vTokens

January 2, 2022

1 Venus vToken Deposits and Withdrawals

Account Behavior in Discrete Time Periods

[17]:
```
import os
import pandas as pd
import dataframe_image as dfi
from dotenv import load_dotenv
from duneanalytics import DuneAnalytics
from matplotlib import ticker
from matplotlib import pyplot as plt

load_dotenv()
DUNE_UN = os.getenv('DUNE_UN')
DUNE_PW = os.getenv('DUNE_PW')

# initialize client
dune = DuneAnalytics(DUNE_UN, DUNE_PW)
dune.login()
dune.fetch_auth_token()
```

1.1 vToken <> Collateral Token Transfer Count Comparison

[18]:
```
data = [
    ['vBTC', 'BTCB', 311563, 264254]
    ,['vUSDC', 'USDC', 290068, 291524]
]
df = pd.DataFrame(data, columns = ['vToken', 'Collateral Token', 'Deposit␣
 ↪Count', 'Withdrawal Count'])
df.style

dfi.export(df, './assets/transfer-count.png')
```

```
[0102/213920.401835:ERROR:xattr.cc(63)] setxattr
org.chromium.crashpad.database.initialized on file
/var/folders/pt/m1djk0z97h95d0v2hlvvmpt80000gn/T/: Operation not permitted (1)
[0102/213920.402598:ERROR:file_io.cc(94)] ReadExactly: expected 8, observed 0
[0102/213920.403120:ERROR:xattr.cc(63)] setxattr
```

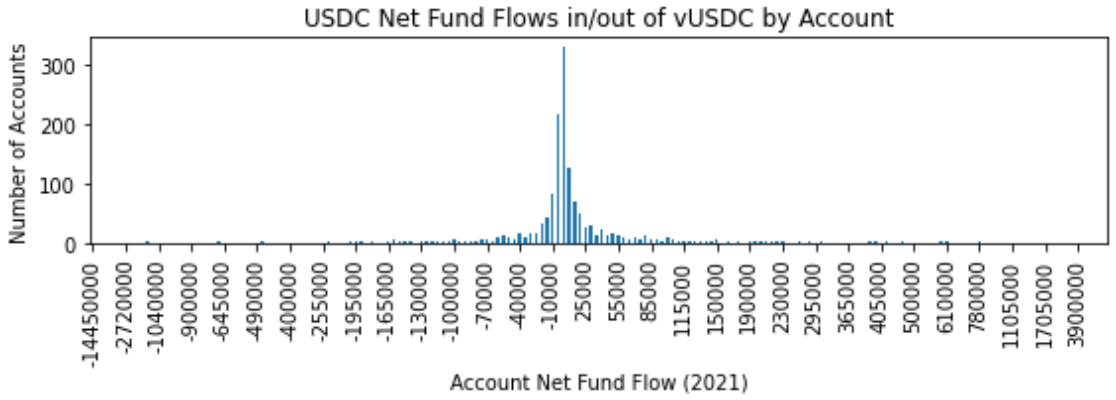
```
org.chromium.crashpad.database.initialized on file
/var/folders/pt/m1djk0z97h95d0v2hlvvmpt80000gn/T/: Operation not permitted (1)
[0102/213921.448795:INFO:headless_shell.cc(653)] Written to file
/var/folders/pt/m1djk0z97h95d0v2hlvvmpt80000gn/T/tmptq085dlf/temp.png.
```

1.2 Full Year Deposit/Withdraw Net Distribution (USDC)

[2]:
```
%%capture output
# fetch query result id using query id & fetch query result
result_id = dune.query_result_id(query_id=326955)
data = dune.query_result(result_id)
result_data = data['data']['get_result_by_result_id']
frame_data = []
for d in result_data:
    frame_data.append(d['data'])
```

[3]:
```
import pandas as pd
df = pd.DataFrame(frame_data, columns =␣
 ↪data['data']['query_results'][0]['columns'])
```

