In [1]:
```
pip install yfinance
```

```
Collecting yfinanceNote: you may need to restart the kernel to use updated packag
es.
  DEPRECATION: Building 'multitasking' using the legacy setup.py bdist_wheel mech
anism, which will be removed in a future version. pip 25.3 will enforce this beha
viour change. A possible replacement is to use the standardized build interface b
y setting the `--use-pep517` option, (possibly combined with `--no-build-isolatio
n`), or adding a `pyproject.toml` file to the source tree of 'multitasking'. Disc
ussion can be found at https://github.com/pypa/pip/issues/6334
```

```
  Downloading yfinance-1.0-py2.py3-none-any.whl.metadata (6.0 kB)
Requirement already satisfied: pandas>=1.3.0 in c:\users\hp\anaconda3\lib\site-pa
ckages (from yfinance) (2.2.3)
Requirement already satisfied: numpy>=1.16.5 in c:\users\hp\anaconda3\lib\site-pa
ckages (from yfinance) (2.1.3)
Requirement already satisfied: requests>=2.31 in c:\users\hp\anaconda3\lib\site-p
ackages (from yfinance) (2.32.3)
Collecting multitasking>=0.0.7 (from yfinance)
  Downloading multitasking-0.0.12.tar.gz (19 kB)
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Requirement already satisfied: platformdirs>=2.0.0 in c:\users\hp\anaconda3\lib\s
ite-packages (from yfinance) (4.3.7)
Requirement already satisfied: pytz>=2022.5 in c:\users\hp\anaconda3\lib\site-pac
kages (from yfinance) (2024.1)
Requirement already satisfied: frozendict>=2.3.4 in c:\users\hp\anaconda3\lib\sit
e-packages (from yfinance) (2.4.2)
Collecting peewee>=3.16.2 (from yfinance)
  Downloading peewee-3.18.3.tar.gz (3.0 MB)
     ---------------------------------------- 0.0/3.0 MB ? eta -:--:--
     ------ --------------------------------- 0.5/3.0 MB 4.2 MB/s eta 0:00:01
     ---------------- ----------------------- 1.3/3.0 MB 3.5 MB/s eta 0:00:01
     -------------------------- ------------ 2.1/3.0 MB 3.7 MB/s eta 0:00:01
     ----------------------------------- - 2.9/3.0 MB 3.9 MB/s eta 0:00:01
     ---------------------------------------- 3.0/3.0 MB 3.7 MB/s eta 0:00:00
  Installing build dependencies: started
  Installing build dependencies: finished with status 'done'
  Getting requirements to build wheel: started
  Getting requirements to build wheel: finished with status 'done'
  Preparing metadata (pyproject.toml): started
  Preparing metadata (pyproject.toml): finished with status 'done'
Requirement already satisfied: beautifulsoup4>=4.11.1 in c:\users\hp\anaconda3\li
b\site-packages (from yfinance) (4.12.3)
Collecting curl_cffi<0.14,>=0.7 (from yfinance)
  Downloading curl_cffi-0.13.0-cp39-abi3-win_amd64.whl.metadata (13 kB)
Requirement already satisfied: protobuf>=3.19.0 in c:\users\hp\anaconda3\lib\site
-packages (from yfinance) (5.29.3)
Collecting websockets>=13.0 (from yfinance)
  Downloading websockets-15.0.1-cp313-cp313-win_amd64.whl.metadata (7.0 kB)
Requirement already satisfied: cffi>=1.12.0 in c:\users\hp\anaconda3\lib\site-pac
kages (from curl_cffi<0.14,>=0.7->yfinance) (1.17.1)
Requirement already satisfied: certifi>=2024.2.2 in c:\users\hp\anaconda3\lib\sit
e-packages (from curl_cffi<0.14,>=0.7->yfinance) (2025.10.5)
Requirement already satisfied: soupsieve>1.2 in c:\users\hp\anaconda3\lib\site-pa
ckages (from beautifulsoup4>=4.11.1->yfinance) (2.5)
Requirement already satisfied: pycparser in c:\users\hp\anaconda3\lib\site-packag
es (from cffi>=1.12.0->curl_cffi<0.14,>=0.7->yfinance) (2.21)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\hp\anaconda3\li
b\site-packages (from pandas>=1.3.0->yfinance) (2.9.0.post0)
Requirement already satisfied: tzdata>=2022.7 in c:\users\hp\anaconda3\lib\site-p
ackages (from pandas>=1.3.0->yfinance) (2025.2)
Requirement already satisfied: six>=1.5 in c:\users\hp\anaconda3\lib\site-package
s (from python-dateutil>=2.8.2->pandas>=1.3.0->yfinance) (1.17.0)
Requirement already satisfied: charset-normalizer<4,>=2 in c:\users\hp\anaconda3
\lib\site-packages (from requests>=2.31->yfinance) (3.3.2)
Requirement already satisfied: idna<4,>=2.5 in c:\users\hp\anaconda3\lib\site-pac
kages (from requests>=2.31->yfinance) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\hp\anaconda3\lib\si
te-packages (from requests>=2.31->yfinance) (2.3.0)
Downloading yfinance-1.0-py2.py3-none-any.whl (127 kB)
```

```
Downloading curl_cffi-0.13.0-cp39-abi3-win_amd64.whl (1.6 MB)
   ---------------------------------------- 0.0/1.6 MB ? eta -:--:--
   ---------------------------------- ------- 1.3/1.6 MB 6.5 MB/s eta 0:00:01
   ---------------------------------------- 1.6/1.6 MB 5.7 MB/s eta 0:00:00
Downloading websockets-15.0.1-cp313-cp313-win_amd64.whl (176 kB)
Building wheels for collected packages: multitasking, peewee
  Building wheel for multitasking (setup.py): started
  Building wheel for multitasking (setup.py): finished with status 'done'
  Created wheel for multitasking: filename=multitasking-0.0.12-py3-none-any.whl s
ize=15618 sha256=119ab5e02d85bdb310e0e9ef0c29f38a3e2088537354ae62d8a1ce3e6a819234
  Stored in directory: c:\users\hp\appdata\local\pip\cache\wheels\1e\df\0f\e2bbb2
2d689b30c681feb5410ab64a2523437b34c8ecfc6476
  Building wheel for peewee (pyproject.toml): started
  Building wheel for peewee (pyproject.toml): finished with status 'done'
  Created wheel for peewee: filename=peewee-3.18.3-py3-none-any.whl size=139181 s
ha256=1ec284b78e72ce93b5d0e26409fa1b512c73e12bc76824606384e3ca0ae2b15f
  Stored in directory: c:\users\hp\appdata\local\pip\cache\wheels\8c\a9\a4\df972c
d49f865ffde174d9c5b26f14f08f8a363ed31e10ff91
Successfully built multitasking peewee
Installing collected packages: peewee, multitasking, websockets, curl_cffi, yfina
nce

   ---------------------------------------- 0/5 [peewee]
   ---------------------------------------- 0/5 [peewee]
   --------------- ------------------------ 2/5 [websockets]
   --------------- ------------------------ 2/5 [websockets]
   --------------- ------------------------ 2/5 [websockets]
   ----------------------- ---------------- 3/5 [curl_cffi]
   ----------------------- ---------------- 3/5 [curl_cffi]
   ----------------------- ---------------- 3/5 [curl_cffi]
   ------------------------------- -------- 4/5 [yfinance]
   ------------------------------- -------- 4/5 [yfinance]
   ------------------------------- -------- 4/5 [yfinance]
   ------------------------------- -------- 4/5 [yfinance]
   ------------------------------- -------- 4/5 [yfinance]
   ------------------------------- -------- 4/5 [yfinance]
   ------------------------------- -------- 4/5 [yfinance]
   ---------------------------------------- 5/5 [yfinance]

Successfully installed curl_cffi-0.13.0 multitasking-0.0.12 peewee-3.18.3 websock
ets-15.0.1 yfinance-1.0
```

