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

```
Requirement already satisfied: yfinance in c:\users\anitha\anaconda3\lib\site-packag
es (1.0)
Requirement already satisfied: pandas>=1.3.0 in c:\users\anitha\anaconda3\lib\site-p
ackages (from yfinance) (2.3.3)
Requirement already satisfied: numpy>=1.16.5 in c:\users\anitha\anaconda3\lib\site-p
ackages (from yfinance) (2.3.5)
Requirement already satisfied: requests>=2.31 in c:\users\anitha\anaconda3\lib\site-
packages (from yfinance) (2.32.5)
Requirement already satisfied: multitasking>=0.0.7 in c:\users\anitha\anaconda3\lib
\site-packages (from yfinance) (0.0.12)
Requirement already satisfied: platformdirs>=2.0.0 in c:\users\anitha\anaconda3\lib
\site-packages (from yfinance) (4.5.0)
Requirement already satisfied: pytz>=2022.5 in c:\users\anitha\anaconda3\lib\site-pa
ckages (from yfinance) (2025.2)
Requirement already satisfied: frozendict>=2.3.4 in c:\users\anitha\anaconda3\lib\si
te-packages (from yfinance) (2.4.6)
Requirement already satisfied: peewee>=3.16.2 in c:\users\anitha\anaconda3\lib\site-
packages (from yfinance) (3.18.3)
Requirement already satisfied: beautifulsoup4>=4.11.1 in c:\users\anitha\anaconda3\l
ib\site-packages (from yfinance) (4.13.5)
Requirement already satisfied: curl_cffi<0.14,>=0.7 in c:\users\anitha\anaconda3\lib
\site-packages (from yfinance) (0.13.0)
Requirement already satisfied: protobuf>=3.19.0 in c:\users\anitha\anaconda3\lib\sit
e-packages (from yfinance) (5.29.3)
Requirement already satisfied: websockets>=13.0 in c:\users\anitha\anaconda3\lib\sit
e-packages (from yfinance) (15.0.1)
Requirement already satisfied: cffi>=1.12.0 in c:\users\anitha\anaconda3\lib\site-pa
ckages (from curl_cffi<0.14,>=0.7->yfinance) (2.0.0)
Requirement already satisfied: certifi>=2024.2.2 in c:\users\anitha\anaconda3\lib\si
te-packages (from curl_cffi<0.14,>=0.7->yfinance) (2025.11.12)
Requirement already satisfied: soupsieve>1.2 in c:\users\anitha\anaconda3\lib\site-p
ackages (from beautifulsoup4>=4.11.1->yfinance) (2.5)
Requirement already satisfied: typing-extensions>=4.0.0 in c:\users\anitha\anaconda3
\lib\site-packages (from beautifulsoup4>=4.11.1->yfinance) (4.15.0)
Requirement already satisfied: pycparser in c:\users\anitha\anaconda3\lib\site-packa
ges (from cffi>=1.12.0->curl_cffi<0.14,>=0.7->yfinance) (2.23)
Requirement already satisfied: python-dateutil>=2.8.2 in c:\users\anitha\anaconda3\l
ib\site-packages (from pandas>=1.3.0->yfinance) (2.9.0.post0)
Requirement already satisfied: tzdata>=2022.7 in c:\users\anitha\anaconda3\lib\site-
packages (from pandas>=1.3.0->yfinance) (2025.2)
Requirement already satisfied: six>=1.5 in c:\users\anitha\anaconda3\lib\site-packag
es (from python-dateutil>=2.8.2->pandas>=1.3.0->yfinance) (1.17.0)
Requirement already satisfied: charset_normalizer<4,>=2 in c:\users\anitha\anaconda3
\lib\site-packages (from requests>=2.31->yfinance) (3.4.4)
Requirement already satisfied: idna<4,>=2.5 in c:\users\anitha\anaconda3\lib\site-pa
ckages (from requests>=2.31->yfinance) (3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in c:\users\anitha\anaconda3\lib\s
ite-packages (from requests>=2.31->yfinance) (2.5.0)
```

In [5]:
```
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.metrics import mean_absolute_error, mean_squared_error
```

