Ethereum On-chain Data Analysis of Transaction Value with Gas Used with Etherscan API

1. Introduction

In this notebook, I will perform an on-chain analysis of Ethereum transactions using the Etherscan API. By querying relevant transaction and contract data, we will extract key metrics that influence tokenomics, particularly the supply and demand aspects of Ethereum-based tokens.

2. Prerequisites

a) Python IDE; Python 3.x
b) Modules/ Libraries for Python:

```
pip install requests pandas matplotlib
```

c) Etherscan API Key: You need an API key from [Etherscan.io] (https://etherscan.io/) to access the data.

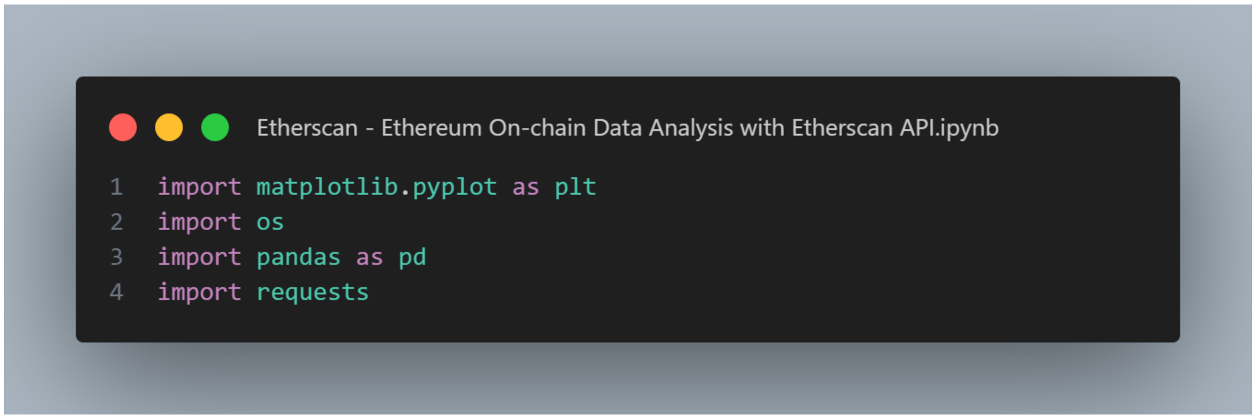
3. Data Collection using Etherscan API

Data Query Parameters in the Project

- Transaction Value: Indicates the liquidity flow and demand for a token.
- Gas Price: Reflects network demand and transaction costs, influencing user behavior and token interaction.
- Gas Used: Helps assess the complexity of transactions and demand for contract interactions.
- Timestamp: Provides a time series for analysis of transaction trends.

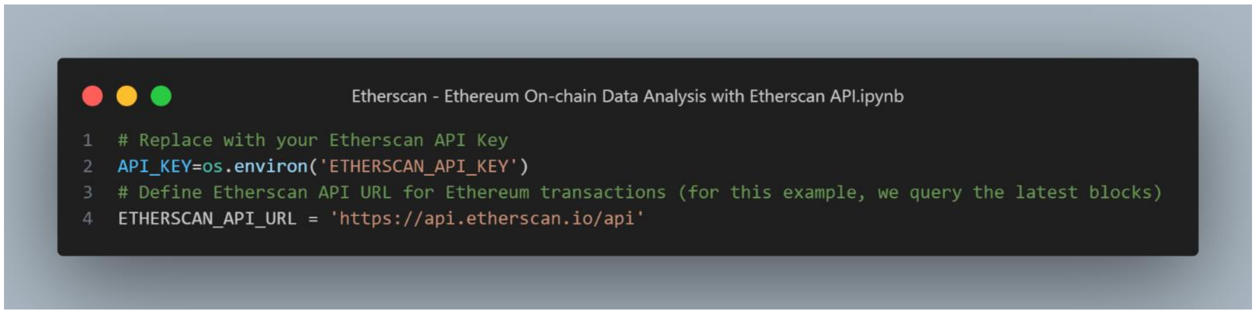
These metrics are valuable for any blockchain-based project aiming to understand the economic principles of supply and demand and their effect on tokenomics. By analyzing these on-chain metrics, developers and analysts can make informed decisions about network health, token distribution, and user engagement strategies.

