In [ ]:
```
!pip install ta
!pip install requests pandas numpy matplotlib ta
!pip install pandas numpy matplotlib scikit-learn requests scipy SQLAlchemy
!pip install statsmodels
!pip install dash
!pip freeze > requirements.txt
!conda install -c conda-forge ta-lib
!pip install sqlalchemy pyodbc
!pip install yfinance
!pip install streamlit
!pip install dash plotly
```

In [ ]:
```
!pip install nbconvert[webpdf]
```

In [ ]:
```
!pip install playwright
```

In [28]:
```
import pandas as pd
import numpy as np
import talib as ta
import yfinance as yf
import matplotlib.pyplot as plt
from datetime import datetime
from sqlalchemy import create_engine
import urllib
import os  # Import the os module for checking file existence

# Database connection configuration
DATABASE_TYPE = 'mssql'
DBAPI = 'pyodbc'
SERVER = 'MARTIN'
DATABASE = 'crypto_data'
DRIVER = 'ODBC Driver 17 for SQL Server'

# Create a connection URI for SQLAlchemy
params = urllib.parse.quote_plus(f"DRIVER={DRIVER};SERVER={SERVER};DATABASE={DATABA
DATABASE_URI = f"{DATABASE_TYPE}+{DBAPI}:///?odbc_connect={params}"

# Create SQLAlchemy engine
engine = create_engine(DATABASE_URI, echo=False)

# Download historical data
def get_crypto_data(symbol, period='7d', interval='1m'):
    data = yf.download(tickers=symbol, period=period, interval=interval)
    return data

# Calculate volatility
def calculate_volatility(df, window):
    df['returns'] = df['Close'].pct_change()
    df['volatility'] = df['returns'].rolling(window=window).std() * np.sqrt(window)
    return df

# Support and resistance levels
def find_support_resistance(df):
```

```
    df['support'] = df['Low'].rolling(window=60).min()
    df['resistance'] = df['High'].rolling(window=60).max()
    return df

# Moving Averages (SMA, EMA)
def calculate_moving_averages(df, short_window=14, long_window=50):
    df['SMA_14'] = ta.SMA(df['Close'], timeperiod=short_window)
    df['EMA_50'] = ta.EMA(df['Close'], timeperiod=long_window)
    return df

# Bollinger Bands
def calculate_bollinger_bands(df, window=20, num_std=2):
    df['BB_upper'], df['BB_middle'], df['BB_lower'] = ta.BBANDS(df['Close'], timepe
    return df

# Relative Strength Index (RSI)
def calculate_rsi(df, period=14):
    df['RSI'] = ta.RSI(df['Close'], timeperiod=period)
    return df

# VWAP calculation
def calculate_vwap(df):
    df['vwap'] = (df['Volume'] * (df['High'] + df['Low'] + df['Close']) / 3).cumsum
    return df

# Fibonacci retracements (simplified)
def calculate_fibonacci_levels(df):
    max_price = df['Close'].max()
    min_price = df['Close'].min()
    diff = max_price - min_price
    df['fib_0.236'] = max_price - 0.236 * diff
    df['fib_0.382'] = max_price - 0.382 * diff
    df['fib_0.5'] = max_price - 0.5 * diff
    df['fib_0.618'] = max_price - 0.618 * diff
    df['fib_1'] = min_price
    return df

# Verify and clean data
def clean_data(df):
    df.dropna(how='all', inplace=True)
    df.ffill(inplace=True)  # Forward fill missing data
    df.bfill(inplace=True)  # Backward fill missing data
    df.replace([np.inf, -np.inf], np.nan, inplace=True)
    df.dropna(inplace=True)
    return df

# Save data to SQL Server
def save_to_sql(df, table_name):
    try:
        if df.empty:
            print("Data is empty after cleaning. Nothing to save.")
            return
        df.to_sql(table_name, con=engine, if_exists='replace', index_label='Date')
        print(f"Data successfully saved to {table_name} in SQL Server.")
    except Exception as e:
        print(f"Error saving to SQL Server: {e}")
```

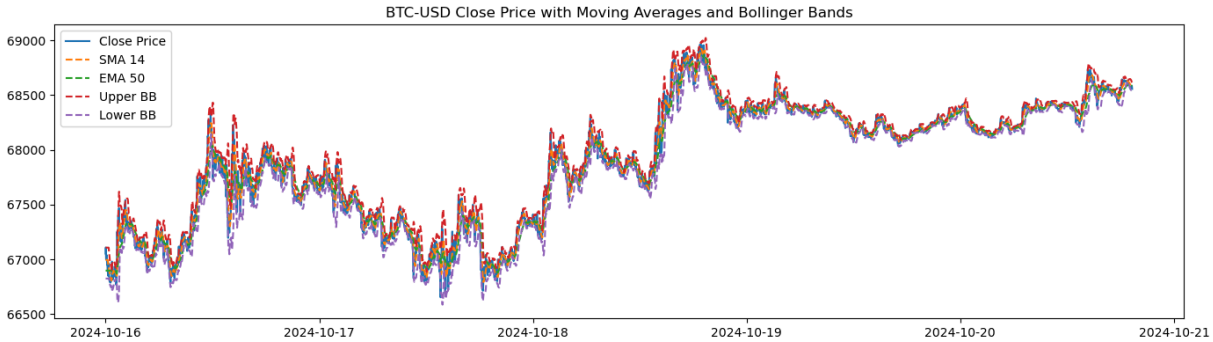
```
    finally:
        engine.dispose()
        print("SQL connection closed.")

# CSV to SQL function
def csv_to_sql(file_name, table_name):
    try:
        df = pd.read_csv(file_name)
        df = clean_data(df)
        save_to_sql(df, table_name)
    except Exception as e:
        print(f"Error moving CSV data to SQL Server: {e}")

# Plot various data points
def plot_data(df, symbol):
    plt.figure(figsize=(14, 8))

    # Plot Close Price, Moving Averages, and Bollinger Bands
    plt.subplot(2, 1, 1)
    plt.plot(df['Close'], label='Close Price')
    plt.plot(df['SMA_14'], label='SMA 14', linestyle='--')
    plt.plot(df['EMA_50'], label='EMA 50', linestyle='--')
    plt.plot(df['BB_upper'], label='Upper BB', linestyle='--')
    plt.plot(df['BB_lower'], label='Lower BB', linestyle='--')
    plt.title(f'{symbol} Close Price with Moving Averages and Bollinger Bands')
    plt.legend()

    # Plot RSI
    plt.subplot(2, 1, 2)
    plt.plot(df['RSI'], label='RSI', color='green')
    plt.axhline(70, color='red', linestyle='--', label='Overbought (70)')
    plt.axhline(30, color='blue', linestyle='--', label='Oversold (30)')
    plt.title(f'{symbol} RSI')
    plt.legend()

    plt.tight_layout()
    plt.show()

    # Plot Returns and Volatility
    plt.figure(figsize=(14, 8))
    plt.subplot(2, 1, 1)
    plt.plot(df.index, df['returns'], label='Returns')
    plt.title(f'{symbol} Returns')
    plt.legend()

    plt.subplot(2, 1, 2)
    plt.plot(df.index, df['volatility'], label='Volatility', color='orange')
    plt.title(f'{symbol} Volatility')
    plt.legend()
    plt.tight_layout()
    plt.show()

# Plot candlestick chart
def plot_candlestick(df, symbol):
    import plotly.graph_objects as go
```

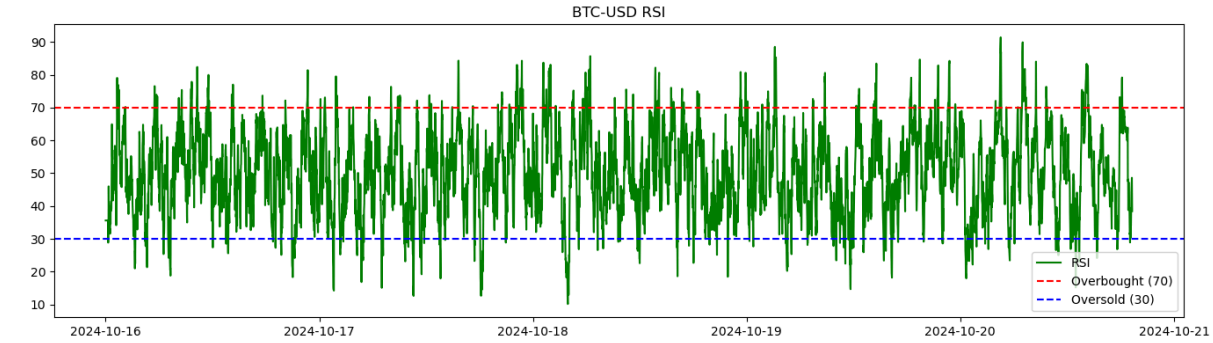
```
    fig = go.Figure(data=[go.Candlestick(x=df.index,
                                         open=df['Open'],
                                         high=df['High'],
                                         low=df['Low'],
                                         close=df['Close'],
                                         name='Candlestick Chart')])

    fig.update_layout(title=f'{symbol} Candlestick Chart',
                      yaxis_title='Price (USD)',
                      xaxis_title='Date')
    fig.show()

# Calculate MACD
def calculate_macd(df):
    df['macd'], df['macdsignal'], df['macdhist'] = ta.MACD(df['Close'], fastperiod=
    return df

# Plot MACD
def plot_macd(df, symbol):
    plt.figure(figsize=(14, 8))
    plt.plot(df['macd'], label='MACD', color='blue')
    plt.plot(df['macdsignal'], label='MACD Signal', color='red')
    plt.fill_between(df.index, df['macdhist'], color='gray', alpha=0.3, label='MACD
    plt.title(f'{symbol} MACD')
    plt.legend()
    plt.show()

# Calculate Average True Range (ATR)
def calculate_atr(df, window):
    df['ATR'] = ta.ATR(df['High'], df['Low'], df['Close'], timeperiod=window)
    return df

# Plot ATR
def plot_atr(df, symbol):
    plt.figure(figsize=(14, 8))
    plt.plot(df['ATR'], label='ATR', color='orange')
    plt.title(f'{symbol} Average True Range (ATR)')
    plt.legend()
    plt.show()

# Plot Volume
def plot_volume(df, symbol):
    plt.figure(figsize=(14, 8))
    plt.bar(df.index, df['Volume'], label='Volume', color='skyblue')
    plt.title(f'{symbol} Trading Volume')
    plt.xlabel('Date')
    plt.ylabel('Volume')
    plt.legend()
    plt.show()

# Create the main function
def main():
    symbol = 'BTC-USD'
    period = '5d'
    interval = '1m'
```

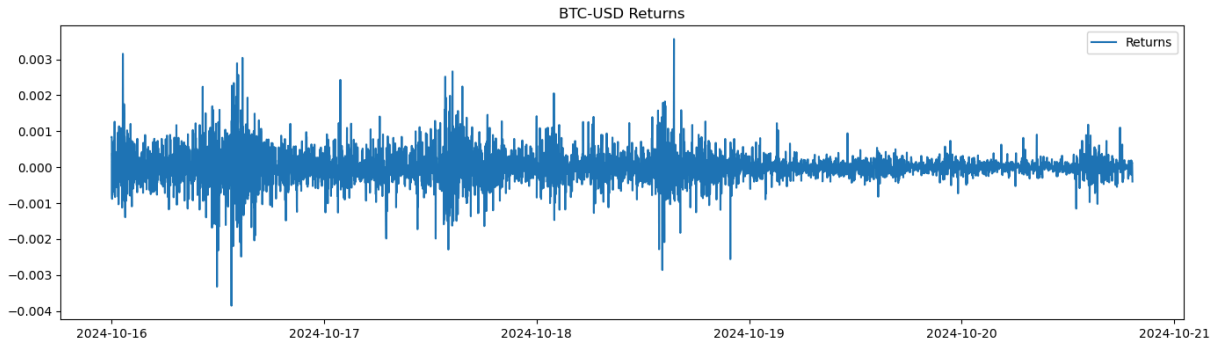
```
    df = get_crypto_data(symbol, period, interval)
    if df.empty:
        print(f"No data found for {symbol}. Please check the period and interval.")
        return

    # Perform technical analysis
    df = calculate_volatility(df, window=60)
    df = find_support_resistance(df)
    df = calculate_moving_averages(df)
    df = calculate_bollinger_bands(df)
    df = calculate_rsi(df)
    df = calculate_vwap(df)
    df = calculate_fibonacci_levels(df)
    df = calculate_macd(df)
    df = calculate_atr(df, window=14)

    df = clean_data(df)
    save_to_sql(df, f"{symbol}_technical_analysis")

    file_name = f"{symbol}_technical_analysis_{datetime.now().strftime('%Y%m%d_%H%M
    df.to_csv(file_name)
    print(f"Data saved to {file_name}")

    # Visualize the data
    plot_data(df, symbol)
    plot_candlestick(df, symbol)
    plot_volume(df, symbol)
    plot_macd(df, symbol)
    plot_atr(df, symbol)

    if os.path.exists(file_name):
        csv_to_sql(file_name, f"{symbol}_technical_analysis")
    else:
        print(f"The file {file_name} does not exist. Please check the file path.")

if __name__ == "__main__":
    main()
```

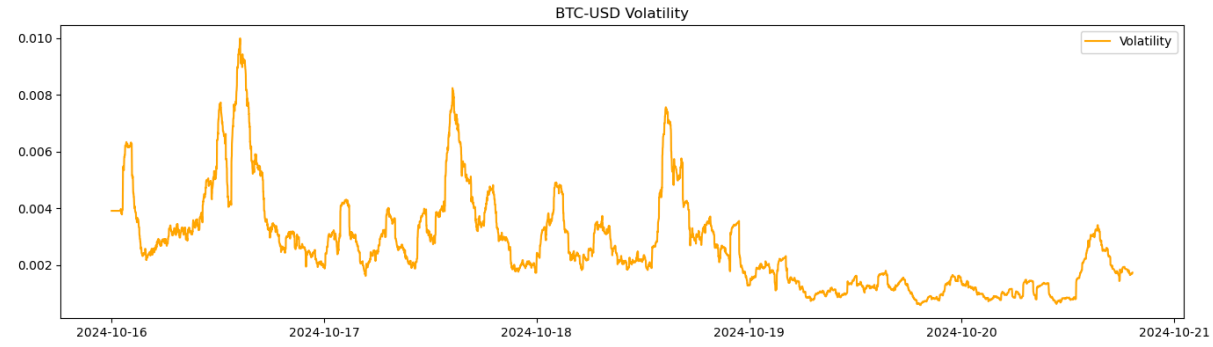
```
[**********100%**********]  1 of 1 completed
```