In [6]:
```
btc = yf.Ticker('BTC-USD')
prices1 = btc.history(period='5y')
prices1.drop(columns=['Open', 'High', 'Low', 'Dividends', 'Stock Splits'], axis = 1

eth = yf.Ticker('ETH-USD')
prices2 = eth.history(period='5y')
prices2.drop(columns=['Open', 'High', 'Low', 'Dividends', 'Stock Splits'], axis = 1

usdt = yf.Ticker('USDT-USD')
prices3 = usdt.history(period='5y')
prices3.drop(columns=['Open', 'High', 'Low', 'Dividends', 'Stock Splits'], axis = 1

bnb = yf.Ticker('BNB-USD')
prices4 = bnb.history(period='5y')
prices4.drop(columns=['Open', 'High', 'Low', 'Dividends', 'Stock Splits'], axis = 1
```

In [7]:
```
p1 = prices1.join(prices2, lsuffix = '(BTC)', rsuffix ='(ETH)' )
p2 = prices3.join(prices4, lsuffix = '(USDT)', rsuffix = '(BNB)')
data = p1.join(p2, lsuffix = '_',rsuffix = '_')

data.head()
```

Out[7]:

Date	Close(BTC)	Volume(BTC)	Close(ETH)	Volume(ETH)	Close(USDT)	Volur
2021-01-05 00:00:00+00:00	33992.429688	67547324782	1100.006104	41535932781	1.002202	1019
2021-01-06 00:00:00+00:00	36824.363281	75289433811	1207.112183	44699914188	1.001528	1161
2021-01-07 00:00:00+00:00	39371.042969	84762141031	1225.678101	40468027280	1.000400	1294
2021-01-08 00:00:00+00:00	40797.609375	88107519480	1224.197144	44334826666	1.000045	1315
2021-01-09 00:00:00+00:00	40254.546875	61984162837	1281.077271	33233105361	1.002947	1032

In [8]:
```
data.tail()
```

Out[8]:

Date	Close(BTC)	Volume(BTC)	Close(ETH)	Volume(ETH)	Close(USDT)	Volur
2026-01-01 00:00:00+00:00	88731.984375	18849043990	3000.394287	10268796662	0.998745	505
2026-01-02 00:00:00+00:00	89944.695312	46398906171	3124.422607	25242778003	0.999672	961
2026-01-03 00:00:00+00:00	90603.187500	20774828592	3125.917480	11460707919	0.999635	556
2026-01-04 00:00:00+00:00	91413.492188	26770491368	3140.710449	13890479323	0.999506	669
2026-01-05 00:00:00+00:00	92475.218750	33017376768	3159.529053	16685405184	0.999341	763

In [9]:
```
data.shape
```

Out[9]: (1827, 8)

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

```
<class 'pandas.core.frame.DataFrame'>
DatetimeIndex: 1827 entries, 2021-01-05 00:00:00+00:00 to 2026-01-05 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 [11]:
```
data.isna().sum()
```

Out[11]:
```
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 [12]:
```
data.describe()
```

Out[12]:

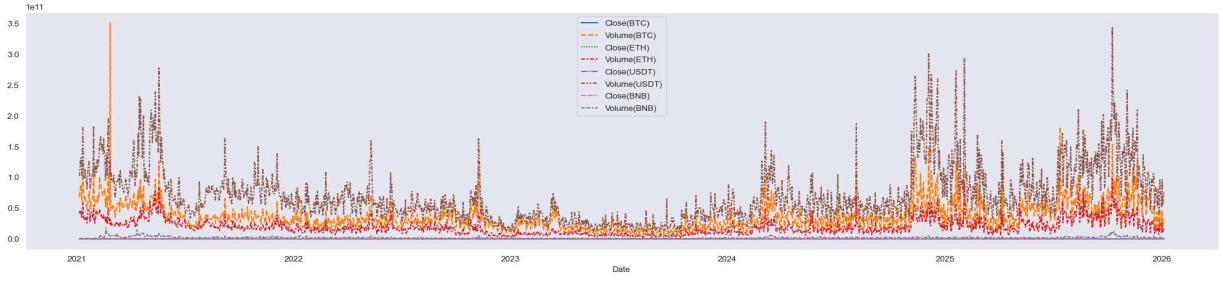
	Close(BTC)	Volume(BTC)	Close(ETH)	Volume(ETH)	Close(USDT)	Volume(USDT
count	1827.000000	1.827000e+03	1827.000000	1.827000e+03	1827.000000	1.827000e+0
mean	54575.500738	3.708813e+10	2539.990056	1.949619e+10	1.000145	7.086426e+1
std	29496.845381	2.299915e+10	908.471406	1.290509e+10	0.000710	4.500425e+1
min	15787.284180	5.331173e+09	993.636780	2.081626e+09	0.995872	9.989859e+0
25%	29412.204102	2.130695e+10	1793.673828	1.025234e+10	0.999893	4.011224e+1
50%	46622.675781	3.175896e+10	2458.723877	1.643428e+10	1.000142	6.038998e+1
75%	69351.074219	4.702226e+10	3234.704468	2.500993e+10	1.000383	8.803484e+1
max	124752.531250	3.509679e+11	4831.348633	9.773662e+10	1.011530	3.443980e+1