In [1]:
```python
import seaborn as sns
import yfinance as yf
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
#from sklearn import metrics
#from sklearn.metrics import accuracy_score, classification_report, confusion_ma
```

In [2]:
```python
#The code fetches historical price data for Bitcoin, Ethereum, Tether, and Binan
#This cleaned data can then be used for further analysis or machine learning tas


btc = yf.Ticker('BTC-USD')
prices1 = btc.history(period='5y')
prices1.drop(columns=['Open', 'High', 'Low', 'Dividends', 'Stock Splits'], axis
```

```
eth = yf.Ticker('ETH-USD')
prices2 = eth.history(period='5y')
prices2.drop(columns=['Open', 'High', 'Low', 'Dividends', 'Stock Splits'], axis

usdt = yf.Ticker('USDT-USD')
prices3 = usdt.history(period='5y')
prices3.drop(columns=['Open', 'High', 'Low', 'Dividends', 'Stock Splits'], axis

bnb = yf.Ticker('BNB-USD')
prices4 = bnb.history(period='5y')
prices4.drop(columns=['Open', 'High', 'Low', 'Dividends', 'Stock Splits'], axis
```

In [3]:
```
#The parameters lsuffix and rsuffix in the join method are used to add suffixes
# This is necessary to avoid column name conflicts when the two DataFrames have

p1 = prices1.join(prices2, lsuffix = ' (BTC)', rsuffix = ' (ETH)')
p2 = prices3.join(prices4, lsuffix = ' (USDT)', rsuffix = ' (BNB)')
data = p1.join(p2, lsuffix = '_', rsuffix = '_')
```

In [4]:
```
data.head()
```

Out[4]:

Date	Close (BTC)	Volume (BTC)	Close (ETH)	Volume (ETH)	Close (USDT)	
2021-01-03 00:00:00+00:00	32782.023438	78665235202	975.507690	45200463368	1.000514	120425
2021-01-04 00:00:00+00:00	31971.914062	81163475344	1040.233032	56945985763	1.000128	125906
2021-01-05 00:00:00+00:00	33992.429688	67547324782	1100.006104	41535932781	1.002202	101918
2021-01-06 00:00:00+00:00	36824.363281	75289433811	1207.112183	44699914188	1.001528	116105
2021-01-07 00:00:00+00:00	39371.042969	84762141031	1225.678101	40468027280	1.000400	129467

In [5]:
```
data.tail()
```

Out[5]:

Date	Close (BTC)	Volume (BTC)	Close (ETH)	Volume (ETH)	Close (USDT)	V (
2025-12-30 00:00:00+00:00	88430.132812	35586356225	2971.416748	18816704381	0.998867	741190
2025-12-31 00:00:00+00:00	87508.828125	33830210616	2967.037598	16451891101	0.998449	702594
2026-01-01 00:00:00+00:00	88731.984375	18849043990	3000.394287	10268796662	0.998745	505486
2026-01-02 00:00:00+00:00	89944.695312	46398906171	3124.422607	25242778003	0.999672	961285
2026-01-03 00:00:00+00:00	89809.804688	42759680000	3103.197998	23285262336	0.999629	927059