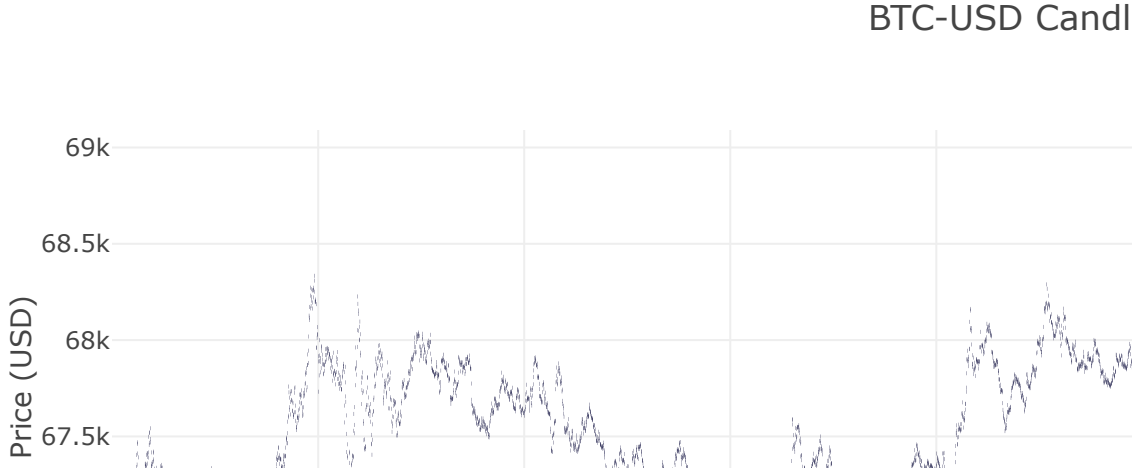
Error saving to SQL Server: (pyodbc.IntegrityError) ('23000', '[23000] [Microsoft][ODBC Driver 17 for SQL Server][SQL Server]Cannot insert an explicit value into a timestamp column. Use INSERT with a column list to exclude the timestamp column, or insert a DEFAULT into the timestamp column. (273) (SQLExecDirectW); [23000] [Microsoft][ODBC Driver 17 for SQL Server][SQL Server]Statement(s) could not be prepared. (8180)')
[SQL: INSERT INTO [BTC-USD_technical_analysis] ([Date], [Open], [High], [Low], [Close], [Adj Close], [Volume], returns, volatility, support, resistance, [SMA_14], [EMA_50], [BB_upper], [BB_middle], [BB_lower], [RSI], vwap, [fib_0.236], [fib_0.382], [fib_0. ... 6355 characters truncated ... ?, ?, ?, ?, ?, ?), (?, ?)]
[parameters: ('2024-10-16 00:00:00.000000 +00:00', 67053.1328125, 67053.1328125, 67053.1328125, 67053.1328125, 67053.1328125, 0, 0.0008399355874615377, 0.003912749250311085, 66789.7109375, 67109.453125, 66995.47544642857, 66898.0621875, 67108.19711410218, 66966.908984375, 66825.62085464782, 35.614266338090054, 67109.453125, 68417.89546875, 68080.892109375, 67808.51953125, 67536.146953125, 66654.3984375, -53.49288715995499, -60.88371191570897, 7.390824755753982, 29.211495535714285, '2024-10-16 00:01:00.000000 +00:00', 67109.453125, 67109.453125, 67109.453125, 67109.453125, 67109.453125, 929792, 0.0008399355874615377, 0.003912749250311085, 66789.7109375, 67109.453125, 66995.47544642857, 66898.0621875, 67108.19711410218, 66966.908984375, 66825.62085464782, 35.614266338090054, 67109.453125, 68417.89546875, 68080.892109375, 67808.51953125, 67536.146953125, 66654.3984375 ... 1979 parameters truncated ... 67108.859375, 67108.859375, 0, 0.0031544135104333293, 0.00505336455189185, 66767.6484375, 67108.859375, 66865.52455357143, 66880.4603614673, 67006.54198145322, 66870.63046875, 66734.71895604677, 73.06286285146648, 66912.28421534634, 68417.89546875, 68080.892109375, 67808.51953125, 67536.146953125, 66654.3984375, 10.124783137041959, -7.525005435853857, 17.649788572895815, 39.229309385598604, '2024-10-16 01:17:00.000000 +00:00', 67255.3046875, 67255.3046875, 67255.3046875, 67255.3046875, 67255.3046875, 39837696, 0.0021822053580389245, 0.00543903346641764, 66767.6484375, 67255.3046875, 66889.688058035571, 66895.16013895877, 67105.51651812762, 66889.594140625, 66673.67176312239, 79.07239851474893, 67027.84821521355, 68417.89546875, 68080.892109375, 67808.519531255, 67536.146953125, 66654.3984375, 39.02048269701481, 1.7840921907198775, 37.236390506294946, 46.88759532234156)]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
SQL connection closed.
Data saved to BTC-USD_technical_analysis_20241020_132410.csv

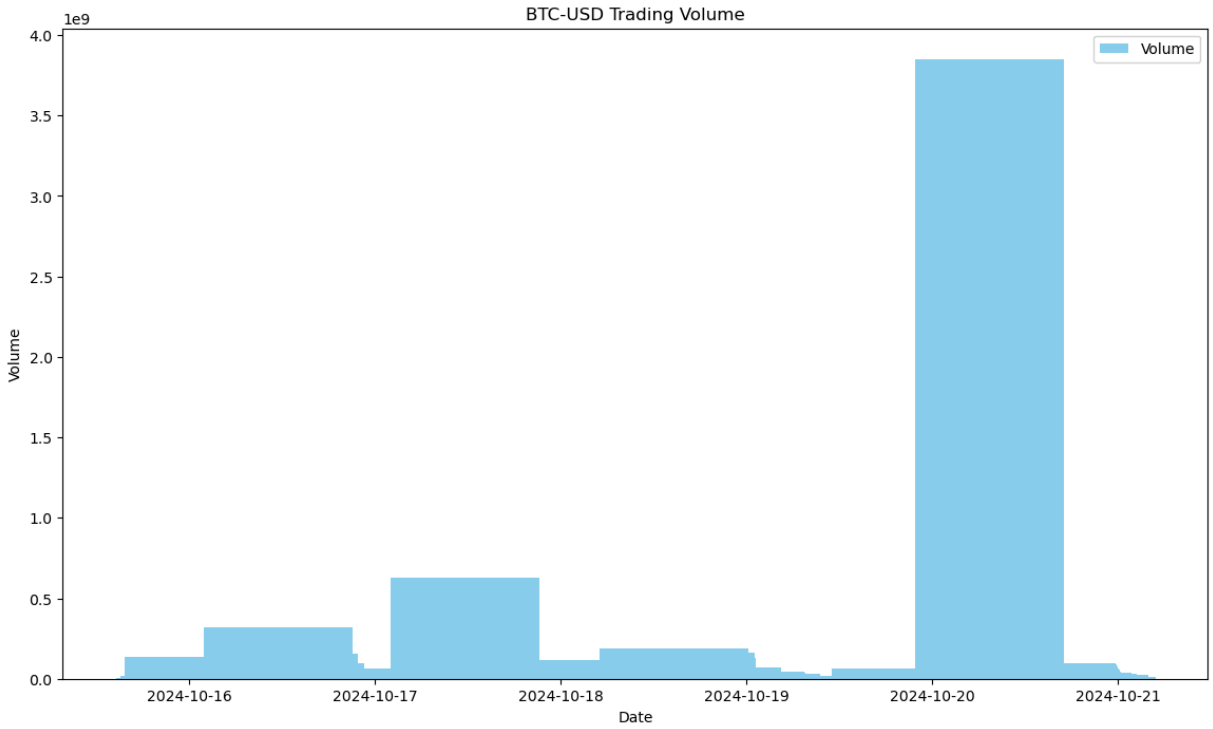

BTC-USD Close Price with Moving Averages and Bollinger Bands
Close Price
SMA 14
EMA 50
Upper BB
Lower BB
69000
68500
68000
67500
67000
66500
2024-10-16
2024-10-17
2024-10-18
2024-10-19
2024-10-20
2024-10-21

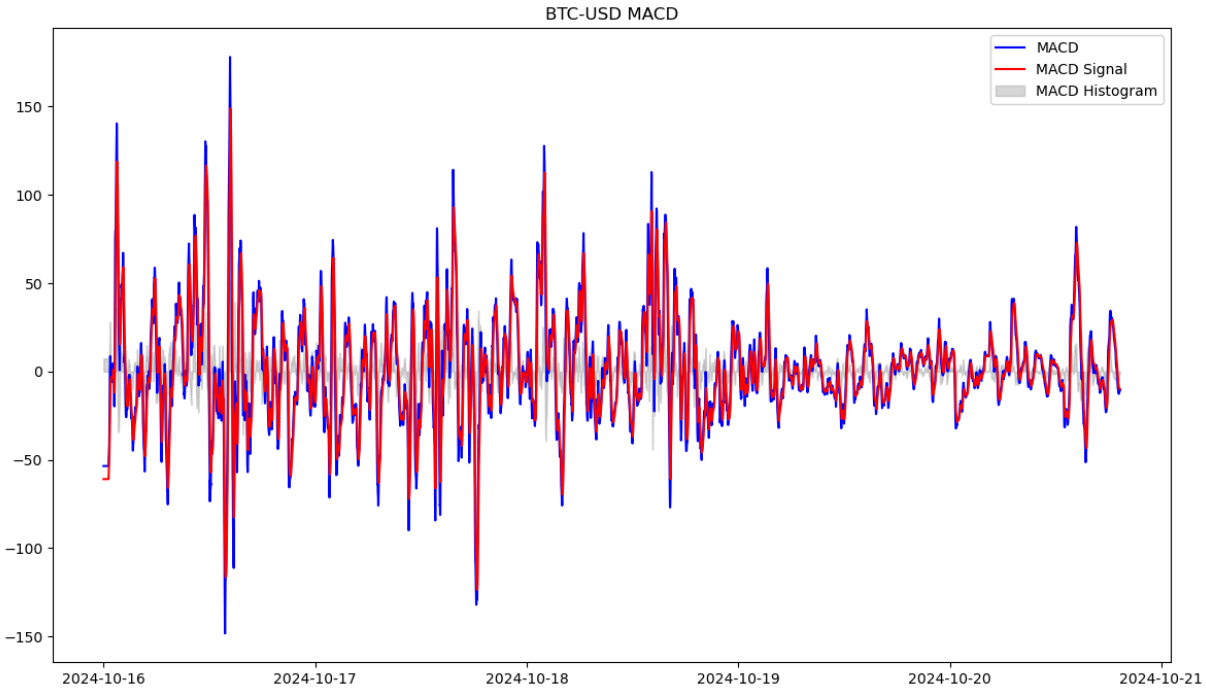

BTC-USD RSI
90
80
70
60
50
40
30
20
10
RSI
Overbought (70)
Oversold (30)
2024-10-16
2024-10-17
2024-10-18
2024-10-19
2024-10-20
2024-10-21

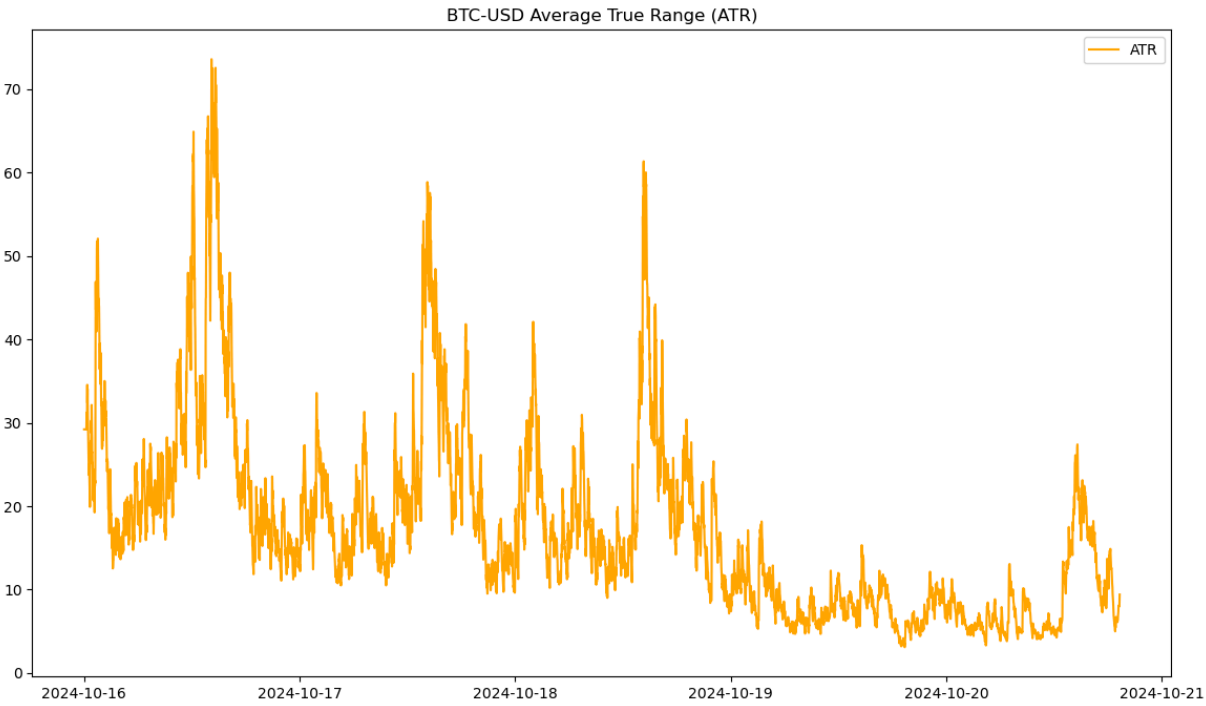

BTC-USD Returns
Returns
0.003
0.002
0.001
0.000
−0.001
−0.002
−0.003
−0.004
2024-10-16
2024-10-17
2024-10-18
2024-10-19
2024-10-20
2024-10-21


BTC-USD Volatility
Volatility
0.010
0.008
0.006
0.004
0.002
2024-10-16
2024-10-17
2024-10-18
2024-10-19
2024-10-20
2024-10-21


BTC-USD Candl
69k
68.5k
68k
67.5k
Price (USD)


BTC-USD Trading Volume
1e9
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Volume
Volume
2024-10-16
2024-10-17
2024-10-18
2024-10-19
2024-10-20
2024-10-21
Date


BTC-USD MACD
MACD
MACD Signal
MACD Histogram
150
100
50
0
−50
−100
−150
2024-10-16
2024-10-17
2024-10-18
2024-10-19
2024-10-20
2024-10-21


BTC-USD Average True Range (ATR)
ATR
70
60
50
40
30
20
10
0
2024-10-16
2024-10-17
2024-10-18
2024-10-19
2024-10-20
2024-10-21

```
Error saving to SQL Server: (pyodbc.ProgrammingError) ('42000', "[42000] [Microsoft]
[ODBC Driver 17 for SQL Server][SQL Server]There is insufficient system memory in re
source pool 'default' to run this query. (701) (SQLExecDirectW)")
[SQL: SELECT [INFORMATION_SCHEMA].[COLUMNS].[COLUMN_NAME], [INFORMATION_SCHEMA].[COL
UMNS].[DATA_TYPE], [INFORMATION_SCHEMA].[COLUMNS].[IS_NULLABLE], [INFORMATION_SCHEM
A].[COLUMNS].[CHARACTER_MAXIMUM_LENGTH], [INFORMATION_SCHEMA].[COLUMNS].[NUMERIC_PRE
CISION], [INFORMATION_SCHEMA].[COLUMNS].[NUMERIC_SCALE], [INFORMATION_SCHEMA].[COLUM
NS].[COLUMN_DEFAULT], [INFORMATION_SCHEMA].[COLUMNS].[COLLATION_NAME], sys.computed_
columns.definition, sys.computed_columns.is_persisted, sys.identity_columns.is_ident
ity, CAST(sys.identity_columns.seed_value AS NUMERIC(38, 0)) AS seed_value, CAST(sy
s.identity_columns.increment_value AS NUMERIC(38, 0)) AS increment_value, CAST(sys.e
xtended_properties.value AS NVARCHAR(max)) AS comment
FROM [INFORMATION_SCHEMA].[COLUMNS] LEFT OUTER JOIN sys.computed_columns ON sys.comp
uted_columns.object_id = object_id([INFORMATION_SCHEMA].[COLUMNS].[TABLE_SCHEMA] + C
AST(? AS NVARCHAR(max)) + [INFORMATION_SCHEMA].[COLUMNS].[TABLE_NAME]) AND sys.compu
ted_columns.name = ([INFORMATION_SCHEMA].[COLUMNS].[COLUMN_NAME] COLLATE DATABASE_DE
FAULT) LEFT OUTER JOIN sys.identity_columns ON sys.identity_columns.object_id = obje
ct_id([INFORMATION_SCHEMA].[COLUMNS].[TABLE_SCHEMA] + CAST(? AS NVARCHAR(max)) + [IN
FORMATION_SCHEMA].[COLUMNS].[TABLE_NAME]) AND sys.identity_columns.name = ([INFORMAT
ION_SCHEMA].[COLUMNS].[COLUMN_NAME] COLLATE DATABASE_DEFAULT) LEFT OUTER JOIN sys.ex
tended_properties ON sys.extended_properties.class = ? AND sys.extended_properties.m
ajor_id = object_id([INFORMATION_SCHEMA].[COLUMNS].[TABLE_SCHEMA] + CAST(? AS NVARCH
AR(max)) + [INFORMATION_SCHEMA].[COLUMNS].[TABLE_NAME]) AND sys.extended_properties.
minor_id = [INFORMATION_SCHEMA].[COLUMNS].[ORDINAL_POSITION] AND sys.extended_proper
ties.name = CAST(? AS NVARCHAR(max))
WHERE [INFORMATION_SCHEMA].[COLUMNS].[TABLE_NAME] = CAST(? AS NVARCHAR(max)) AND [IN
FORMATION_SCHEMA].[COLUMNS].[TABLE_SCHEMA] = CAST(? AS NVARCHAR(max)) ORDER BY [INFO
RMATION_SCHEMA].[COLUMNS].[ORDINAL_POSITION]]
[parameters: ('.', '.', 1, '.', 'MS_Description', 'BTC-USD_technical_analysis', 'db
o')]
(Background on this error at: https://sqlalche.me/e/20/f405)
SQL connection closed.
```

