In [ ]:
```
import pandas as pd
df_btc=pd.read_csv("BTC_1min.csv")
df_ada=pd.read_csv("ADA_1min.csv")
df_eth=pd.read_csv("ETH_1min.csv")
import numpy as np
```

In [ ]:
```
for i in range (15):
  df_btc[f'bids_distance_{i}'] = df_btc['midpoint']- np.exp(df_btc[f'bids_dist
  df_ada[f'bids_distance_{i}'] = df_ada['midpoint']- np.exp(df_ada[f'bids_dist
  df_eth[f'bids_distance_{i}'] = df_eth['midpoint']- np.exp(df_eth[f'bids_dist
  df_btc[f'asks_distance_{i}'] = df_btc['midpoint']- np.exp(df_btc[f'asks_dist
  df_ada[f'asks_distance_{i}'] = df_ada['midpoint']- np.exp(df_ada[f'asks_dist
  df_eth[f'asks_distance_{i}'] = df_eth['midpoint']- np.exp(df_eth[f'asks_dist
df_ada
```

Out[ ]:

	Unnamed: 0	system_time	midpoint	spread	buys	
0	0	2021-04-07 11:33:59.055697+00:00	1.16205	0.0001	56936.467913	2582
1	1	2021-04-07 11:34:59.055697+00:00	1.16800	0.0022	56491.336799	786
2	2	2021-04-07 11:35:59.055697+00:00	1.17530	0.0012	52859.493359	484
3	3	2021-04-07 11:36:59.055697+00:00	1.16585	0.0017	50772.386336	326
4	4	2021-04-07 11:37:59.055697+00:00	1.17255	0.0009	113579.364184	825
...	...	...	...	...	...	
17104	17104	2021-04-19 09:45:00.442103+00:00	1.27325	0.0001	13671.251598	253
17105	17105	2021-04-19 09:46:00.442103+00:00	1.27200	0.0008	9916.946518	336
17106	17106	2021-04-19 09:47:00.442103+00:00	1.27255	0.0007	32589.054204	434
17107	17107	2021-04-19 09:48:00.442103+00:00	1.27305	0.0001	3437.251449	79
17108	17108	2021-04-19 09:49:00.442103+00:00	1.27105	0.0007	10510.439494	68

17109 rows × 156 columns

In [ ]:
```
df_btc
```

Out[ ]:

	Unnamed: 0	system_time	midpoint	spread	buys	
0	0	2021-04-07 11:33:41.122161+00:00	55896.285	0.01	4.448599e+06	3.89
1	1	2021-04-07 11:34:41.122161+00:00	55948.685	1.43	1.243244e+06	3.60
2	2	2021-04-07 11:35:41.122161+00:00	56013.785	0.01	3.069094e+06	1.57
3	3	2021-04-07 11:36:41.122161+00:00	55903.575	7.17	1.220819e+06	1.32
4	4	2021-04-07 11:37:41.122161+00:00	55899.995	0.01	2.011287e+06	3.08
...	...	...	...	...	...	
17108	17108	2021-04-19 09:50:00.386544+00:00	56878.090	6.54	7.205687e+04	1.54
17109	17109	2021-04-19 09:51:00.386544+00:00	56944.085	2.57	9.383907e+04	3.08
17110	17110	2021-04-19 09:52:00.386544+00:00	56873.165	0.01	3.366408e+05	1.19
17111	17111	2021-04-19 09:53:00.386544+00:00	56820.445	0.01	3.118859e+04	8.35
17112	17112	2021-04-19 09:54:00.386544+00:00	56862.695	2.03	1.471420e+05	4.96

17113 rows × 156 columns

In [ ]:
```
df_eth
```

Out[ ]:

	Unnamed: 0	system_time	midpoint	spread	buys	
0	0	2021-04-07 11:33:49.861733+00:00	1965.845	0.01	875154.482918	1.68
1	1	2021-04-07 11:34:49.861733+00:00	1969.645	0.65	514168.079888	8.58
2	2	2021-04-07 11:35:49.861733+00:00	1975.595	0.29	729915.129243	1.44
3	3	2021-04-07 11:36:49.861733+00:00	1969.335	0.19	611826.976792	5.98
4	4	2021-04-07 11:37:49.861733+00:00	1970.965	0.49	429786.641273	4.14
...	...	...	...	...	...	
17105	17105	2021-04-19 09:49:00.345392+00:00	2238.505	0.01	47067.406991	7.25
17106	17106	2021-04-19 09:50:00.345392+00:00	2238.005	0.01	19950.489155	3.17
17107	17107	2021-04-19 09:51:00.345392+00:00	2240.405	0.01	35268.328086	2.64
17108	17108	2021-04-19 09:52:00.345392+00:00	2236.795	0.01	82036.022512	1.62
17109	17109	2021-04-19 09:53:00.345392+00:00	2235.225	0.01	44542.528617	1.41

17110 rows × 156 columns

In [ ]:
```
ts_btc=23000
for j in range (len (df_btc)):
  for i in range (1,15):
    k=i-1
    a=df_btc[f'bids_distance_{i}'][j] - df_btc[f'bids_distance_{k}'][j]
    if (a < ts_btc) and (a>0):
      ts_btc=a
ts_btc
```

Out[ ]:
```
np.float64(0.009966180543415248)
```

In [ ]:
```
ts_ada=23
for j in range (len (df_ada)):
  for i in range (1,15):
    k=i-1
    a=df_ada[f'bids_distance_{i}'][j] - df_ada[f'bids_distance_{k}'][j]
    if (a < ts_ada) and (a>0):
      ts_ada=a
```

```
ts_ada
```

Out[ ]:
```
np.float64(9.700497390707774e-05)
```

In [ ]:
```
ts_eth=23
for j in range (len (df_eth)):
  for i in range (1,15):
    k=i-1
    a=df_eth[f'bids_distance_{i}'][j] - df_eth[f'bids_distance_{k}'][j]
    if (a < ts_eth) and (a>0):
      ts_eth=a

ts_eth
```

Out[ ]:
```
np.float64(0.00988892104533079)
```

In [ ]:
```
df_btc
```

Out[ ]:

	Unnamed: 0	system_time	midpoint	spread	buys	
0	0	2021-04-07 11:33:41.122161+00:00	55896.285	0.01	4.448599e+06	3.89
1	1	2021-04-07 11:34:41.122161+00:00	55948.685	1.43	1.243244e+06	3.60
2	2	2021-04-07 11:35:41.122161+00:00	56013.785	0.01	3.069094e+06	1.57
3	3	2021-04-07 11:36:41.122161+00:00	55903.575	7.17	1.220819e+06	1.32
4	4	2021-04-07 11:37:41.122161+00:00	55899.995	0.01	2.011287e+06	3.08
...	...	...	...	...	...	
17108	17108	2021-04-19 09:50:00.386544+00:00	56878.090	6.54	7.205687e+04	1.54
17109	17109	2021-04-19 09:51:00.386544+00:00	56944.085	2.57	9.383907e+04	3.08
17110	17110	2021-04-19 09:52:00.386544+00:00	56873.165	0.01	3.366408e+05	1.19
17111	17111	2021-04-19 09:53:00.386544+00:00	56820.445	0.01	3.118859e+04	8.35
17112	17112	2021-04-19 09:54:00.386544+00:00	56862.695	2.03	1.471420e+05	4.96

17113 rows × 156 columns

In [ ]:
```
df_ada
```

Out[ ]:

	Unnamed: 0	system_time	midpoint	spread	buys	
0	0	2021-04-07 11:33:59.055697+00:00	1.16205	0.0001	56936.467913	2582
1	1	2021-04-07 11:34:59.055697+00:00	1.16800	0.0022	56491.336799	786
2	2	2021-04-07 11:35:59.055697+00:00	1.17530	0.0012	52859.493359	484
3	3	2021-04-07 11:36:59.055697+00:00	1.16585	0.0017	50772.386336	326
4	4	2021-04-07 11:37:59.055697+00:00	1.17255	0.0009	113579.364184	825
...	...	...	...	...	...	
17104	17104	2021-04-19 09:45:00.442103+00:00	1.27325	0.0001	13671.251598	253
17105	17105	2021-04-19 09:46:00.442103+00:00	1.27200	0.0008	9916.946518	336
17106	17106	2021-04-19 09:47:00.442103+00:00	1.27255	0.0007	32589.054204	434
17107	17107	2021-04-19 09:48:00.442103+00:00	1.27305	0.0001	3437.251449	79
17108	17108	2021-04-19 09:49:00.442103+00:00	1.27105	0.0007	10510.439494	68

17109 rows × 156 columns

In [ ]:
```
df_eth
```

Out[ ]:

	Unnamed: 0	system_time	midpoint	spread	buys	
0	0	2021-04-07 11:33:49.861733+00:00	1965.845	0.01	875154.482918	1.68
1	1	2021-04-07 11:34:49.861733+00:00	1969.645	0.65	514168.079888	8.58
2	2	2021-04-07 11:35:49.861733+00:00	1975.595	0.29	729915.129243	1.44
3	3	2021-04-07 11:36:49.861733+00:00	1969.335	0.19	611826.976792	5.98
4	4	2021-04-07 11:37:49.861733+00:00	1970.965	0.49	429786.641273	4.14
...	...	...	...	...	...	
17105	17105	2021-04-19 09:49:00.345392+00:00	2238.505	0.01	47067.406991	7.25
17106	17106	2021-04-19 09:50:00.345392+00:00	2238.005	0.01	19950.489155	3.17
17107	17107	2021-04-19 09:51:00.345392+00:00	2240.405	0.01	35268.328086	2.64
17108	17108	2021-04-19 09:52:00.345392+00:00	2236.795	0.01	82036.022512	1.62
17109	17109	2021-04-19 09:53:00.345392+00:00	2235.225	0.01	44542.528617	1.41

17110 rows × 156 columns

In [ ]:
```
df_eth['system_time']
```

Out[ ]:

	system_time
0	2021-04-07 11:33:49.861733+00:00
1	2021-04-07 11:34:49.861733+00:00
2	2021-04-07 11:35:49.861733+00:00
3	2021-04-07 11:36:49.861733+00:00
4	2021-04-07 11:37:49.861733+00:00
...	...
17105	2021-04-19 09:49:00.345392+00:00
17106	2021-04-19 09:50:00.345392+00:00
17107	2021-04-19 09:51:00.345392+00:00
17108	2021-04-19 09:52:00.345392+00:00
17109	2021-04-19 09:53:00.345392+00:00