In [ ]:
```
data.columns
```

In [ ]:
```
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()
```

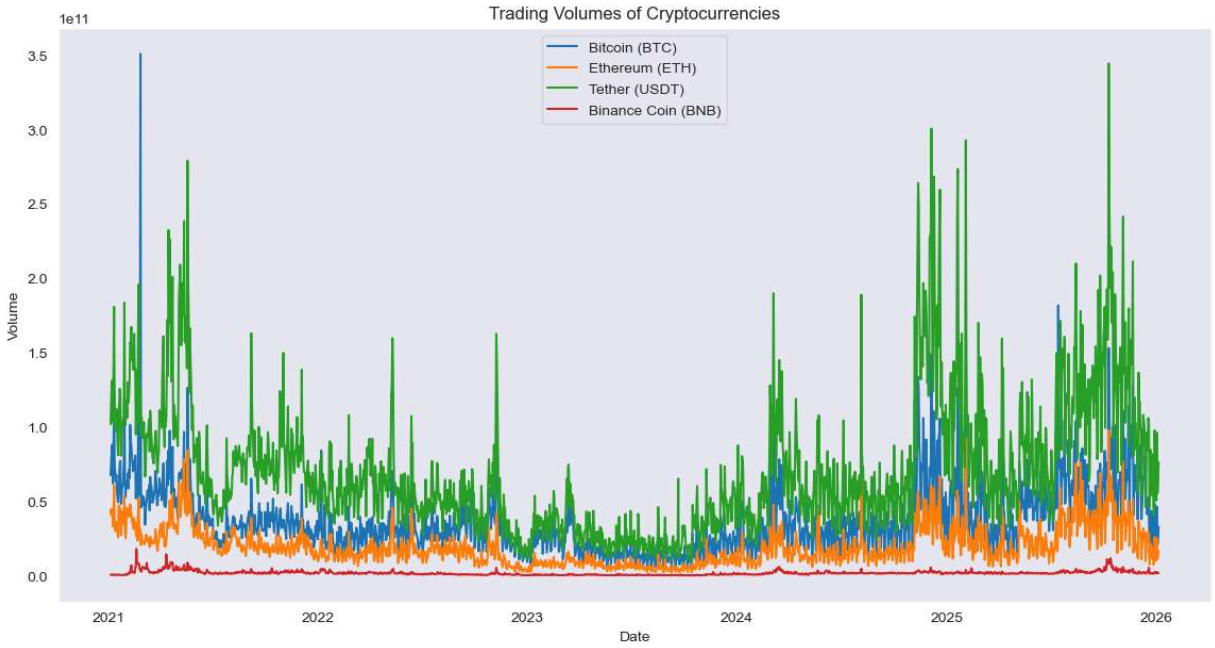
In [13]:
```
plt.figure(figsize = (25, 5))
sns.set_style('dark')
sns.lineplot(data=data)
```