[13]:
```
plt.rcParams["figure.figsize"] = [8, 3]
plt.rcParams["figure.autolayout"] = True

all_data = []
for d in frame_data:
    if d['net'] is not None:
        net_rounded = round(d['net']/5000)*5000
        if net_rounded != 0:
            all_data.append(net_rounded)
all_data.sort()

if len(all_data) > 0:
    fig, ax = plt.subplots()
    fig.patch.set_facecolor('white')
    fig.patch.set_alpha(1)

    df = pd.DataFrame({'numbers': all_data})
    df['numbers'].value_counts().sort_index().plot(ax=ax, kind='bar',␣
 ↪xlabel='Account Net Fund Flow (2021)', ylabel='Number of Accounts')
    ax.xaxis.set_major_locator(ticker.MultipleLocator(6))
    # ax.xaxis.set_major_formatter(ticker.FuncFormatter(lambda x, p:␣
 ↪format(int(x), ',')))
    # ax.xaxis.set_major_formatter(ticker.StrMethodFormatter('{x:,.0f}'))
    plt.title(f"USDC Net Fund Flows in/out of vUSDC by Account")
    plt.show()
    fig.savefig('./assets/net-fundflow-accountdist-usdc-2021')
```



[101]:
```
%%capture output
# fetch query result id using query id & fetch query result
result_id = dune.query_result_id(query_id=326786) #vBTC Dune query
data = dune.query_result(result_id)
result_data = data['data']['get_result_by_result_id']
frame_data = []
for d in result_data:
    frame_data.append(d['data'])
```

[102]:
```
df = pd.DataFrame(frame_data, columns =␣
 ↪data['data']['query_results'][0]['columns'])
```

[103]:
```
df.query('withdraw > 0 and deposit <= 0')
```

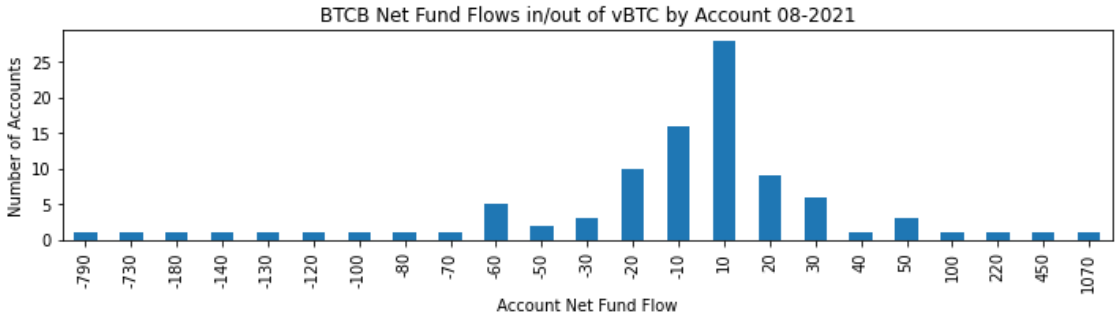
[103]:

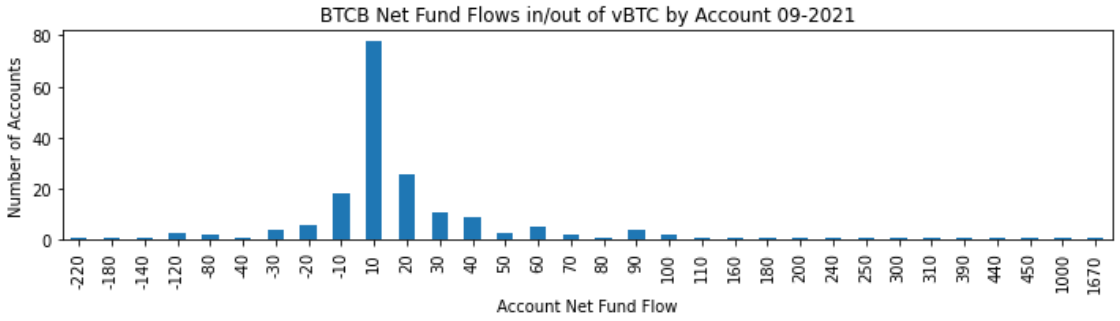
	month	account	deposit	withdraw	\
1766	08-21	\xa8c50e9f552886612109fe27cb94111a2f8006de	0.0	0.226897	
4460	09-21	\xa8c50e9f552886612109fe27cb94111a2f8006de	0.0	0.002745	
7542	10-21	\xa8c50e9f552886612109fe27cb94111a2f8006de	0.0	0.018529	
10465	11-21	\xa8c50e9f552886612109fe27cb94111a2f8006de	0.0	0.086873	