17110 rows × 1 columns

dtype: object

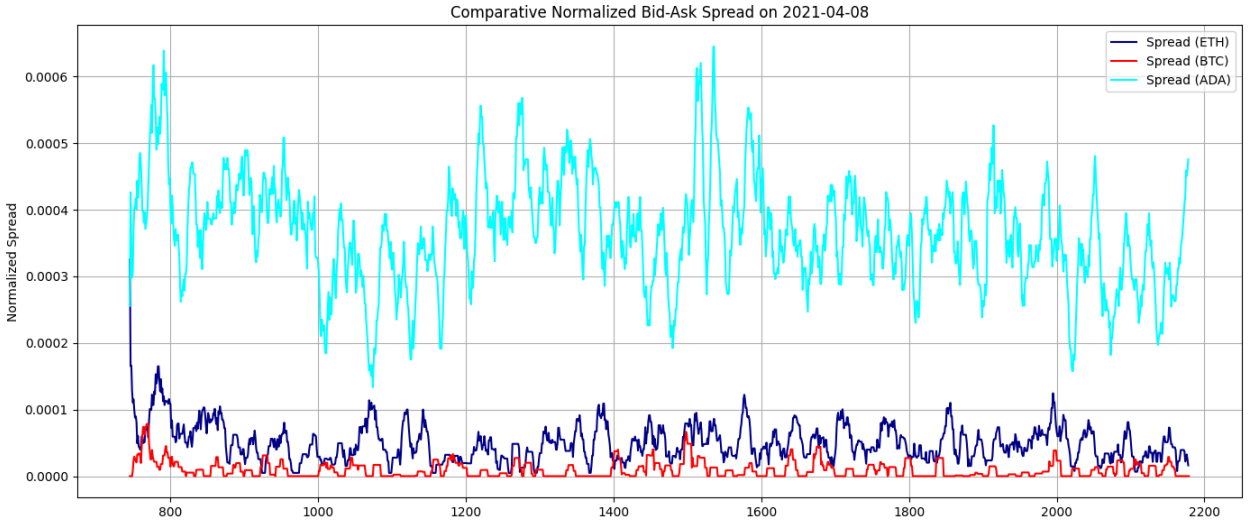
In [ ]:
```
df_eth['date'] = pd.to_datetime(df_eth['system_time']).dt.date
df_btc['date'] = pd.to_datetime(df_btc['system_time']).dt.date
df_ada['date'] = pd.to_datetime(df_ada['system_time']).dt.date
```

In [ ]:
```
from datetime import date
df_eth['spread1']=df_eth['spread']/df_eth['midpoint']
df_btc['spread1']=df_btc['spread']/df_btc['midpoint']
df_ada['spread1']=df_ada['spread']/df_ada['midpoint']
target_date = date(2021, 4, 8)
df_one_day = df_eth[df_eth['date'] == target_date]
df_two_day = df_btc[df_btc['date'] == target_date]
df_three_day = df_ada[df_ada['date'] == target_date]
```

In [ ]:
```
import matplotlib.pyplot as plt
df_one_day['spread_smooth1'] = df_one_day['spread1'].rolling(window=10, min_pe
df_two_day['spread_smooth2'] = df_two_day['spread1'].rolling(window=10, min_pe
df_three_day['spread_smooth3'] = df_three_day['spread1'].rolling(window=10, mi

plt.figure(figsize=(14, 6))
plt.plot( df_one_day['spread_smooth1'], label='Spread (ETH)', color='darkblue'
plt.plot( df_two_day['spread_smooth2'], label='Spread (BTC)', color='red', lin
plt.plot( df_three_day['spread_smooth3'], label='Spread (ADA)', color='cyan',

plt.title('Comparative Normalized Bid-Ask Spread on 2021-04-08')
plt.ylabel('Normalized Spread')
plt.grid(True)
plt.legend()
```

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

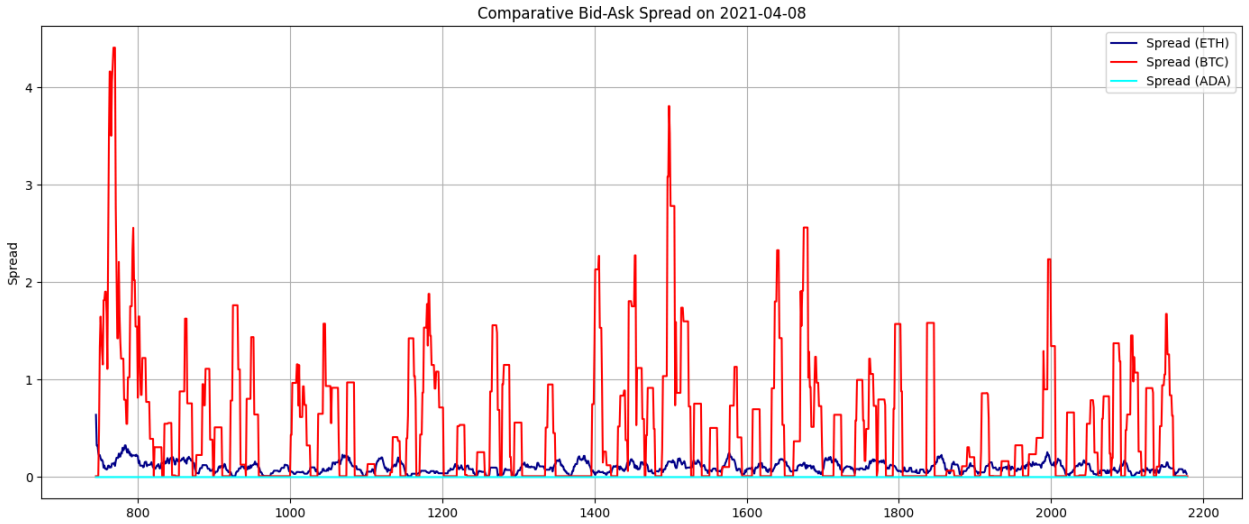
```
/tmp/ipython-input-37-4244041925.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_one_day['spread_smooth1'] = df_one_day['spread1'].rolling(window=10, min_p
eriods=1).mean()
/tmp/ipython-input-37-4244041925.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_two_day['spread_smooth2'] = df_two_day['spread1'].rolling(window=10, min_p
eriods=1).mean()
/tmp/ipython-input-37-4244041925.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_three_day['spread_smooth3'] = df_three_day['spread1'].rolling(window=10, m
in_periods=1).mean()
```



In [ ]:
```
import matplotlib.pyplot as plt
df_one_day['spread_smooth1'] = df_one_day['spread'].rolling(window=10, min_per
df_two_day['spread_smooth2'] = df_two_day['spread'].rolling(window=10, min_per
df_three_day['spread_smooth3'] = df_three_day['spread'].rolling(window=10, mir

plt.figure(figsize=(14, 6))
plt.plot( df_one_day['spread_smooth1'], label='Spread (ETH)', color='darkblue'
plt.plot( df_two_day['spread_smooth2'], label='Spread (BTC)', color='red', lir
plt.plot( df_three_day['spread_smooth3'], label='Spread (ADA)', color='cyan',
```

```
plt.title('Comparative Bid-Ask Spread on 2021-04-08')
plt.ylabel('Spread')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-38-833663482.py:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_one_day['spread_smooth1'] = df_one_day['spread'].rolling(window=10, min_pe
riods=1).mean()
/tmp/ipython-input-38-833663482.py:3: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_two_day['spread_smooth2'] = df_two_day['spread'].rolling(window=10, min_pe
riods=1).mean()
/tmp/ipython-input-38-833663482.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_three_day['spread_smooth3'] = df_three_day['spread'].rolling(window=10, mi
n_periods=1).mean()
```



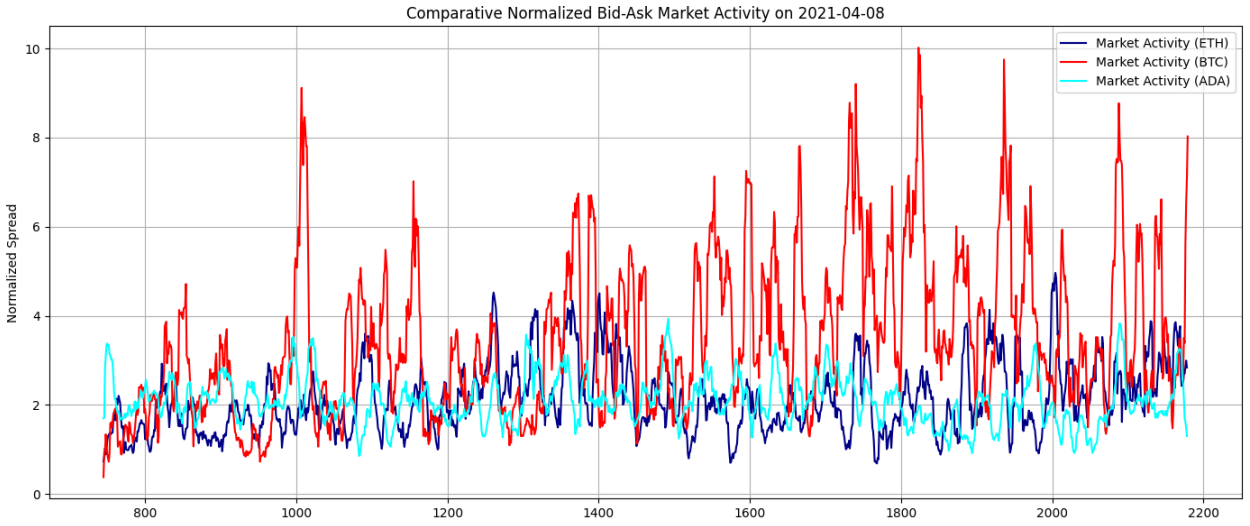
In [ ]:
```
btc_market_notional = df_btc[[f'bids_market_notional_{i}' for i in range(15)]
                             [f'asks_market_notional_{i}' for i in range(15)]]

btc_cancel_notional = df_btc[[f'bids_cancel_notional_{i}' for i in range(15)]
                              [f'asks_cancel_notional_{i}' for i in range(15)]
```

```
btc_depth = df_btc[[f'bids_notional_{i}' for i in range(15)] +
                   [f'asks_notional_{i}' for i in range(15)]].sum(axis=1)

df_btc['market_activity_flow'] = (btc_market_notional + btc_cancel_notional) /

eth_market_notional = df_eth[[f'bids_market_notional_{i}' for i in range(15)]
                             [f'asks_market_notional_{i}' for i in range(15)]]

eth_cancel_notional = df_eth[[f'bids_cancel_notional_{i}' for i in range(15)]
                              [f'asks_cancel_notional_{i}' for i in range(15)]

eth_depth = df_eth[[f'bids_notional_{i}' for i in range(15)] +
                   [f'asks_notional_{i}' for i in range(15)]].sum(axis=1)

df_eth['market_activity_flow'] = (eth_market_notional + eth_cancel_notional) /
ada_market_notional = df_ada[[f'bids_market_notional_{i}' for i in range(15)]
                             [f'asks_market_notional_{i}' for i in range(15)]]

ada_cancel_notional = df_ada[[f'bids_cancel_notional_{i}' for i in range(15)]
                              [f'asks_cancel_notional_{i}' for i in range(15)]

ada_depth = df_ada[[f'bids_notional_{i}' for i in range(15)] +
                   [f'asks_notional_{i}' for i in range(15)]].sum(axis=1)

df_ada['market_activity_flow'] = (ada_market_notional + ada_cancel_notional) /
```

In [ ]:
```
target_date = date(2021, 4, 8)
df_one_day = df_eth[df_eth['date'] == target_date]
df_two_day = df_btc[df_btc['date'] == target_date]
df_three_day = df_ada[df_ada['date'] == target_date]
```

In [ ]:
```
target_date = date(2021, 4, 8)
df_one_day = df_eth[df_eth['date'] == target_date]
df_two_day = df_btc[df_btc['date'] == target_date]
df_three_day = df_ada[df_ada['date'] == target_date]

import matplotlib.pyplot as plt
df_one_day['spread_smooth1'] = df_one_day['market_activity_flow'].rolling(winc
df_two_day['spread_smooth2'] = df_two_day['market_activity_flow'].rolling(winc
df_three_day['spread_smooth3'] = df_three_day['market_activity_flow'].rolling(

plt.figure(figsize=(14, 6))
plt.plot( df_one_day['spread_smooth1'], label='Market Activity (ETH)', color='
plt.plot( df_two_day['spread_smooth2'], label='Market Activity (BTC)', color='
plt.plot( df_three_day['spread_smooth3'], label='Market Activity (ADA)', color

plt.title('Comparative Normalized Bid-Ask Market Activity on 2021-04-08')
plt.ylabel('Normalized Spread')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-41-144711461.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_one_day['spread_smooth1'] = df_one_day['market_activity_flow'].rolling(win
dow=10, min_periods=1).mean()
/tmp/ipython-input-41-144711461.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_two_day['spread_smooth2'] = df_two_day['market_activity_flow'].rolling(win
dow=10, min_periods=1).mean()
/tmp/ipython-input-41-144711461.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_three_day['spread_smooth3'] = df_three_day['market_activity_flow'].rollin
g(window=10, min_periods=1).mean()
```