I will use the Etherscan API to query Ethereum transaction data. Below is an example of how to get data for Ethereum transactions.


Etherscan - Ethereum On-chain Data Analysis with Etherscan API.ipynb

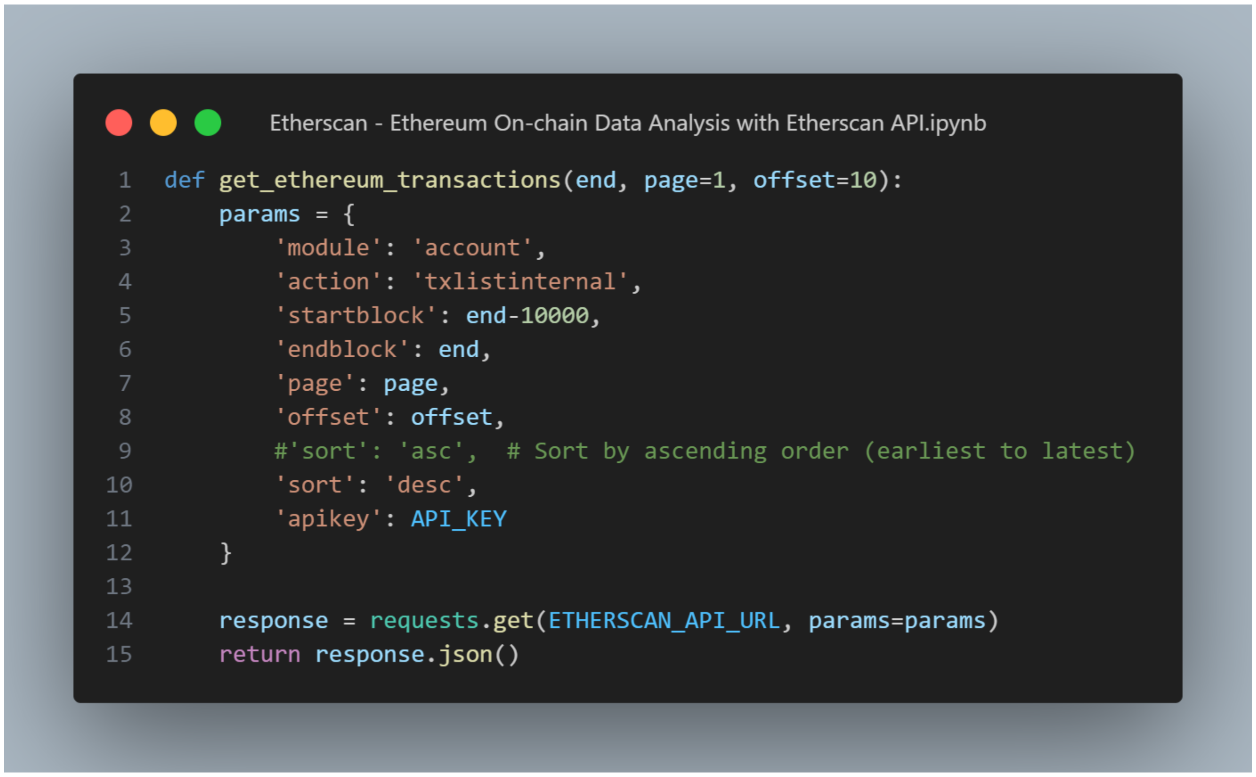
```python
import matplotlib.pyplot as plt
import os
import pandas as pd
import requests
```