In [6]:
```
data.shape
```

Out[6]:
```
(1827, 8)
```

In [7]:
```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1827 entries, 2021-01-03 00:00:00+00:00 to 2026-01-03 00:00:00+00:
00
Data columns (total 8 columns):
 #   Column         Non-Null Count  Dtype  
---  ------         --------------  -----  
 0   Close (BTC)    1827 non-null   float64
 1   Volume (BTC)   1827 non-null   int64  
 2   Close (ETH)    1827 non-null   float64
 3   Volume (ETH)   1827 non-null   int64  
 4   Close (USDT)   1827 non-null   float64
 5   Volume (USDT)  1827 non-null   int64  
 6   Close (BNB)    1827 non-null   float64
 7   Volume (BNB)   1827 non-null   int64  
dtypes: float64(4), int64(4)
memory usage: 128.5 KB
```

In [9]:
```
data.isna().sum()
```

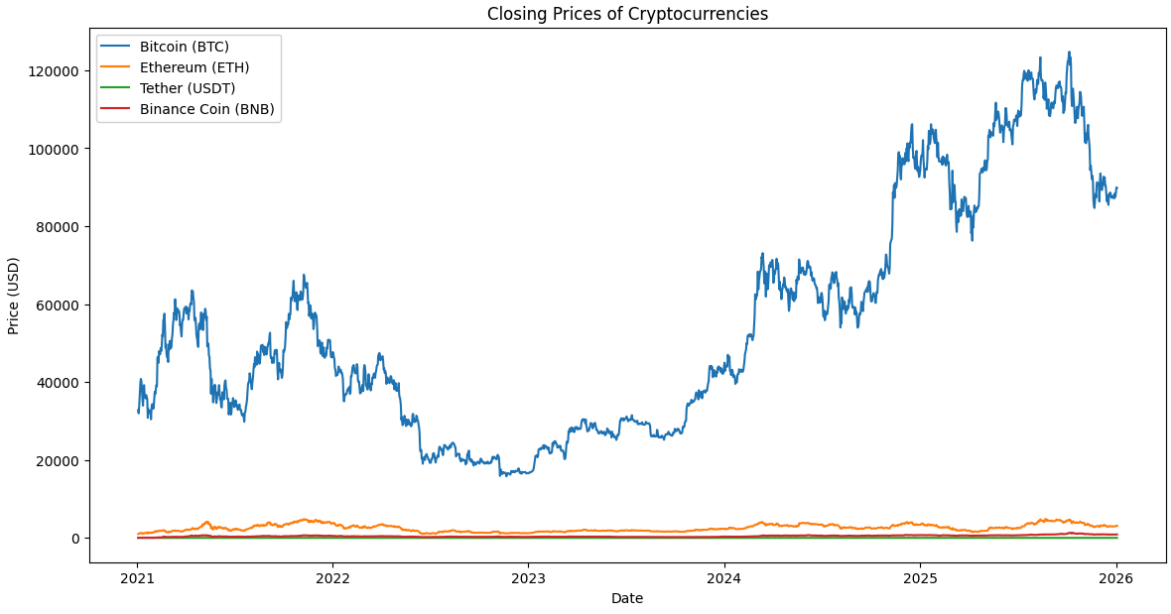
Out[9]:
```
Close (BTC)      0
Volume (BTC)     0
Close (ETH)      0
Volume (ETH)     0
Close (USDT)     0
Volume (USDT)    0
Close (BNB)      0
Volume (BNB)     0
dtype: int64
```

In [10]:
```
data.describe()
```

Out[10]:

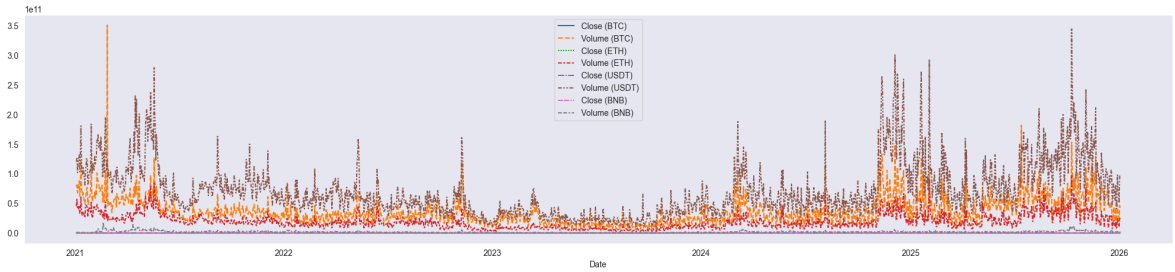
	Close (BTC)	Volume (BTC)	Close (ETH)	Volume (ETH)	Close (USDT)	Volu (US
count	1827.000000	1.827000e+03	1827.000000	1.827000e+03	1827.000000	1.827000e
mean	54509.858616	3.715492e+10	2537.632520	1.954183e+10	1.000146	7.094093e
std	29479.462652	2.303848e+10	909.650938	1.294682e+10	0.000710	4.503878e
min	15787.284180	5.331173e+09	975.507690	2.081626e+09	0.995872	9.989859e
25%	29412.204102	2.132182e+10	1792.933411	1.025234e+10	0.999894	4.011224e
50%	46481.105469	3.180847e+10	2455.935059	1.645097e+10	1.000142	6.045086e
75%	69297.523438	4.714782e+10	3234.704468	2.505416e+10	1.000385	8.832358e
max	124752.531250	3.509679e+11	4831.348633	9.773662e+10	1.011530	3.443980e