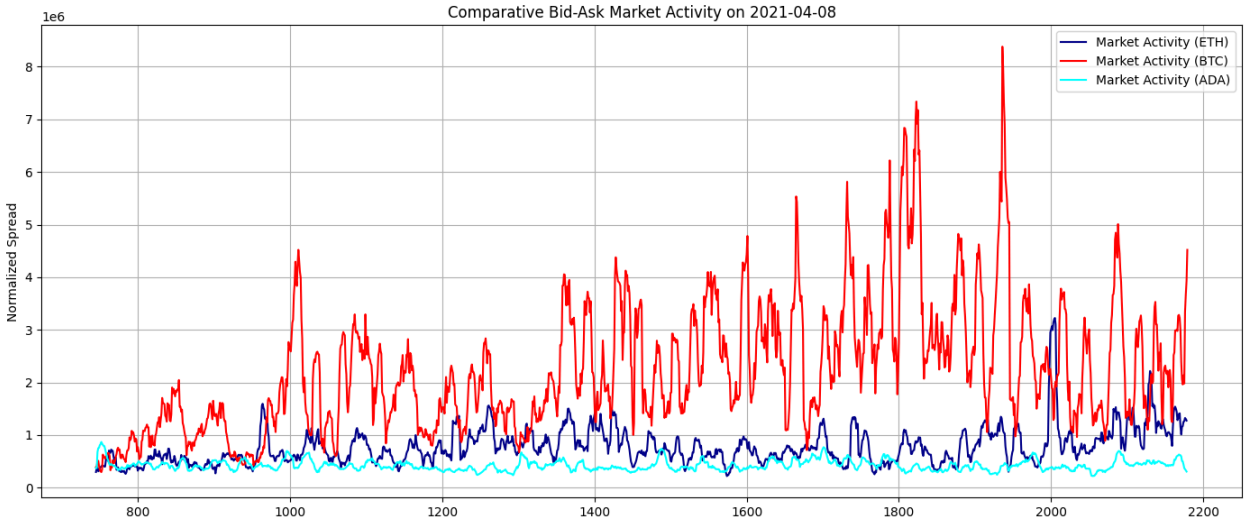


In [ ]:
```
df_btc['market_activity_flow1'] = (btc_market_notional + btc_cancel_notional)
df_eth['market_activity_flow1'] = (eth_market_notional + eth_cancel_notional)
df_ada['market_activity_flow1'] = (ada_market_notional + ada_cancel_notional)
```

In [ ]:
```
target_date = date(2021, 4, 8)
df_one_day = df_eth[df_eth['date'] == target_date]
df_two_day = df_btc[df_btc['date'] == target_date]
df_three_day = df_ada[df_ada['date'] == target_date]
import matplotlib.pyplot as plt
df_one_day['spread_smooth1'] = df_one_day['market_activity_flow1'].rolling(wir
df_two_day['spread_smooth2'] = df_two_day['market_activity_flow1'].rolling(wir
df_three_day['spread_smooth3'] = df_three_day['market_activity_flow1'].rolling
```

```
plt.figure(figsize=(14, 6))
plt.plot( df_one_day['spread_smooth1'], label='Market Activity (ETH)', color='
plt.plot( df_two_day['spread_smooth2'], label='Market Activity (BTC)', color='
plt.plot( df_three_day['spread_smooth3'], label='Market Activity (ADA)', color

plt.title('Comparative Bid-Ask Market Activity on 2021-04-08')
plt.ylabel('Normalized Spread')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

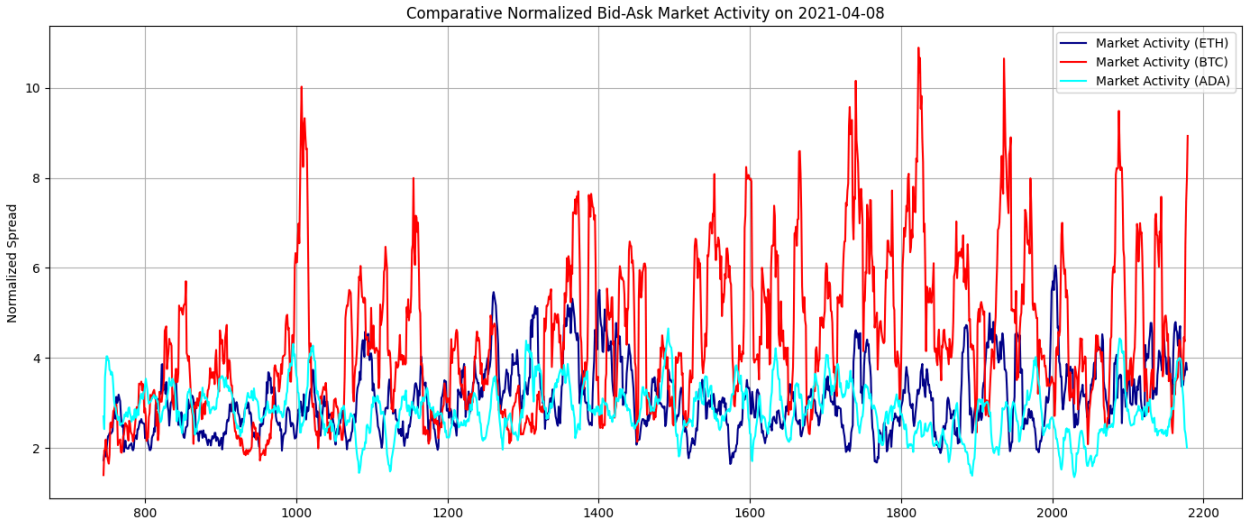
```
/tmp/ipython-input-43-369707001.py:6: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_one_day['spread_smooth1'] = df_one_day['market_activity_flow1'].rolling(wi
ndow=10, min_periods=1).mean()
/tmp/ipython-input-43-369707001.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_two_day['spread_smooth2'] = df_two_day['market_activity_flow1'].rolling(wi
ndow=10, min_periods=1).mean()
/tmp/ipython-input-43-369707001.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_three_day['spread_smooth3'] = df_three_day['market_activity_flow1'].rollin
g(window=10, min_periods=1).mean()
```



In [ ]:
```
btc_market_notional = df_btc[[f'bids_market_notional_{i}' for i in range(15)]
                             [f'asks_market_notional_{i}' for i in range(15)]]

btc_limit_notional = df_btc[[f'bids_limit_notional_{i}' for i in range(15)] +
                            [f'asks_limit_notional_{i}' for i in range(15)]]

btc_depth = df_btc[[f'bids_notional_{i}' for i in range(15)] +
                   [f'asks_notional_{i}' for i in range(15)]].sum(axis=1)

df_btc['market_activity_flow'] = (btc_market_notional + btc_limit_notional) /

eth_market_notional = df_eth[[f'bids_market_notional_{i}' for i in range(15)]
                             [f'asks_market_notional_{i}' for i in range(15)]]

eth_limit_notional = df_eth[[f'bids_limit_notional_{i}' for i in range(15)] +
                            [f'asks_limit_notional_{i}' for i in range(15)]]

eth_depth = df_eth[[f'bids_notional_{i}' for i in range(15)] +
                   [f'asks_notional_{i}' for i in range(15)]].sum(axis=1)

df_eth['market_activity_flow'] = (eth_market_notional + eth_limit_notional) /
ada_market_notional = df_ada[[f'bids_market_notional_{i}' for i in range(15)]
                             [f'asks_market_notional_{i}' for i in range(15)]]

ada_limit_notional = df_ada[[f'bids_limit_notional_{i}' for i in range(15)] +
                            [f'asks_limit_notional_{i}' for i in range(15)]]

ada_depth = df_ada[[f'bids_notional_{i}' for i in range(15)] +
                   [f'asks_notional_{i}' for i in range(15)]].sum(axis=1)

df_ada['market_activity_flow'] = (ada_market_notional + ada_limit_notional) /
```

In [ ]:
```
target_date = date(2021, 4, 8)
df_one_day = df_eth[df_eth['date'] == target_date]
df_two_day = df_btc[df_btc['date'] == target_date]
df_three_day = df_ada[df_ada['date'] == target_date]

import matplotlib.pyplot as plt
df_one_day['spread_smooth1'] = df_one_day['market_activity_flow'].rolling(windo
df_two_day['spread_smooth2'] = df_two_day['market_activity_flow'].rolling(windo
df_three_day['spread_smooth3'] = df_three_day['market_activity_flow'].rolling(

plt.figure(figsize=(14, 6))
plt.plot( df_one_day['spread_smooth1'], label='Market Activity (ETH)', color='
plt.plot( df_two_day['spread_smooth2'], label='Market Activity (BTC)', color='
plt.plot( df_three_day['spread_smooth3'], label='Market Activity (ADA)', color

plt.title('Comparative Normalized Bid-Ask Market Activity on 2021-04-08')
plt.ylabel('Normalized Spread')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-45-144711461.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_one_day['spread_smooth1'] = df_one_day['market_activity_flow'].rolling(win
dow=10, min_periods=1).mean()
/tmp/ipython-input-45-144711461.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_two_day['spread_smooth2'] = df_two_day['market_activity_flow'].rolling(win
dow=10, min_periods=1).mean()
/tmp/ipython-input-45-144711461.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/sta
ble/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_three_day['spread_smooth3'] = df_three_day['market_activity_flow'].rollin
g(window=10, min_periods=1).mean()
```



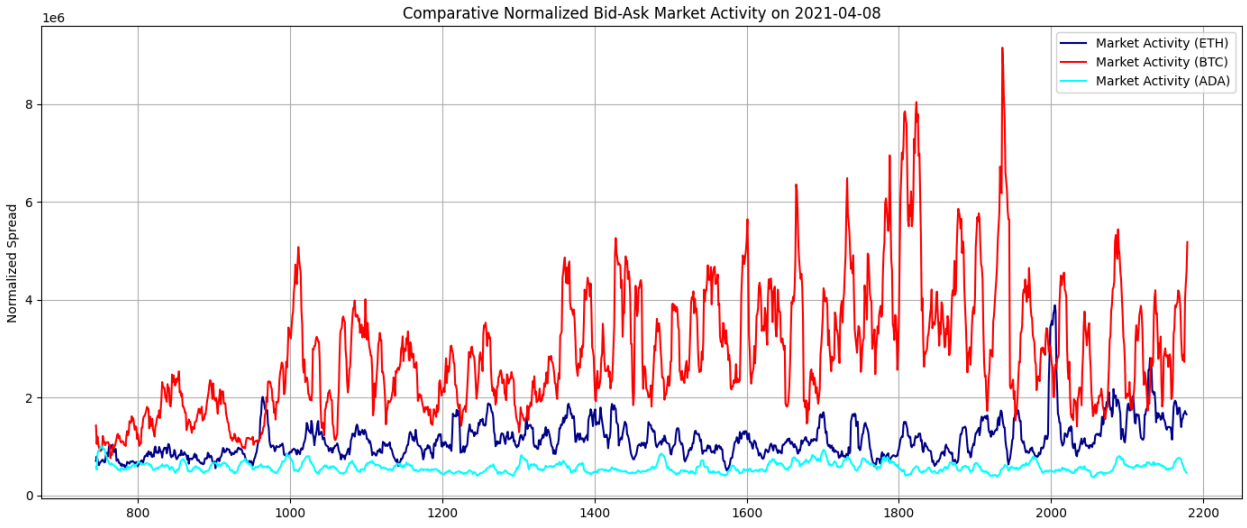
In [ ]:
```python
btc_market_notional = df_btc[[f'bids_market_notional_{i}' for i in range(15)]
                             [f'asks_market_notional_{i}' for i in range(15)]]

btc_limit_notional = df_btc[[f'bids_limit_notional_{i}' for i in range(15)] +
                            [f'asks_limit_notional_{i}' for i in range(15)]]

btc_depth = df_btc[[f'bids_notional_{i}' for i in range(15)] +
                   [f'asks_notional_{i}' for i in range(15)]].sum(axis=1)

df_btc['market_activity_flow'] = (btc_market_notional + btc_limit_notional)

eth_market_notional = df_eth[[f'bids_market_notional_{i}' for i in range(15)]
                             [f'asks_market_notional_{i}' for i in range(15)]]
```