Etherscan API Key


Etherscan - Ethereum On-chain Data Analysis with Etherscan API.ipynb

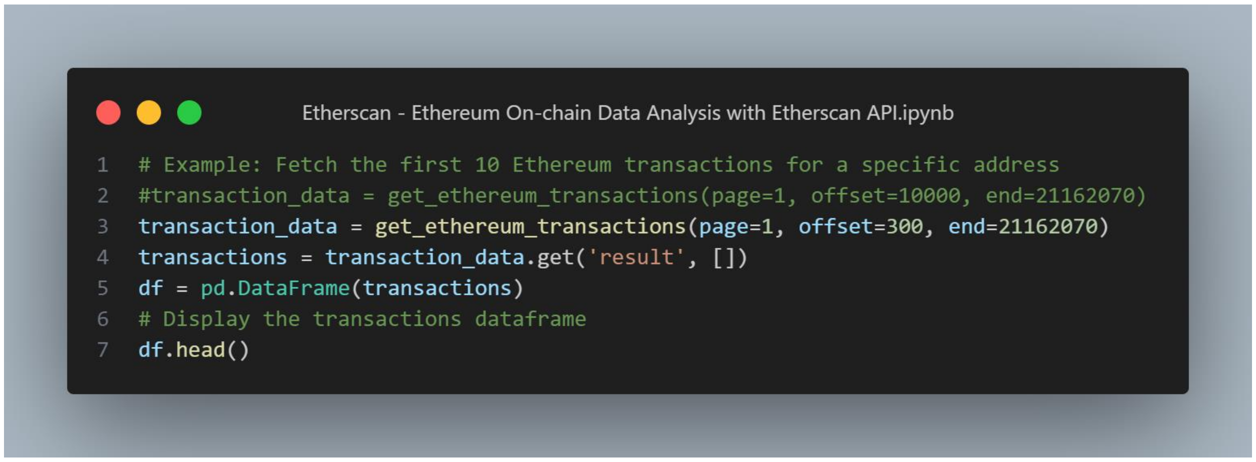
```python
# Replace with your Etherscan API Key
API_KEY=os.environ('ETHERSCAN_API_KEY')
# Define Etherscan API URL for Ethereum transactions (for this example, we query the latest blocks)
ETHERSCAN_API_URL = 'https://api.etherscan.io/api'
```

def get_ethereum_transactions (page=1, offset=10):


Etherscan - Ethereum On-chain Data Analysis with Etherscan API.ipynb

```python
def get_ethereum_transactions(end, page=1, offset=10):
    params = {
        'module': 'account',
        'action': 'txlistinternal',
        'startblock': end-10000,
        'endblock': end,
        'page': page,
        'offset': offset,
        #'sort': 'asc',  # Sort by ascending order (earliest to latest)
        'sort': 'desc',
        'apikey': API_KEY
    }

    response = requests.get(ETHERSCAN_API_URL, params=params)
    return response.json()
```

# Example: Fetch the first 10 Ethereum transactions for a specific address


Etherscan - Ethereum On-chain Data Analysis with Etherscan API.ipynb

```python
# Example: Fetch the first 10 Ethereum transactions for a specific address
#transaction_data = get_ethereum_transactions(page=1, offset=10000, end=21162070)
transaction_data = get_ethereum_transactions(page=1, offset=300, end=21162070)
transactions = transaction_data.get('result', [])
df = pd.DataFrame(transactions)
# Display the transactions dataframe
df.head()
```

This code will fetch the latest 10 transactions for a given Ethereum address.