In [34]:

```python
import pandas as pd

# Load your cryptocurrency data
df = pd.read_csv('BTC-USD_technical_analysis_20241020_132410.csv')

# Example of calculating a 20-day Bollinger Band
df['BB_upper'] = df['Close'].rolling(window=20).mean() + (df['Close'].rolling(windo
df['BB_lower'] = df['Close'].rolling(window=20).mean() - (df['Close'].rolling(windo

# Example RSI calculation (14-day)
delta = df['Close'].diff(1)
gain = delta.where(delta > 0, 0)
loss = -delta.where(delta < 0, 0)
avg_gain = gain.rolling(window=14).mean()
avg_loss = loss.rolling(window=14).mean()
rs = avg_gain / avg_loss
df['RSI'] = 100 - (100 / (1 + rs))

# Example MACD calculation
df['EMA12'] = df['Close'].ewm(span=12, adjust=False).mean()
df['EMA26'] = df['Close'].ewm(span=26, adjust=False).mean()
df['MACD'] = df['EMA12'] - df['EMA26']
df['MACD_signal'] = df['MACD'].ewm(span=9, adjust=False).mean()
```

```
df['MACD_hist'] = df['MACD'] - df['MACD_signal']

# Save the enhanced dataset with technical indicators
df.to_csv('enhanced_crypto_data.csv', index=False)
```

In [32]:
```
import pandas as pd
import os

# Define the file path
file_path = 'BTC-USD.csv'

# Check if the file exists
if not os.path.exists(file_path):
    print(f"File not found: {file_path}")
else:
    # Load your cryptocurrency data
    df = pd.read_csv(file_path)

    # Example of calculating a 20-day Bollinger Band
    df['BB_upper'] = df['Close'].rolling(window=20).mean() + (df['Close'].rolling(w
    df['BB_lower'] = df['Close'].rolling(window=20).mean() - (df['Close'].rolling(w

    # Example RSI calculation (14-day)
    delta = df['Close'].diff(1)
    gain = delta.where(delta > 0, 0)
    loss = -delta.where(delta < 0, 0)
    avg_gain = gain.rolling(window=14).mean()
    avg_loss = loss.rolling(window=14).mean()
    rs = avg_gain / avg_loss
    df['RSI'] = 100 - (100 / (1 + rs))

    # Example MACD calculation
    df['EMA12'] = df['Close'].ewm(span=12, adjust=False).mean()
    df['EMA26'] = df['Close'].ewm(span=26, adjust=False).mean()
    df['MACD'] = df['EMA12'] - df['EMA26']
    df['MACD_signal'] = df['MACD'].ewm(span=9, adjust=False).mean()
    df['MACD_hist'] = df['MACD'] - df['MACD_signal']

    # Save the enhanced dataset with technical indicators
    df.to_csv('enhanced_crypto_data.csv', index=False)
    print("Enhanced data saved to enhanced_crypto_data.csv")
```

```
File not found: BTC-USD.csv
```

In [2]:
```
import pandas as pd
import numpy as np
import talib as ta
import yfinance as yf
import matplotlib.pyplot as plt
from datetime import datetime
from sqlalchemy import create_engine
import urllib
import os  # Import the os module for checking file existence

# Database connection configuration
DATABASE_TYPE = 'mssql'
```

```
DBAPI = 'pyodbc'
SERVER = 'MARTIN'
DATABASE = 'crypto_data'
DRIVER = 'ODBC Driver 17 for SQL Server'

# Create a connection URI for SQLAlchemy
params = urllib.parse.quote_plus(f"DRIVER={DRIVER};SERVER={SERVER};DATABASE={DATABA
DATABASE_URI = f"{DATABASE_TYPE}+{DBAPI}:///?odbc_connect={params}"

# Create SQLAlchemy engine
engine = create_engine(DATABASE_URI, echo=False)

# Download historical data
def get_crypto_data(symbol, period='7d', interval='1m'):
    data = yf.download(tickers=symbol, period=period, interval=interval)
    return data

# Calculate volatility
def calculate_volatility(df, window):
    df['returns'] = df['Close'].pct_change()
    df['volatility'] = df['returns'].rolling(window=window).std() * np.sqrt(window)
    return df

# Support and resistance levels
def find_support_resistance(df):
    df['support'] = df['Low'].rolling(window=60).min()
    df['resistance'] = df['High'].rolling(window=60).max()
    return df

# Moving Averages (SMA, EMA)
def calculate_moving_averages(df, short_window=14, long_window=50):
    df['SMA_14'] = ta.SMA(df['Close'], timeperiod=short_window)
    df['EMA_50'] = ta.EMA(df['Close'], timeperiod=long_window)
    return df

# Bollinger Bands
def calculate_bollinger_bands(df, window=20, num_std=2):
    df['BB_upper'], df['BB_middle'], df['BB_lower'] = ta.BBANDS(df['Close'], timepe
    return df

# Relative Strength Index (RSI)
def calculate_rsi(df, period=14):
    df['RSI'] = ta.RSI(df['Close'], timeperiod=period)
    return df

# VWAP calculation
def calculate_vwap(df):
    df['vwap'] = (df['Volume'] * (df['High'] + df['Low'] + df['Close']) / 3).cumsum
    return df

# Fibonacci retracements (simplified)
def calculate_fibonacci_levels(df):
    max_price = df['Close'].max()
    min_price = df['Close'].min()
    diff = max_price - min_price
    df['fib_0.236'] = max_price - 0.236 * diff
```

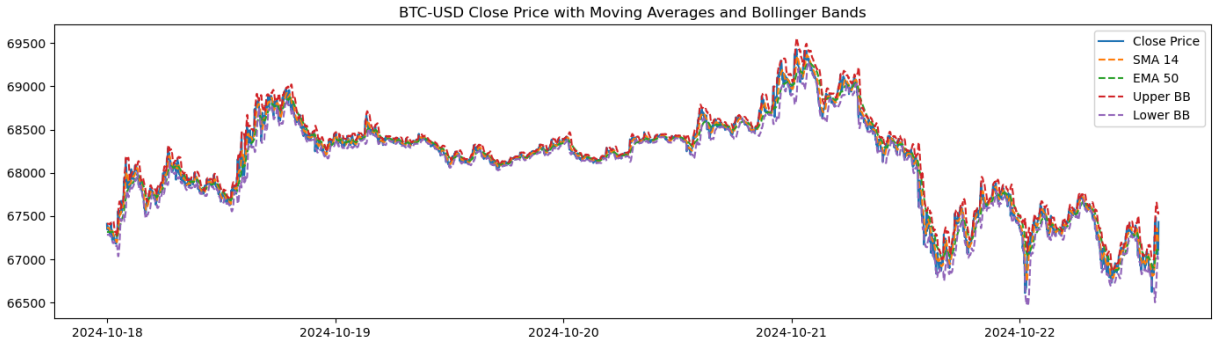
```
    df['fib_0.382'] = max_price - 0.382 * diff
    df['fib_0.5'] = max_price - 0.5 * diff
    df['fib_0.618'] = max_price - 0.618 * diff
    df['fib_1'] = min_price
    return df

# Verify and clean data
def clean_data(df):
    df.dropna(how='all', inplace=True)
    df.ffill(inplace=True)  # Forward fill missing data
    df.bfill(inplace=True)  # Backward fill missing data
    df.replace([np.inf, -np.inf], np.nan, inplace=True)
    df.dropna(inplace=True)
    return df

# Save data to SQL Server
def save_to_sql(df, table_name):
    try:
        if df.empty:
            print("Data is empty after cleaning. Nothing to save.")
            return
        df.to_sql(table_name, con=engine, if_exists='replace', index_label='Date')
        print(f"Data successfully saved to {table_name} in SQL Server.")
    except Exception as e:
        print(f"Error saving to SQL Server: {e}")
    finally:
        engine.dispose()
        print("SQL connection closed.")

# CSV to SQL function
def csv_to_sql(file_name, table_name):
    try:
        df = pd.read_csv(file_name)
        df = clean_data(df)
        save_to_sql(df, table_name)
    except Exception as e:
        print(f"Error moving CSV data to SQL Server: {e}")

# Plot various data points
def plot_data(df, symbol):
    plt.figure(figsize=(14, 8))

    # Plot Close Price, Moving Averages, and Bollinger Bands
    plt.subplot(2, 1, 1)
    plt.plot(df['Close'], label='Close Price')
    plt.plot(df['SMA_14'], label='SMA 14', linestyle='--')
    plt.plot(df['EMA_50'], label='EMA 50', linestyle='--')
    plt.plot(df['BB_upper'], label='Upper BB', linestyle='--')
    plt.plot(df['BB_lower'], label='Lower BB', linestyle='--')
    plt.title(f'{symbol} Close Price with Moving Averages and Bollinger Bands')
    plt.legend()

    # Plot RSI
    plt.subplot(2, 1, 2)
    plt.plot(df['RSI'], label='RSI', color='green')
    plt.axhline(70, color='red', linestyle='--', label='Overbought (70)')
```

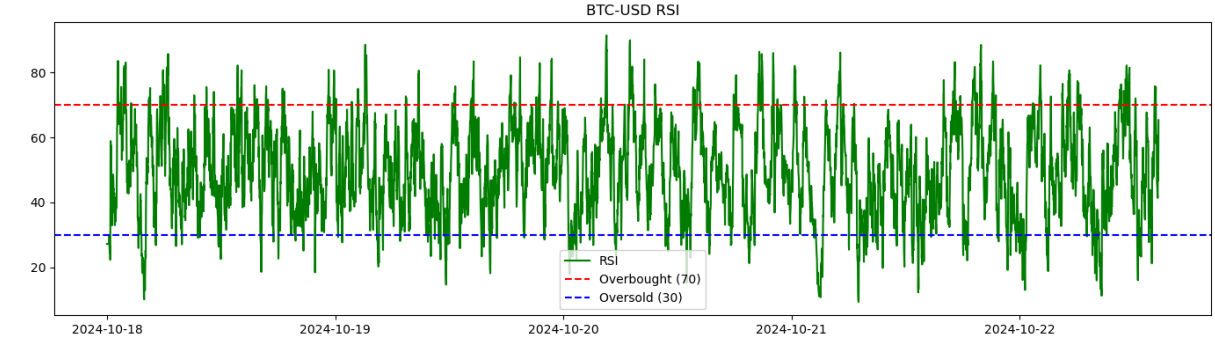
```
    plt.axhline(30, color='blue', linestyle='--', label='Oversold (30)')
    plt.title(f'{symbol} RSI')
    plt.legend()

    plt.tight_layout()
    plt.show()

    # Plot Returns and Volatility
    plt.figure(figsize=(14, 8))
    plt.subplot(2, 1, 1)
    plt.plot(df.index, df['returns'], label='Returns')
    plt.title(f'{symbol} Returns')
    plt.legend()

    plt.subplot(2, 1, 2)
    plt.plot(df.index, df['volatility'], label='Volatility', color='orange')
    plt.title(f'{symbol} Volatility')
    plt.legend()
    plt.tight_layout()
    plt.show()

# Plot candlestick chart
def plot_candlestick(df, symbol):
    import plotly.graph_objects as go

    fig = go.Figure(data=[go.Candlestick(x=df.index,
                                         open=df['Open'],
                                         high=df['High'],
                                         low=df['Low'],
                                         close=df['Close'],
                                         name='Candlestick Chart')])

    fig.update_layout(title=f'{symbol} Candlestick Chart',
                      yaxis_title='Price (USD)',
                      xaxis_title='Date')
    fig.show()

# Calculate MACD
def calculate_macd(df):
    df['macd'], df['macdsignal'], df['macdhist'] = ta.MACD(df['Close'], fastperiod=
    return df

# Plot MACD
def plot_macd(df, symbol):
    plt.figure(figsize=(14, 8))
    plt.plot(df['macd'], label='MACD', color='blue')
    plt.plot(df['macdsignal'], label='MACD Signal', color='red')
    plt.fill_between(df.index, df['macdhist'], color='gray', alpha=0.3, label='MACD
    plt.title(f'{symbol} MACD')
    plt.legend()
    plt.show()

# Calculate Average True Range (ATR)
def calculate_atr(df, window):
    df['ATR'] = ta.ATR(df['High'], df['Low'], df['Close'], timeperiod=window)
    return df
```

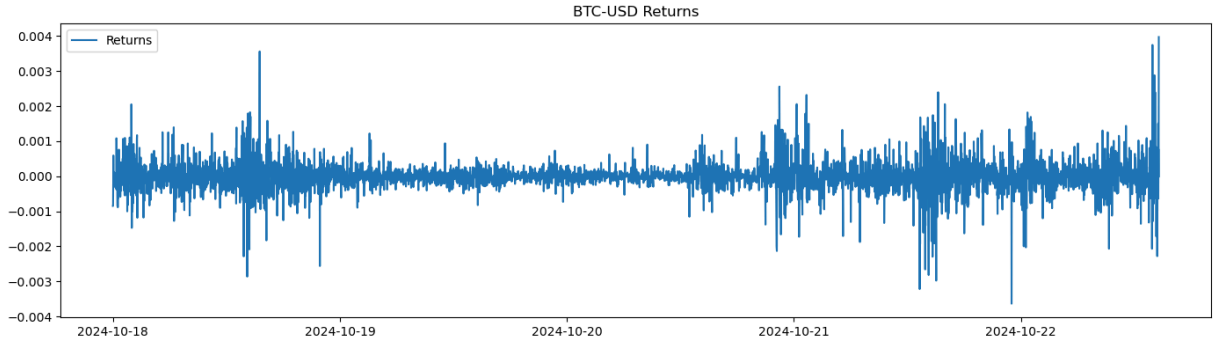
```python
# Plot ATR
def plot_atr(df, symbol):
    plt.figure(figsize=(14, 8))
    plt.plot(df['ATR'], label='ATR', color='orange')
    plt.title(f'{symbol} Average True Range (ATR)')
    plt.legend()
    plt.show()

# Plot Volume
def plot_volume(df, symbol):
    plt.figure(figsize=(14, 8))
    plt.bar(df.index, df['Volume'], label='Volume', color='skyblue')
    plt.title(f'{symbol} Trading Volume')
    plt.xlabel('Date')
    plt.ylabel('Volume')
    plt.legend()
    plt.show()

# Create the main function
def main():
    symbols = [
        'BTC-USD', 'ETH-USD', 'XRP-USD', 'LTC-USD', 'BNB-USD', 'MATIC-USD', 'SOL-US
        'TRX-USD', 'BLZ-USD', 'LEVER-USD', 'EURUSD=X', 'NZDUSD=X', 'GBPUSD=X', 'JPY
        'AUDUSD=X', 'AUDJPY=X', 'GBPJPY=X', 'XAUUSD=X', 'XAGUSD=X'
    ]

    period = '5d'
    interval = '1m'

    for symbol in symbols:
        print(f"Processing data for {symbol}...")
        df = get_crypto_data(symbol, period, interval)
        if df.empty:
            print(f"No data found for {symbol}. Please check the period and interva
            continue

        # Perform technical analysis
        df = calculate_volatility(df, window=60)
        df = find_support_resistance(df)
        df = calculate_moving_averages(df)
        df = calculate_bollinger_bands(df)
        df = calculate_rsi(df)
        df = calculate_vwap(df)
        df = calculate_fibonacci_levels(df)
        df = calculate_macd(df)
        df = calculate_atr(df, window=14)

        df = clean_data(df)
        save_to_sql(df, f"{symbol}_technical_analysis")

        file_name = f"{symbol}_technical_analysis_{datetime.now().strftime('%Y%m%d_
        df.to_csv(file_name)
        print(f"Data saved to {file_name}")

        # Visualize the data
```

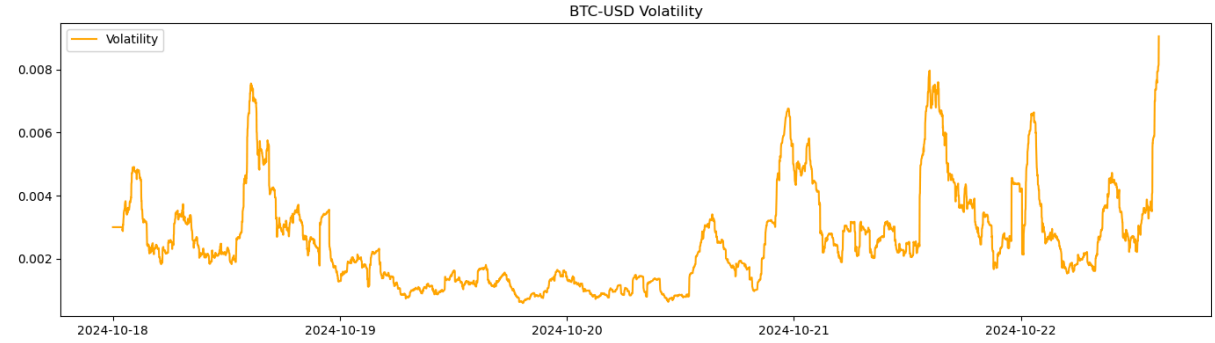
```
        plot_data(df, symbol)
        plot_candlestick(df, symbol)
        plot_volume(df, symbol)
        plot_macd(df, symbol)
        plot_atr(df, symbol)

        if os.path.exists(file_name):
            csv_to_sql(file_name, f"{symbol}_technical_analysis")
        else:
            print(f"The file {file_name} does not exist. Please check the file path

if __name__ == "__main__":
    main()
```