Out[13]: <Axes: xlabel='Date'>



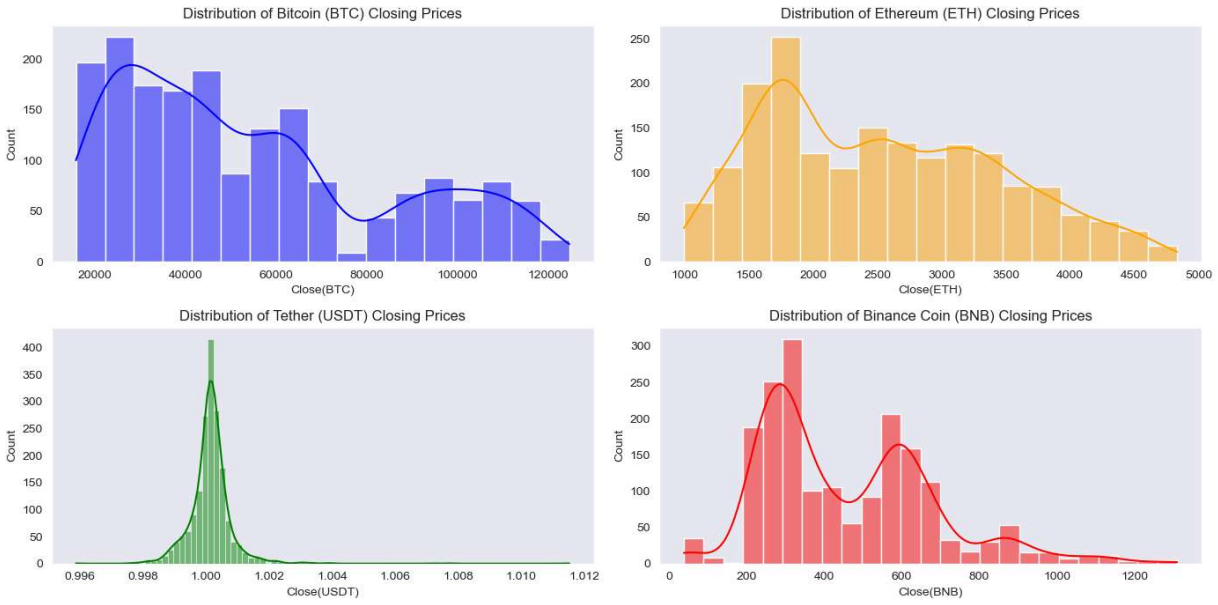
In [14]:
```
plt.figure(figsize=(14, 7))
plt.plot(data.index, data['Volume(BTC)'], label='Bitcoin (BTC)')
plt.plot(data.index, data['Volume(ETH)'], label='Ethereum (ETH)')
plt.plot(data.index, data['Volume(USDT)'], label='Tether (USDT)')
plt.plot(data.index, data['Volume(BNB)'], label='Binance Coin (BNB)')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

```
plt.legend()
plt.show()
```



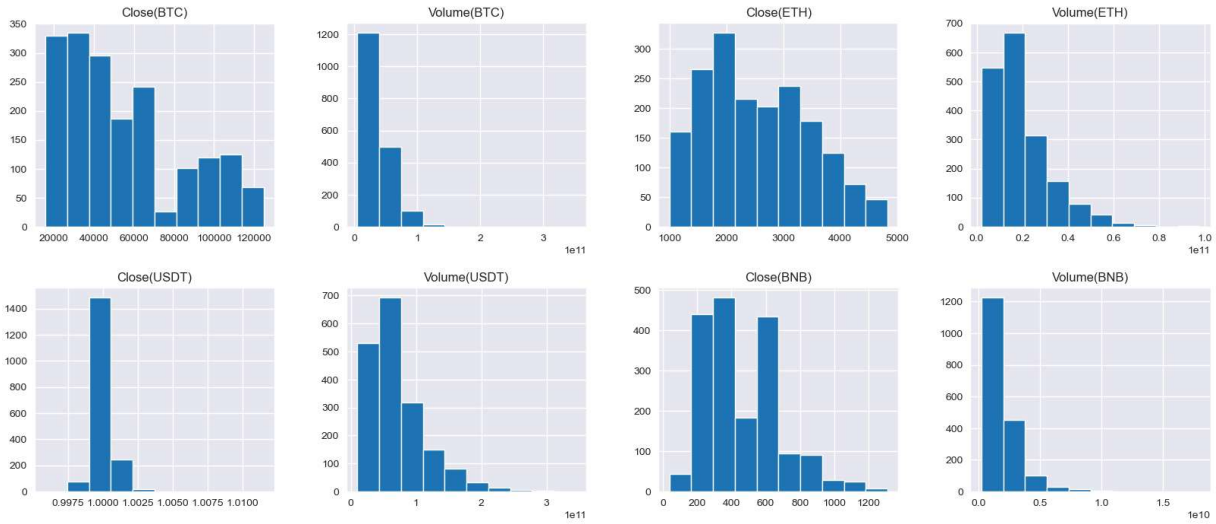
In [15]:
```
corr_matrix = data[['Close(BTC)', 'Close(ETH)', 'Close(USDT)', 'Close(BNB)']].corr(

# 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 [16]:
```
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()
```