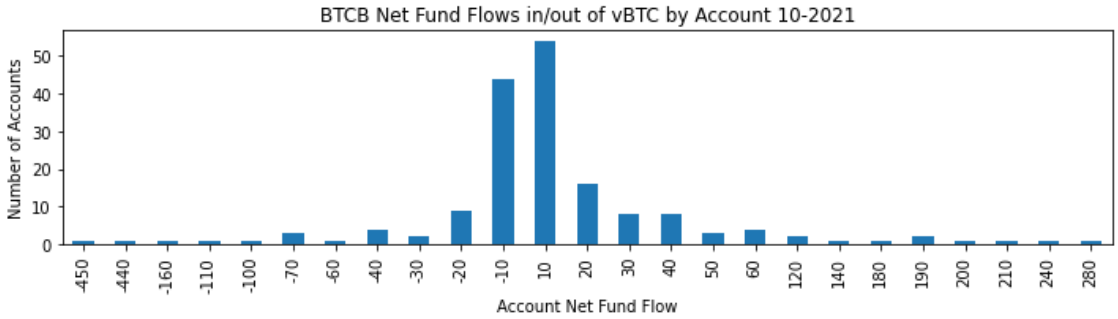
	net	net_abs
1766	0	0
4460	0	0
7542	0	0
10465	0	0

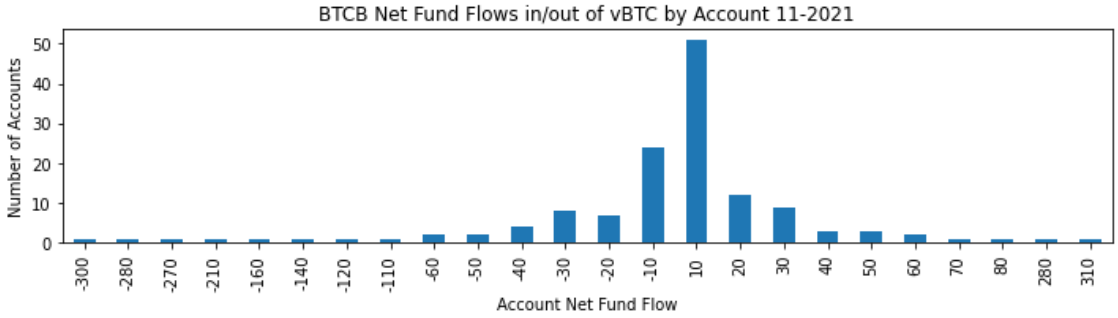
[108]:
```
plt.rcParams["figure.figsize"] = [10, 3]
plt.rcParams["figure.autolayout"] = True

for m in range(1,13):
    month_data = []
    for d in frame_data:
        if d['month'] == f"{m:02d}-21" and d['net'] is not None:
```