```
eth_limit_notional = df_eth[[f'bids_limit_notional_{i}' for i in range(15)] +
                            [f'asks_limit_notional_{i}' for i in range(15)]]

eth_depth = df_eth[[f'bids_notional_{i}' for i in range(15)] +
                   [f'asks_notional_{i}' for i in range(15)]].sum(axis=1)

df_eth['market_activity_flow'] = (eth_market_notional + eth_limit_notional)
ada_market_notional = df_ada[[f'bids_market_notional_{i}' for i in range(15)]
                             [f'asks_market_notional_{i}' for i in range(15)]]

ada_limit_notional = df_ada[[f'bids_limit_notional_{i}' for i in range(15)] +
                            [f'asks_limit_notional_{i}' for i in range(15)]]

ada_depth = df_ada[[f'bids_notional_{i}' for i in range(15)] +
                   [f'asks_notional_{i}' for i in range(15)]].sum(axis=1)

df_ada['market_activity_flow'] = (ada_market_notional + ada_limit_notional)
```

In [ ]:

```
target_date = date(2021, 4, 8)
df_one_day = df_eth[df_eth['date'] == target_date]
df_two_day = df_btc[df_btc['date'] == target_date]
df_three_day = df_ada[df_ada['date'] == target_date]

import matplotlib.pyplot as plt
df_one_day['spread_smooth1'] = df_one_day['market_activity_flow'].rolling(wind
df_two_day['spread_smooth2'] = df_two_day['market_activity_flow'].rolling(wind
df_three_day['spread_smooth3'] = df_three_day['market_activity_flow'].rolling(

plt.figure(figsize=(14, 6))
plt.plot( df_one_day['spread_smooth1'], label='Market Activity (ETH)', color='
plt.plot( df_two_day['spread_smooth2'], label='Market Activity (BTC)', color='
plt.plot( df_three_day['spread_smooth3'], label='Market Activity (ADA)', color

plt.title('Comparative Normalized Bid-Ask Market Activity on 2021-04-08')
plt.ylabel('Normalized Spread')
plt.grid(True)
plt.legend()
plt.tight_layout()
plt.show()
```

```
/tmp/ipython-input-47-144711461.py:7: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_one_day['spread_smooth1'] = df_one_day['market_activity_flow'].rolling(window=10, min_periods=1).mean()
/tmp/ipython-input-47-144711461.py:8: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_two_day['spread_smooth2'] = df_two_day['market_activity_flow'].rolling(window=10, min_periods=1).mean()
/tmp/ipython-input-47-144711461.py:9: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
  df_three_day['spread_smooth3'] = df_three_day['market_activity_flow'].rolling(window=10, min_periods=1).mean()
```



In [ ]:
```
df_ada
```

Out[ ]:

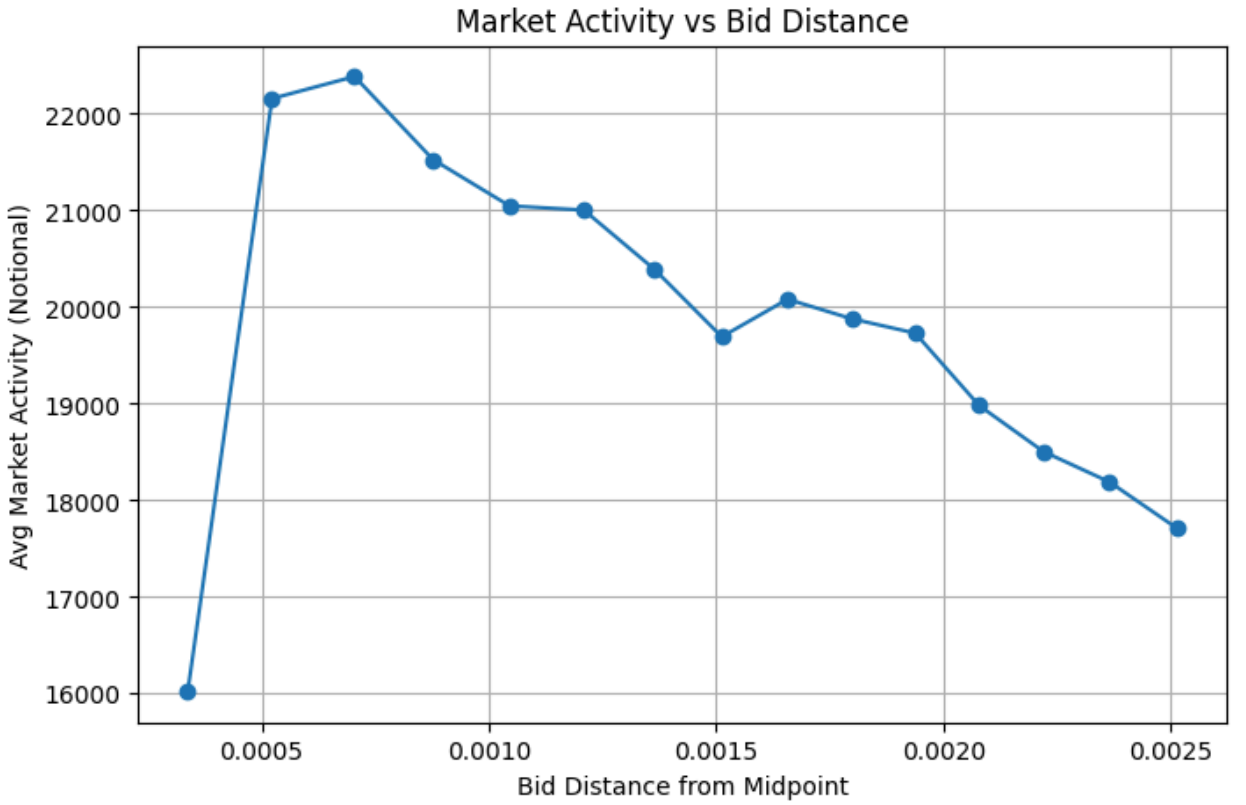
	Unnamed: 0	system_time	midpoint	spread	buys	
0	0	2021-04-07 11:33:59.055697+00:00	1.16205	0.0001	56936.467913	2582
1	1	2021-04-07 11:34:59.055697+00:00	1.16800	0.0022	56491.336799	786
2	2	2021-04-07 11:35:59.055697+00:00	1.17530	0.0012	52859.493359	484
3	3	2021-04-07 11:36:59.055697+00:00	1.16585	0.0017	50772.386336	326
4	4	2021-04-07 11:37:59.055697+00:00	1.17255	0.0009	113579.364184	825
...	...	...	...	...	...	
17104	17104	2021-04-19 09:45:00.442103+00:00	1.27325	0.0001	13671.251598	253
17105	17105	2021-04-19 09:46:00.442103+00:00	1.27200	0.0008	9916.946518	336
17106	17106	2021-04-19 09:47:00.442103+00:00	1.27255	0.0007	32589.054204	434
17107	17107	2021-04-19 09:48:00.442103+00:00	1.27305	0.0001	3437.251449	79
17108	17108	2021-04-19 09:49:00.442103+00:00	1.27105	0.0007	10510.439494	68

17109 rows × 161 columns

In [ ]:

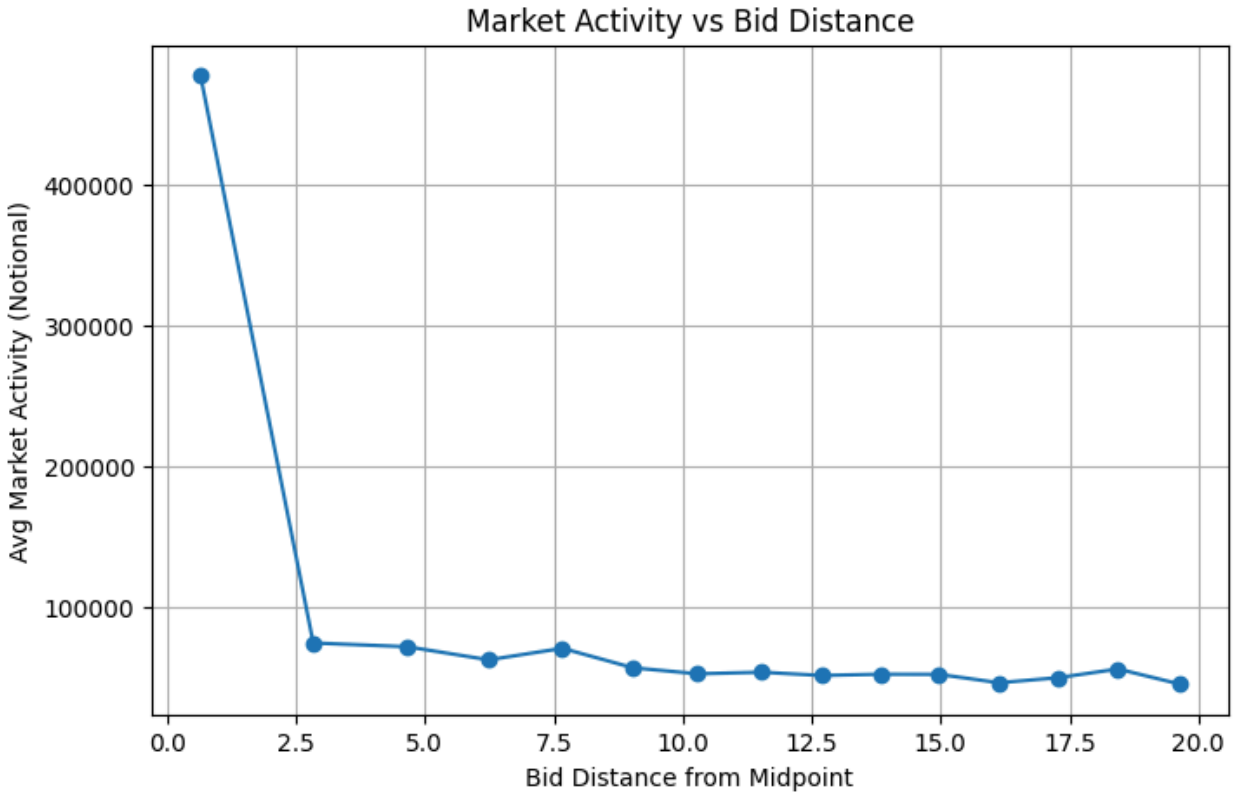
```
levels = list(range(15))