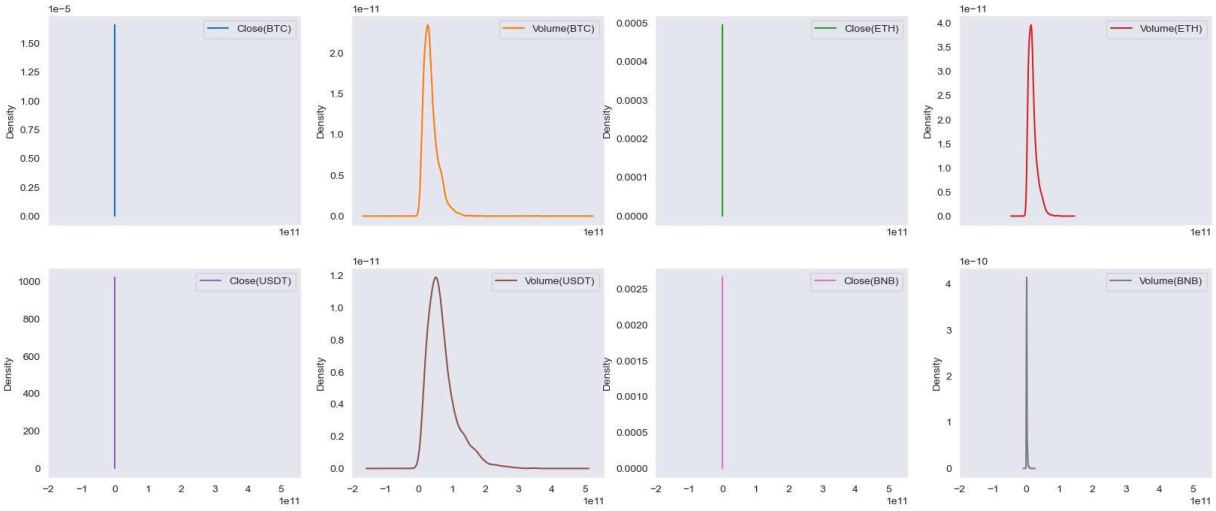
In [17]:
```
data.hist(figsize=(20, 8), layout=(2, 4))
```