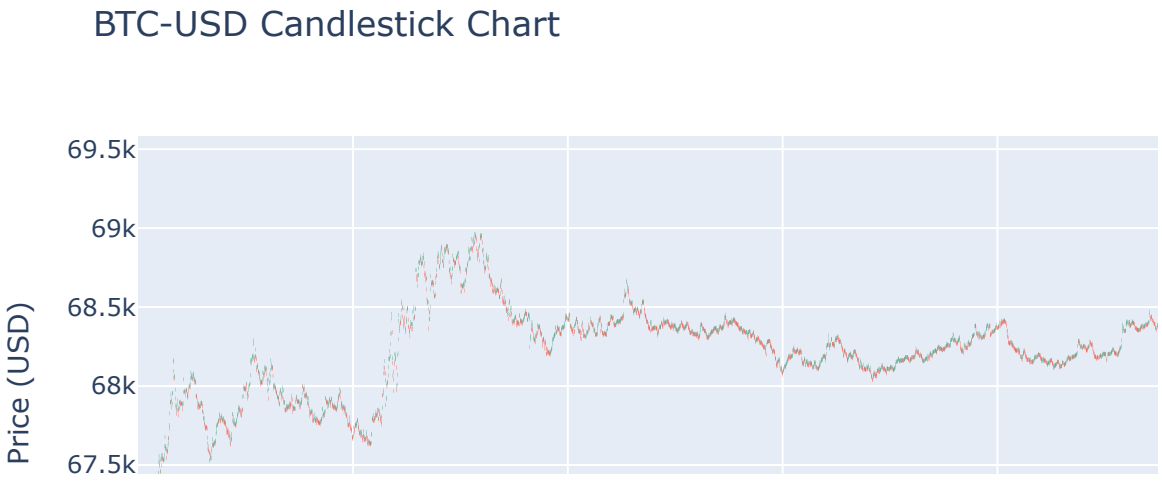
```
Processing data for BTC-USD...
[**********************100%***********************]  1 of 1 completed
Error saving to SQL Server: (pyodbc.IntegrityError) ('23000', '[23000] [Microsoft][O
DBC Driver 17 for SQL Server][SQL Server]Cannot insert an explicit value into a time
stamp column. Use INSERT with a column list to exclude the timestamp column, or inse
rt a DEFAULT into the timestamp column. (273) (SQLExecDirectW); [23000] [Microsoft]
[ODBC Driver 17 for SQL Server][SQL Server]Statement(s) could not be prepared. (818
0)')
[SQL: INSERT INTO [BTC-USD_technical_analysis] ([Date], [Open], [High], [Low], [Clos
e], [Adj Close], [Volume], returns, volatility, support, resistance, [SMA_14], [EMA_
50], [BB_upper], [BB_middle], [BB_lower], [RSI], vwap, [fib_0.236], [fib_0.382], [fi
b_0. ... 6355 characters truncated ... ?, ?, ?, ?, ?, ?), (?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)]
[parameters: ('2024-10-18 00:00:00.000000 +00:00', 67411.1875, 67411.1875, 67411.187
5, 67411.1875, 67411.1875, 0, -0.0008466000172449029, 0.0030020209485352913, 67181.9
84375, 67414.5078125, 67364.38560267857, 67317.34171875, 67411.30011366823, 67347.97
8515625, 67284.65691758177, 27.212585720048406, 67354.1171875, 68762.6320625, 68351.
56678125, 68019.3359375, 67687.10509375, 66611.578125, -22.511126657569548, -11.2061
26110194923, -11.305000547374625, 13.833705357142858, '2024-10-18 00:01:00.000000 +0
0:00', 67354.1171875, 67354.1171875, 67354.1171875, 67354.1171875, 67354.1171875, 79
76960, -0.0008466000172449029, 0.0030020209485352913, 67181.984375, 67414.5078125, 6
7364.38560267857, 67317.34171875, 67411.30011366823, 67347.978515625, 67284.65691758
177, 27.212585720048406, 67354.1171875, 68762.6320625, 68351.56678125, 68019.335937
5, 67687.10509375, 66611.578125 ... 1979 parameters truncated ... 67459.359375, 6745
9.359375, 2564096, 3.3585148599968306e-06, 0.0035800064342346016, 67181.984375, 6755
7.734375, 67415.19196428571, 67338.71793332987, 67620.63387004453, 67348.58203125, 6
7076.53019245547, 64.14847435516555, 67376.27682568878, 68762.6320625, 68351.5667812
5, 68019.3359375, 67687.10509375, 66611.578125, 67.32424454057764, 53.5372362245363
4, 13.787008316041295, 25.413476087922664, '2024-10-18 01:16:00.000000 +00:00', 6753
3.3828125, 67533.3828125, 67533.3828125, 67533.3828125, 67533.3828125, 3016704, 0.00
1097304187081205, 0.00371613650978, 67181.984375, 67557.734375, 67437.31752232143, 6
7346.35185016008, 67638.28556793832, 67366.151953125, 67094.01833831167, 70.71094265
502178, 67377.32433642144, 68762.6320625, 68351.56678125, 68019.3359375, 67687.10509
375, 66611.578125, 70.14147647879145, 56.85808427538736, 13.283392203404084, 28.8856
1618878533)]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
SQL connection closed.
Data saved to BTC-USD_technical_analysis_20241022_083907.csv
```

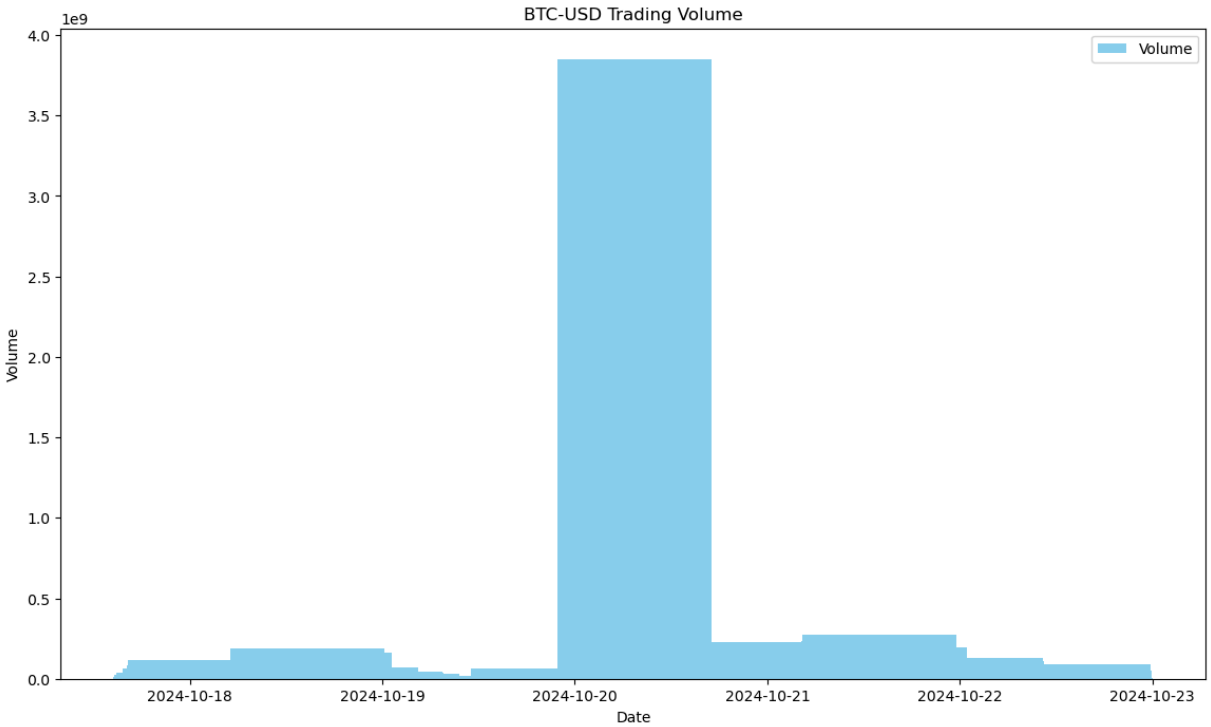

BTC-USD Close Price with Moving Averages and Bollinger Bands
Close Price
SMA 14
EMA 50
Upper BB
Lower BB
69500
69000
68500
68000
67500
67000
66500
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22

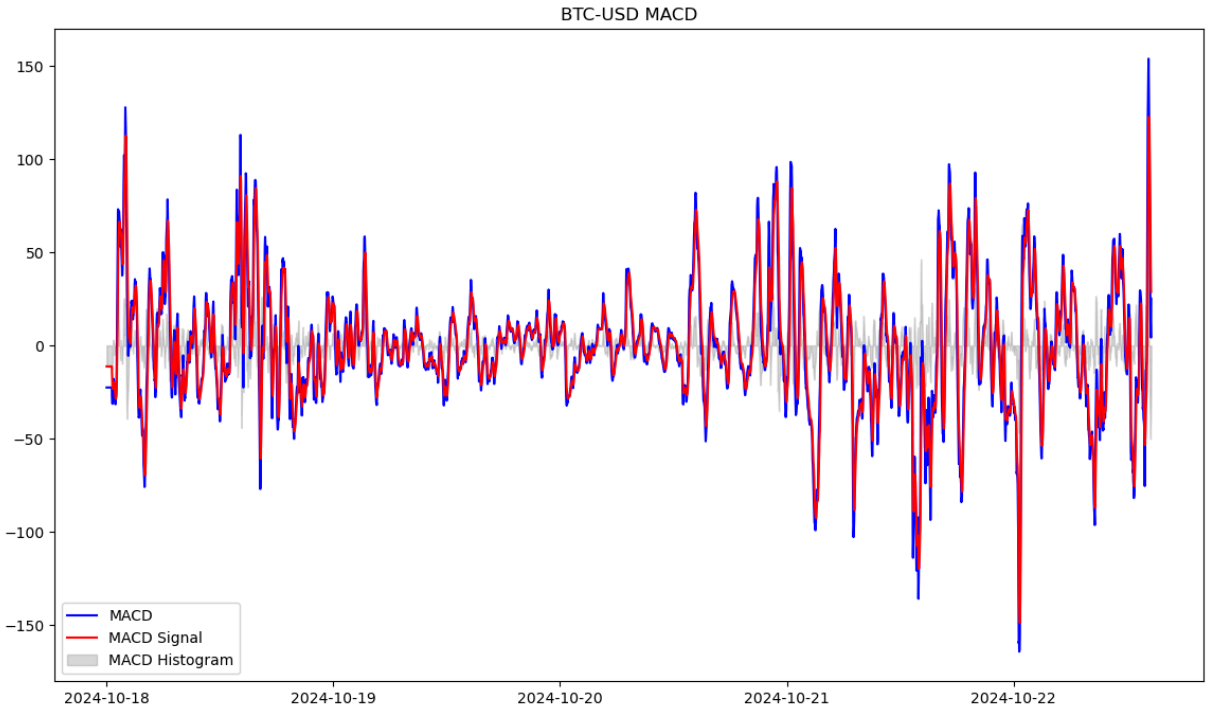

BTC-USD RSI
RSI
Overbought (70)
Oversold (30)
80
60
40
20
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22

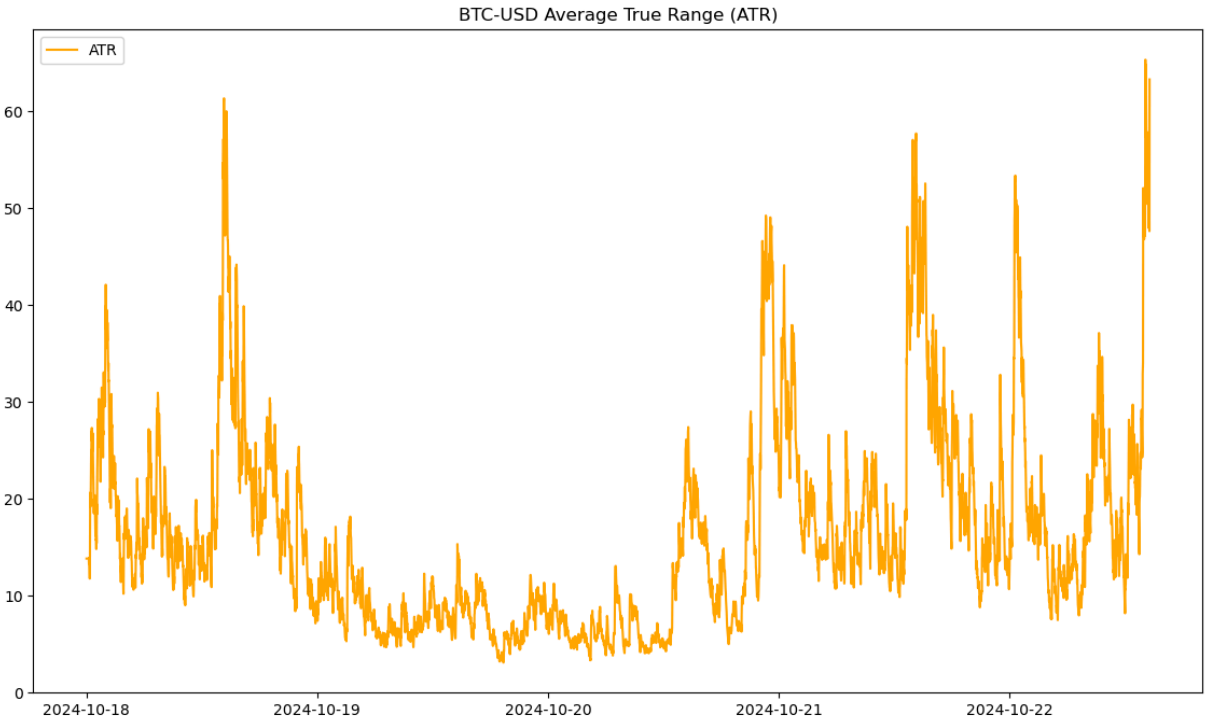

BTC-USD Returns
Returns
0.004
0.003
0.002
0.001
0.000
−0.001
−0.002
−0.003
−0.004
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22


BTC-USD Volatility
Volatility
0.008
0.006
0.004
0.002
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22


BTC-USD Candlestick Chart
Price (USD)
69.5k
69k
68.5k
68k
67.5k

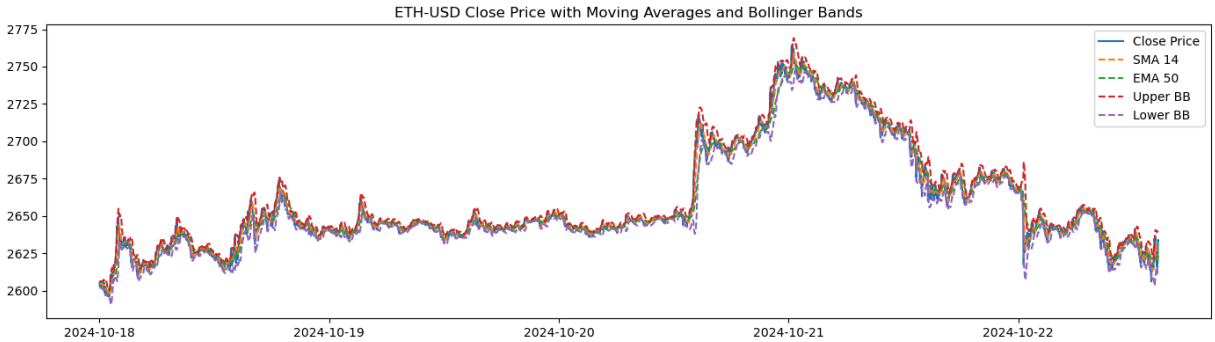

BTC-USD Trading Volume
1e9
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Volume
Volume
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22
2024-10-23
Date