In [11]:
```
#Visualize the Closing Prices
# create a line plot to visualize the closing prices of all four cryptocurrencie
plt.figure(figsize=(14, 7))
plt.plot(data.index, data['Close (BTC)'], label='Bitcoin (BTC)')
plt.plot(data.index, data['Close (ETH)'], label='Ethereum (ETH)')
plt.plot(data.index, data['Close (USDT)'], label='Tether (USDT)')
plt.plot(data.index, data['Close (BNB)'], label='Binance Coin (BNB)')
plt.title('Closing Prices of Cryptocurrencies')
plt.xlabel('Date')
plt.ylabel('Price (USD)')
plt.legend()
plt.show()
```

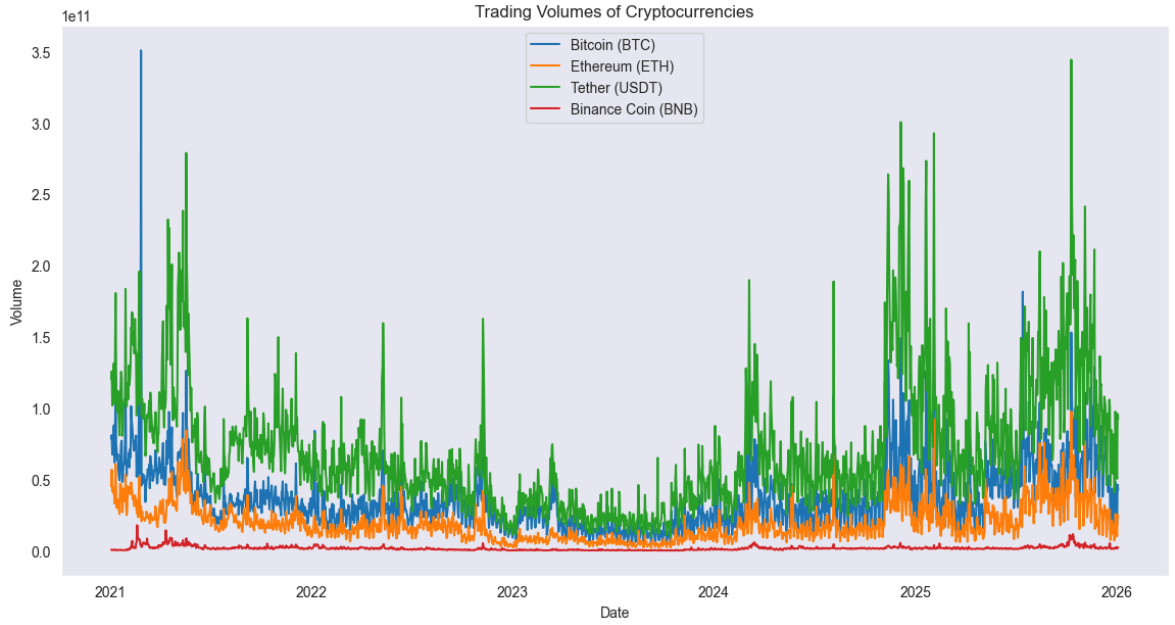


In [12]:
```
plt.figure(figsize= (25, 5))
sns.set_style('dark')
sns.lineplot(data= data)
```