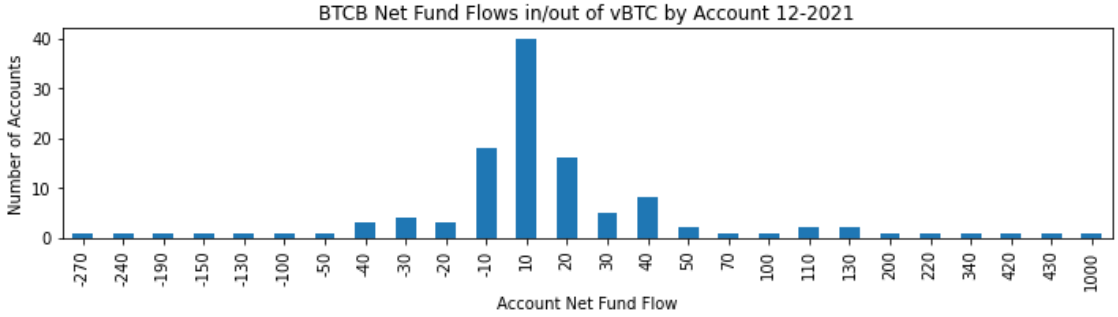
```
            net_rounded = round(d['net']/10)*10
            if net_rounded != 0:
                month_data.append(net_rounded)
    month_data.sort()

    if len(month_data) > 0:
        fig, ax = plt.subplots()
        fig.patch.set_facecolor('white')
        fig.patch.set_alpha(1)

        df = pd.DataFrame({'numbers': month_data})
        df['numbers'].value_counts().sort_index().plot(ax=ax, kind='bar',␣
↪xlabel='Account Net Fund Flow', ylabel='Number of Accounts')
        ax.xaxis.set_major_locator(ticker.MultipleLocator(1))
        # ax.xaxis.set_major_formatter(ticker.FuncFormatter(lambda x, p:␣
↪format(int(x), ',')))
        # ax.xaxis.set_major_formatter(ticker.StrMethodFormatter('{x:,.0f}'))
        plt.title(f"BTCB Net Fund Flows in/out of vBTC by Account {m:02d}-2021")
        plt.show()
```