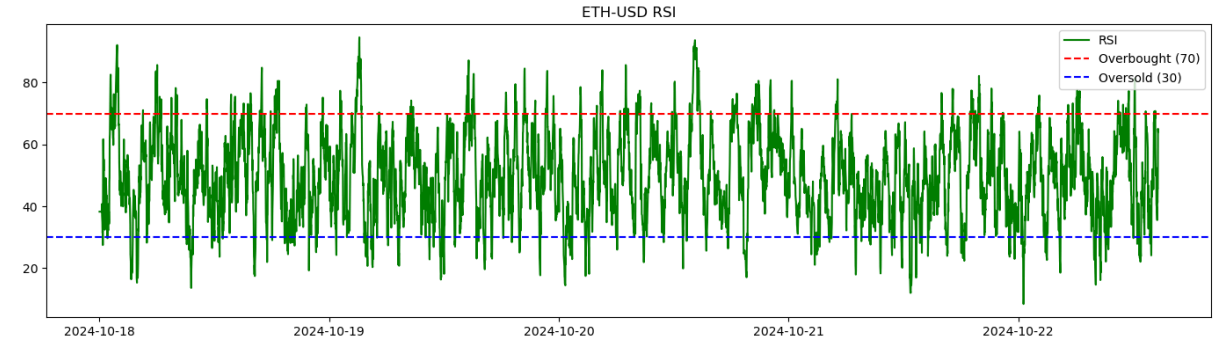


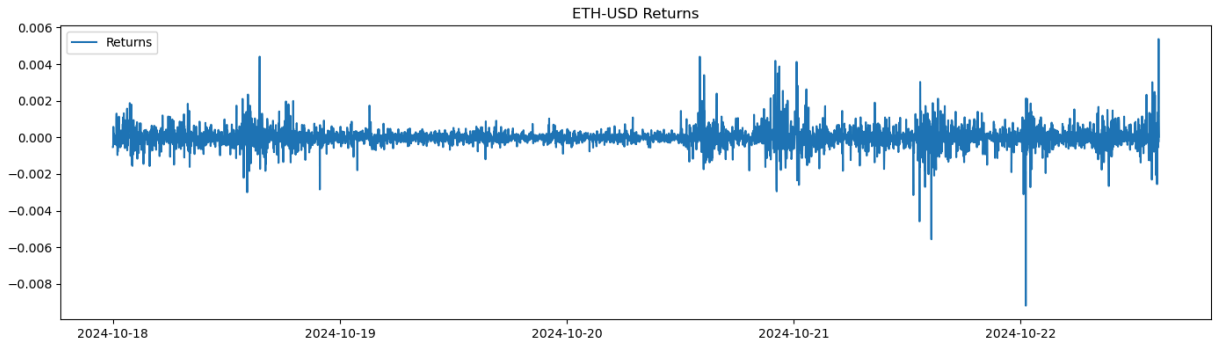


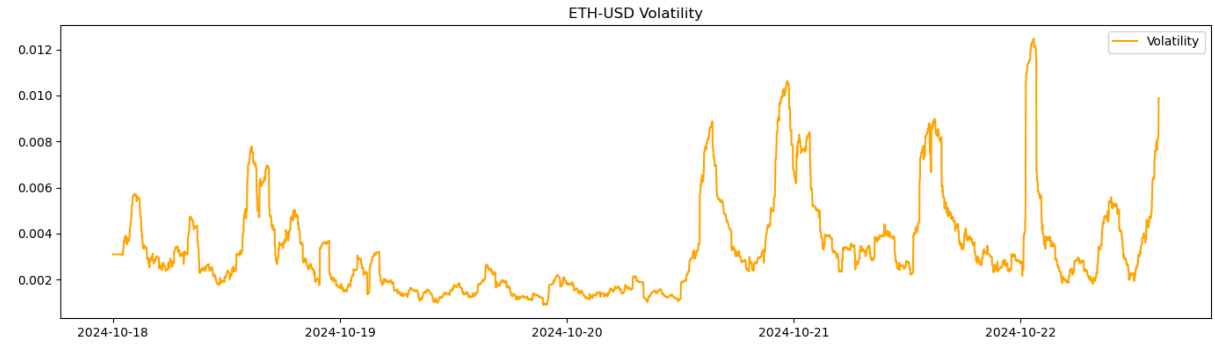

```
Data successfully saved to BTC-USD_technical_analysis in SQL Server.
SQL connection closed.
Processing data for ETH-USD...
[**********100%**********]  1 of 1 completed
```

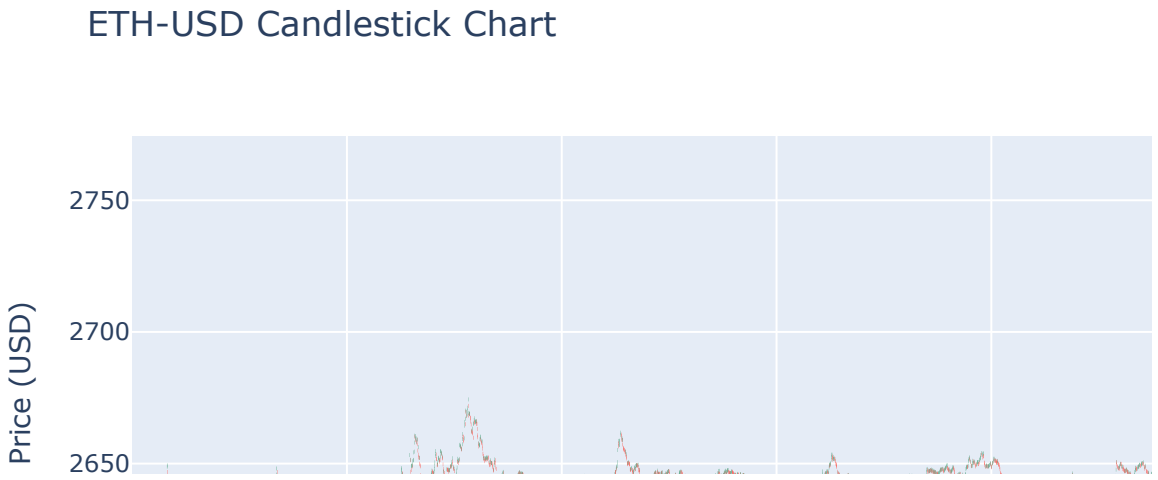
```
Error saving to SQL Server: (pyodbc.IntegrityError) ('23000', '[23000] [Microsoft][O
DBC Driver 17 for SQL Server][SQL Server]Cannot insert an explicit value into a time
stamp column. Use INSERT with a column list to exclude the timestamp column, or inse
rt a DEFAULT into the timestamp column. (273) (SQLExecDirectW); [23000] [Microsoft]
[ODBC Driver 17 for SQL Server][SQL Server]Statement(s) could not be prepared. (818
0)')
[SQL: INSERT INTO [ETH-USD_technical_analysis] ([Date], [Open], [High], [Low], [Clos
e], [Adj Close], [Volume], returns, volatility, support, resistance, [SMA_14], [EMA_
50], [BB_upper], [BB_middle], [BB_lower], [RSI], vwap, [fib_0.236], [fib_0.382], [fi
b_0. ... 6355 characters truncated ... ?, ?, ?, ?, ?, ?), (?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)]
[parameters: ('2024-10-18 00:00:00.000000 +00:00', 2605.232177734375, 2605.232177734
375, 2605.232177734375, 2605.232177734375, 2605.232177734375, 0, -0.0005293771520296
087, 0.003093705225276315, 2596.484375, 2606.732666015625, 2604.6680733816966, 2602.
564619140625, 2606.3406067763854, 2604.1176147460938, 2601.894622715802, 38.26333816
767982, 2603.85302734375, 2725.18848046875, 2700.535154296875, 2680.60986328125, 266
0.684572265625, 2596.1806640625, -1.09595170856619, -0.5472606465719865, -0.54869106
19942036, 0.5651332310267857, '2024-10-18 00:01:00.000000 +00:00', 2603.85302734375,
2603.85302734375, 2603.85302734375, 2603.85302734375, 2603.85302734375, 3012608, -0.
0005293771520296087, 0.003093705225276315, 2596.484375, 2606.732666015625, 2604.6680
733816966, 2602.564619140625, 2606.3406067763854, 2604.1176147460938, 2601.894622715
802, 38.26333816767982, 2603.85302734375, 2725.18848046875, 2700.535154296875, 2680.
60986328125, 2660.684572265625, 2596.1806640625 ... 1979 parameters truncated ... 26
07.800537109375, 2607.800537109375, 0, 4.484568590124027e-05, 0.0037224523138428183,
2596.1806640625, 2610.908447265625, 2605.5428466796875, 2602.830247152519, 2613.5775
62743677, 2602.9274658203126, 2592.2773688969482, 66.3773415921197, 2603.25400008948
4, 2725.18848046875, 2700.535154296875, 2680.60986328125, 2660.684572265625, 2596.18
06640625, 2.6217511935701623, 1.9934459872645511, 0.6283052063056112, 0.966061341349
12, '2024-10-18 01:16:00.000000 +00:00', 2610.265380859375, 2610.265380859375, 2610.
265380859375, 2610.265380859375, 2610.265380859375, 89088, 0.000945181088401803, 0.0
038205702611483184, 2596.1806640625, 2610.908447265625, 2606.407557896205, 2603.1218
21023376, 2614.3132701599566, 2603.6041259765625, 2592.8949817931684, 71.89362079504
619, 2603.2583156095266, 2725.18848046875, 2700.535154296875, 2680.60986328125, 266
0.684572265625, 2596.1806640625, 2.7352915644623863, 2.1418151027041183, 0.593476461
758268, 1.0731172276813257)]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
SQL connection closed.
Data saved to ETH-USD_technical_analysis_20241022_083954.csv
```

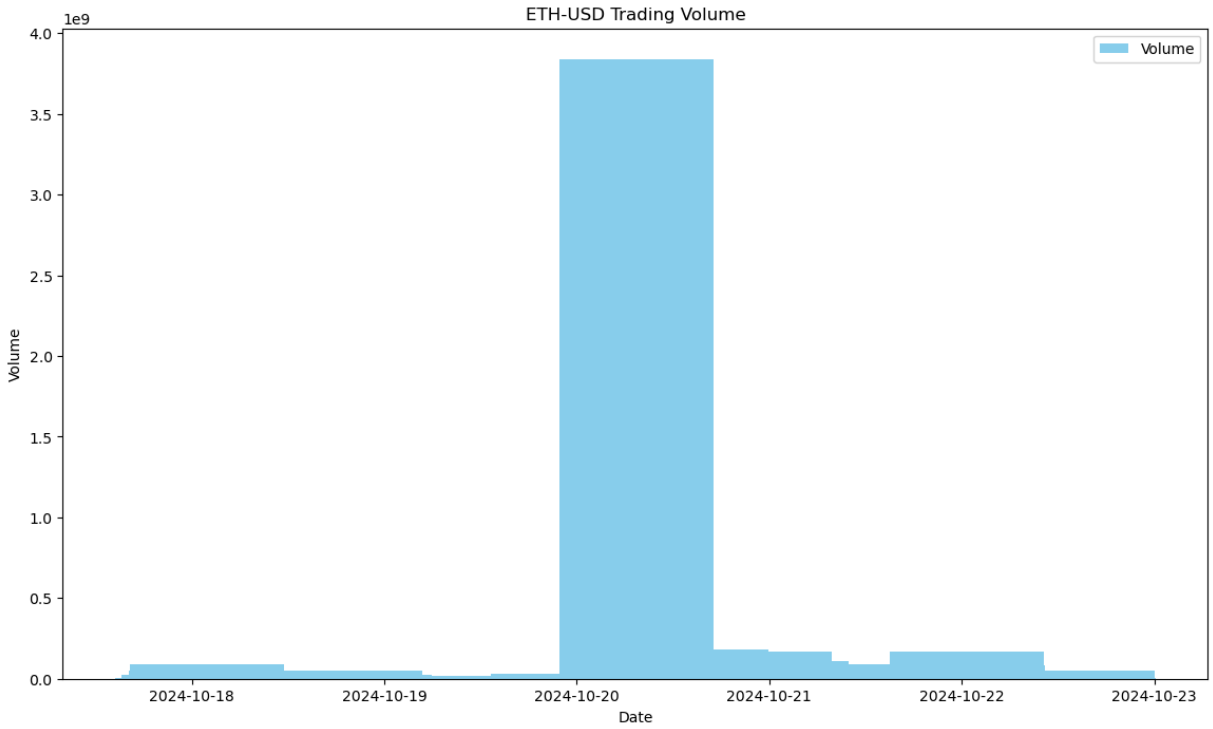

ETH-USD Close Price with Moving Averages and Bollinger Bands
Close Price
SMA 14
EMA 50
Upper BB
Lower BB
2775
2750
2725
2700
2675
2650
2625
2600
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22

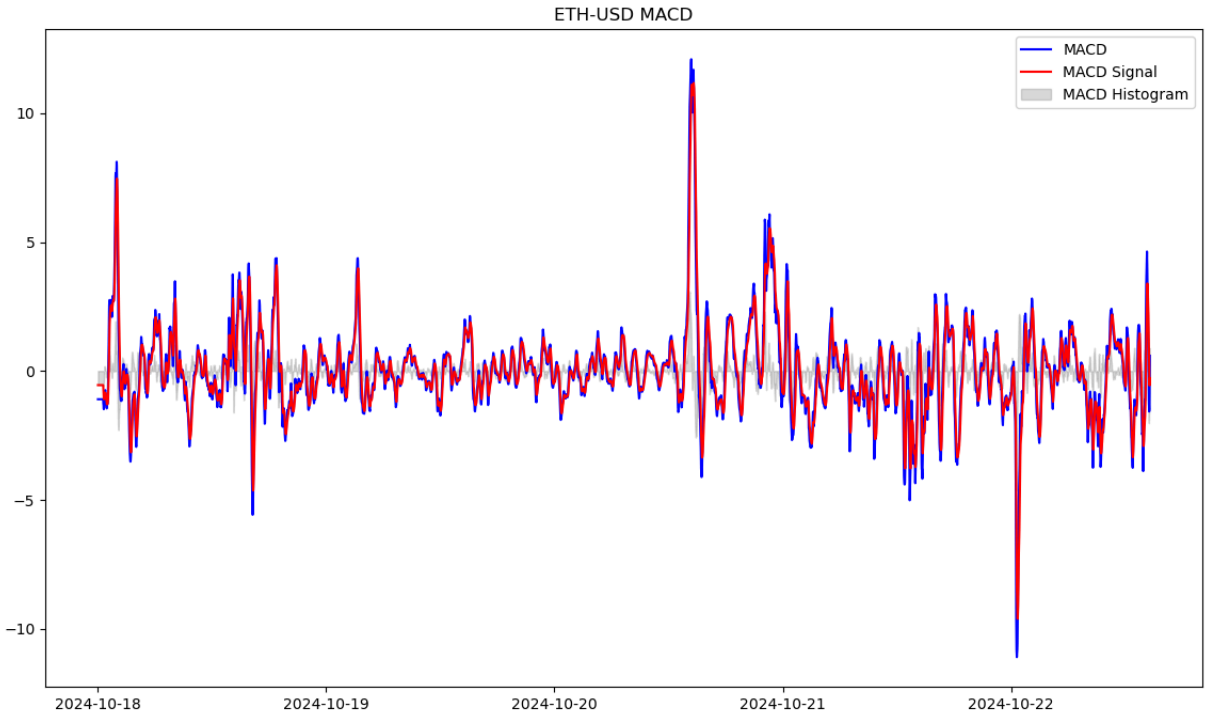

ETH-USD RSI
RSI
Overbought (70)
Oversold (30)
80
60
40
20
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22