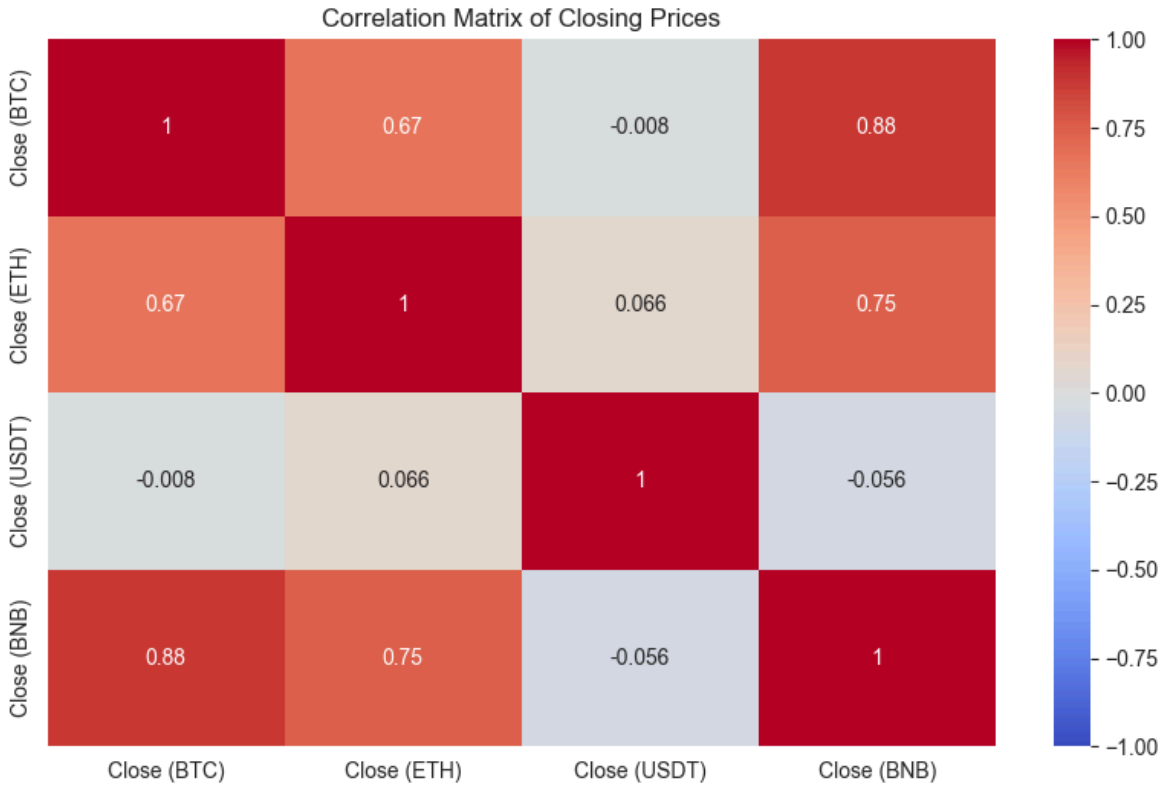
Out[12]: <Axes: xlabel='Date'>



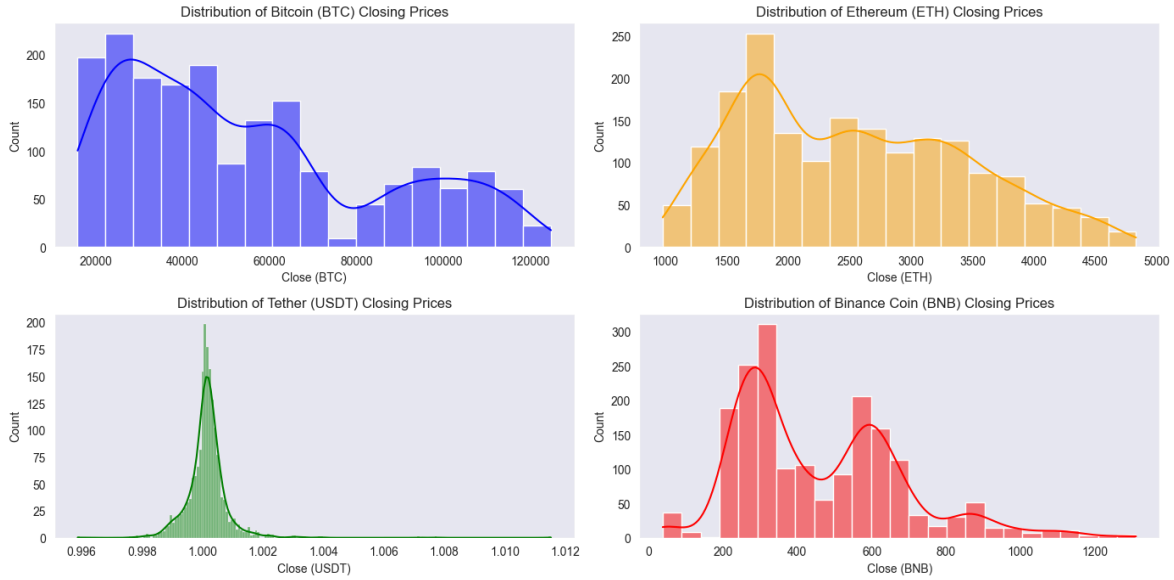
In [13]:
```
plt.figure(figsize=(14, 7))
plt.plot(data.index, data['Volume (BTC)'], label='Bitcoin (BTC)')
plt.plot(data.index, data['Volume (ETH)'], label='Ethereum (ETH)')
plt.plot(data.index, data['Volume (USDT)'], label='Tether (USDT)')
plt.plot(data.index, data['Volume (BNB)'], label='Binance Coin (BNB)')
plt.title('Trading Volumes of Cryptocurrencies')
plt.xlabel('Date')
plt.ylabel('Volume')
plt.legend()
plt.show()
```



In [14]:
```
 #Correlation Analysis
#We'll analyze the correlation between the closing prices of the cryptocurrencie
# Calculate the correlation matrix
corr_matrix = data[['Close (BTC)', 'Close (ETH)', 'Close (USDT)', 'Close (BNB)']

# Plot the heatmap
plt.figure(figsize=(10, 6))
sns.heatmap(corr_matrix, annot=True, cmap='coolwarm', vmin=-1, vmax=1)
plt.title('Correlation Matrix of Closing Prices')
plt.show()
```