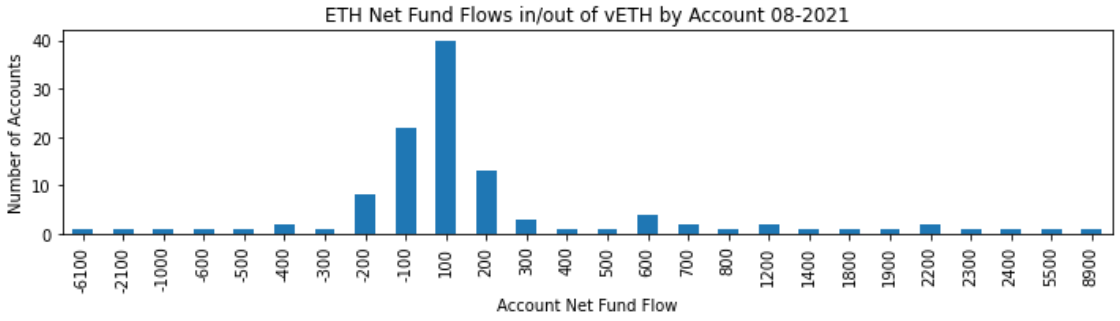


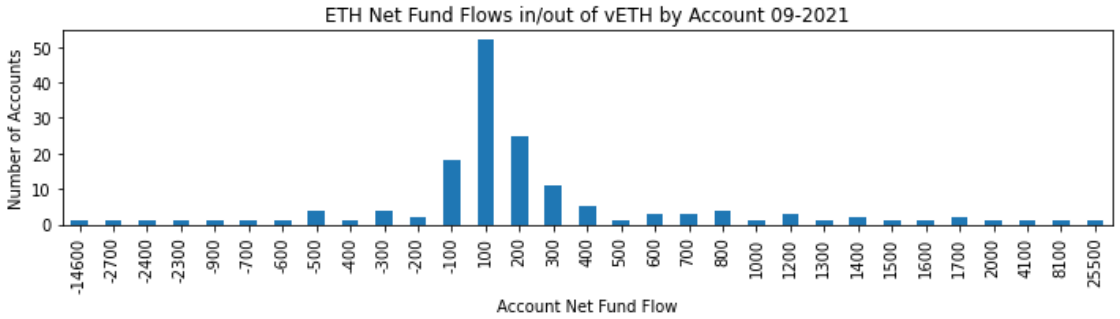


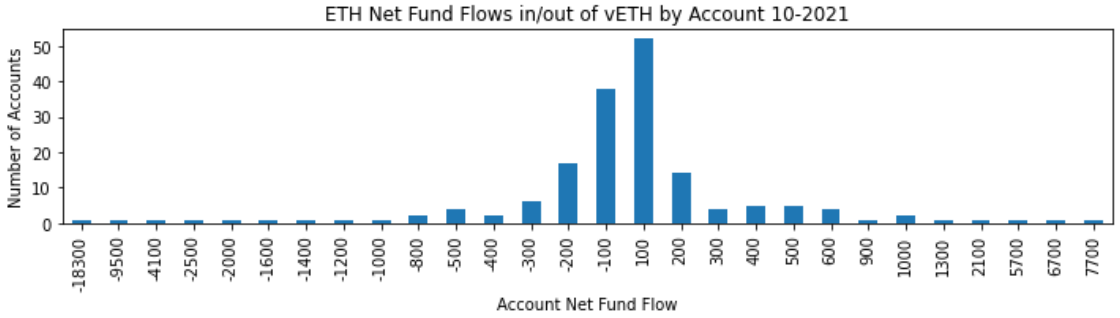
[86]:
```
%%capture output
# fetch query result id using query id & fetch query result
result_id = dune.query_result_id(query_id=326631) #vETH Dune query
data = dune.query_result(result_id)
result_data = data['data']['get_result_by_result_id']
frame_data = []
for d in result_data:
    frame_data.append(d['data'])
```

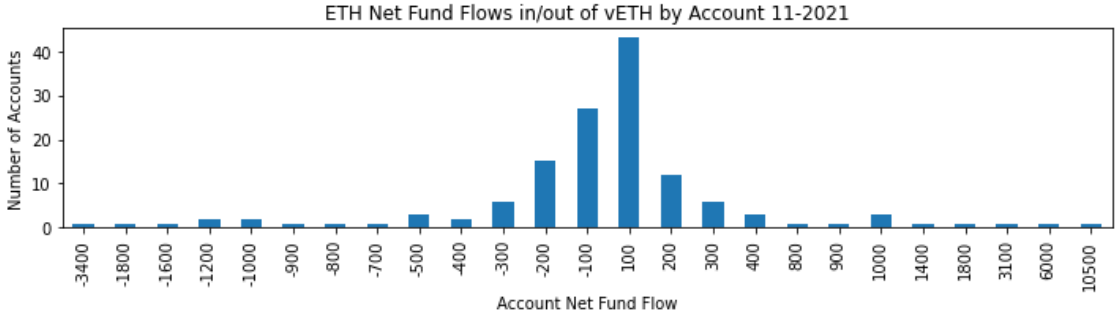
[87]:
```
df = pd.DataFrame(frame_data, columns =␣
↪data['data']['query_results'][0]['columns'])
```

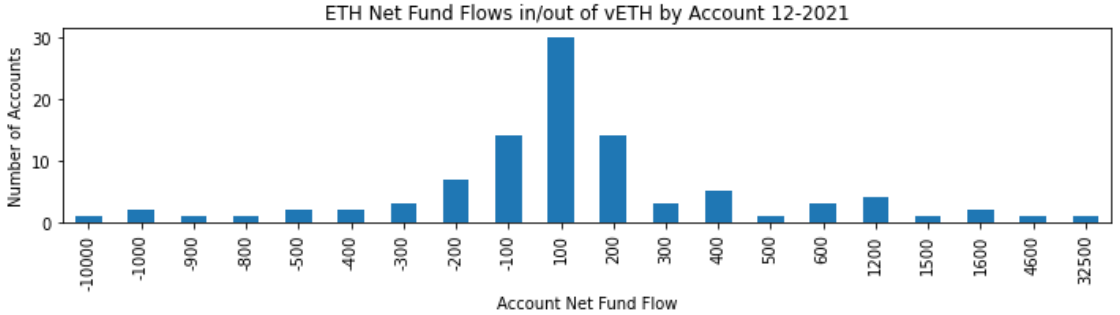
[88]:
```
plt.rcParams["figure.figsize"] = [10, 3]
plt.rcParams["figure.autolayout"] = True

for m in range(1,13):
    month_data = []
    for d in frame_data:
        if d['month'] == f"{m:02d}-21" and d['net'] is not None:
            net_rounded = round(d['net']/100)*100
            if net_rounded != 0:
                month_data.append(net_rounded)
    month_data.sort()

    if len(month_data) > 0:
        fig, ax = plt.subplots()
        fig.patch.set_facecolor('white')
        fig.patch.set_alpha(1)

        df = pd.DataFrame({'numbers': month_data})
        df['numbers'].value_counts().sort_index().plot(ax=ax, kind='bar',␣
↪xlabel='Account Net Fund Flow', ylabel='Number of Accounts')
        ax.xaxis.set_major_locator(ticker.MultipleLocator(1))
        plt.title(f"ETH Net Fund Flows in/out of vETH by Account {m:02d}-2021")
        plt.show()
```