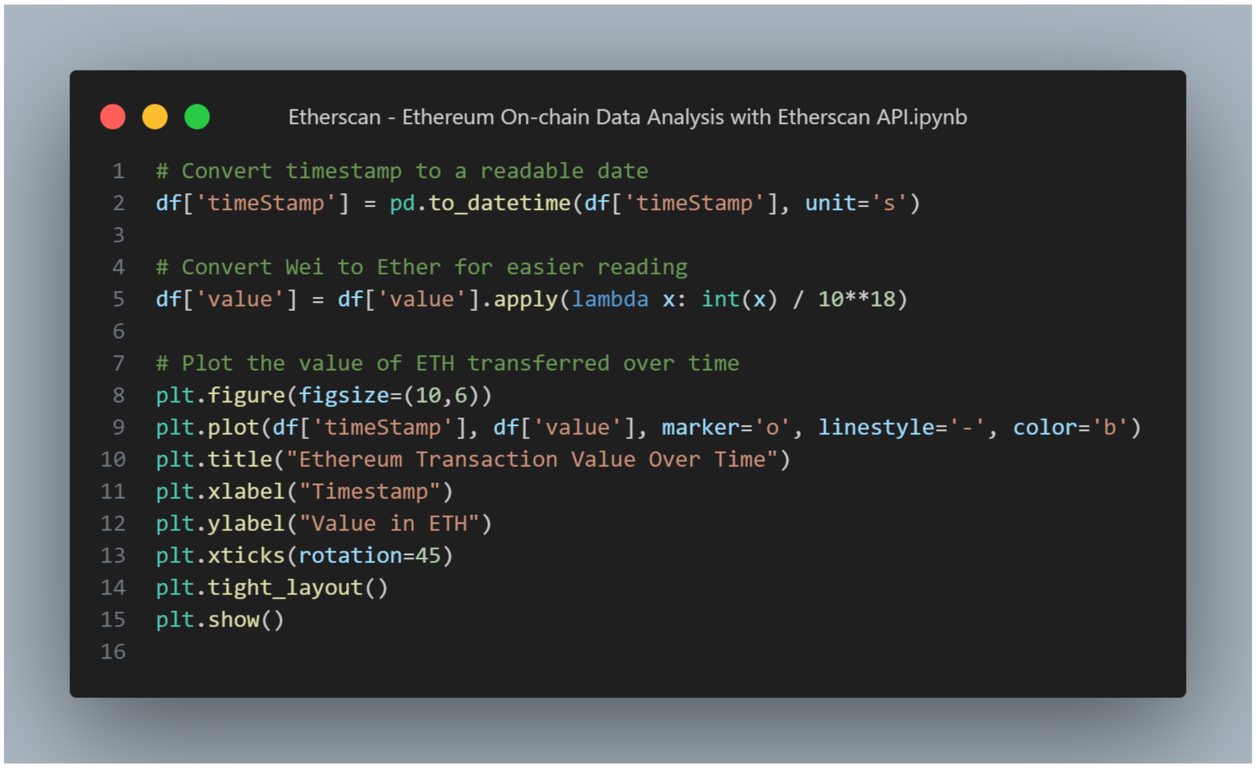
4. Data Exploration and Metrics

Let's explore some key metrics you can extract from Ethereum transactions and analyze how they might affect the tokenomics of a Ethereum Project.

4.1 Key Metrics from Transaction Data

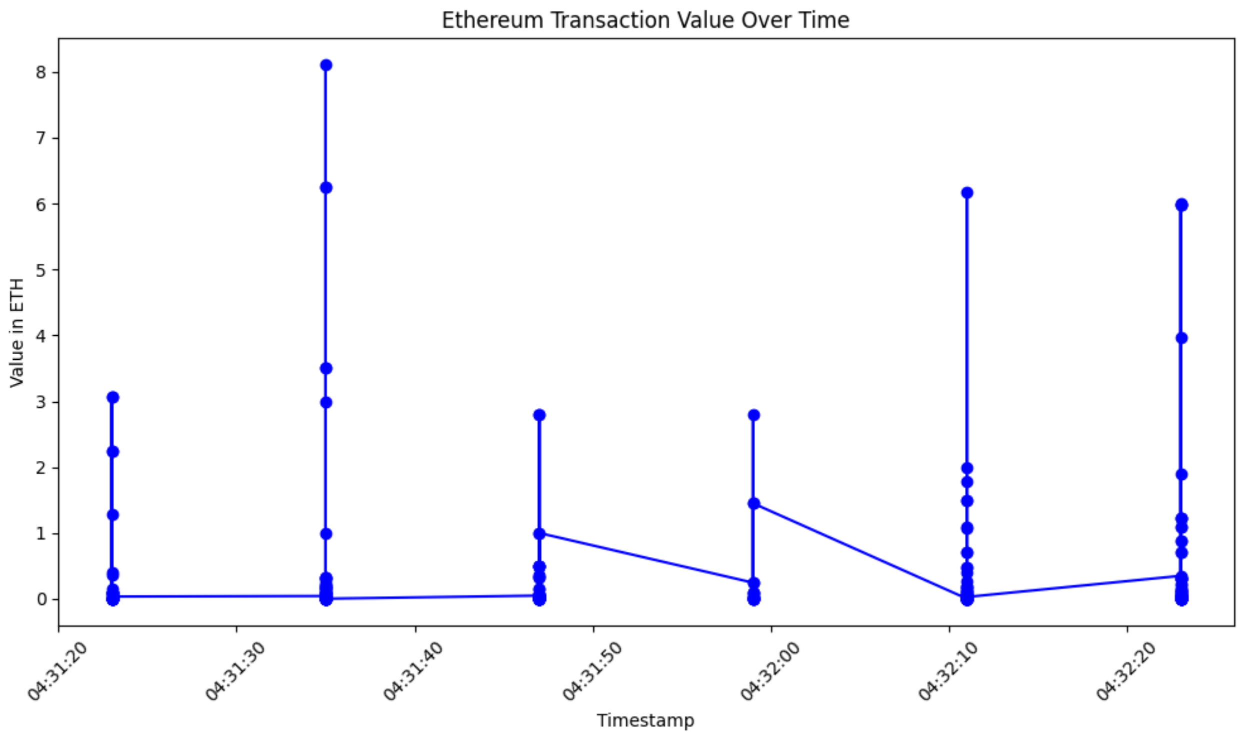
a) Transaction Hash (`hash`): The unique identifier for a transaction. It's useful for tracing individual transactions on the blockchain.
b) Block Number (`blockNumber`): The block in which the transaction was included. Useful for analyzing transaction activity in a specific block.
c) From Address (`from`): The sender's address in a transaction. Can help identify whale wallets or significant market participants.
d) To Address (`to`): The recipient's address in a transaction. Helps to track token transfers, including contract interactions.
e) Value (`value`): The value of ETH transferred in the transaction, expressed in Wei (1 ETH = 10^18 Wei). This can be used to assess the liquidity in the network.
f) Gas Price (`gasPrice`): The price per unit of gas paid for a transaction. This is an important metric for understanding transaction costs and network congestion.
g) Gas Used (`gasUsed`): The total gas used in a transaction. It can indicate the complexity of the transaction, such as whether it was a simple transfer or a contract interaction.
h) Timestamp (`timestamp`): The time at which the transaction was mined. Useful for analyzing transaction volume over time.
i) Input Data (`input`): For contract-based transactions, this field contains the encoded data that specifies what function is being called. It is essential for analyzing interaction with smart contracts.

