```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit

# Step 1: Load data
data = pd.read_csv("C:\\Users\\scott\\OneDrive\\Desktop\\Code\\Python\\BTC Optimal
Leverage\\INDEX_BTCUSD, 1D_(2021-2024).csv", parse_dates=['Date_Time'])

# Step 1.1: Filter data
data = data[data['Date_Time'] > '2023-01-01']

# Step 2: Calculate daily returns, mean daily return, and daily volatility
data['Daily_Return'] = data['close'].pct_change()
mean_daily_return = data['Daily_Return'].mean()
daily_volatility = data['Daily_Return'].std()

# Step 3: Estimate compound daily growth rate adjusting for leverage and risk
def compound_daily_growth_rate(leverage, mean_daily_return, daily_volatility):
    k = leverage
    μ = mean_daily_return
    σ = daily_volatility
    R = k * μ - 0.5 * (k ** 2) * (σ ** 2) / (1 + k * μ)
    return R

leverage_values = np.arange(1, 15)
growth_rates = [compound_daily_growth_rate(leverage, mean_daily_return, daily_volatility) for
leverage in leverage_values]

# Step 4: Fit a quadratic curve to the data points
def quadratic_curve(x, a, b, c):
    return a * x ** 2 + b * x + c

popt, _ = curve_fit(quadratic_curve, leverage_values, growth_rates)

# Step 5: Plot the curve along with data points
plt.figure(figsize=(10, 6))
plt.plot(leverage_values, growth_rates, 'bo', label='Data Points')
plt.plot(leverage_values, quadratic_curve(leverage_values, *popt), 'r-', label='Quadratic Curve
Fit')
plt.title('Risk-Adjusted Return vs. Leverage')
plt.xlabel('Leverage')
plt.ylabel('Risk-Adjusted Return (%)')
plt.legend()
```

```
plt.grid(True)
plt.show()

# Step 6: Find the optimal leverage based on the peak of the curve
optimal_leverage = -popt[1] / (2 * popt[0])
max_growth_rate = quadratic_curve(optimal_leverage, *popt)
print(f'Optimal Leverage: {optimal_leverage:.2f}\nMax Growth Rate: {max_growth_rate:.2f}')
```