ETH Net Fund Flows in/out of vETH by Account 09-2021
Number of Accounts
0
10
20
30
40
50
-14600
-2700
-2400
-2300
-900
-700
-600
-500
-400
-300
-200
-100
100
200
300
400
500
600
700
800
1000
1200
1300
1400
1500
1600
1700
2000
4100
8100
25500
Account Net Fund Flow


ETH Net Fund Flows in/out of vETH by Account 10-2021
Number of Accounts
0
10
20
30
40
50
-18300
-9500
-4100
-2500
-2000
-1600
-1400
-1200
-1000
-800
-500
-400
-300
-200
-100
100
200
300
400
500
600
900
1000
1300
2100
5700
6700
7700
Account Net Fund Flow


ETH Net Fund Flows in/out of vETH by Account 11-2021
Number of Accounts
0
10
20
30
40
-3400
-1800
-1600
-1200
-1000
-900
-800
-700
-500
-400
-300
-200
-100
100
200
300
400
800
900
1000
1400
1800
3100
6000
10500
Account Net Fund Flow


ETH Net Fund Flows in/out of vETH by Account 12-2021
Number of Accounts
0
10
20
30
-10000
-1000
-900
-800
-500
-400
-300
-200
-100
100
200
300
400
500
600
1200
1500
1600
4600
32500
Account Net Fund Flow

[ ]:
```
%%capture output
# fetch query result id using query id & fetch query result
result_id = dune.query_result_id(query_id=326728) #vBNB Dune query
data = dune.query_result(result_id)
result_data = data['data']['get_result_by_result_id']
frame_data = []
for d in result_data:
    frame_data.append(d['data'])
```

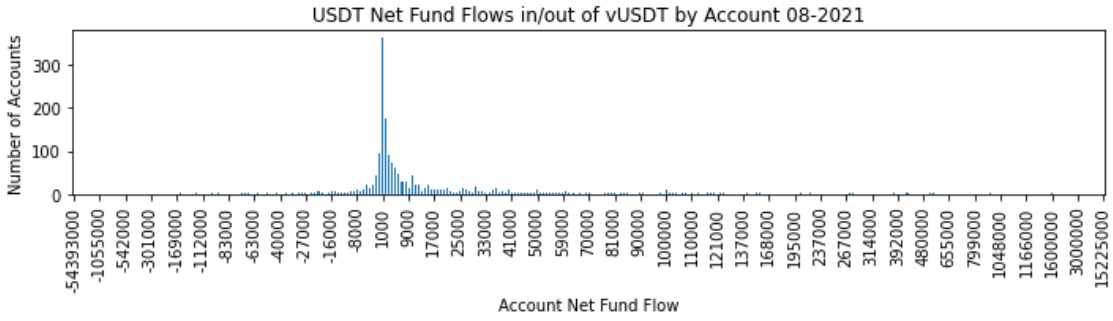
[ ]:
```
df = pd.DataFrame(frame_data, columns =␣
↪data['data']['query_results'][0]['columns'])
```