4.2 Example of Transaction Data Analysis


Etherscan - Ethereum On-chain Data Analysis with Etherscan API.ipynb

```python
# Convert timestamp to a readable date
df['timeStamp'] = pd.to_datetime(df['timeStamp'], unit='s')

# Convert Wei to Ether for easier reading
df['value'] = df['value'].apply(lambda x: int(x) / 10**18)

# Plot the value of ETH transferred over time
plt.figure(figsize=(10,6))
plt.plot(df['timeStamp'], df['value'], marker='o', linestyle='-', color='b')
plt.title("Ethereum Transaction Value Over Time")
plt.xlabel("Timestamp")
plt.ylabel("Value in ETH")
plt.xticks(rotation=45)
plt.tight_layout()
plt.show()

```

This code will create a plot of the ETH value transferred in each of the transactions, which allows you to visually analyze transaction activity over time.



5. Metrics Impact on Tokenomics: Supply & Demand

On-chain metrics can directly influence tokenomics by affecting the supply and demand of tokens.

5.1 Supply Impact

- New Token Supply: If a contract or token is programmed to mint new tokens upon certain conditions (e.g., staking, rewards), transaction data will show the minting activity. A high volume of transactions with significant value could indicate increased token issuance.

- Burn Mechanism: Some tokens have a "burn" feature that removes tokens from circulation during transactions. The total burned can be tracked using transaction data, which directly impacts the token's supply and inflation rate.

