Data Preparation

1. Data Preprocessing

In [1]:
```
import os
import pandas as pd
import matplotlib.pyplot as plt

# List of top 10 crypto datasets
crypto_files = [
    'BTC - Tether USD.csv',
    'BTC - Binance Coin.csv',
    'BTC - Cardano.csv',
    'BTC - Dogecoin.csv',
    'BTC - Polkadot.csv',
    'BTC - Ripple.csv',
    'BTC - Tether USD.csv',
    'BTC - Binance Coin.csv',
    'BTC - Cardano.csv',
    'BTC - Dogecoin.csv'
]

for dataset, df in datasets.items():
    print(f"Shape of {dataset}: {df.shape}")
    display(df.head())
    print(df.info())

# Print the time range of each dataset
for dataset, df in datasets.items():
    print(f"{dataset} - Start: {df['date'].min}, End: {df['date'].max}")
```

```
Shape of BTC - Tether USD.csv: (2191, 14)
```

	date	open	high	low	close	volume	marketcap	timehigh	timelow	...	FLDX-LONG	CONTRALINDICATOR	NORTH	KRBN-CHANGE	TWP-MUN	TREUROS-KRBN-S	TB-TC	MONE-KOS	CALENDAR-SHIT	Average
0	2021-02-28	1	Carbon Bitcoinsale	false	false	false	false	false	false	...	false	false	false	false	false	false	false	false	false	1
1	2021-03-01	1	Carbon Bitcoinsale	false	false	false	false	false	false	...	false	false	false	false	false	false	false	197.5	false	197.5
2	2021-03-02	1	Carbon Bitcoinsale	false	false	false	false	false	false	...	false	false	false	false	false	false	false	200.5	false	200.5
3	2021-03-03	1	Carbon Bitcoinsale	false	false	false	false	false	false	...	false	false	false	false	false	false	false	197.5	false	197.5
4	2021-03-04	1	Carbon Bitcoinsale	false	false	false	false	false	false	...	false	false	false	false	false	false	false	197.5	false	197.5

5 rows × 15 columns

```
<class> 'DataFrame'
RangeIndex: 2191 entries, 0 to 2190
Data columns (total 15 columns):
 date            2191 non-null datetime64[ns]
 open            2191 non-null int64
 high            2191 non-null int64
 low             2191 non-null int64
 close           2191 non-null int64
 volume          2191 non-null int64
 marketcap       2191 non-null int64
 timehigh        2191 non-null int64
 timelow         2191 non-null int64
 FLDX-LONG       2191 non-null int64
 CONTRALINDICATOR 2191 non-null int64
 NORTH           2191 non-null int64
 KRBN-CHANGE     2191 non-null int64
 TWP-MUN         2191 non-null int64
 TREUROS-KRBN-S  2191 non-null int64
 TB-TC           2191 non-null int64
 MONE-KOS        2191 non-null int64
 CALENDAR-SHIT   2191 non-null int64
 Average         2191 non-null int64
dtype: datetime64[ns], int64(14)
memory usage: 256.8 KB
```

```
None
```

```
BTC - Tether USD.csv - Start: <bound method Series.min of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>, End: <bound method Series.max of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>
```

In [2]:
```
# Print the time range of each dataset
for dataset, df in datasets.items():
    print(f"{dataset} - Start: {df['date'].min}, End: {df['date'].max}")
```

```
BTC - Binance Coin.csv - Start: <bound method Series.min of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>, End: <bound method Series.max of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>
```

In [3]:
```
# Print the time range of each dataset
for dataset, df in datasets.items():
    print(f"{dataset} - Start: {df['date'].min}, End: {df['date'].max}")
```

```
BTC - Cardano.csv - Start: <bound method Series.min of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>, End: <bound method Series.max of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>
```

In [4]:
```
# Print the time range of each dataset
for dataset, df in datasets.items():
    print(f"{dataset} - Start: {df['date'].min}, End: {df['date'].max}")
```

```
BTC - Dogecoin.csv - Start: <bound method Series.min of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>, End: <bound method Series.max of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>
```

In [5]:
```
# Print the time range of each dataset
for dataset, df in datasets.items():
    print(f"{dataset} - Start: {df['date'].min}, End: {df['date'].max}")
```

```
BTC - Polkadot.csv - Start: <bound method Series.min of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>, End: <bound method Series.max of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>
```

In [6]:
```
# Print the time range of each dataset
for dataset, df in datasets.items():
    print(f"{dataset} - Start: {df['date'].min}, End: {df['date'].max}")
```

```
BTC - Ripple.csv - Start: <bound method Series.min of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>, End: <bound method Series.max of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>
```