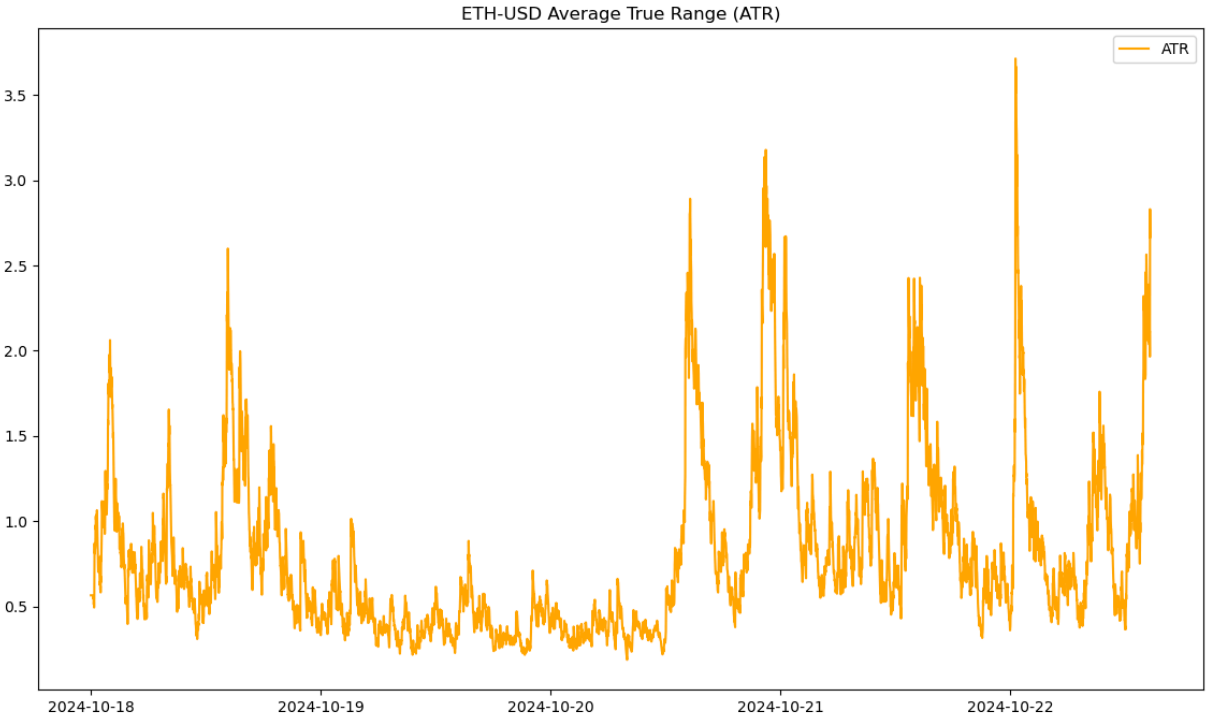

ETH-USD Returns
Returns
0.006
0.004
0.002
0.000
−0.002
−0.004
−0.006
−0.008
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22


ETH-USD Volatility
Volatility
0.012
0.010
0.008
0.006
0.004
0.002
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22


ETH-USD Candlestick Chart
Price (USD)
2750
2700
2650

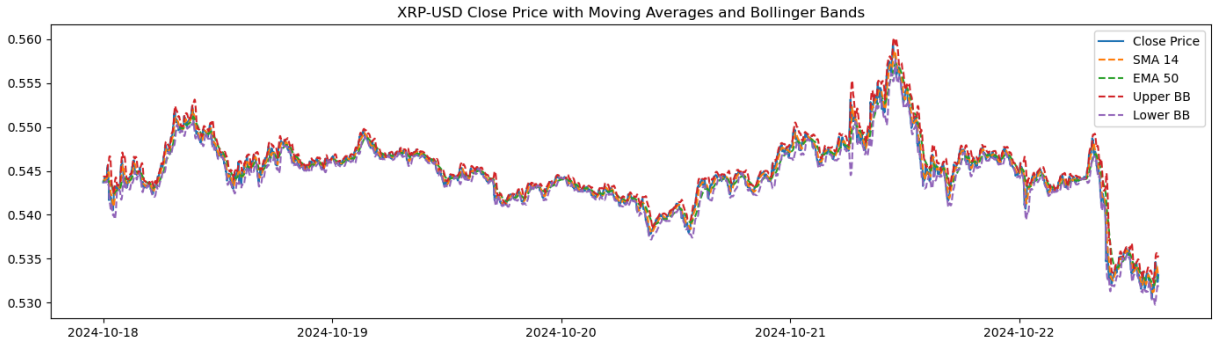

ETH-USD Trading Volume
1e9
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Volume
Volume
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22
2024-10-23
Date

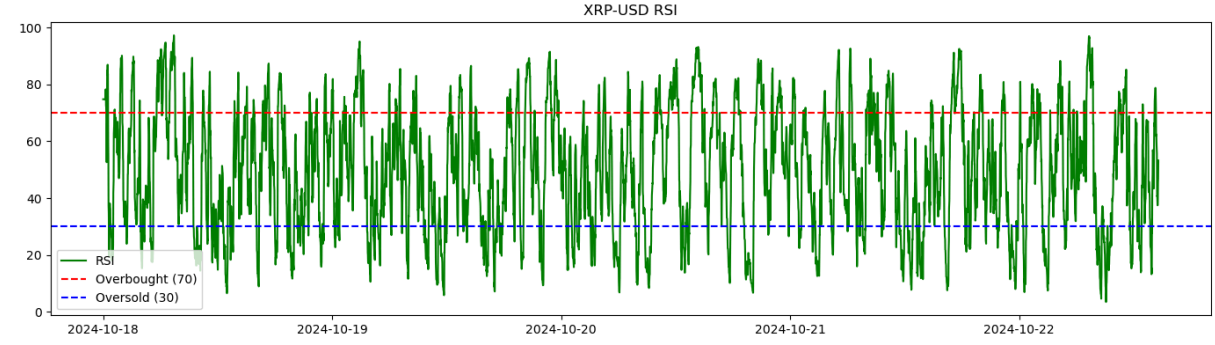


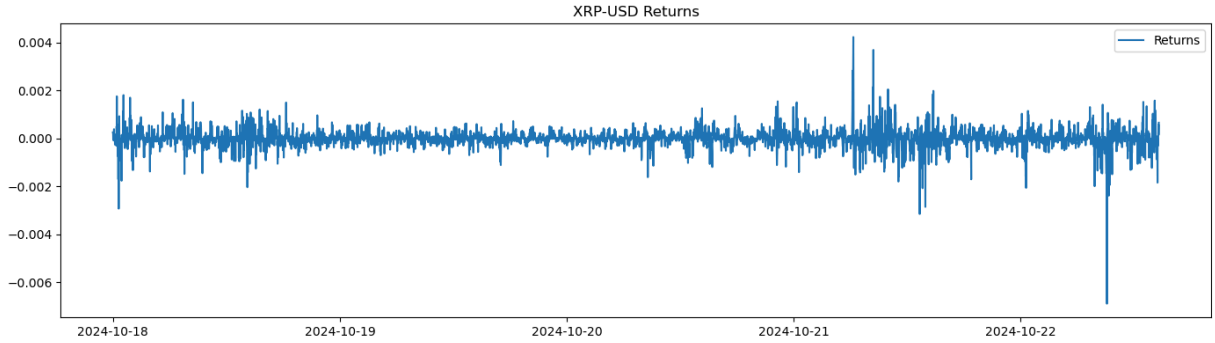


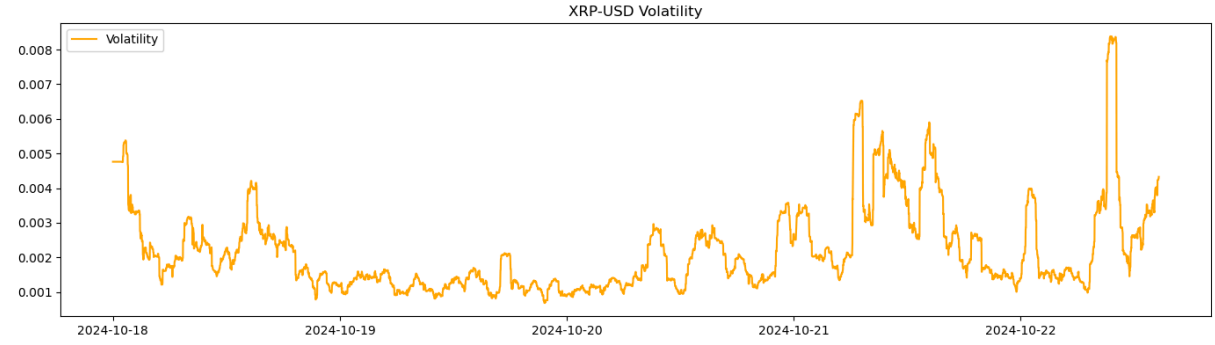

```
Data successfully saved to ETH-USD_technical_analysis in SQL Server.
SQL connection closed.
Processing data for XRP-USD...
[**********100%**********]  1 of 1 completed
```

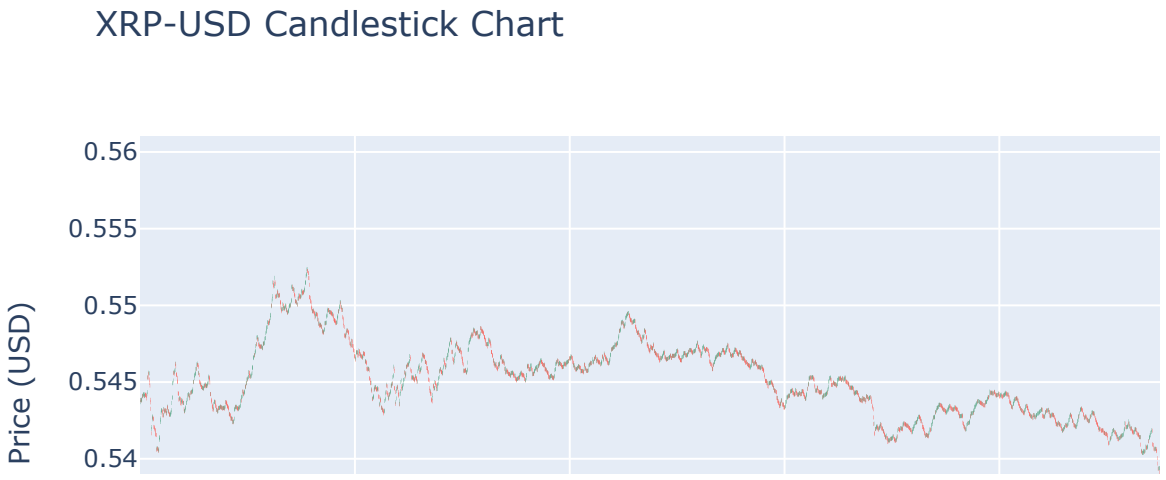
```
Error saving to SQL Server: (pyodbc.IntegrityError) ('23000', '[23000] [Microsoft][O
DBC Driver 17 for SQL Server][SQL Server]Cannot insert an explicit value into a time
stamp column. Use INSERT with a column list to exclude the timestamp column, or inse
rt a DEFAULT into the timestamp column. (273) (SQLExecDirectW); [23000] [Microsoft]
[ODBC Driver 17 for SQL Server][SQL Server]Statement(s) could not be prepared. (818
0)')
[SQL: INSERT INTO [XRP-USD_technical_analysis] ([Date], [Open], [High], [Low], [Clos
e], [Adj Close], [Volume], returns, volatility, support, resistance, [SMA_14], [EMA_
50], [BB_upper], [BB_middle], [BB_lower], [RSI], vwap, [fib_0.236], [fib_0.382], [fi
b_0. ... 6355 characters truncated ... ?, ?, ?, ?, ?, ?), (?, ?, ?, ?, ?, ?, ?, ?,
?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)]
[parameters: ('2024-10-18 00:00:00.000000 +00:00', 0.5436916351318359, 0.54369163513
18359, 0.5436916351318359, 0.5436916351318359, 0.5436916351318359, 0, 0.000256533041
45567546, 0.004761914295680228, 0.5405896902084351, 0.545610249042511, 0.54397172587
25848, 0.5437828052043915, 0.5443588804641374, 0.5440260171890259, 0.543693153913914
4, 74.8015873015873, 0.5438311100006104, 0.5525914545059204, 0.5483601632118225, 0.5
449403524398804, 0.5415205416679383, 0.530449628829956, 0.0003223999078150186, 0.000
3175163651974024, 4.883542617616218e-06, 5.793571472167969e-05, '2024-10-18 00:01:0
0.000000 +00:00', 0.5438311100006104, 0.5438311100006104, 0.5438311100006104, 0.5438
311100006104, 0.5438311100006104, 44800, 0.00025653304145567546, 0.00476191429568022
8, 0.5405896902084351, 0.545610249042511, 0.5439717258725848, 0.5437828052043915, 0.
5443588804641374, 0.5440260171890259, 0.5436931539139144, 74.8015873015873, 0.543831
1100006104, 0.5525914545059204, 0.5483601632118225, 0.5449403524398804, 0.5415205416
679383, 0.530449628829956 ... 1979 parameters truncated ... 0.5430562496185303, 0.54
30562496185303, 39040, -8.242129397484366e-05, 0.005313912880020317, 0.5405094623565
674, 0.545610249042511, 0.5422391976628985, 0.5425747505827132, 0.5439296747093277,
0.5417558282613755, 0.5395819818134232, 68.09205788325094, 0.5426232419355025, 0.552
5914545059204, 0.5483601632118225, 0.5449403524398804, 0.5415205416679383, 0.5304496
28829956, 0.00031814626384119027, 6.571033948934742e-05, 0.00025243592435184286, 0.0
0017088201191557863, '2024-10-18 01:16:00.000000 +00:00', 0.5428360104560852, 0.5428
360104560852, 0.5428360104560852, 0.5428360104560852, 0.5428360104560852, 0, -0.0004
055549726198482, 0.005323213307659341, 0.5405094623565674, 0.545610249042511, 0.5424
023142882756, 0.5425849960679435, 0.5440265147546747, 0.5418641924858093, 0.53970187
02169439, 61.950228363192814, 0.5426232419355025, 0.5525914545059204, 0.548360163211
8225, 0.5449403524398804, 0.5415205416679383, 0.530449628829956, 0.00031563794035238
68, 0.0001156958596619553, 0.0001999420806904315, 0.00017440752266768505)]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
SQL connection closed.
Data saved to XRP-USD_technical_analysis_20241022_084042.csv
```

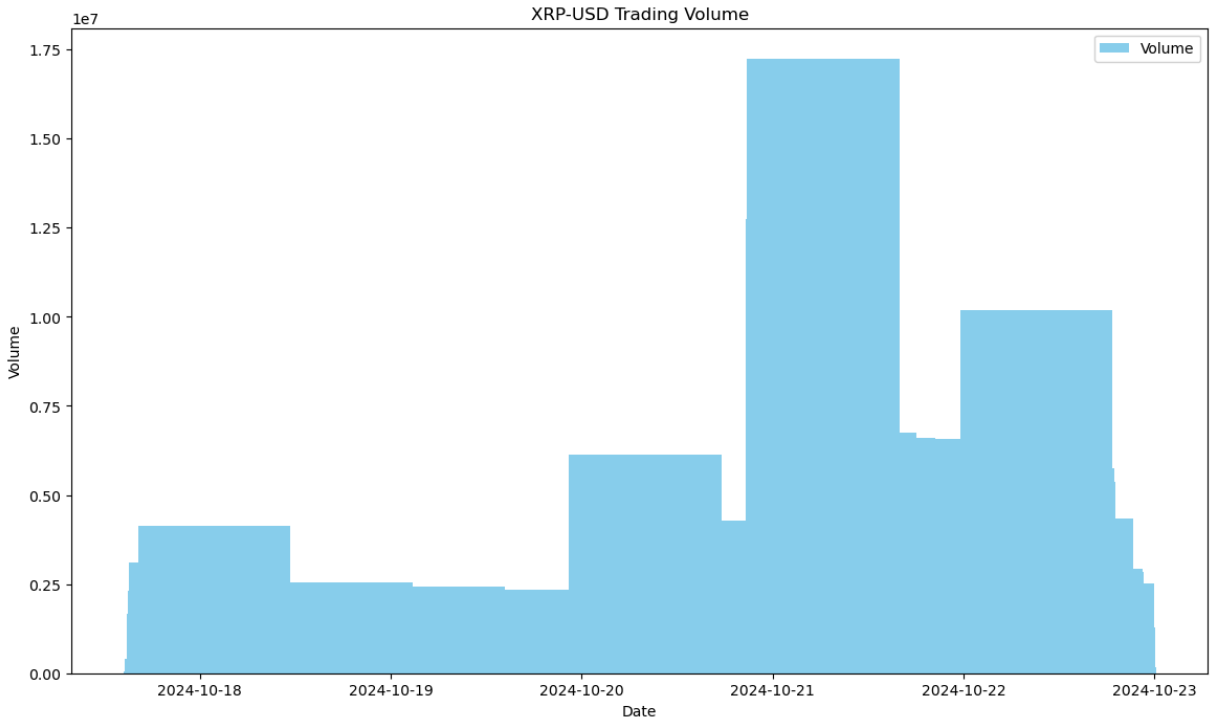

XRP-USD Close Price with Moving Averages and Bollinger Bands
Close Price
SMA 14
EMA 50
Upper BB
Lower BB
0.560
0.555
0.550
0.545
0.540
0.535
0.530
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22