market_activities = []
distances = []

for i in levels:
    mkt_col = f"bids_market_notional_{i}"
    lim_col = f"bids_limit_notional_{i}"
    dist_col = f"bids_distance_{i}"


    activity_i = (df_ada[mkt_col] + df_ada[lim_col]).mean()
    distance_i = df_ada[dist_col].mean()

    market_activities.append(activity_i)
    distances.append(distance_i)


plt.figure(figsize=(8,5))
plt.plot(distances, market_activities, marker='o')
```

```
plt.xlabel("Bid Distance from Midpoint")
plt.ylabel("Avg Market Activity (Notional)")
plt.title("Market Activity vs Bid Distance")
plt.grid(True)
plt.show()
```

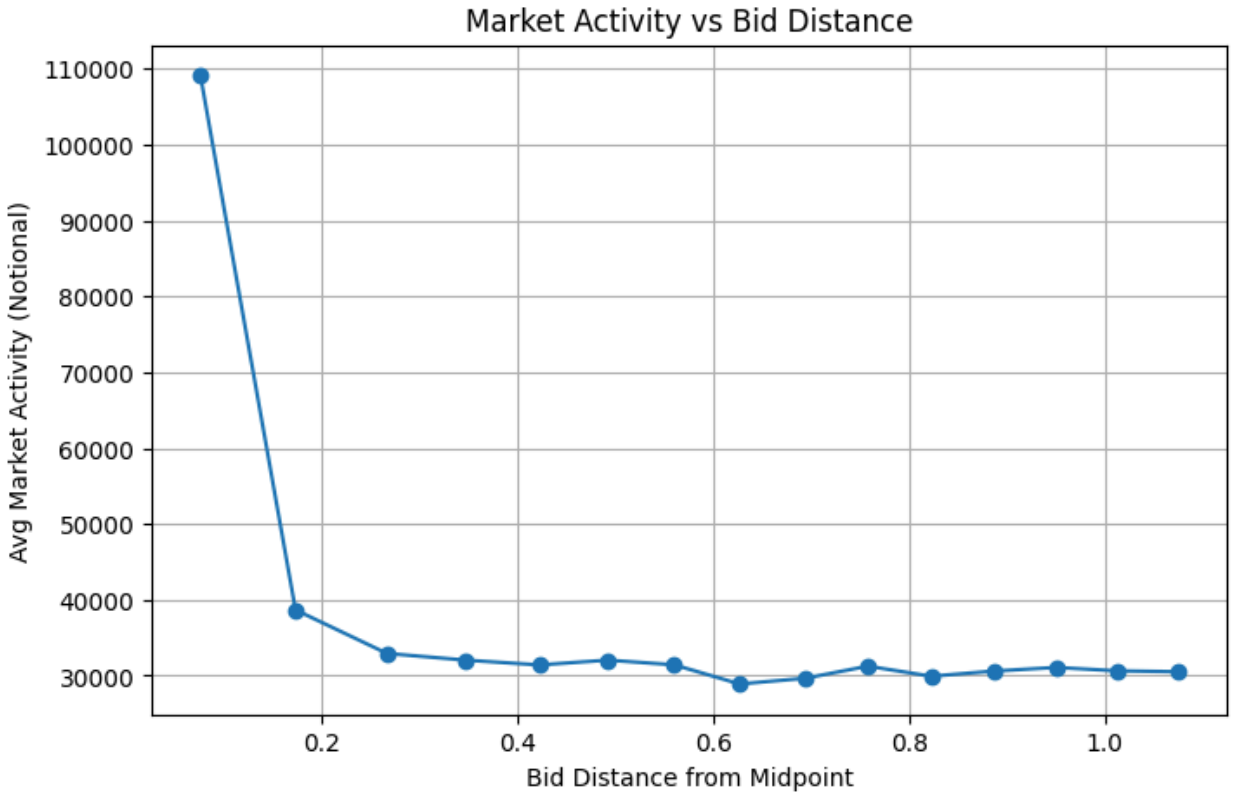


In [ ]:
```
levels = list(range(15))

market_activities = []
distances = []

for i in levels:
    mkt_col = f"bids_market_notional_{i}"
    lim_col = f"bids_limit_notional_{i}"
    dist_col = f"bids_distance_{i}"


    activity_i = (df_btc[mkt_col] + df_btc[lim_col]).mean()
    distance_i = df_btc[dist_col].mean()

    market_activities.append(activity_i)
    distances.append(distance_i)


plt.figure(figsize=(8,5))
plt.plot(distances, market_activities, marker='o')
plt.xlabel("Bid Distance from Midpoint")
plt.ylabel("Avg Market Activity (Notional)")
```

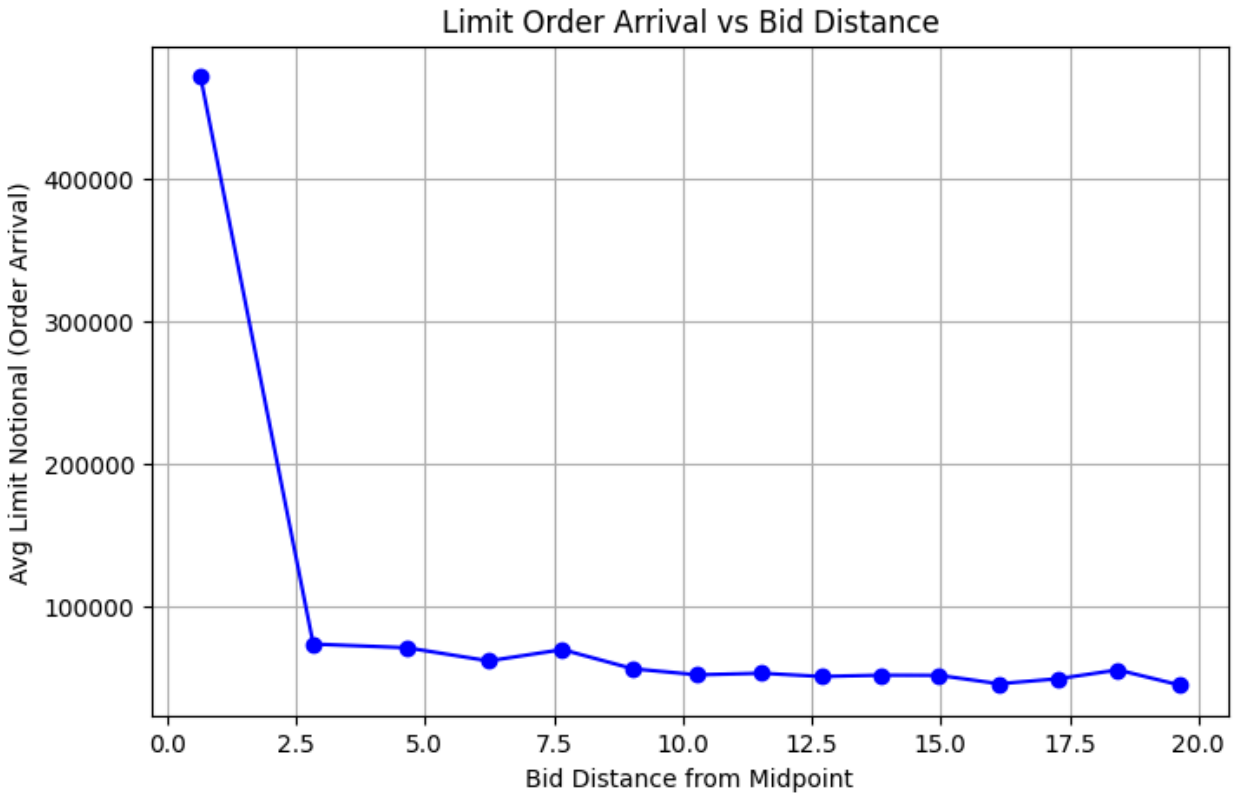
```
plt.title("Market Activity vs Bid Distance")
plt.grid(True)
plt.show()
```



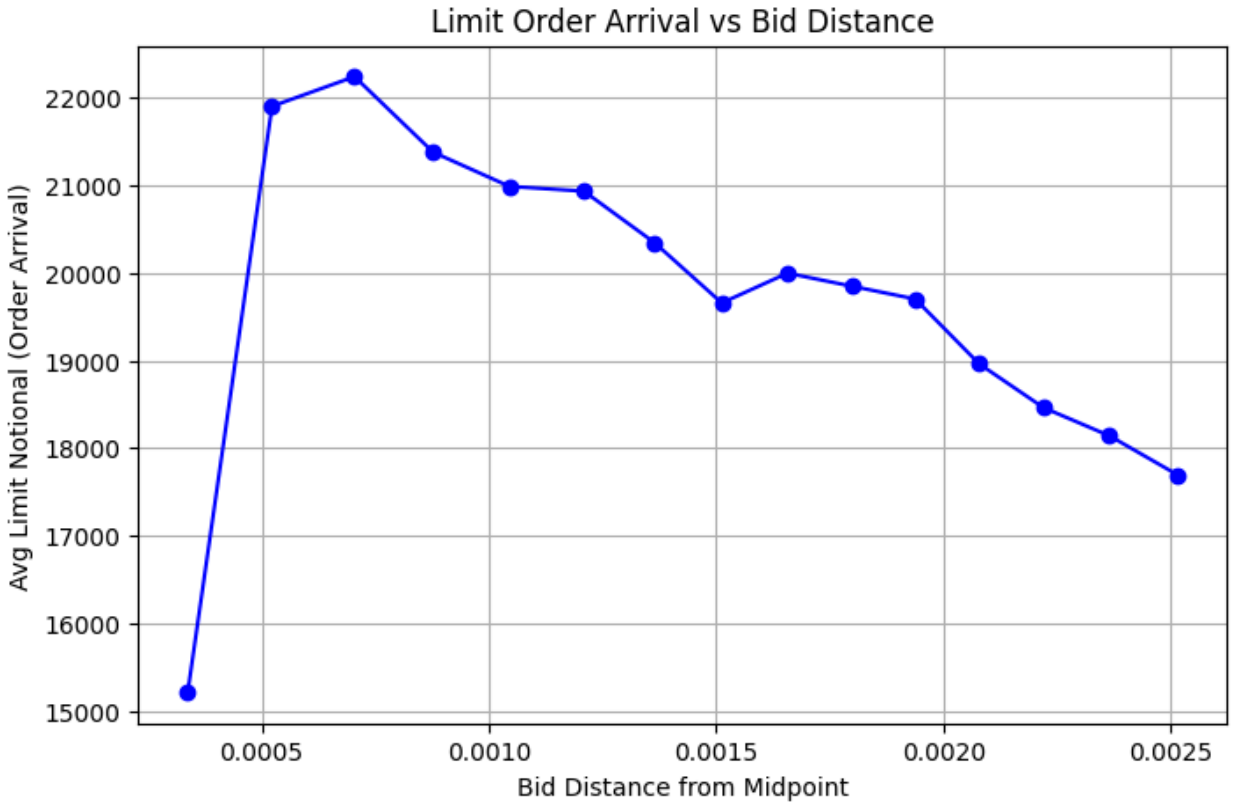
In [ ]:
```
levels = list(range(15))

market_activities = []
distances = []

for i in levels:
    mkt_col = f"bids_market_notional_{i}"
    lim_col = f"bids_limit_notional_{i}"
    dist_col = f"bids_distance_{i}"