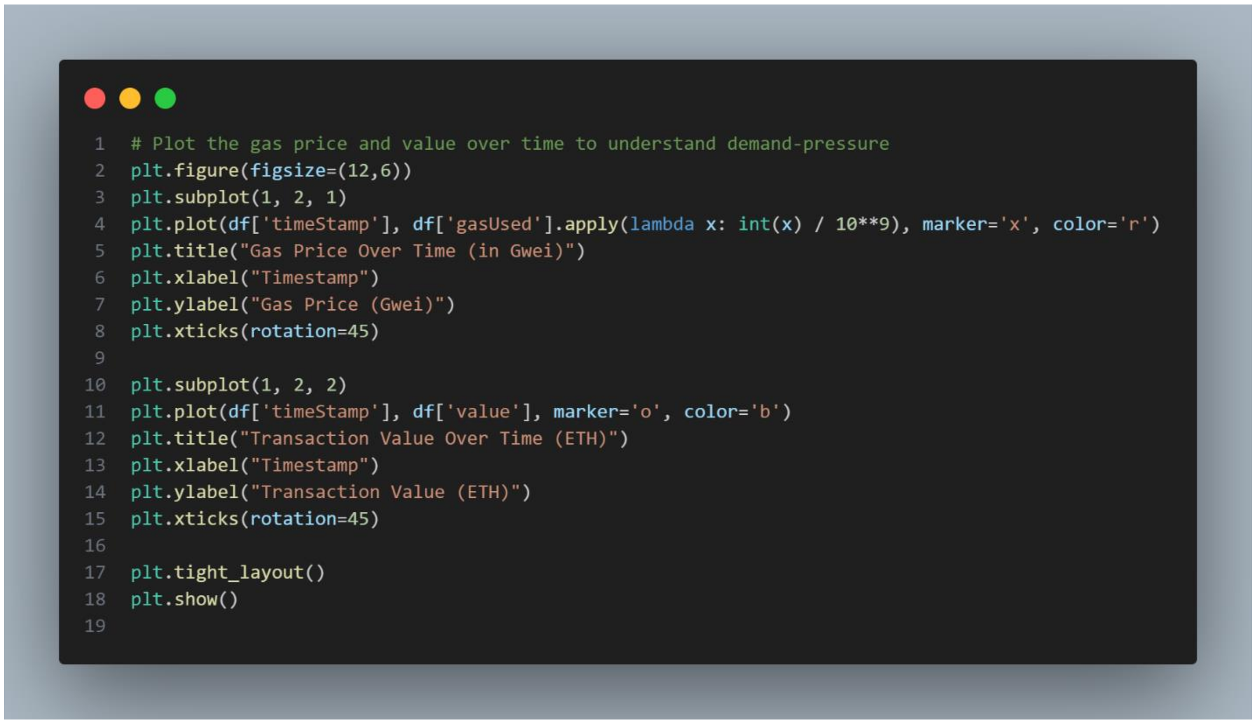
5.2 Demand Impact

- Transaction Volume and Activity: High transaction volumes can signal strong demand for a token, as it indicates active user interaction. This can drive up token price if demand outpaces supply.

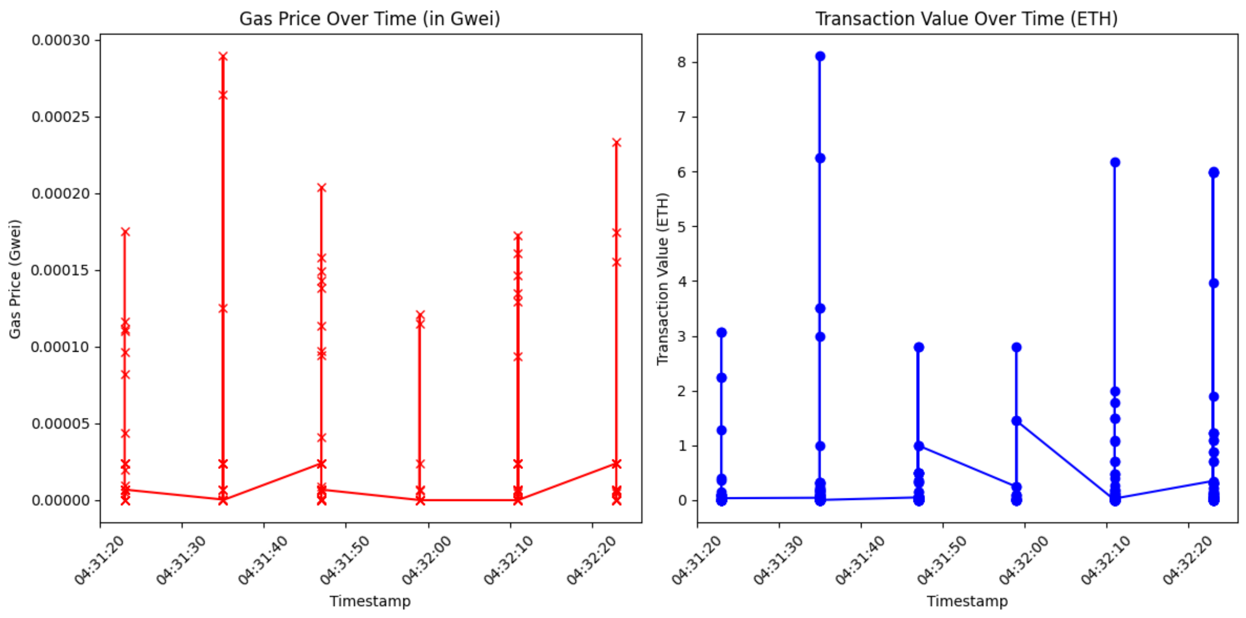
- Gas Price and Network Congestion: When the gas price increases, it may signal high demand for block space, often due to more complex smart contract interactions (e.g., DeFi protocols, NFTs). This can indicate market participants are competing for transaction space, which might correlate with increased demand for the token.