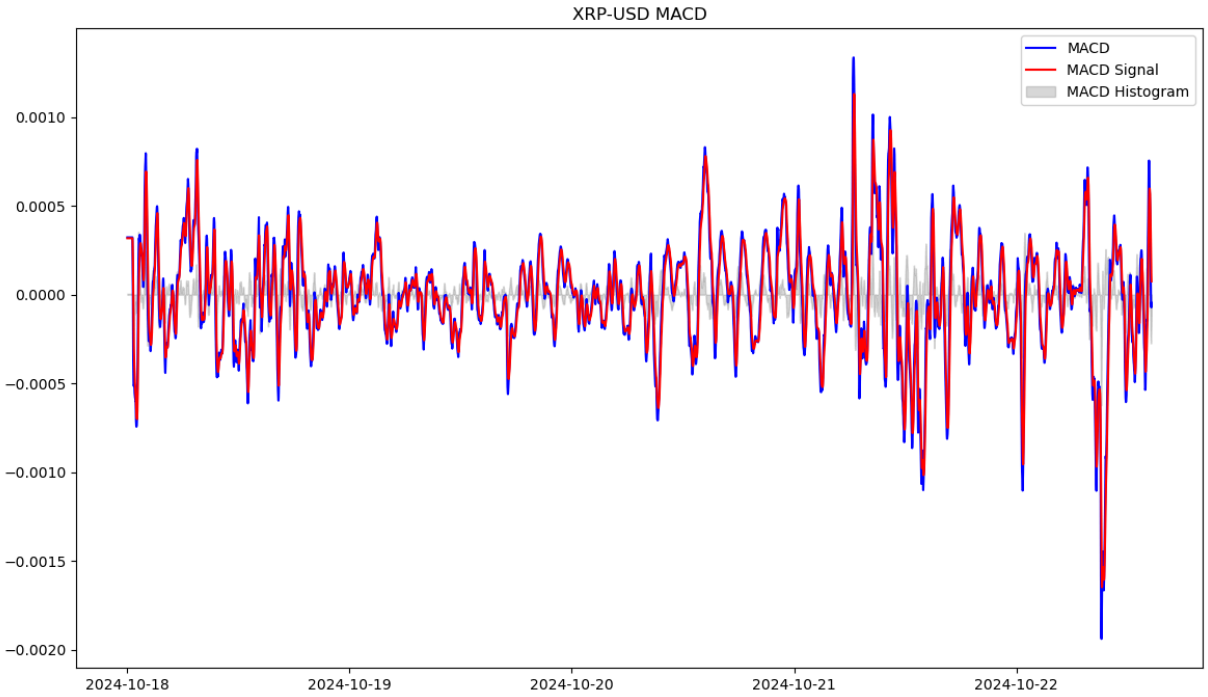

XRP-USD RSI
RSI
Overbought (70)
Oversold (30)
100
80
60
40
20
0
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22

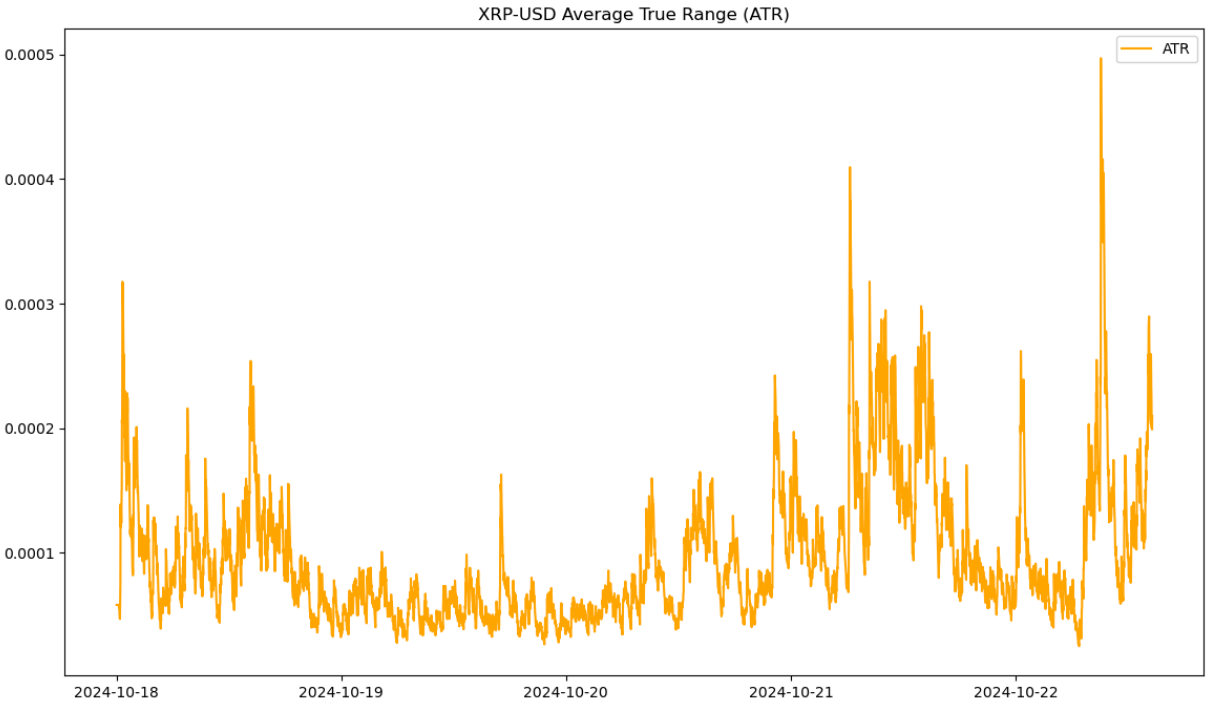

XRP-USD Returns
Returns
0.004
0.002
0.000
−0.002
−0.004
−0.006
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22


XRP-USD Volatility
Volatility
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22


XRP-USD Candlestick Chart
Price (USD)
0.56
0.555
0.55
0.545
0.54

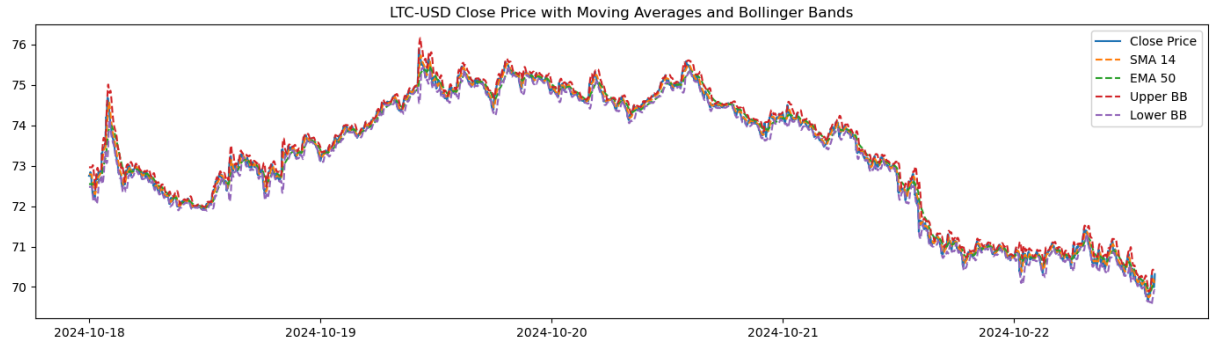

XRP-USD Trading Volume
1e7
Volume
1.75
1.50
1.25
1.00
0.75
0.50
0.25
0.00
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22
2024-10-23
Date

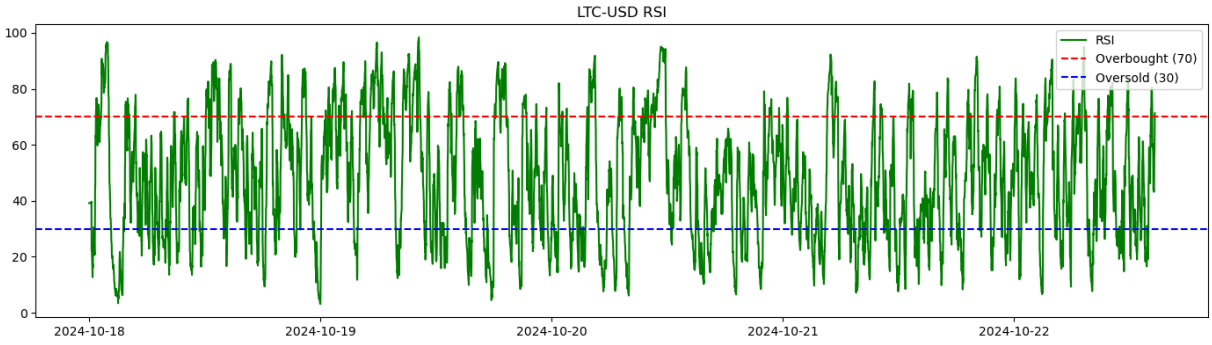


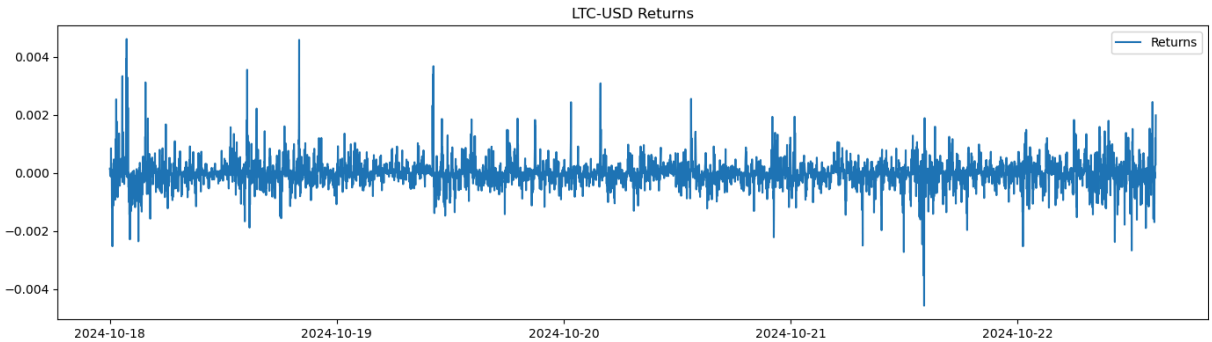


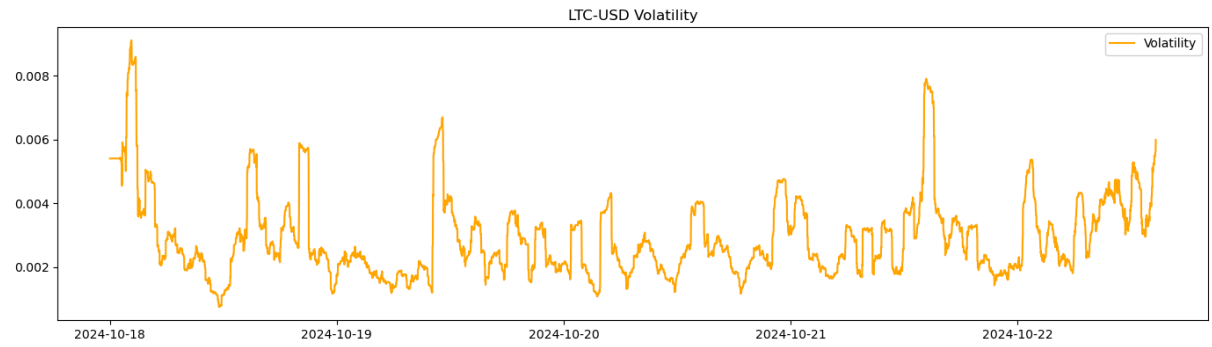

```
Data successfully saved to XRP-USD_technical_analysis in SQL Server.
SQL connection closed.
Processing data for LTC-USD...
[**********100%**********]  1 of 1 completed
```

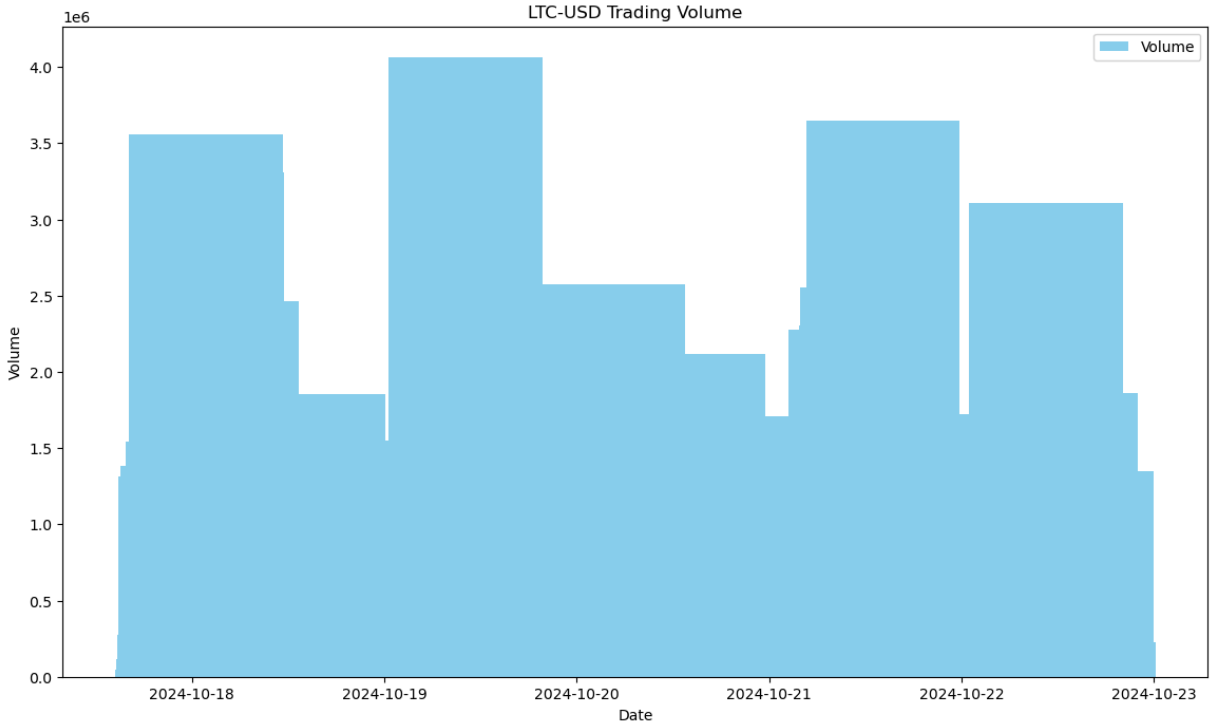
Error saving to SQL Server: (pyodbc.IntegrityError) ('23000', '[23000] [Microsoft][ODBC Driver 17 for SQL Server][SQL Server]Cannot insert an explicit value into a timestamp column. Use INSERT with a column list to exclude the timestamp column, or insert a DEFAULT into the timestamp column. (273) (SQLExecDirectW); [23000] [Microsoft][ODBC Driver 17 for SQL Server][SQL Server]Statement(s) could not be prepared. (8180)')
[SQL: INSERT INTO [LTC-USD_technical_analysis] ([Date], [Open], [High], [Low], [Close], [Adj Close], [Volume], returns, volatility, support, resistance, [SMA_14], [EMA_50], [BB_upper], [BB_middle], [BB_lower], [RSI], vwap, [fib_0.236], [fib_0.382], [fib_0. ... 6355 characters truncated ... ?, ?, ?, ?, ?, ?), (?, ?)]
[parameters: ('2024-10-18 00:00:00.000000 +00:00', 72.7493667602539, 72.7493667602539, 72.7493667602539, 72.7493667602539, 72.7493667602539, 0, 0.00015143562334274385, 0.005405351601748548, 72.21712493896484, 72.84589385986328, 72.78707122802734, 72.54617279052735, 72.96120100377932, 72.71843910217285, 72.47567720056638, 39.19540558301666, 72.76038360595703, 74.41001495361328, 73.50155838012695, 72.76732635498047, 72.03309432983399, 69.65617370605469, -0.136700703846941, -0.1593637793670821, 0.0226630755201411, 0.01907294137137277, '2024-10-18 00:01:00.000000 +00:00', 72.76038360595703, 72.76038360595703, 72.76038360595703, 72.76038360595703, 72.76038360595703, 47328, 0.00015143562334274385, 0.005405351601748548, 72.21712493896484, 72.84589385986328, 72.78707122802734, 72.54617279052735, 72.96120100377932, 72.71843910217285, 72.47567720056638, 39.19540558301666, 72.76038360595703, 74.41001495361328, 73.50155838012695, 72.76732635498047, 72.03309432983399, 69.65617370605469 ... 1979 parameters truncated ... 72.870361328125, 72.870361328125, 27168, 9.03626978594474e-05, 0.005241790834511336, 72.21712493896484, 72.870361328125, 72.81968797956195, 72.70993142461606, 72.89967158415756, 72.81402359008788, 72.72837559601821, 70.33293898144912, 72.57936447350299, 74.41001495361328, 73.50155838012695, 72.76732635498047, 72.03309432983399, 69.65617370605469, 0.053581329294729585, 0.05547836711741101, -0.0018970378226814272, 0.018519027681884157, '2024-10-18 01:16:00.000000 +00:00', 72.89740753173828, 72.89740753173828, 72.89740753173828, 72.89740753173828, 72.89740753173828, 23264, 0.00037115506387430663, 0.004555122135628794, 72.21712493896484, 72.89740753173828, 72.82502583095005, 72.71728342881693, 72.90341830854908, 72.8231273651123, 72.74283642167552, 73.329207310288, 72.5799253198114, 74.41001495361328, 73.50155838012695, 72.76732635498047, 72.03309432983399, 69.65617370605469, 0.05486055988585292, 0.05535480567109939, -0.0004942457852464693, 0.019128111676983948)]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
SQL connection closed.
Data saved to LTC-USD_technical_analysis_20241022_084127.csv