    activity_i = (df_eth[mkt_col] + df_eth[lim_col]).mean()
    distance_i = df_eth[dist_col].mean()

    market_activities.append(activity_i)
    distances.append(distance_i)


plt.figure(figsize=(8,5))
plt.plot(distances, market_activities, marker='o')
plt.xlabel("Bid Distance from Midpoint")
plt.ylabel("Avg Market Activity (Notional)")
plt.title("Market Activity vs Bid Distance")
plt.grid(True)
plt.show()
```



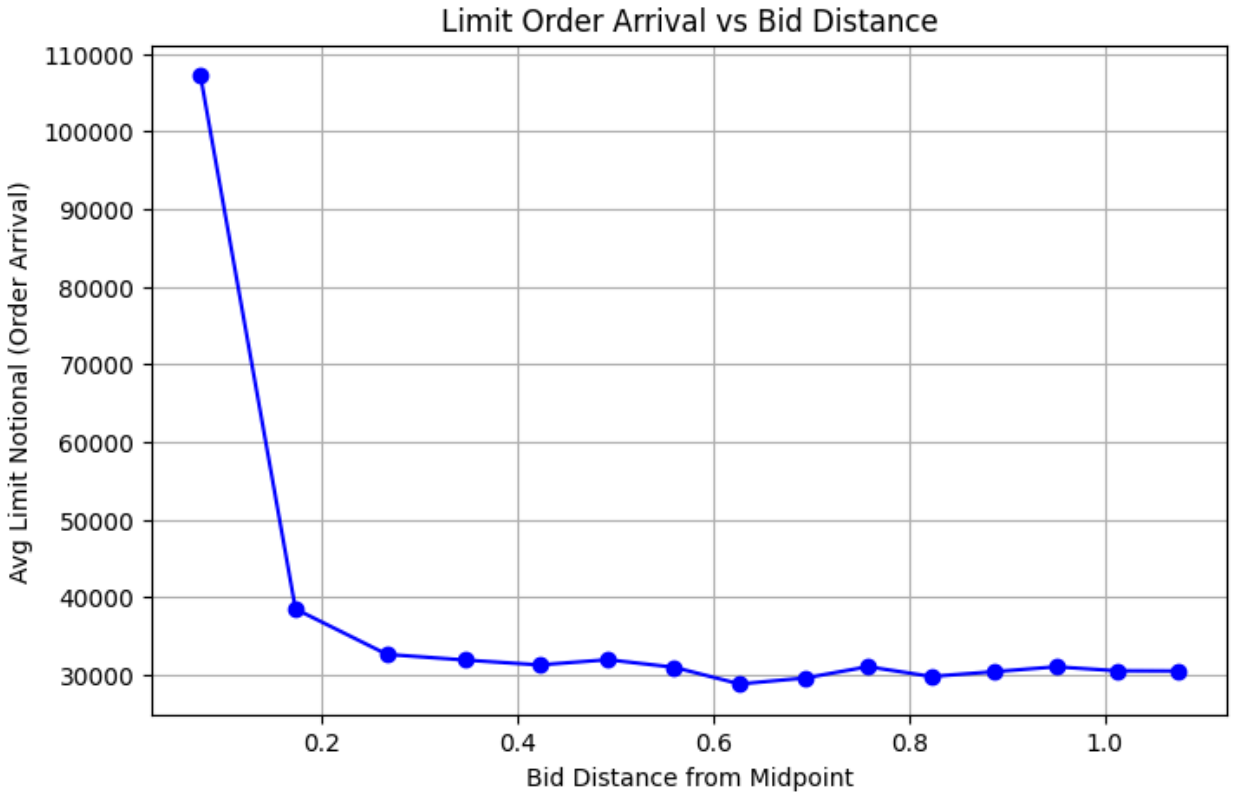
In [ ]:
```python
import matplotlib.pyplot as plt

limit_arrivals = []
distances = []

for i in range(15):
    lim_col = f"bids_limit_notional_{i}"
    dist_col = f"bids_distance_{i}"

    limit_arrival_i = df_btc[lim_col].mean()
    distance_i = df_btc[dist_col].mean()

    limit_arrivals.append(limit_arrival_i)
    distances.append(distance_i)

# Plotting
plt.figure(figsize=(8,5))
plt.plot(distances, limit_arrivals, marker='o', linestyle='-', color='blue')
plt.xlabel("Bid Distance from Midpoint")
plt.ylabel("Avg Limit Notional (Order Arrival)")
plt.title("Limit Order Arrival vs Bid Distance")
plt.grid(True)
plt.show()
```



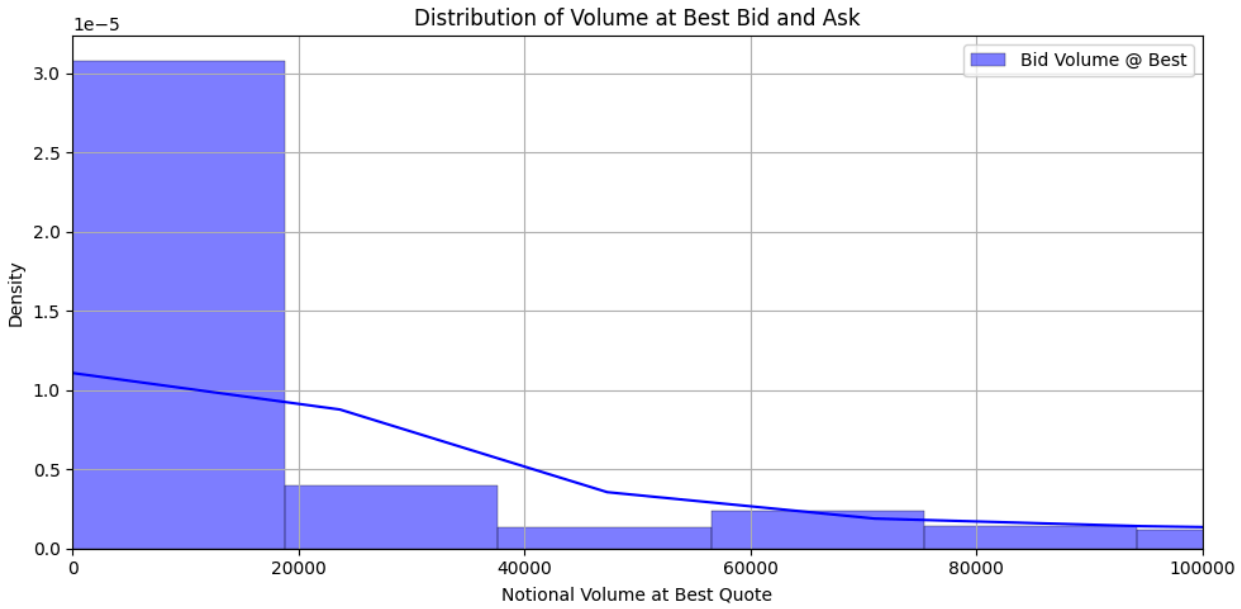
In [ ]:
```python
import matplotlib.pyplot as plt

limit_arrivals = []
distances = []

for i in range(15):
    lim_col = f"bids_limit_notional_{i}"
    dist_col = f"bids_distance_{i}"

    limit_arrival_i = df_ada[lim_col].mean()
    distance_i = df_ada[dist_col].mean()

    limit_arrivals.append(limit_arrival_i)
    distances.append(distance_i)

# Plotting
plt.figure(figsize=(8,5))
plt.plot(distances, limit_arrivals, marker='o', linestyle='-', color='blue')
plt.xlabel("Bid Distance from Midpoint")
plt.ylabel("Avg Limit Notional (Order Arrival)")
plt.title("Limit Order Arrival vs Bid Distance")
plt.grid(True)
plt.show()
```



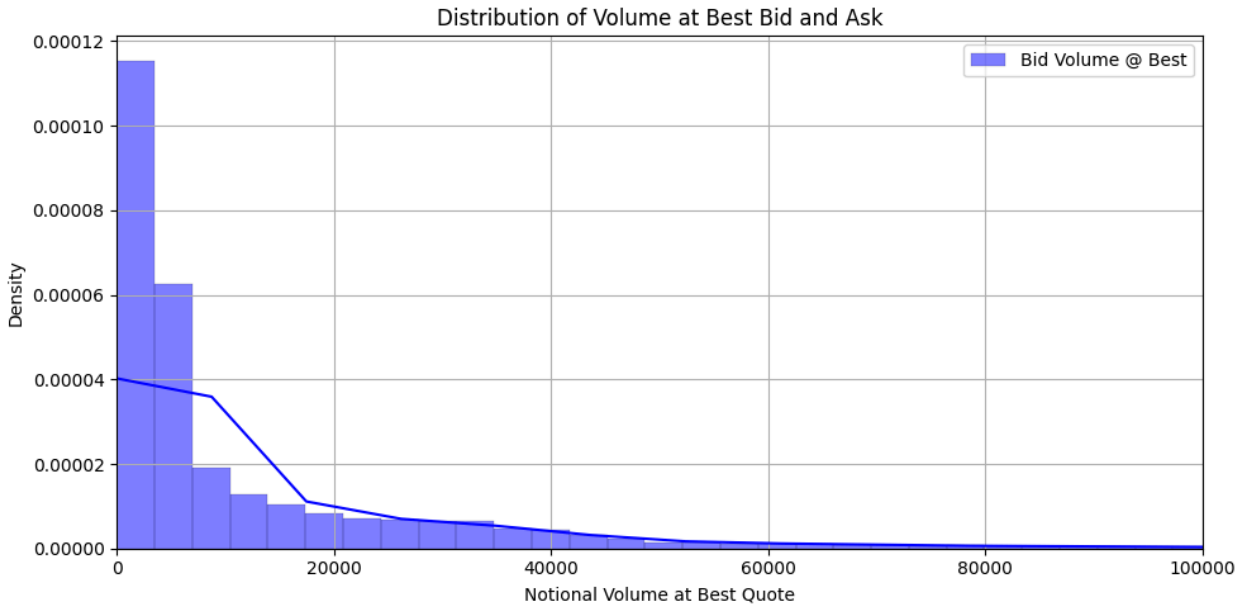
In [ ]:
```python
import matplotlib.pyplot as plt

limit_arrivals = []
distances = []

for i in range(15):
    lim_col = f"bids_limit_notional_{i}"
    dist_col = f"bids_distance_{i}"

    limit_arrival_i = df_eth[lim_col].mean()
    distance_i = df_eth[dist_col].mean()

    limit_arrivals.append(limit_arrival_i)
    distances.append(distance_i)