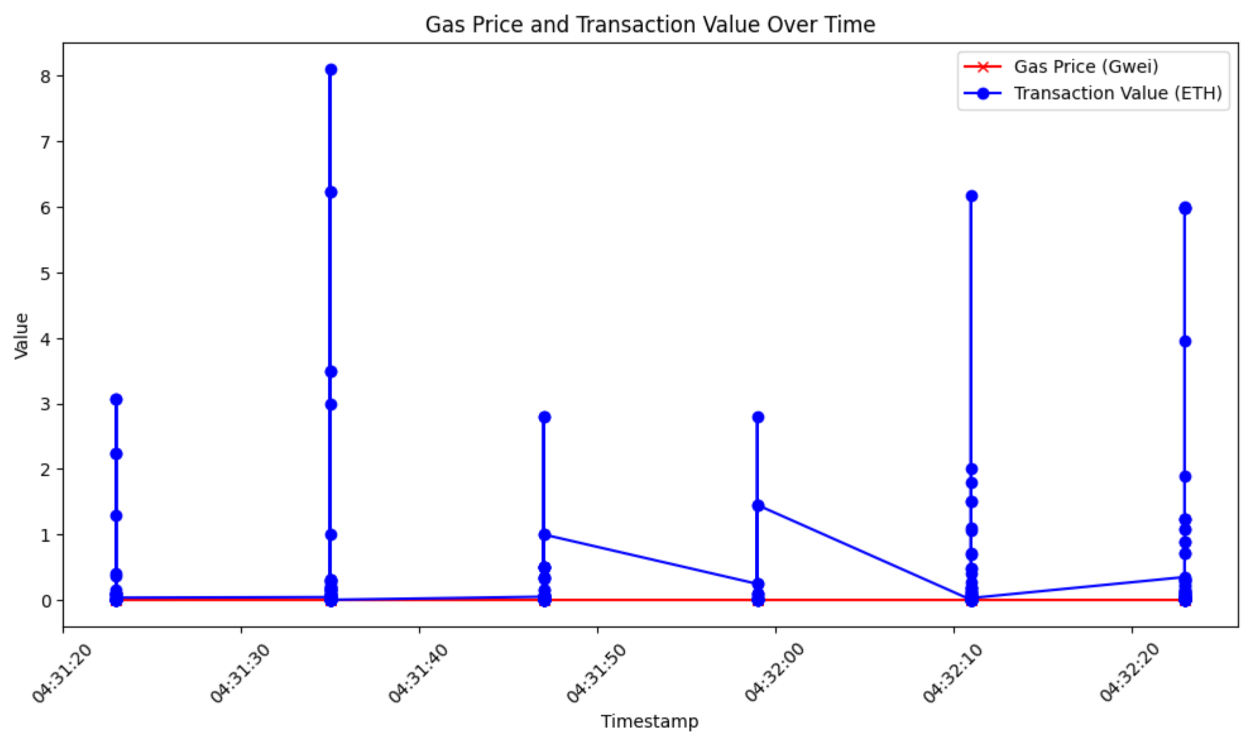
5.3 Gas Prices and Transaction Value Analysis