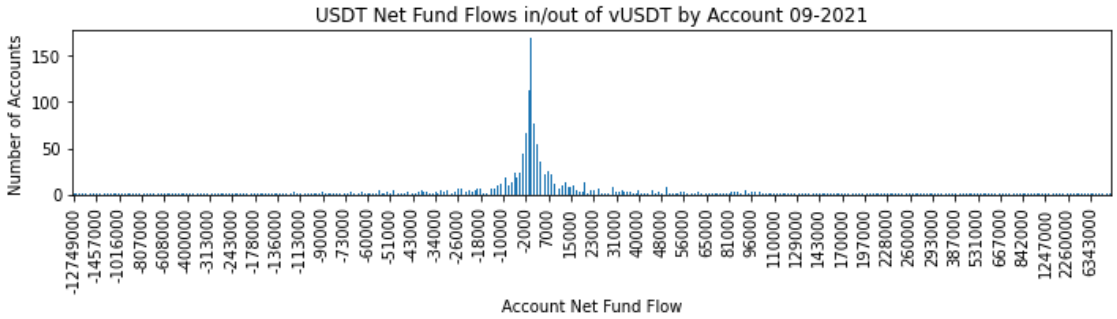
[ ]:
```
plt.rcParams["figure.figsize"] = [10, 3]
plt.rcParams["figure.autolayout"] = True

for m in range(1,13):
    month_data = []
    for d in frame_data:
        if d['month'] == f"{m:02d}-21" and d['net'] is not None:
            net_rounded = round(d['net']/1000)*1000
            if net_rounded != 0:
                month_data.append(net_rounded)
    month_data.sort()

    if len(month_data) > 0:
        fig, ax = plt.subplots()
        fig.patch.set_facecolor('white')
        fig.patch.set_alpha(1)

        df = pd.DataFrame({'numbers': month_data})
        df['numbers'].value_counts().sort_index().plot(ax=ax, kind='bar',␣
↪xlabel='Account Net Fund Flow', ylabel='Number of Accounts')
        ax.xaxis.set_major_locator(ticker.MultipleLocator(8))
        plt.title(f"BNB Net Fund Flows in/out of vBNB by Account {m:02d}-2021")
        plt.show()
```

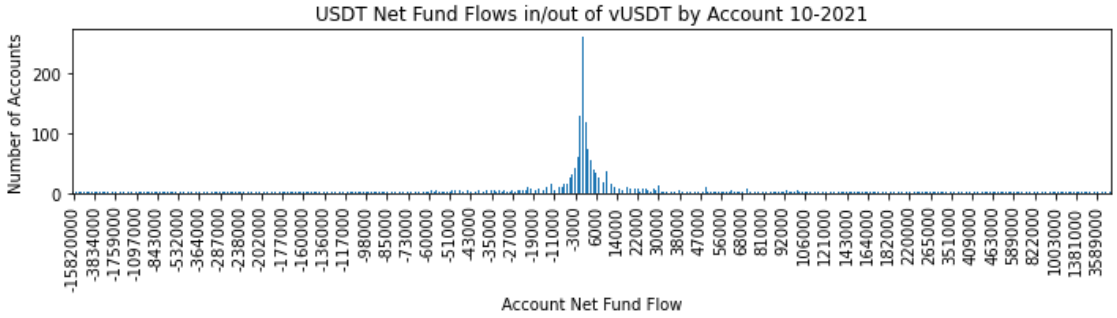
[89]:
```
%%capture output
# fetch query result id using query id & fetch query result
result_id = dune.query_result_id(query_id=326728) #vUSDT Dune query
data = dune.query_result(result_id)
result_data = data['data']['get_result_by_result_id']
frame_data = []
for d in result_data:
    frame_data.append(d['data'])
```

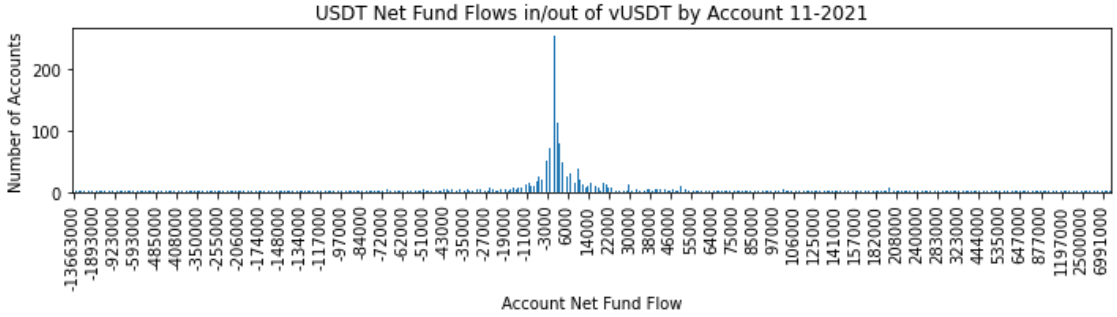
[90]:
```
df = pd.DataFrame(frame_data, columns =␣
↪data['data']['query_results'][0]['columns'])
```

