In [11]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from scipy.stats import chisquare
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
def extract_first_digits(data):
    """
    Extracts the first significant digit from an array of positive intege
    """
    data = np.abs(data)  # Ensure all values are positive
    first_digits = np.array([int(str(num)[0]) for num in data if num > 0]
    return first_digits

def benford_distribution():
    """
    Calculates the expected Benford distribution (log10(1 + 1/d)).
    """
    return np.log10(1 + 1 / np.arange(1, 10))

def empirical_distribution(first_digits):
    """
    Calculates the empirical distribution of the first digits.
    Returns Array of probabilities for digits 1-9.
    """
    counts = np.array([np.sum(first_digits == d) for d in range(1, 10)])
    return counts / counts.sum()

def benford_law(data):
    """
    Main function to calculate and plot the Benford and empirical distrib
    and compute the p-value using a chi-square goodness-of-fit test.
    """
    # Extract first digits from the data
    first_digits = extract_first_digits(data)

    # Calculate Benford and empirical distributions
    benford_probs = benford_distribution()
    empirical_probs = empirical_distribution(first_digits)

    # Perform Chi-Square test for goodness-of-fit
    observed = np.array([np.sum(first_digits == d) for d in range(1, 10)]
    expected = benford_probs * observed.sum()
    chi2_stat, p_value = chisquare(observed, f_exp=expected)

    # Print results
    print(f"Chi-Square Statistic: {chi2_stat:.4f}")
    print(f"P-Value: {p_value:.4f}")

    # Dynamic comment based on p-value
    if p_value < 0.05:
        print("Anomaly Found")
    else:
        print("No Anomaly Found")

    # Plot the distributions
    plot_distributions(benford_probs, empirical_probs)
```

```
def plot_distributions(benford_probs, empirical_probs):
    """
    Plots the Benford distribution vs the empirical distribution.
    """
    digits = np.arange(1, 10)

    plt.figure(figsize=(10, 6))

    # Plot Benford Distribution as a bar chart
    plt.bar(digits - 0.2, benford_probs, width=0.4,
            label='Benford Distribution', align='center', color='lightblu

    # Plot Empirical Distribution as a line chart, aligned with bar cente
    plt.plot(digits, empirical_probs, marker='o', label='Empirical Distri
             color='orange', linewidth=2)

    plt.xlabel('First Digit', fontsize=14)
    plt.ylabel('Probability', fontsize=14)
    plt.title('Benford vs Empirical Distribution', fontsize=16)
    plt.xticks(digits)
    plt.legend()
    plt.grid()
    plt.show()
```

In [3]:

```
# Load the dataset

file_path = "/Users/mdshamiulislam/Downloads/US_County_Level_Presidential
df = pd.read_csv(file_path)
```

In [5]:

```
# Data exploration

df.head(3)
```

Out[5]:

	fips_code	county	total_2008	dem_2008	gop_2008	oth_2008	total_2012
0	26041	Delta County	19064	9974	8763	327	18043
1	48295	Lipscomb County	1256	155	1093	8	1168
2	1127	Walker County	28652	7420	20722	510	28497

In [6]:

```
# Preparing data by selecting relevent columns

data = df[['total_2016','dem_2016','gop_2016','oth_2016']]
data.head(3)
```

Out[6]:

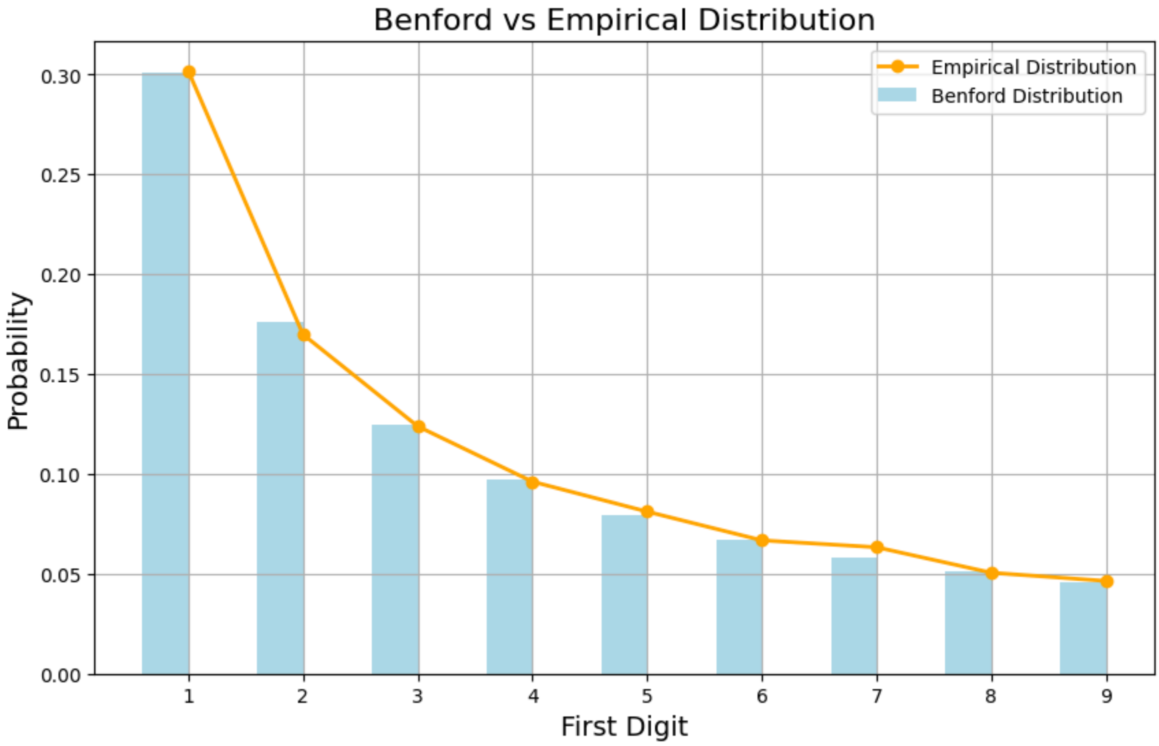
	total_2016	dem_2016	gop_2016	oth_2016
0	18467	6431	11112	924
1	1322	135	1159	28
2	29243	4486	24208	549

```
In [9]:  # flattening the values to feed it to our function

         data = data.values.astype(int).flatten()
```

```
In [10]: # Calling of benford_law function and feed the data
         benford_law(data)
```

```
Chi-Square Statistic: 9.8019
P-Value: 0.2792
No Anomaly Found
```


Benford vs Empirical Distribution
Empirical Distribution
Benford Distribution
Probability
First Digit