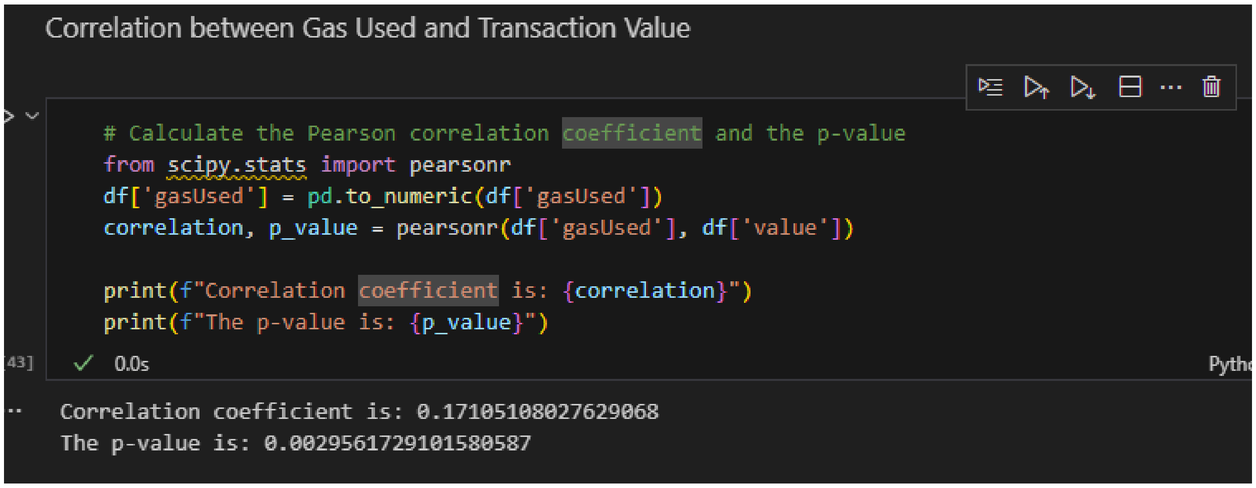
Let's analyze gas prices and transaction value to understand demand:



```python
# Plot the gas price and value over time to understand demand-pressure
plt.figure(figsize=(12,6))
plt.subplot(1, 2, 1)
plt.plot(df['timeStamp'], df['gasUsed'].apply(lambda x: int(x) / 10**9), marker='x', color='r')
plt.title("Gas Price Over Time (in Gwei)")
plt.xlabel("Timestamp")
plt.ylabel("Gas Price (Gwei)")
plt.xticks(rotation=45)

plt.subplot(1, 2, 2)
plt.plot(df['timeStamp'], df['value'], marker='o', color='b')
plt.title("Transaction Value Over Time (ETH)")
plt.xlabel("Timestamp")
plt.ylabel("Transaction Value (ETH)")
plt.xticks(rotation=45)

plt.tight_layout()
plt.show()

```






Correlation between Gas Used and Transaction Value

```
# Calculate the Pearson correlation coefficient and the p-value
from scipy.stats import pearsonr
df['gasUsed'] = pd.to_numeric(df['gasUsed'])
correlation, p_value = pearsonr(df['gasUsed'], df['value'])

print(f"Correlation coefficient is: {correlation}")
print(f"The p-value is: {p_value}")
```

```
Correlation coefficient is: 0.17105108027629068
The p-value is: 0.0029561729101580587
```

This plot shows both gas price and transaction value over time. The correlation coefficient between gas used and transaction value of 0.17105 or 17.1% shows a weak positive correlation between gas prices and transaction values. This suggests that, generally, as the gas used in transactions increases, the transaction value also tends to increase, but this trend is not very pronounced. The p-value of 0.0029 is quite low, which suggests that the correlation is statistically significant, meaning the relationship observed is likely to be real and not due to random variation.

Conclusion

In this analysis, I've demonstrated how to query Ethereum transactions using the Etherscan API and extract meaningful metrics such as transaction value, gas price, and transaction volume. These metrics are crucial for understanding the supply and demand dynamics in the Ethereum ecosystem and for evaluating the health and performance of Ethereum-based tokens.