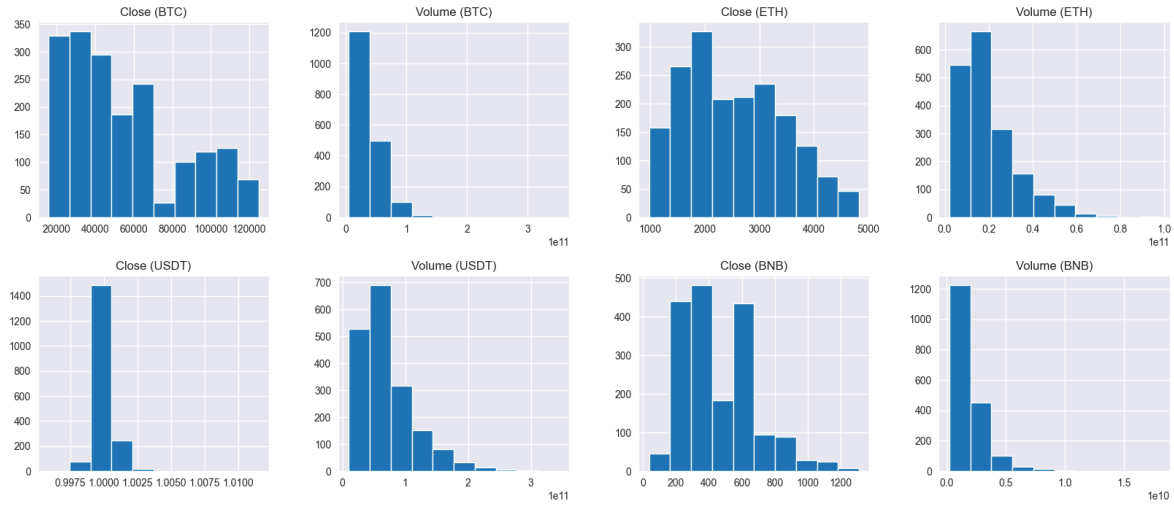


In [15]:
```
# Distribution of Closing Prices
#Let's plot the distribution of closing prices for each cryptocurrency:
plt.figure(figsize=(14, 7))

plt.subplot(2, 2, 1)
sns.histplot(data['Close (BTC)'], kde=True, color='blue')
plt.title('Distribution of Bitcoin (BTC) Closing Prices')

plt.subplot(2, 2, 2)
sns.histplot(data['Close (ETH)'], kde=True, color='orange')
plt.title('Distribution of Ethereum (ETH) Closing Prices')

plt.subplot(2, 2, 3)
sns.histplot(data['Close (USDT)'], kde=True, color='green')
plt.title('Distribution of Tether (USDT) Closing Prices')

plt.subplot(2, 2, 4)
sns.histplot(data['Close (BNB)'], kde=True, color='red')
plt.title('Distribution of Binance Coin (BNB) Closing Prices')

plt.tight_layout()
plt.show()
```



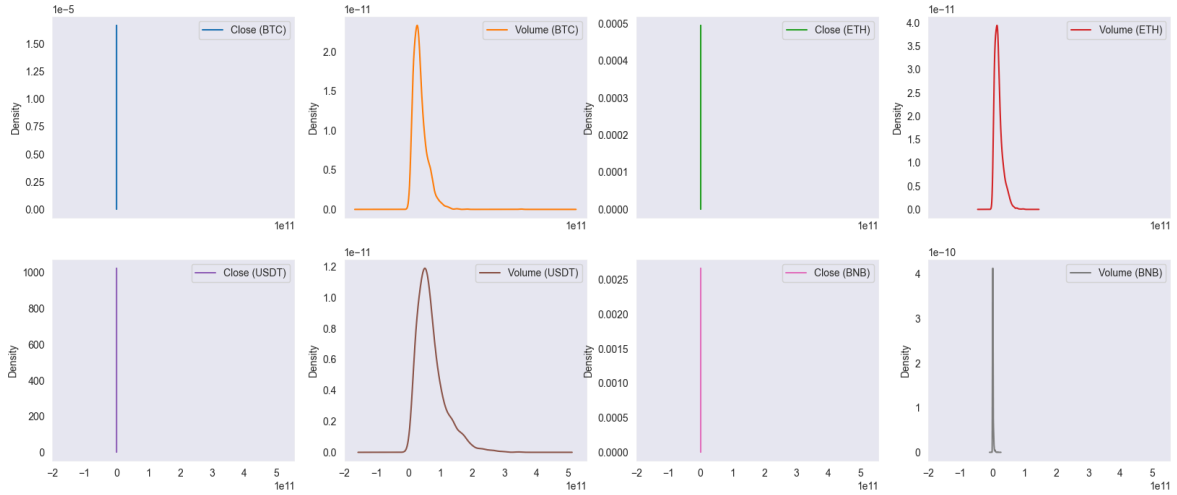
In [16]:
```
data.hist(figsize=(20, 8), layout=(2, 4))
```

Out[16]:
```
array([[<Axes: title={'center': 'Close (BTC)'}>,
        <Axes: title={'center': 'Volume (BTC)'}>,
        <Axes: title={'center': 'Close (ETH)'}>,
        <Axes: title={'center': 'Volume (ETH)'}>],
       [<Axes: title={'center': 'Close (USDT)'}>,
        <Axes: title={'center': 'Volume (USDT)'}>,
        <Axes: title={'center': 'Close (BNB)'}>,
        <Axes: title={'center': 'Volume (BNB)'}>]], dtype=object)
```



In [17]:
```
data.plot(kind = "kde", subplots = True, layout = (2, 4), figsize = (20, 8))
```

Out[17]:
```
array([[<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>, <Axes: ylabel='Density'>],
       [<Axes: ylabel='Density'>, <Axes: ylabel='Density'>,
        <Axes: ylabel='Density'>, <Axes: ylabel='Density'>]], dtype=object)
```



In [18]:
```
sns.pairplot(data.sample(n=100));
```

In [20]:
```
X = data.drop(columns = ['Close (BTC)'], axis = 1)
Y = data.loc[:, 'Close (BTC)']
```

In [21]:
```
X.head()
```

Out[21]:

Date	Volume (BTC)	Close (ETH)	Volume (ETH)	Close (USDT)	Volume (USDT)	C (B
2021-01-03 00:00:00+00:00	78665235202	975.507690	45200463368	1.000514	120425679796	41.148
2021-01-04 00:00:00+00:00	81163475344	1040.233032	56945985763	1.000128	125906387011	40.926
2021-01-05 00:00:00+00:00	67547324782	1100.006104	41535932781	1.002202	101918715244	41.734
2021-01-06 00:00:00+00:00	75289433811	1207.112183	44699914188	1.001528	116105139289	42.165
2021-01-07 00:00:00+00:00	84762141031	1225.678101	40468027280	1.000400	129467601516	43.449

In [22]:
```
X.tail()
```

Out[22]:

Date	Volume (BTC)	Close (ETH)	Volume (ETH)	Close (USDT)	Volume (USDT)	C (B
2025-12-30 00:00:00+00:00	35586356225	2971.416748	18816704381	0.998867	74119035312	860.553
2025-12-31 00:00:00+00:00	33830210616	2967.037598	16451891101	0.998449	70259461189	863.257
2026-01-01 00:00:00+00:00	18849043990	3000.394287	10268796662	0.998745	50548666268	863.054
2026-01-02 00:00:00+00:00	46398906171	3124.422607	25242778003	0.999672	96128566387	880.844
2026-01-03 00:00:00+00:00	42759680000	3103.197998	23285262336	0.999629	92705955840	873.671