Out[17]:
```
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)
```

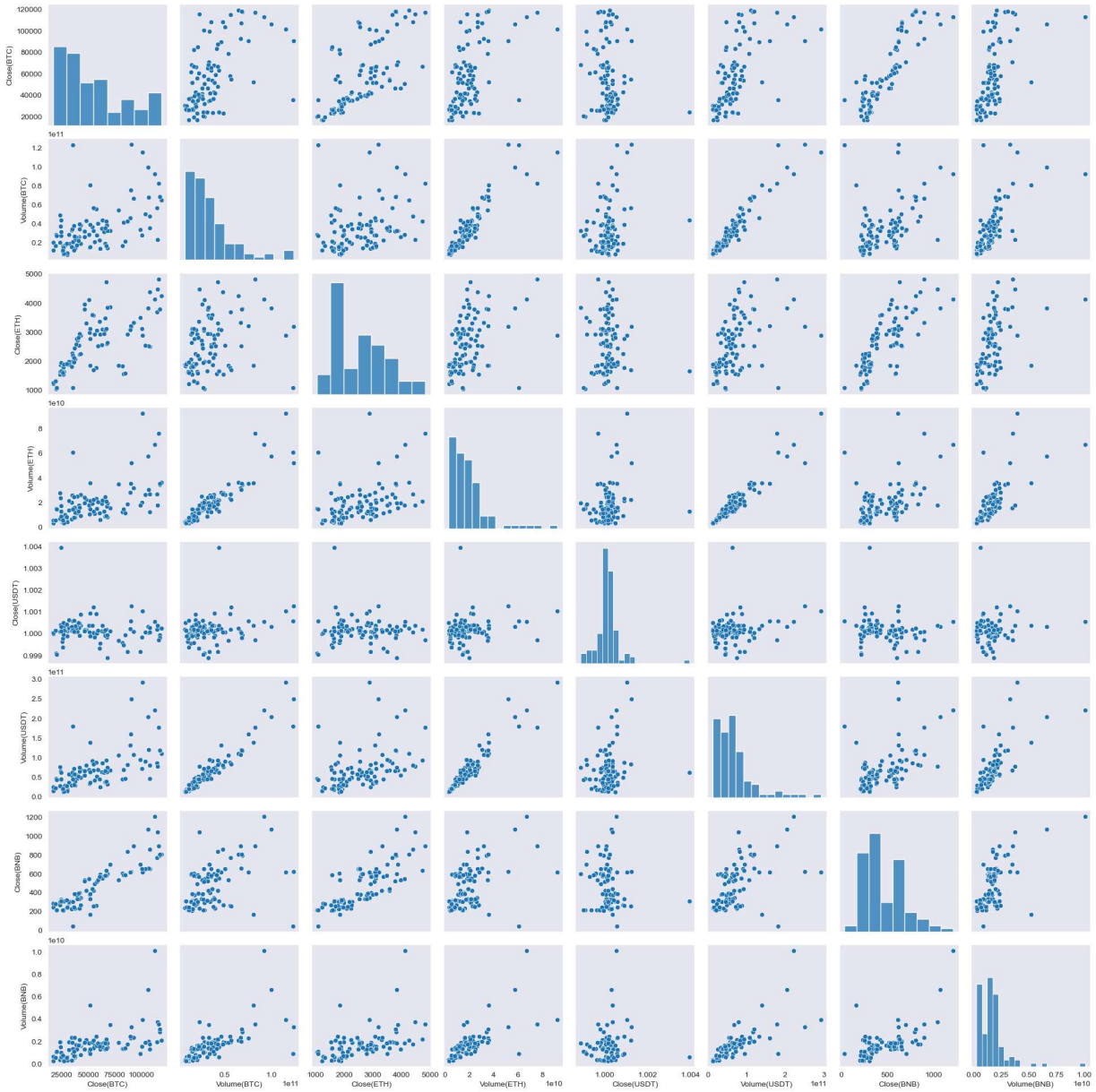


In [18]:
```
data.plot(kind = "kde", subplots = True, layout = (2, 4), figsize = (20, 8))
```

Out[18]:
```
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 [19]:
```
sns.pairplot(data.sample(n=100));
```



In [ ]:

Data Pre-processing

In [21]:
```
X = data.drop (columns = ['Close(BTC)'], axis = 1)
Y = data.loc[:,'Close(BTC)']
```

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

Out[22]:

Date	Volume(BTC)	Close(ETH)	Volume(ETH)	Close(USDT)	Volume(USDT)	Clos
2021-01-05 00:00:00+00:00	67547324782	1100.006104	41535932781	1.002202	101918715244	41
2021-01-06 00:00:00+00:00	75289433811	1207.112183	44699914188	1.001528	116105139289	42
2021-01-07 00:00:00+00:00	84762141031	1225.678101	40468027280	1.000400	129467601516	43
2021-01-08 00:00:00+00:00	88107519480	1224.197144	44334826666	1.000045	131555961745	42
2021-01-09 00:00:00+00:00	61984162837	1281.077271	33233105361	1.002947	103292241007	43

In [23]:
```
X.tail()
```

Out[23]:

Date	Volume(BTC)	Close(ETH)	Volume(ETH)	Close(USDT)	Volume(USDT)	Clos
2026-01-01 00:00:00+00:00	18849043990	3000.394287	10268796662	0.998745	50548666268	863
2026-01-02 00:00:00+00:00	46398906171	3124.422607	25242778003	0.999672	96128566387	880
2026-01-03 00:00:00+00:00	20774828592	3125.917480	11460707919	0.999635	55660104876	878
2026-01-04 00:00:00+00:00	26770491368	3140.710449	13890479323	0.999506	66989138853	894
2026-01-05 00:00:00+00:00	33017376768	3159.529053	16685405184	0.999341	76312829952	898

In [24]:
```
Y.head()
```

Out[24]:
```
Date
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
2021-01-08 00:00:00+00:00    40797.609375
2021-01-09 00:00:00+00:00    40254.546875
Name: Close(BTC), dtype: float64
```

In [25]:
```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2, random_state=
```

In [26]:
```
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 [27]:
```
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\ANITHA\anaconda3\Lib\site-packages\sklearn\feature_selection\_univariate_se
lection.py:107: RuntimeWarning: invalid value encountered in divide
  msw = sswn / float(dfwn)
```

In [28]:
```
import numpy as np

print(np.isnan(X_train).sum())
print(np.isinf(X_train).sum())
```

```
0
0
```

In [29]:
```
X_train = np.nan_to_num(X_train)
X_test = np.nan_to_num(X_test)
```

In [30]:
```
from sklearn.feature_selection import SelectKBest, f_classif
import numpy as np

X_train = np.nan_to_num(X_train)
X_test = np.nan_to_num(X_test)

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

```
C:\Users\ANITHA\anaconda3\Lib\site-packages\sklearn\feature_selection\_univariate_se
lection.py:107: RuntimeWarning: invalid value encountered in divide
  msw = sswn / float(dfwn)
```

In [31]:
```
mask = fs.get_support()
selected_features = X.columns[mask]
print("Selected Features:", selected_features)
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[31], line 2
      1 mask = fs.get_support()
----> 2 selected_features = X.columns[mask]
      3 print("Selected Features:", selected_features)