# Plotting
plt.figure(figsize=(8,5))
plt.plot(distances, limit_arrivals, marker='o', linestyle='-', color='blue')
plt.xlabel("Bid Distance from Midpoint")
plt.ylabel("Avg Limit Notional (Order Arrival)")
plt.title("Limit Order Arrival vs Bid Distance")
plt.grid(True)
plt.show()
```



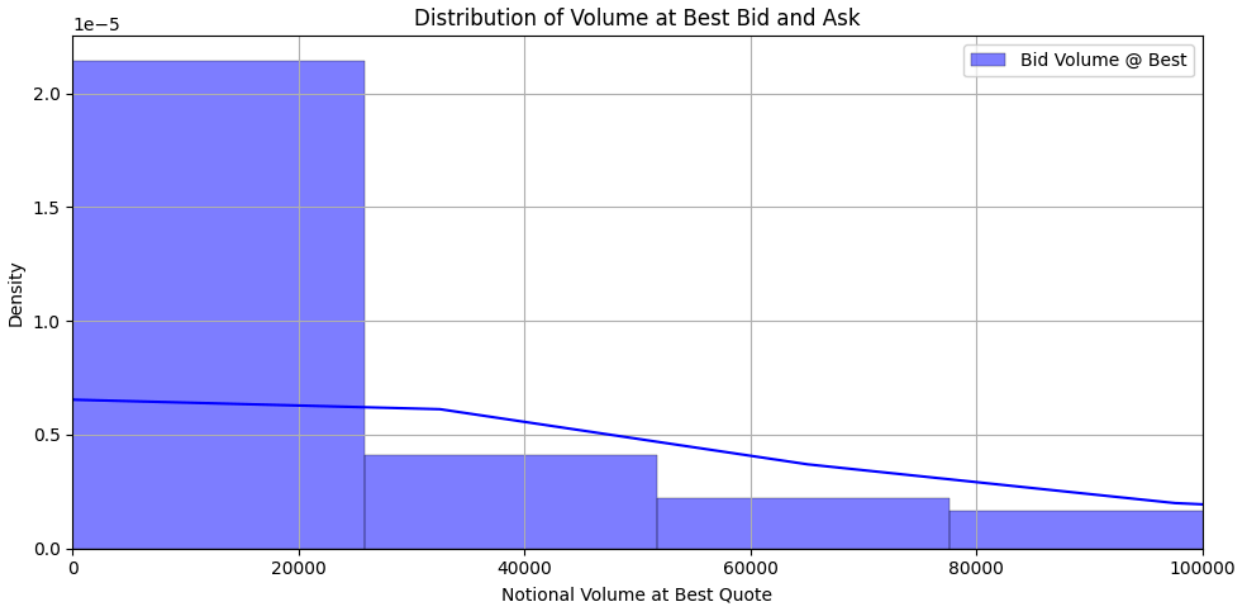
In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 5))

sns.histplot(df_btc['bids_notional_0'], kde=True, bins=250, color='blue', labe



plt.xlabel('Notional Volume at Best Quote')
plt.ylabel('Density')

plt.title('Distribution of Volume at Best Bid and Ask')
plt.legend()
plt.grid(True)
plt.xlim(0,1e5)
plt.tight_layout()
plt.show()
```



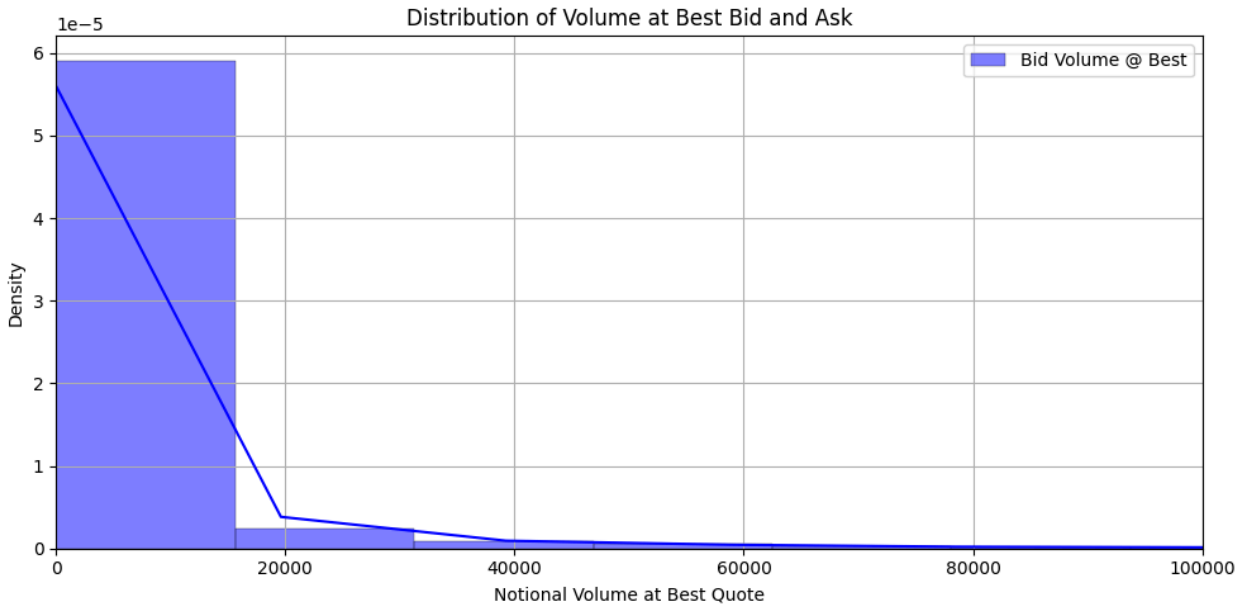
In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 5))

sns.histplot(df_ada['bids_limit_notional_0'], kde=True, bins=500, color='blue'



plt.xlabel('Notional Volume at Best Quote')
plt.ylabel('Density')

plt.title('Distribution of Volume at Best Bid and Ask')
plt.legend()
plt.grid(True)
plt.xlim(0,100000)
plt.tight_layout()
plt.show()
```



In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 5))

sns.histplot(df_eth['bids_limit_notional_0'], kde=True, bins=250, color='blue'



plt.xlabel('Notional Volume at Best Quote')
plt.ylabel('Density')

plt.title('Distribution of Volume at Best Bid and Ask')
plt.legend()
plt.grid(True)
plt.xlim(0,1e5)
plt.tight_layout()
plt.show()
```

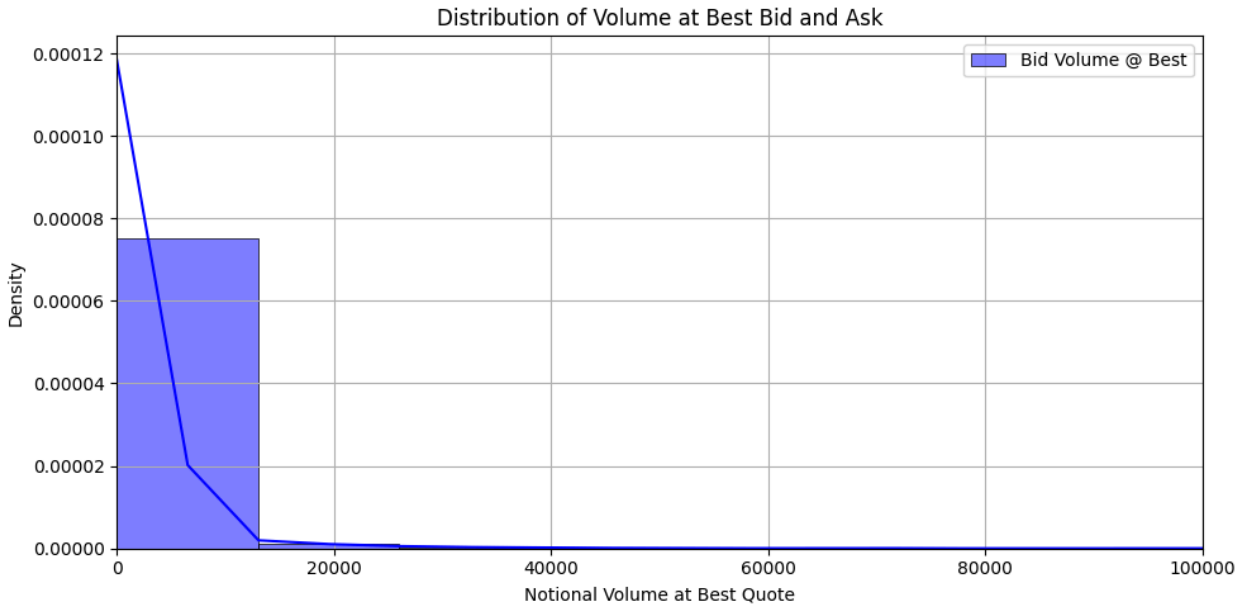


In [ ]:
```
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 5))

sns.histplot(df_btc['bids_market_notional_0'], kde=True, bins=250, color='blue



plt.xlabel('Notional Volume at Best Quote')
plt.ylabel('Density')

plt.title('Distribution of Volume at Best Bid and Ask')
plt.legend()
plt.grid(True)
plt.xlim(0,1e5)
plt.tight_layout()
plt.show()
```



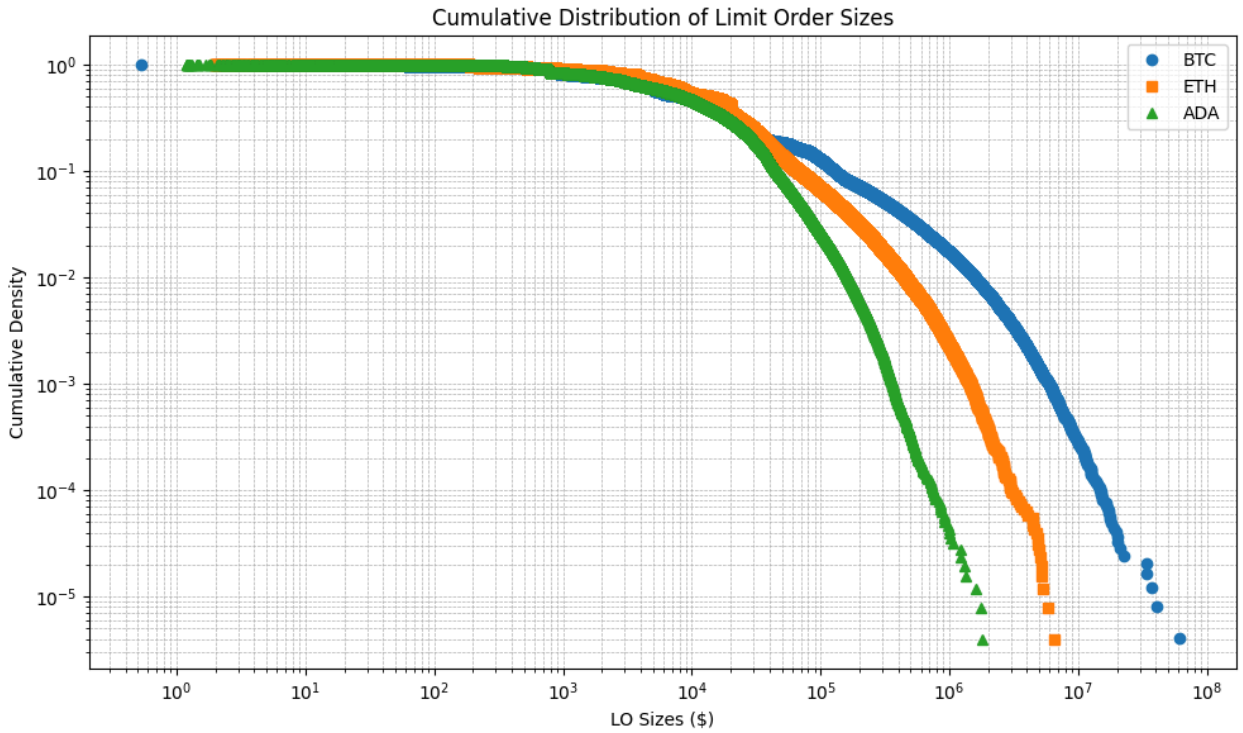
In [ ]:

```python
import matplotlib.pyplot as plt
import seaborn as sns

plt.figure(figsize=(10, 5))

sns.histplot(df_eth['bids_market_notional_0'], kde=True, bins=100, color='blue



plt.xlabel('Notional Volume at Best Quote')
plt.ylabel('Density')

plt.title('Distribution of Volume at Best Bid and Ask')
plt.legend()
plt.grid(True)
plt.xlim(0,1e5)
plt.tight_layout()
plt.show()
```