In [23]:
```
Y.head()
```

Out[23]:
```
Date
2021-01-03 00:00:00+00:00    32782.023438
2021-01-04 00:00:00+00:00    31971.914062
2021-01-05 00:00:00+00:00    33992.429688
2021-01-06 00:00:00+00:00    36824.363281
2021-01-07 00:00:00+00:00    39371.042969
Name: Close (BTC), dtype: float64
```

In [24]:
```
# Split the data into training and testing sets
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_
```

In [25]:
```
# Print the shapes of the resulting datasets
print(f'X_train shape: {X_train.shape}')
```

```
print(f'X_test shape: {X_test.shape}')
print(f'y_train shape: {Y_train.shape}')
print(f'y_test shape: {Y_test.shape}')
```

```
X_train shape: (1461, 7)
X_test shape: (366, 7)
y_train shape: (1461,)
y_test shape: (366,)
```

In [26]:
```
#SelectKBest
#SelectKBest is a feature selection method provided by scikit-learn (sklearn) th
#This function evaluates each feature independently and selects those that have

#Parameters
#k: Specifies the number of top features to select. In your case, k=4 indicates

from sklearn.feature_selection import SelectKBest

fs = SelectKBest(k=4)
X_train = fs.fit_transform(X_train, Y_train)
X_test = fs.transform(X_test)
```

```
C:\Users\Hp\anaconda3\Lib\site-packages\sklearn\feature_selection\_univariate_sel
ection.py:108: RuntimeWarning: invalid value encountered in divide
  msw = sswn / float(dfwn)
```

In [27]:
```
mask = fs.get_support()
selected_features = X.columns[mask]
print("Selected Features:", selected_features)
```

```
Selected Features: Index(['Close (USDT)', 'Volume (USDT)', 'Close (BNB)', 'Volume
(BNB)'], dtype='object')
```

In [28]:
```
X_train
```

Out[28]:
```
array([[1.00015497e+00, 6.59355365e+10, 3.87057343e+02, 1.79623565e+09],
       [1.00047398e+00, 7.95623781e+10, 3.98251465e+02, 2.28811705e+09],
       [1.00028098e+00, 5.35244431e+10, 2.96969055e+02, 8.51195253e+08],
       ...,
       [1.00002396e+00, 1.39888129e+11, 6.89416748e+02, 2.36227653e+09],
       [9.99773979e-01, 4.41267496e+10, 2.49593872e+02, 1.03098597e+09],
       [9.99162972e-01, 3.48790989e+10, 2.73919983e+02, 6.85307608e+08]])
```

In [29]:
```
#MinMaxScaler is a preprocessing method in scikit-learn that transforms features
# It's often used when your data needs to be normalized within a specific range
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

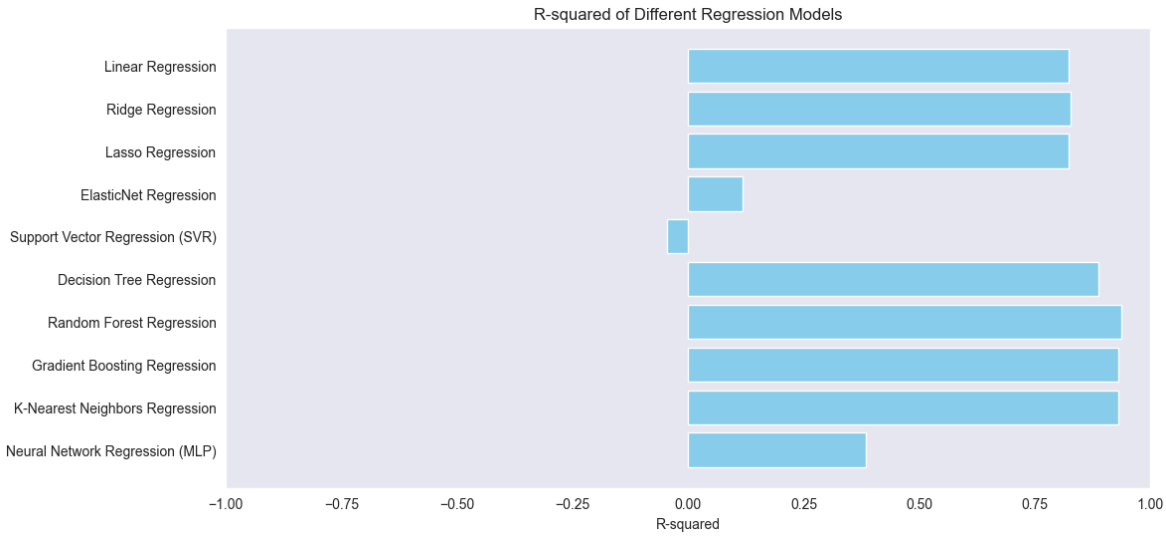
In [30]:
```
# implementation of 10 different regression algorithms using scikit-learn. Each

#Import Libraries and Generate Sample Data

from sklearn.datasets import make_regression
from sklearn.linear_model import LinearRegression, Ridge, Lasso, ElasticNet
from sklearn.svm import SVR
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor
from sklearn.neighbors import KNeighborsRegressor
```