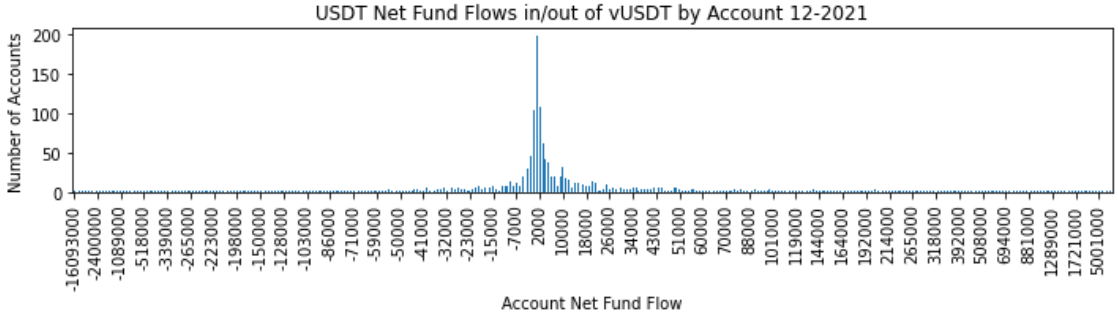
[94]:
```
plt.rcParams["figure.figsize"] = [10, 3]
plt.rcParams["figure.autolayout"] = True

for m in range(1,13):
    month_data = []
    for d in frame_data:
        if d['month'] == f"{m:02d}-21" and d['net'] is not None:
            net_rounded = round(d['net']/1000)*1000
            if net_rounded != 0:
                month_data.append(net_rounded)
    month_data.sort()

    if len(month_data) > 0:
        fig, ax = plt.subplots()
        fig.patch.set_facecolor('white')
        fig.patch.set_alpha(1)

        df = pd.DataFrame({'numbers': month_data})
        df['numbers'].value_counts().sort_index().plot(ax=ax, kind='bar',␣
↪xlabel='Account Net Fund Flow', ylabel='Number of Accounts')
        ax.xaxis.set_major_locator(ticker.MultipleLocator(8))
        plt.title(f"USDT Net Fund Flows in/out of vUSDT by Account {m:02d}-2021")
        plt.show()
```




USDT Net Fund Flows in/out of vUSDT by Account 09-2021
Number of Accounts
Account Net Fund Flow


USDT Net Fund Flows in/out of vUSDT by Account 10-2021
Number of Accounts
Account Net Fund Flow


USDT Net Fund Flows in/out of vUSDT by Account 11-2021
Number of Accounts
Account Net Fund Flow


USDT Net Fund Flows in/out of vUSDT by Account 12-2021
Number of Accounts
Account Net Fund Flow

[ ]:
```
%%capture output
# fetch query result id using query id & fetch query result
result_id = dune.query_result_id(query_id=323328)
data = dune.query_result(result_id)
result_data = data['data']['get_result_by_result_id']
frame_data = []
for d in result_data:
    frame_data.append(d['data'])
```

[ ]:
```
df = pd.DataFrame(frame_data, columns =␣
↪data['data']['query_results'][0]['columns'])
```

[ ]:
```
plt.rcParams["figure.figsize"] = [10, 3]
plt.rcParams["figure.autolayout"] = True

for m in range(1,13):
    month_data = []
    for d in frame_data:
        if d['month'] == f"{m:02d}-21" and d['net'] is not None:
            net_rounded = round(d['net']/1000)*1000
            if net_rounded != 0:
                month_data.append(net_rounded)
    month_data.sort()

    if len(month_data) > 0:
        fig, ax = plt.subplots()
        fig.patch.set_facecolor('white')
        fig.patch.set_alpha(1)

        df = pd.DataFrame({'numbers': month_data})
        df['numbers'].value_counts().sort_index().plot(ax=ax, kind='bar',␣
↪xlabel='Account Net Fund Flow', ylabel='Number of Accounts')
        ax.xaxis.set_major_locator(ticker.MultipleLocator(8))
        plt.title(f"USDC Net Fund Flows in/out of vUSDC by Account {m:02d}-2021")
        plt.show()
```