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:5428, in Index.__getitem__(self, key)
   5419         if len(key) == 0 and len(key) != len(self):
   5420             warnings.warn(
   5421                 "Using a boolean indexer with length 0 on an Index with "
   5422                 "length greater than 0 is deprecated and will raise in a "
   (...)   5425                 stacklevel=find_stack_level(),
   5426             )
-> 5428 result = getitem(key)
   5429 # Because we ruled out integer above, we always get an arraylike here
   5430 if result.ndim > 1:

IndexError: boolean index did not match indexed array along axis 0; size of axis is 7 but size of corresponding boolean axis is 4
```

In [32]:
```
feature_names = X.columns
```

In [33]:
```
from sklearn.feature_selection import SelectKBest, f_classif
import numpy as np

X_train = np.nan_to_num(X_train)
X_test = np.nan_to_num(X_test)

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

```
C:\Users\ANITHA\anaconda3\Lib\site-packages\sklearn\feature_selection\_univariate_selection.py:107: RuntimeWarning: invalid value encountered in divide
  msw = sswn / float(dfwn)
```

In [34]:
```
mask = fs.get_support()
selected_features = feature_names[mask]

print("Selected Features:", selected_features)
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[34], line 2
      1 mask = fs.get_support()
----> 2 selected_features = feature_names[mask]
      4 print("Selected Features:", selected_features)

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:5428, in Index.__geti
tem__(self, key)
   5419         if len(key) == 0 and len(key) != len(self):
   5420             warnings.warn(
   5421                 "Using a boolean indexer with length 0 on an Index with "
   5422                 "length greater than 0 is deprecated and will raise in a "
   (...)   5425                 stacklevel=find_stack_level(),
   5426             )
-> 5428 result = getitem(key)
   5429 # Because we ruled out integer above, we always get an arraylike here
   5430 if result.ndim > 1:

IndexError: boolean index did not match indexed array along axis 0; size of axis is
7 but size of corresponding boolean axis is 4
```

In [35]:
```
mask = fs.get_support()
selected_features = X.columns[mask]
print("Selected Features:", selected_features)
```

```
---------------------------------------------------------------------------
IndexError                                Traceback (most recent call last)
Cell In[35], line 2
      1 mask = fs.get_support()
----> 2 selected_features = X.columns[mask]
      3 print("Selected Features:", selected_features)

File ~\anaconda3\Lib\site-packages\pandas\core\indexes\base.py:5428, in Index.__geti
tem__(self, key)
   5419         if len(key) == 0 and len(key) != len(self):
   5420             warnings.warn(
   5421                 "Using a boolean indexer with length 0 on an Index with "
   5422                 "length greater than 0 is deprecated and will raise in a "
   (...)   5425                 stacklevel=find_stack_level(),
   5426             )
-> 5428 result = getitem(key)
   5429 # Because we ruled out integer above, we always get an arraylike here
   5430 if result.ndim > 1:

IndexError: boolean index did not match indexed array along axis 0; size of axis is
7 but size of corresponding boolean axis is 4
```

In [36]:
```
X_train
```

Out[36]:
```
array([[1.00028396e+00, 6.77472974e+10, 4.28098328e+02, 1.74093742e+09],
       [1.00066805e+00, 7.50076536e+10, 4.54936310e+02, 2.45157869e+09],
       [1.00020397e+00, 6.68346391e+10, 2.93658722e+02, 9.98624528e+08],
       ...,
       [1.00041604e+00, 1.36735818e+11, 7.08126099e+02, 2.29864945e+09],
       [9.99777019e-01, 7.03906069e+10, 2.64886047e+02, 1.63232454e+09],
       [9.98982012e-01, 3.38348894e+10, 2.64235138e+02, 7.20361905e+08]],
      shape=(1461, 4))
```