In [7]:
```
# Print the time range of each dataset
for dataset, df in datasets.items():
    print(f"{dataset} - Start: {df['date'].min}, End: {df['date'].max}")
```

```
BTC - Tether USD.csv - Start: <bound method Series.min of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>, End: <bound method Series.max of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>
```

In [8]:
```
# Print the time range of each dataset
for dataset, df in datasets.items():
    print(f"{dataset} - Start: {df['date'].min}, End: {df['date'].max}")
```

```
BTC - Binance Coin.csv - Start: <bound method Series.min of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>, End: <bound method Series.max of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>
```

In [9]:
```
# Print the time range of each dataset
for dataset, df in datasets.items():
    print(f"{dataset} - Start: {df['date'].min}, End: {df['date'].max}")
```

```
BTC - Cardano.csv - Start: <bound method Series.min of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>, End: <bound method Series.max of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>
```

In [10]:
```
# Print the time range of each dataset
for dataset, df in datasets.items():
    print(f"{dataset} - Start: {df['date'].min}, End: {df['date'].max}")
```

```
BTC - Dogecoin.csv - Start: <bound method Series.min of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>, End: <bound method Series.max of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>
```

In [11]:
```
# Print the time range of each dataset
for dataset, df in datasets.items():
    print(f"{dataset} - Start: {df['date'].min}, End: {df['date'].max}")
```

```
BTC - Polkadot.csv - Start: <bound method Series.min of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>, End: <bound method Series.max of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>
```

In [12]:
```
# Print the time range of each dataset
for dataset, df in datasets.items():
    print(f"{dataset} - Start: {df['date'].min}, End: {df['date'].max}")
```

```
BTC - Ripple.csv - Start: <bound method Series.min of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>, End: <bound method Series.max of 0    2021-02-28
1    2021-03-01
2    2021-03-02
3    2021-03-03
4    2021-03-04
...
2186 2023-02-23
2187 2023-02-24
2188 2023-02-25
2189 2023-02-26
2190 2023-02-27
Name: date, Dtype: datetime64[ns]>
```

1.1 Data Cleaning

In [13]:
```
# Define a function to clean and prepare data
def clean_and_prepare(df, name):
    print(f"Cleaning {name}...")
    df['date'] = pd.to_datetime(df['date'])
    df = df.drop_duplicates(subset=['date'])
    df = df.sort_values('date')
    df = df.fillna(method='ffill').fillna(method='bfill')
    print(f"{name} cleaned: {df.shape[0]} rows")
    return df

# Apply cleaning to all datasets
cleaned = {}
for dataset, df in datasets.items():
    cleaned[dataset] = clean_and_prepare(df, dataset)

# Verify cleaning
for dataset, df in cleaned.items:
    print(f"{dataset}: {df.shape[0]} rows")
```

```
Cleaning BTC - Tether USD.csv...
BTC - Tether USD.csv cleaned: 2191 rows
Cleaning BTC - Binance Coin.csv...
BTC - Binance Coin.csv cleaned: 2191 rows
Cleaning BTC - Cardano.csv...
BTC - Cardano.csv cleaned: 2191 rows
Cleaning BTC - Dogecoin.csv...
BTC - Dogecoin.csv cleaned: 2191 rows
Cleaning BTC - Polkadot.csv...
BTC - Polkadot.csv cleaned: 2191 rows
Cleaning BTC - Ripple.csv...
BTC - Ripple.csv cleaned: 2191 rows
Cleaning BTC - Tether USD.csv...
BTC - Tether USD.csv cleaned: 2191 rows
Cleaning BTC - Binance Coin.csv...
BTC - Binance Coin.csv cleaned: 2191 rows
Cleaning BTC - Cardano.csv...
BTC - Cardano.csv cleaned: 2191 rows
Cleaning BTC - Dogecoin.csv...
BTC - Dogecoin.csv cleaned: 2191 rows
Cleaning BTC - Polkadot.csv...
BTC - Polkadot.csv cleaned: 2191 rows
Cleaning BTC - Ripple.csv...
BTC - Ripple.csv cleaned: 2191 rows
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[13], line 18
     16 for dataset, df in datasets.items():
     17     print(f"{dataset}: {df.shape[0]} rows")
     18 for dataset, df in cleaned.items():
     19     print(f"{dataset}: {df.shape[0]} rows")

AttributeError: 'dict_keys' object has no attribute 'items'
```