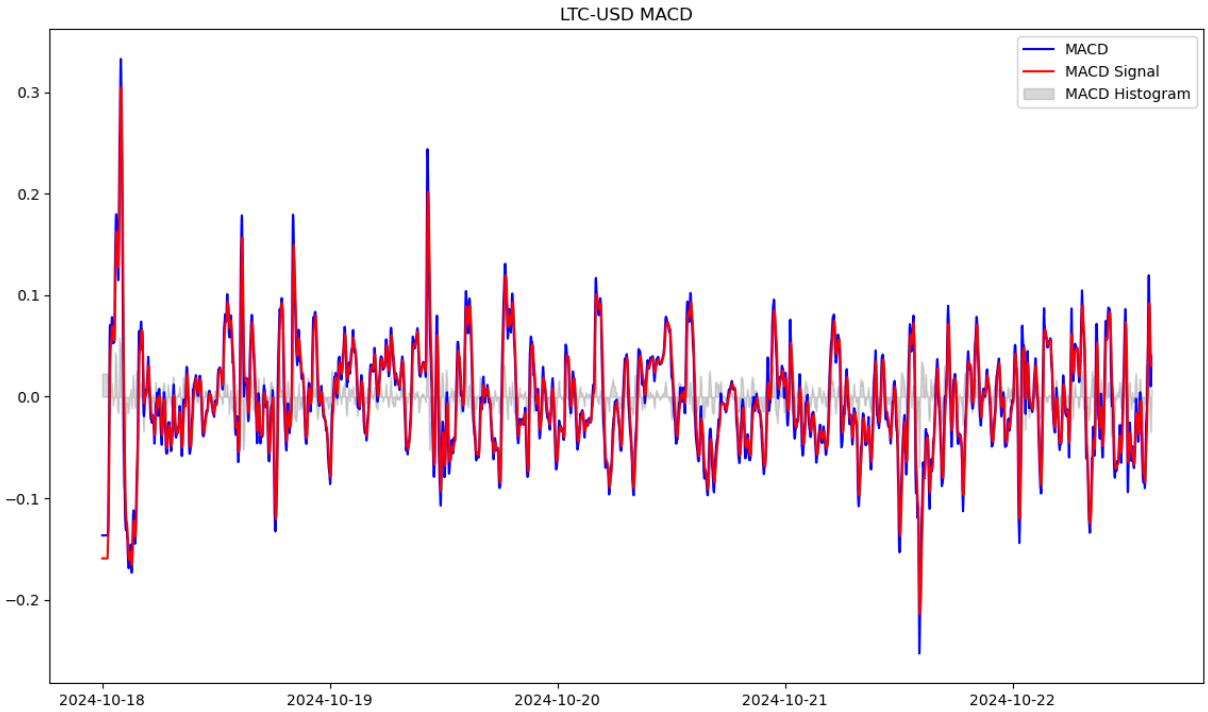

LTC-USD Close Price with Moving Averages and Bollinger Bands
Close Price
SMA 14
EMA 50
Upper BB
Lower BB
76
75
74
73
72
71
70
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22

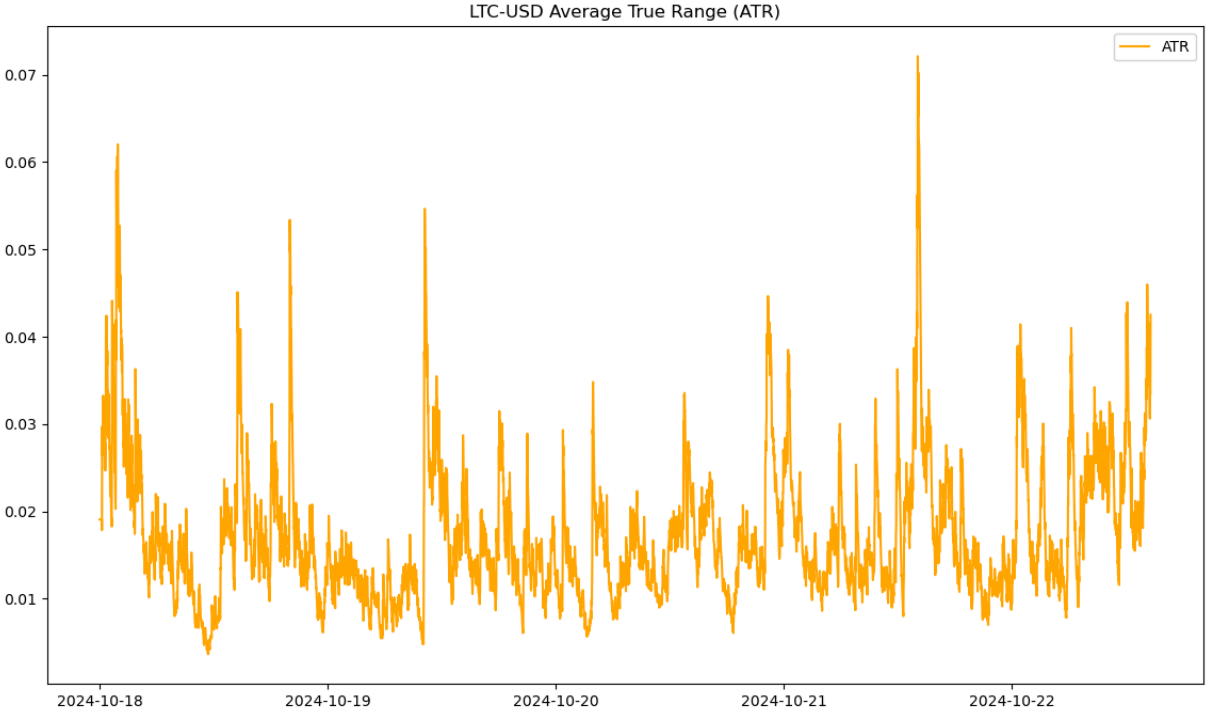

LTC-USD RSI
RSI
Overbought (70)
Oversold (30)
100
80
60
40
20
0
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22


LTC-USD Returns
Returns
0.004
0.002
0.000
−0.002
−0.004
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22


LTC-USD Volatility
Volatility
0.008
0.006
0.004
0.002
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22

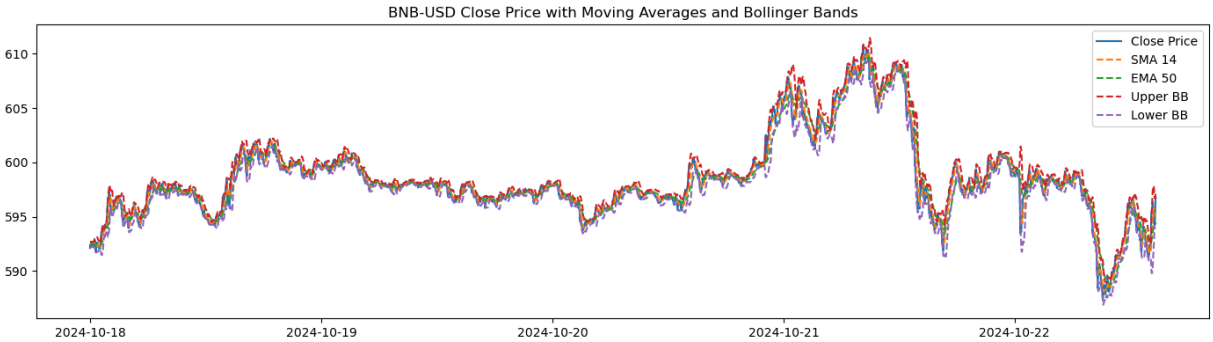

LTC-USD Trading Volume
1e6
Volume
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
Volume
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22
2024-10-23
Date

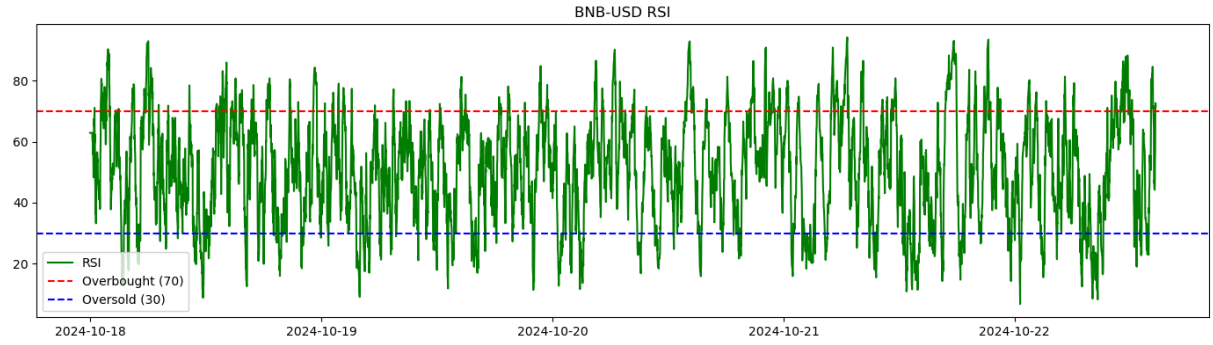


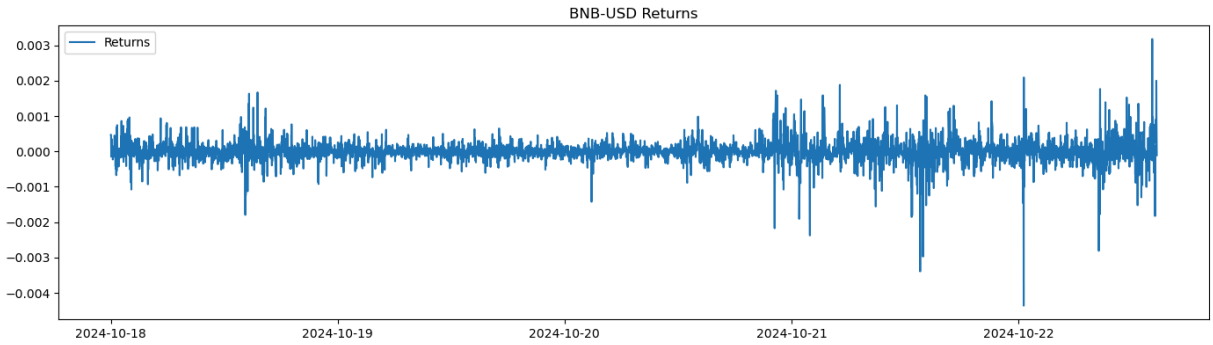


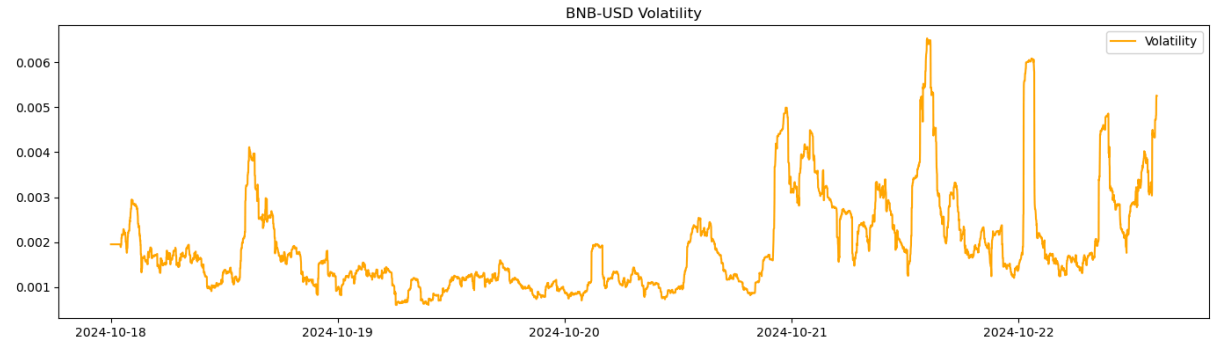

```
Data successfully saved to LTC-USD_technical_analysis in SQL Server.
SQL connection closed.
Processing data for BNB-USD...
[**********************100%**********************]  1 of 1 completed
```

Error saving to SQL Server: (pyodbc.IntegrityError) ('23000', '[23000] [Microsoft][ODBC Driver 17 for SQL Server][SQL Server]Cannot insert an explicit value into a timestamp column. Use INSERT with a column list to exclude the timestamp column, or insert a DEFAULT into the timestamp column. (273) (SQLExecDirectW); [23000] [Microsoft][ODBC Driver 17 for SQL Server][SQL Server]Statement(s) could not be prepared. (8180)')
[SQL: INSERT INTO [BNB-USD_technical_analysis] ([Date], [Open], [High], [Low], [Close], [Adj Close], [Volume], returns, volatility, support, resistance, [SMA_14], [EMA_50], [BB_upper], [BB_middle], [BB_lower], [RSI], vwap, [fib_0.236], [fib_0.382], [fib_0. ... 6355 characters truncated ... ?, ?, ?, ?, ?, ?), (?, ?)]
[parameters: ('2024-10-18 00:00:00.000000 +00:00', 592.1283569335938, 592.1283569335938, 592.1283569335938, 592.1283569335938, 592.1283569335938, 0, 0.00047127068161390007, 0.0019521613338283046, 591.7011108398438, 592.9518432617188, 592.4655412946429, 592.4451965332031, 592.6980158576399, 592.4477630615235, 592.197510265407, 62.8692380056444, 592.4074096679688, 605.2469274902344, 601.8556622314453, 599.1147766113281, 596.3738909912109, 587.5008544921875, 0.022043336294132132, 0.03859926968676872, -0.016555933392636585, 0.09268624441964286, '2024-10-18 00:01:00.000000 +00:00', 592.4074096679688, 592.4074096679688, 592.4074096679688, 592.4074096679688, 592.4074096679688, 82816, 0.00047127068161390007, 0.0019521613338283046, 591.7011108398438, 592.9518432617188, 592.4655412946429, 592.4451965332031, 592.6980158576399, 592.4477630615235, 592.197510265407, 62.8692380056444, 592.4074096679688, 605.2469274902344, 601.8556622314453, 599.1147766113281, 596.3738909912109, 587.5008544921875 ... 1979 parameters truncated ... 593.470703125, 593.470703125, 0, -5.820655924160256e-05, 0.0021644732982699253, 591.7011108398438, 593.6032104492188, 592.9919869559152, 592.6609921296971, 593.9631633945504, 592.7443695068359, 591.5255756191215, 72.54610525666871, 592.5299181753975, 605.2469274902344, 601.8556622314453, 599.1147766113281, 596.3738909912109, 587.5008544921875, 0.3324650931205042, 0.23040893764271678, 0.10205615547778743, 0.10795032941255156, '2024-10-18 01:16:00.000000 +00:00', 593.4104614257812, 593.4104614257812, 593.4104614257812, 593.4104614257812, 593.4104614257812, 0, -0.0001015074525188008 9, 0.0021678920220948674, 591.7011108398438, 593.6032104492188, 593.0905500139509, 592.6903830824847, 594.0259723752029, 592.8083282470703, 591.5906841189378, 69.56010317183079, 592.5299181753975, 605.2469274902344, 601.8556622314453, 599.1147766113281, 596.3738909912109, 587.5008544921875, 0.3256011980082576, 0.24944738971582495, 0.07615380829243265, 0.10454257011299431)]
(Background on this error at: https://sqlalche.me/e/20/gkpj)
SQL connection closed.
Data saved to BNB-USD_technical_analysis_20241022_084206.csv


BNB-USD Close Price with Moving Averages and Bollinger Bands
Close Price
SMA 14
EMA 50
Upper BB
Lower BB
610
605
600
595
590
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22


BNB-USD RSI
RSI
Overbought (70)
Oversold (30)
80
60
40
20
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22


BNB-USD Returns
Returns
0.003
0.002
0.001
0.000
−0.001
−0.002
−0.003
−0.004
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22


BNB-USD Volatility
Volatility
0.006
0.005
0.004
0.003
0.002
0.001
2024-10-18
2024-10-19
2024-10-20
2024-10-21
2024-10-22