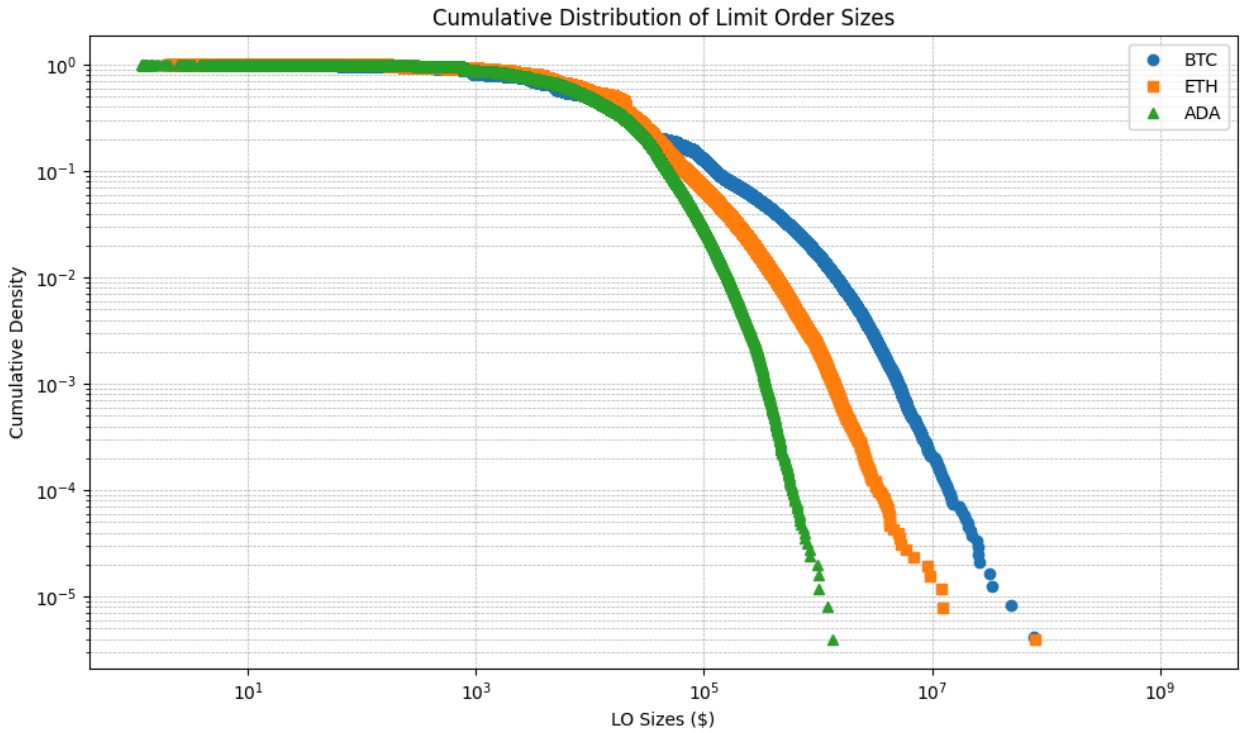


In [ ]:

```python
import numpy as np
import matplotlib.pyplot as plt

def get_cdf_log_data(df, prefix='bids_limit_notional'):
    cols = [f'{prefix}_{i}' for i in range(15)]
    data = df[cols].values.flatten()
    data = data[~np.isnan(data) & (data > 0)]
    sorted_data = np.sort(data)
    cdf = 1.0 - np.arange(1, len(sorted_data)+1) / len(sorted_data)
    return sorted_data, cdf

x_btc, y_btc = get_cdf_log_data(df_btc)
x_eth, y_eth = get_cdf_log_data(df_eth)
x_ada, y_ada = get_cdf_log_data(df_ada)

plt.figure(figsize=(10, 6))
plt.loglog(x_btc, y_btc, label='BTC', marker='o', linestyle='None')
plt.loglog(x_eth, y_eth, label='ETH', marker='s', linestyle='None')
plt.loglog(x_ada, y_ada, label='ADA', marker='^', linestyle='None')

# Optional: plot a power-law guide



plt.xlabel("LO Sizes ($)")
plt.ylabel("Cumulative Density")
plt.title("Cumulative Distribution of Limit Order Sizes")
plt.legend()
plt.grid(True, which="both", ls="--", lw=0.5)
plt.tight_layout()
plt.show()
```

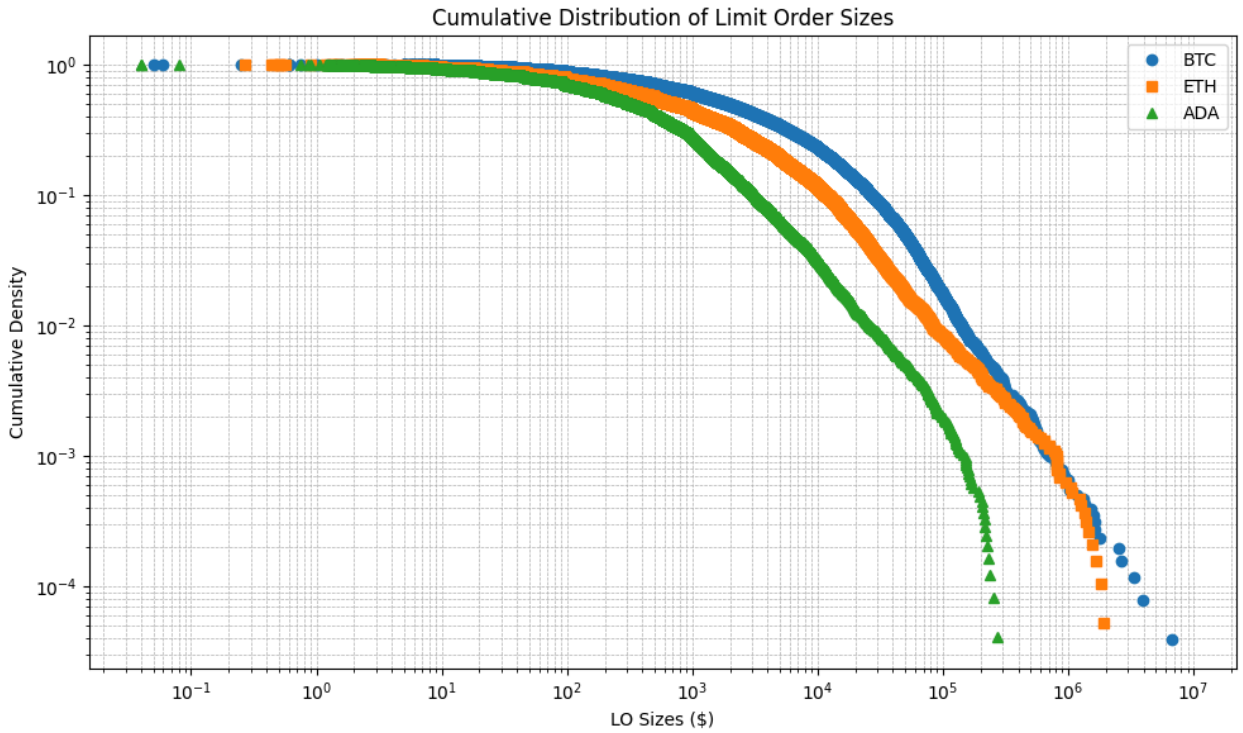


In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt

def get_cdf_log_data(df, prefix='asks_limit_notional'):
    cols = [f'{prefix}_{i}' for i in range(15)]
    data = df[cols].values.flatten()
    data = data[~np.isnan(data) & (data > 0)]
    sorted_data = np.sort(data)
    cdf = 1.0 - np.arange(1, len(sorted_data)+1) / len(sorted_data)
    return sorted_data, cdf

x_btc, y_btc = get_cdf_log_data(df_btc)
x_eth, y_eth = get_cdf_log_data(df_eth)
x_ada, y_ada = get_cdf_log_data(df_ada)

plt.figure(figsize=(10, 6))
plt.loglog(x_btc, y_btc, label='BTC', marker='o', linestyle='None')
plt.loglog(x_eth, y_eth, label='ETH', marker='s', linestyle='None')
plt.loglog(x_ada, y_ada, label='ADA', marker='^', linestyle='None')

# Optional: plot a power-law guide


plt.xlabel("LO Sizes ($)")
plt.ylabel("Cumulative Density")
plt.title("Cumulative Distribution of Limit Order Sizes")
plt.legend()
plt.grid(True, which="both", ls="--", lw=0.5)
plt.tight_layout()
plt.show()
```



In [ ]:

```
import numpy as np
import matplotlib.pyplot as plt

def get_cdf_log_data(df, prefix='asks_market_notional'):
    cols = [f'{prefix}_{i}' for i in range(15)]
    data = df[cols].values.flatten()
    data = data[~np.isnan(data) & (data > 0)]
    sorted_data = np.sort(data)
    cdf = 1.0 - np.arange(1, len(sorted_data)+1) / len(sorted_data)
    return sorted_data, cdf

x_btc, y_btc = get_cdf_log_data(df_btc)
x_eth, y_eth = get_cdf_log_data(df_eth)
x_ada, y_ada = get_cdf_log_data(df_ada)

plt.figure(figsize=(10, 6))
plt.loglog(x_btc, y_btc, label='BTC', marker='o', linestyle='None')
plt.loglog(x_eth, y_eth, label='ETH', marker='s', linestyle='None')
plt.loglog(x_ada, y_ada, label='ADA', marker='^', linestyle='None')


plt.xlabel("LO Sizes ($)")
plt.ylabel("Cumulative Density")
plt.title("Cumulative Distribution of Limit Order Sizes")
plt.legend()
plt.grid(True, which="both", ls="--", lw=0.5)
plt.tight_layout()
plt.show()
```



In [ ]:
```python
import numpy as np
import matplotlib.pyplot as plt

def get_cdf_log_data(df, prefix='bids_limit_notional'):
    cols = [f'{prefix}_{i}' for i in range(15)]
    data = df[cols].values.flatten()
    data = data[~np.isnan(data) & (data > 0)]
    sorted_data = np.sort(data)
    cdf = 1.0 - np.arange(1, len(sorted_data)+1) / len(sorted_data)
    return sorted_data, cdf

x_btc, y_btc = get_cdf_log_data(df_btc)
x_eth, y_eth = get_cdf_log_data(df_eth)
x_ada, y_ada = get_cdf_log_data(df_ada)

plt.figure(figsize=(10, 6))
plt.loglog(x_btc, y_btc, label='BTC', marker='o', linestyle='None')
plt.loglog(x_eth, y_eth, label='ETH', marker='s', linestyle='None')
plt.loglog(x_ada, y_ada, label='ADA', marker='^', linestyle='None')

# Optional: plot a power-law guide
x_line = np.logspace(2, 7, 100)
y_line = x_line**(-2.5)
y_line = y_line / y_line[0]  # normalize to fit in range
plt.loglog(x_line, y_line * y_btc[0], 'k--', label=r'$x^{-5/2}$')

plt.xlabel("LO Sizes ($)")
plt.ylabel("Cumulative Density")
plt.title("Cumulative Distribution of Limit Order Sizes")
plt.legend()
```

```
plt.grid(True, which="both", ls="--", lw=0.5)
plt.tight_layout()
plt.show()
```