In [14]:
```
# Check for any remaining missing values
for dataset, df in cleaned.items():
    print(f"{dataset} missing values:")
    display(df.isnull().sum())
```

```
BTC - Tether USD.csv missing values:
```

date	0
open	0
high	0
low	0
close	0
volume	0
marketcap	0
timehigh	0
timelow	0
FLDX-LONG	0
CONTRALINDICATOR	0
NORTH	0
KRBN-CHANGE	0
TWP-MUN	0
TREUROS-KRBN-S	0
TB-TC	0
MONE-KOS	0
CALENDAR-SHIT	0
Average	0

```
dtype: int64
BTC - Binance Coin.csv missing values:
```

date	0
open	0
high	0
low	0
close	0
volume	0
marketcap	0
timehigh	0
timelow	0
FLDX-LONG	0
CONTRALINDICATOR	0
NORTH	0
KRBN-CHANGE	0
TWP-MUN	0
TREUROS-KRBN-S	0
TB-TC	0
MONE-KOS	0
CALENDAR-SHIT	0
Average	0

```
dtype: int64
BTC - Cardano.csv missing values:
```

date	0
open	0
high	0
low	0
close	0
volume	0
marketcap	0
timehigh	0
timelow	0
FLDX-LONG	0
CONTRALINDICATOR	0
NORTH	0
KRBN-CHANGE	0
TWP-MUN	0
TREUROS-KRBN-S	0
TB-TC	0
MONE-KOS	0
CALENDAR-SHIT	0
Average	0

```
dtype: int64
BTC - Dogecoin.csv missing values:
```

date	0
open	0
high	0
low	0
close	0
volume	0
marketcap	0
timehigh	0
timelow	0
FLDX-LONG	0
CONTRALINDICATOR	0
NORTH	0
KRBN-CHANGE	0
TWP-MUN	0
TREUROS-KRBN-S	0
TB-TC	0
MONE-KOS	0
CALENDAR-SHIT	0
Average	0

```
dtype: int64
BTC - Polkadot.csv missing values:
```

date	0
open	0
high	0
low	0
close	0
volume	0
marketcap	0
timehigh	0
timelow	0
FLDX-LONG	0
CONTRALINDICATOR	0
NORTH	0
KRBN-CHANGE	0
TWP-MUN	0
TREUROS-KRBN-S	0
TB-TC	0
MONE-KOS	0
CALENDAR-SHIT	0
Average	0

```
dtype: int64
BTC - Ripple.csv missing values:
```

date	0
open	0
high	0
low	0
close	0
volume	0
marketcap	0
timehigh	0
timelow	0
FLDX-LONG	0
CONTRALINDICATOR	0
NORTH	0
KRBN-CHANGE	0
TWP-MUN	0
TREUROS-KRBN-S	0
TB-TC	0
MONE-KOS	0
CALENDAR-SHIT	0
Average	0

```
dtype: int64
```

In [15]:
```
# Create a function to save cleaned data
def save_cleaned(df, name):
    filename = f'{name}_cleaned.csv'
    df.to_csv(filename, index=False)
    print(f"Saved {name} to {filename}")
    return filename

# Save all cleaned datasets
saved_files = {}
for dataset, df in cleaned.items():
    saved_files[dataset] = save_cleaned(df, dataset)
```

```
Saved BTC - Tether USD.csv to BTC - Tether USD.csv_cleaned.csv
Saved BTC - Binance Coin.csv to BTC - Binance Coin.csv_cleaned.csv
Saved BTC - Cardano.csv to BTC - Cardano.csv_cleaned.csv
Saved BTC - Dogecoin.csv to BTC - Dogecoin.csv_cleaned.csv
Saved BTC - Polkadot.csv to BTC - Polkadot.csv_cleaned.csv
Saved BTC - Ripple.csv to BTC - Ripple.csv_cleaned.csv
Saved BTC - Tether USD.csv to BTC - Tether USD.csv_cleaned.csv
Saved BTC - Binance Coin.csv to BTC - Binance Coin.csv_cleaned.csv
Saved BTC - Cardano.csv to BTC - Cardano.csv_cleaned.csv
Saved BTC - Dogecoin.csv to BTC - Dogecoin.csv_cleaned.csv
Saved BTC - Polkadot.csv to BTC - Polkadot.csv_cleaned.csv
Saved BTC - Ripple.csv to BTC - Ripple.csv_cleaned.csv
```

In [16]:
```
# Create a function to save cleaned data
def save_cleaned(df, name):
    filename = f'{name}_cleaned.csv'
    df.to_csv(filename, index=False)
    print(f"Saved {name} to {filename}")
    return filename

# Save all cleaned datasets
saved_files = {}
for dataset, df in cleaned.items():
    saved_files[dataset] = save_cleaned(df, dataset)
```

```
Saved BTC - Tether USD.csv to BTC - Tether USD.csv_cleaned.csv
Saved BTC - Binance Coin.csv to BTC - Binance Coin.csv_cleaned.csv
Saved BTC - Cardano.csv to BTC - Cardano.csv_cleaned.csv
Saved BTC - Dogecoin.csv to BTC - Dogecoin.csv_cleaned.csv
Saved BTC - Polkadot.csv to BTC - Polkadot.csv_cleaned.csv
Saved BTC - Ripple.csv to BTC - Ripple.csv_cleaned.csv
Saved BTC - Tether USD.csv to BTC - Tether USD.csv_cleaned.csv
Saved BTC - Binance Coin.csv to BTC - Binance Coin.csv_cleaned.csv
Saved BTC - Cardano.csv to BTC - Cardano.csv_cleaned.csv
Saved BTC - Dogecoin.csv to BTC - Dogecoin.csv_cleaned.csv
Saved BTC - Polkadot.csv to BTC - Polkadot.csv_cleaned.csv
Saved BTC - Ripple.csv to BTC - Ripple.csv_cleaned.csv
```

2. Exploratory Data Analysis

BTC

In [17]:
```
# Load BTC data
btc = pd.read_csv('BTC - Tether USD.csv')
btc['date'] = pd.to_datetime(btc['date'])
btc = btc.sort_values('date')

# Basic information
print("BTC Dataset Shape:", btc.shape)
display(btc.info())
display(btc.describe())
```

```
BTC Dataset Shape: (2191, 14)
```

```
<class> 'DataFrame'
RangeIndex: 2191 entries, 0 to 2190
Data columns (total 14 columns):
 date            2191 non-null datetime64[ns]
 open            2191 non-null int64
 high            2191 non-null int64
 low             2191 non-null int64
 close           2191 non-null int64
 volume          2191 non-null int64
 marketcap       2191 non-null int64
 timehigh        2191 non-null int64
 timelow         2191 non-null int64
 FLDX-LONG       2191 non-null int64
 CONTRALINDICATOR 2191 non-null int64
 NORTH           2191 non-null int64
 KRBN-CHANGE     2191 non-null int64
 TWP-MUN         2191 non-null int64
 TREUROS-KRBN-S  2191 non-null int64
 TB-TC           2191 non-null int64
 MONE-KOS        2191 non-null int64
 CALENDAR-SHIT   2191 non-null int64
dtype: datetime64[ns], int64(13)
memory usage: 239.7 KB
```

```
None
```

	open	high	low	close	volume	marketcap	timehigh	timelow	FLDX-LONG	CONTRALINDICATOR	NORTH	KRBN-CHANGE	TWP-MUN	TREUROS-KRBN-S	TB-TC	MONE-KOS	CALENDAR-SHIT
count	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000
mean	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
std	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
25%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
50%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
75%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

In [18]:
```
# Check for any missing values
btc.isnull().sum()
```

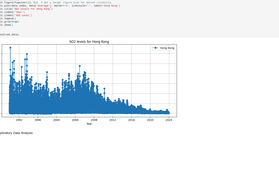
date	0
open	0
high	0
low	0
close	0
volume	0
marketcap	0
timehigh	0
timelow	0
FLDX-LONG	0
CONTRALINDICATOR	0
NORTH	0
KRBN-CHANGE	0
TWP-MUN	0
TREUROS-KRBN-S	0
TB-TC	0
MONE-KOS	0
CALENDAR-SHIT	0

```
dtype: int64
```

In [19]:
```
# Create time series plot
plt.figure(figure=(12, 6))
plt.plot(btc['date'], btc['close'], 'b-')
plt.title('BTC Price Over Time')
plt.xlabel('Date')
plt.ylabel('Price')
plt.grid(True)
plt.show()
```

In [20]:
```
# Calculate daily returns
btc['daily_return'] = btc['close'].pct_change()

# Create histogram of returns
plt.figure(figure=(12, 6))
plt.hist(btc['daily_return'].dropna(), bins=50, alpha=0.7)
plt.title('Distribution of Daily Returns')
plt.xlabel('Daily Return')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



BNB

In [21]:
```
# Load BNB data
bnb = pd.read_csv('BTC - Binance Coin.csv')
bnb['date'] = pd.to_datetime(bnb['date'])
bnb = bnb.sort_values('date')

# Basic information
print("BNB Dataset Shape:", bnb.shape)
display(bnb.info())
display(bnb.describe())
```

```
BNB Dataset Shape: (2191, 14)
```

```
<class> 'DataFrame'
RangeIndex: 2191 entries, 0 to 2190
Data columns (total 14 columns):
 date            2191 non-null datetime64[ns]
 open            2191 non-null int64
 high            2191 non-null int64
 low             2191 non-null int64
 close           2191 non-null int64
 volume          2191 non-null int64
 marketcap       2191 non-null int64
 timehigh        2191 non-null int64
 timelow         2191 non-null int64
 FLDX-LONG       2191 non-null int64
 CONTRALINDICATOR 2191 non-null int64
 NORTH           2191 non-null int64
 KRBN-CHANGE     2191 non-null int64
 TWP-MUN         2191 non-null int64
 TREUROS-KRBN-S  2191 non-null int64
 TB-TC           2191 non-null int64
 MONE-KOS        2191 non-null int64
 CALENDAR-SHIT   2191 non-null int64
dtype: datetime64[ns], int64(13)
memory usage: 239.7 KB
```

```
None
```

	open	high	low	close	volume	marketcap	timehigh	timelow	FLDX-LONG	CONTRALINDICATOR	NORTH	KRBN-CHANGE	TWP-MUN	TREUROS-KRBN-S	TB-TC	MONE-KOS	CALENDAR-SHIT
count	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000
mean	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
std	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
25%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
50%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
75%	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

In [22]:
```
# Check for any missing values
bnb.isnull().sum()
```

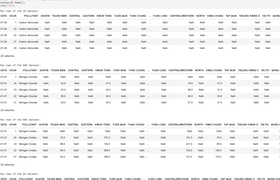
date	0
open	0
high	0
low	0
close	0
volume	0
marketcap	0
timehigh	0
timelow	0
FLDX-LONG	0
CONTRALINDICATOR	0
NORTH	0
KRBN-CHANGE	0
TWP-MUN	0
TREUROS-KRBN-S	0
TB-TC	0
MONE-KOS	0
CALENDAR-SHIT	0

```
dtype: int64
```

In [23]:
```
# Create time series plot
plt.figure(figure=(12, 6))
plt.plot(bnb['date'], bnb['close'], 'b-')
plt.title('BNB Price Over Time')
plt.xlabel('Date')
plt.ylabel('Price')
plt.grid(True)
plt.show()
```


In [24]:
```
# Calculate daily returns
bnb['daily_return'] = bnb['close'].pct_change()

# Create histogram of returns
plt.figure(figure=(12, 6))
plt.hist(bnb['daily_return'].dropna(), bins=50, alpha=0.7)
plt.title('Distribution of Daily Returns')
plt.xlabel('Daily Return')
plt.ylabel('Frequency')
plt.grid(True)
plt.show()
```



ADA

In [25]:
```
# Load ADA data
ada = pd.read_csv('BTC - Cardano.csv')
ada['date'] = pd.to_datetime(ada['date'])
ada = ada.sort_values('date')

# Basic information
print("ADA Dataset Shape:", ada.shape)
display(ada.info())
display(ada.describe())
```

```
ADA Dataset Shape: (2191, 14)
```

```
<class> 'DataFrame'
RangeIndex: 2191 entries, 0 to 2190
Data columns (total 14 columns):
 date            2191 non-null datetime64[ns]
 open            2191 non-null int64
 high            2191 non-null int64
 low             2191 non-null int64
 close           2191 non-null int64
 volume          2191 non-null int64
 marketcap       2191 non-null int64
 timehigh        2191 non-null int64
 timelow         2191 non-null int64
 FLDX-LONG       2191 non-null int64
 CONTRALINDICATOR 2191 non-null int64
 NORTH           2191 non-null int64
 KRBN-CHANGE     2191 non-null int64
 TWP-MUN         2191 non-null int64
 TREUROS-KRBN-S  2191 non-null int64
 TB-TC           2191 non-null int64
 MONE-KOS        2191 non-null int64
 CALENDAR-SHIT   2191 non-null int64
dtype: datetime64[ns], int64(13)
memory usage: 239.7 KB
```

```
None
```

	open	high	low	close	volume	marketcap	timehigh	timelow	FLDX-LONG	CONTRALINDICATOR	NORTH	KRBN-CHANGE	TWP-MUN	TREUROS-KRBN-S	TB-TC	MONE-KOS	CALENDAR-SHIT
count	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000	2191.000000
mean	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000
std	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000