In [37]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

In [38]:
```
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
```

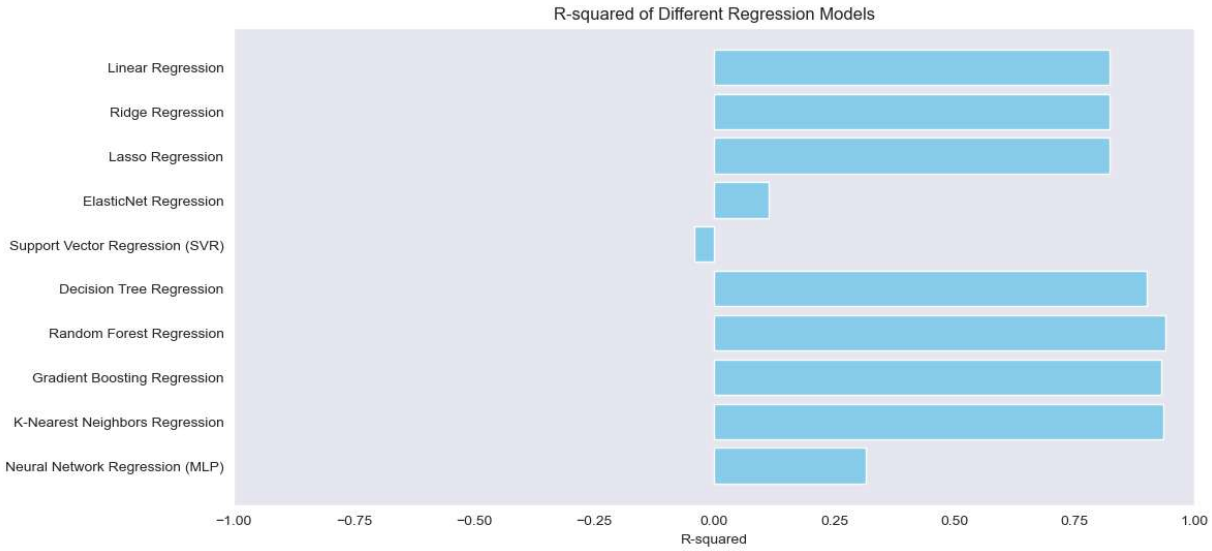
In [39]:
```
#Define Models and Perform Training and Evaluation
models = {
    'Linear Regression': LinearRegression(),
    'Ridge Regression': Ridge(alpha=1.0),
    'Lasso Regression': Lasso(alpha=1.0),
    'ElasticNet Regression': ElasticNet(alpha=1.0, l1_ratio=0.5),
    'Support Vector Regression (SVR)': SVR(kernel='rbf'),
    'Decision Tree Regression': DecisionTreeRegressor(),
    'Random Forest Regression': RandomForestRegressor(n_estimators=100),
    'Gradient Boosting Regression': GradientBoostingRegressor(n_estimators=100, lea
    'K-Nearest Neighbors Regression': KNeighborsRegressor(n_neighbors=5),
    'Neural Network Regression (MLP)': MLPRegressor(hidden_layer_sizes=(100, 50), a
}

# 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): 152064435.4541398
R-squared: 0.8224818829482098

----- Ridge Regression -----
Mean Squared Error (MSE): 152254399.0362988
R-squared: 0.8222601218420539

----- Lasso Regression -----
Mean Squared Error (MSE): 152039671.37944096
R-squared: 0.8225107922187959

----- ElasticNet Regression -----
Mean Squared Error (MSE): 759323191.1872752
R-squared: 0.11357561858062992

----- Support Vector Regression (SVR) -----
Mean Squared Error (MSE): 892004027.4757841
R-squared: -0.041314327621789104

----- Decision Tree Regression -----
Mean Squared Error (MSE): 86061479.58239356
R-squared: 0.8995329068198524

----- Random Forest Regression -----
Mean Squared Error (MSE): 51744342.72691239
R-squared: 0.9395943024972845

----- Gradient Boosting Regression -----
Mean Squared Error (MSE): 58701513.782728724
R-squared: 0.9314725881585744

----- K-Nearest Neighbors Regression -----
Mean Squared Error (MSE): 55166798.4151865
R-squared: 0.9355989705995849

----- Neural Network Regression (MLP) -----
Mean Squared Error (MSE): 587204865.4273624
R-squared: 0.314504395962117
```

	Model	MSE	R-squared
0	Linear Regression	1.520644e+08	0.822482
1	Ridge Regression	1.522544e+08	0.822260
2	Lasso Regression	1.520397e+08	0.822511
3	ElasticNet Regression	7.593232e+08	0.113576
4	Support Vector Regression (SVR)	8.920040e+08	-0.041314
5	Decision Tree Regression	8.606148e+07	0.899533
6	Random Forest Regression	5.174434e+07	0.939594
7	Gradient Boosting Regression	5.870151e+07	0.931473
8	K-Nearest Neighbors Regression	5.516680e+07	0.935599
9	Neural Network Regression (MLP)	5.872049e+08	0.314504

```
C:\Users\ANITHA\anaconda3\Lib\site-packages\sklearn\neural_network\_multilayer_perce
ptron.py:785: ConvergenceWarning: Stochastic Optimizer: Maximum iterations (200) rea
ched and the optimization hasn't converged yet.
  warnings.warn(
```



In [40]:
```python
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): 52249372.2130668
Loaded Random Forest Regression - R-squared: 0.9390047374016063
```

In [ ]: