================================================================

Warnings:
[1] Covariance matrix calculated using the outer product of gradients (complex-step).
[2] Covariance matrix is singular or near-singular, with condition number 1.01e+27. Standard errors may be unstable.
```

```
C:\Users\HELLO\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency YS-JAN will be used.
  self._init_dates(dates, freq)
C:\Users\HELLO\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency YS-JAN will be used.
  self._init_dates(dates, freq)
C:\Users\HELLO\anaconda3\Lib\site-packages\statsmodels\tsa\base\tsa_model.py:473: ValueWarning: No frequency
information was provided, so inferred frequency YS-JAN will be used.
  self._init_dates(dates, freq)
C:\Users\HELLO\anaconda3\Lib\site-packages\statsmodels\base\model.py:607: ConvergenceWarning: Maximum
Likelihood optimization failed to converge. Check mle_retvals
  warnings.warn("Maximum Likelihood optimization failed to "
```

In [18]:

```
# To Forecast Next 10 Years
forecast = model_fit.get_forecast(steps=10)
forecast_data= forecast.conf_int()
forecast_data['Forecast'] = forecast.predicted_mean
forecast_data.index = pd.date_range(start='2023', periods=10, freq='Y')
forecast_data['Forecast']
```

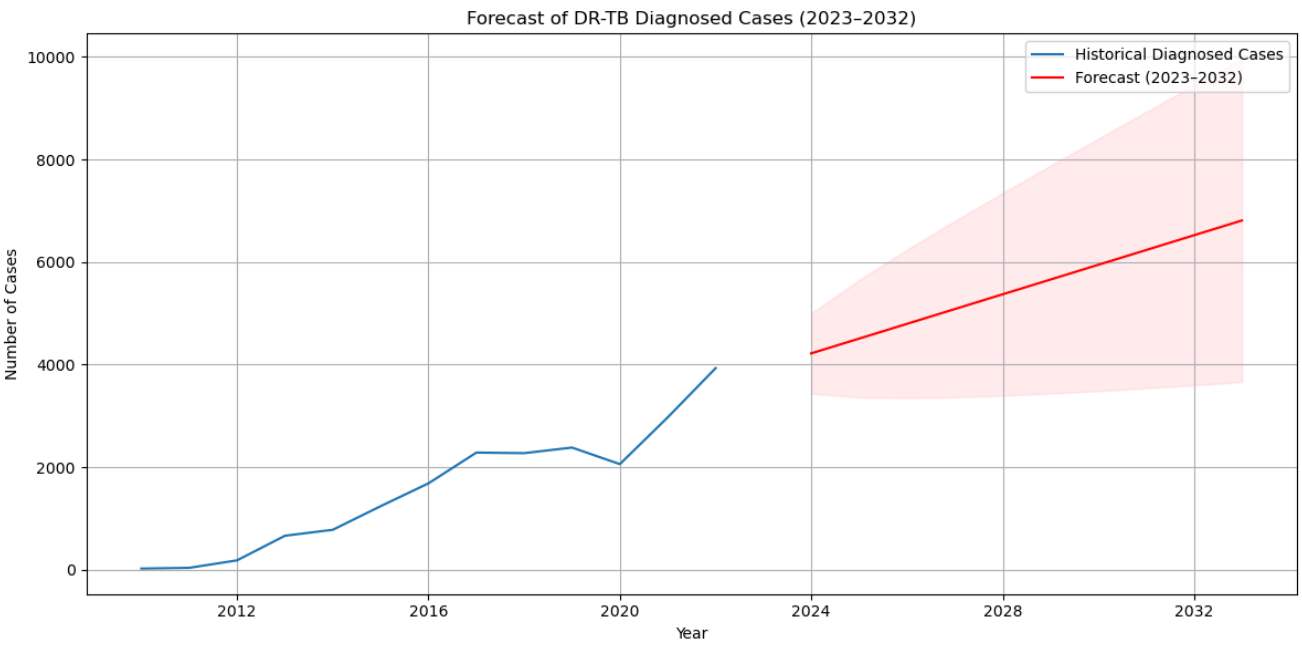
```
C:\Users\HELLO\AppData\Local\Temp\ipykernel_8116\776392553.py:5: FutureWarning: 'Y' is deprecated and will be
removed in a future version, please use 'YE' instead.
  forecast_data.index = pd.date_range(start='2023', periods=10, freq='Y')
```

Out [18]:

```
2023-12-31    4219.989455
2024-12-31    4507.964007
2025-12-31    4795.923654
2026-12-31    5083.868398
2027-12-31    5371.798240
2028-12-31    5659.713181
2029-12-31    5947.613221
2030-12-31    6235.498362
2031-12-31    6523.368603
2032-12-31    6811.223946
Freq: YE-DEC, Name: Forecast, dtype: float64
```

In [19]:

```
# To Plot Forecast
plt.figure(figsize=(12, 6))
plt.plot(data['Diagnosed'], label='Historical Diagnosed Cases')
plt.plot(forecast_data['Forecast'], label='Forecast (2023-2032)', color='red')
plt.fill_between(forecast_data.index,
                 forecast_data.iloc[:, 0],
                 forecast_data.iloc[:, 1],
                 color='pink', alpha=0.3)
plt.title("Forecast of DR-TB Diagnosed Cases (2023-2032)")
plt.xlabel("Year")
plt.ylabel("Number of Cases")
plt.legend()
plt.grid(True)
plt.tight_layout()
plt.show()
```


Forecast of DR-TB Diagnosed Cases (2023–2032)
Historical Diagnosed Cases
Forecast (2023–2032)
Number of Cases
Year
0
2000
4000
6000
8000
10000
2012
2016
2020
2024
2028
2032