```
from sklearn.neural_network import MLPRegressor
from sklearn.metrics import mean_squared_error, r2_score
```

In [31]:
```
#Define Models and Perform Training and Evaluation
models = {
    'Linear Regression': LinearRegression(),
    'Ridge Regression': Ridge(alpha=1.0),
    'Lasso Regression': Lasso(alpha=1.0),
    'ElasticNet Regression': ElasticNet(alpha=1.0, l1_ratio=0.5),
    'Support Vector Regression (SVR)': SVR(kernel='rbf'),
    'Decision Tree Regression': DecisionTreeRegressor(),
    'Random Forest Regression': RandomForestRegressor(n_estimators=100),
    'Gradient Boosting Regression': GradientBoostingRegressor(n_estimators=100,
    'K-Nearest Neighbors Regression': KNeighborsRegressor(n_neighbors=5),
    'Neural Network Regression (MLP)': MLPRegressor(hidden_layer_sizes=(100, 50)
}

# Train and evaluate each model
results = {'Model': [], 'MSE': [], 'R-squared': []}

for name, model in models.items():
    # Train the model
    model.fit(X_train, Y_train)

    # Predict on test set
    Y_pred = model.predict(X_test)

    # Evaluate model
    mse = mean_squared_error(Y_test, Y_pred)
    r2 = r2_score(Y_test, Y_pred)

    # Store results
    results['Model'].append(name)
    results['MSE'].append(mse)
    results['R-squared'].append(r2)

    # Print results
    print(f"----- {name} -----")
    print(f"Mean Squared Error (MSE): {mse}")
    print(f"R-squared: {r2}")
    print()

# Convert results to DataFrame for visualization
results_df = pd.DataFrame(results)
print(results_df)

# Plotting the results
plt.figure(figsize=(12, 6))
plt.barh(results_df['Model'], results_df['R-squared'], color='skyblue')
plt.xlabel('R-squared')
plt.title('R-squared of Different Regression Models')
plt.xlim(-1, 1)
plt.gca().invert_yaxis()
plt.show()
```

```
----- Linear Regression -----
Mean Squared Error (MSE): 152155760.24799544
R-squared: 0.8230895798921432

----- Ridge Regression -----
Mean Squared Error (MSE): 148532002.7200102
R-squared: 0.8273028969929876

----- Lasso Regression -----
Mean Squared Error (MSE): 152042233.98714682
R-squared: 0.8232215760680833

----- ElasticNet Regression -----
Mean Squared Error (MSE): 759034925.7388706
R-squared: 0.11747549110110933

----- Support Vector Regression (SVR) -----
Mean Squared Error (MSE): 900335178.4723414
R-squared: -0.046813307638299495

----- Decision Tree Regression -----
Mean Squared Error (MSE): 97403387.20108831
R-squared: 0.8867497745626768

----- Random Forest Regression -----
Mean Squared Error (MSE): 54447325.75665544
R-squared: 0.9366944816439432

----- Gradient Boosting Regression -----
Mean Squared Error (MSE): 59152994.00142416
R-squared: 0.9312232346486709

----- K-Nearest Neighbors Regression -----
Mean Squared Error (MSE): 59994065.13298272
R-squared: 0.9302453272268151

C:\Users\Hp\anaconda3\Lib\site-packages\sklearn\neural_network\_multilayer_perceptron.py:691: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) reached and the optimization hasn't converged yet.
  warnings.warn(
----- Neural Network Regression (MLP) -----
Mean Squared Error (MSE): 529766573.3000398
R-squared: 0.38404417362276955
```

	Model	MSE	R-squared
0	Linear Regression	1.521558e+08	0.823090
1	Ridge Regression	1.485320e+08	0.827303
2	Lasso Regression	1.520422e+08	0.823222
3	ElasticNet Regression	7.590349e+08	0.117475
4	Support Vector Regression (SVR)	9.003352e+08	-0.046813
5	Decision Tree Regression	9.740339e+07	0.886750
6	Random Forest Regression	5.444733e+07	0.936694
7	Gradient Boosting Regression	5.915299e+07	0.931223
8	K-Nearest Neighbors Regression	5.999407e+07	0.930245
9	Neural Network Regression (MLP)	5.297666e+08	0.384044


R-squared of Different Regression Models
Linear Regression
Ridge Regression
Lasso Regression
ElasticNet Regression
Support Vector Regression (SVR)
Decision Tree Regression
Random Forest Regression
Gradient Boosting Regression
K-Nearest Neighbors Regression
Neural Network Regression (MLP)
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
R-squared

In [32]:
```
import pickle
import numpy as np
from sklearn.datasets import make_regression
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error, r2_score

# Generate sample data
X, Y = make_regression(n_samples=1000, n_features=10, noise=0.1, random_state=0)


# Scale the features (optional but recommended for some algorithms)
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# Initialize Random Forest Regressor
model_rf = RandomForestRegressor(n_estimators=100, random_state=0)

# Train the model
model_rf.fit(X_train, Y_train)

# Save the model to a file
filename = 'random_forest_model.pkl'
pickle.dump(model_rf, open(filename, 'wb'))

# Save scaler to a file
with open('scaler.pkl', 'wb') as f:
    pickle.dump(scaler, f)

# Load the model from the file
loaded_model = pickle.load(open(filename, 'rb'))

# Predict using the loaded model
Y_pred = loaded_model.predict(X_test)

# Evaluate the loaded model
mse = mean_squared_error(Y_test, Y_pred)
r2 = r2_score(Y_test, Y_pred)

print(f"Loaded Random Forest Regression - Mean Squared Error (MSE): {mse}")
print(f"Loaded Random Forest Regression - R-squared: {r2}")
```

```
Loaded Random Forest Regression - Mean Squared Error (MSE): 53560343.460057266
Loaded Random Forest Regression - R-squared: 0.9377257696508098
```

In [ ]: