This notebook demonstrates basic functionality offered by the Coin Metrics Python API Client and **Market Data Feed.**

Coin Metrics offers a vast assortment of data for hundreds of cryptoassets. The Python API Client allows for easy access to this data using Python without needing to create your own wrappers using `requests` and other such libraries.

Resources

To understand the data that Coin Metrics offers, feel free to peruse the resources below.

- The Coin Metrics API v4 website contains the full set of endpoints and data offered by Coin Metrics.
- The Coin Metrics Knowledge Base gives detailed, conceptual explanations of the data that Coin Metrics offers.
- The API Spec contains a full list of functions.

Notebook Setup

In [1]:
```
import os
from os import environ
import sys
import pandas as pd
import numpy as np
import seaborn as sns
import logging
from datetime import date, datetime, timedelta
from coinmetrics.api_client import CoinMetricsClient
import json
import logging
from pytz import timezone as timezone_conv
from datetime import timezone as timezone_info
import matplotlib.ticker as mticker
from matplotlib.ticker import ScalarFormatter
from matplotlib.ticker import FuncFormatter
from matplotlib.dates import DateFormatter
import matplotlib.pyplot as plt
import matplotlib.dates as mdates
%matplotlib inline
```

In [2]:
```
sns.set_theme()
sns.set(rc={'figure.figsize':(12,8)})
```

In [3]:
```
logging.basicConfig(
    format='%(asctime)s %(levelname)-8s %(message)s',
    level=logging.INFO,
    datefmt='%Y-%m-%d %H:%M:%S'
)
now = datetime.utcnow()
last_day_date_time = now - timedelta(hours = 24)
```

In [4]:
```
# We recommend privately storing your API key in your local environment.
try:
    api_key = environ["CM_API_KEY"]
    logging.info("Using API key found in environment")
except KeyError:
    api_key = ""
    logging.info("API key not found. Using community client")

client = CoinMetricsClient(api_key)
```

```
2024-01-26 12:46:56 INFO     Using API key found in environment
```

Futures Catalog

Futures contracts are standardized contracts that allow counterparties to enter into an agreement to buy or sell a standardized asset under contract specifications that are defined by the exchange. Each specific futures contract offered by a specific exchange will have identical contract specifications regardless of who is the counterparty.

The contract specifications include information such as the underlying base and quote asset, the margin asset, the contract size, the listing time, expiration time, and other terms.

Coin Metrics offers contract specifications for both futures and options. Here we define futures to include both non-perpetual futures that expire and perpetual futures (sometimes called perpetual swaps).

In [5]:
```
market_catalog = client.catalog_markets(
    market_type='future',
).to_dataframe()
```

In [6]:
```
print('Total number of supported futures markets: ' + str(len(market_catalog)))
```

```
Total number of supported futures markets: 15958
```

In [7]:
```
# Perpetual futures markets are any futures market with null expiration
print('Total number of perpetual futures markets: ' + str(len(market_catalog.loc[market_catalog['expiration'].isna()])))
```

```
Total number of perpetual futures markets: 3287
```

In [8]:
```
# Filter by base or quote asset
print('Total number of supported BTC futures markets: ' + str(len(market_catalog.loc[market_catalog['base'] == 'btc'])))
```

```
Total number of supported BTC futures markets: 2674
```

In [9]:
```
# Select first BTC futures market as an example
market_catalog.loc[market_catalog['base'] == 'btc'].iloc[0]
```

Out[9]:
```
market                                          binance-BTCBUSD-future
min_time                                       2021-01-11 16:00:00+00:00
max_time                                       2023-12-11 05:32:00+00:00
exchange                                                         binance
type                                                              future
orderbooks                {'min_time': '2021-08-18T16:07:20.000000000Z',...
quotes                    {'min_time': '2021-08-18T16:07:20.000000000Z',...
base                                                                 btc
quote                                                               busd
symbol                                                           BTCBUSD
size_asset                                                           btc
margin_asset                                                        busd
contract_size                                                          1
tick_size                                                            0.1
listing                                        2021-01-11 08:00:00+00:00
order_amount_increment                                               0.001
order_amount_min                                                     0.001
order_amount_max                                                       500
order_price_increment                                                  0.1
order_price_min                                                      557.6
order_price_max                                                    4529890
order_size_min                                                           5
expiration                                                             NaT
order_taker_fee                                                        <NA>
order_maker_fee                                                        <NA>
status                                                                 <NA>
min_time_trades                          2021-01-12 07:00:55.912000+00:00
max_time_trades                          2023-12-11 05:30:08.211000+00:00
min_time_funding_rates                         2021-01-11 16:00:00+00:00
max_time_funding_rates                         2023-12-11 00:00:00+00:00
min_time_openinterest                    2021-01-12 07:15:48.339000+00:00
max_time_openinterest                          2023-12-11 05:32:00+00:00
min_time_liquidations                    2021-01-12 07:32:41.554000+00:00
max_time_liquidations                    2023-12-11 02:13:30.958000+00:00
Name: 128, dtype: object
```

Open Interest

Open interest represents the number of contracts that are currently outstanding and not settled for a specific derivatives market.

Open Interest is available at various levels:

- Assets level (i.e btc)
- Asset Pair level (i.e. btc-usd)
- Exchange level (i.e. binance)
- Exchange-Asset level (i.e. binance-btc)
- Market level (i.e. binance-BTCUSDT-future)

BTC Open Interest at the Market Level

In [10]:
```
binance_btcusdt_oi = client.get_market_open_interest(
    markets = 'binance-BTCUSDT-future',
    start_time = datetime.utcnow() - timedelta(days=1),
).to_dataframe()
```

In [11]:
```
binance_btcusdt_oi
```

Out[11]:

	market	time	contract_count	value_usd	database_time	exchange_time
0	binance-BTCUSDT-future	2024-01-25 17:47:00+00:00	78408.64	3120656031.136	2024-01-25 17:47:48.330257+00:00	2024-01-25 17:47:00+00:00
1	binance-BTCUSDT-future	2024-01-25 17:48:00+00:00	78383.23	3118594380.395	2024-01-25 17:48:34.432171+00:00	2024-01-25 17:48:00+00:00
2	binance-BTCUSDT-future	2024-01-25 17:49:00+00:00	78373.068	3120619635.09	2024-01-25 17:49:27.972942+00:00	2024-01-25 17:49:00+00:00
3	binance-BTCUSDT-future	2024-01-25 17:50:00+00:00	78396.74	3120056977.542	2024-01-25 17:50:53.304366+00:00	2024-01-25 17:50:00+00:00
4	binance-BTCUSDT-future	2024-01-25 17:51:00+00:00	78393.619	3120497201.1045	2024-01-25 17:51:39.631374+00:00	2024-01-25 17:51:00+00:00
...	...	...	...	...	...	...
1435	binance-BTCUSDT-future	2024-01-26 17:42:00+00:00	79164.134	3318275506.3976	2024-01-26 17:42:39.040577+00:00	2024-01-26 17:42:00+00:00
1436	binance-BTCUSDT-future	2024-01-26 17:43:00+00:00	79225.411	3319544720.9	2024-01-26 17:43:26.140326+00:00	2024-01-26 17:43:00+00:00
1437	binance-BTCUSDT-future	2024-01-26 17:44:00+00:00	79242.427	3320376554.9405	2024-01-26 17:44:27.333970+00:00	2024-01-26 17:44:00+00:00
1438	binance-BTCUSDT-future	2024-01-26 17:45:00+00:00	79249.821	3320068226.0277	2024-01-26 17:45:44.252456+00:00	2024-01-26 17:45:00+00:00
1439	binance-BTCUSDT-future	2024-01-26 17:46:00+00:00	79282.988	3321164367.32	2024-01-26 17:46:30.644799+00:00	2024-01-26 17:46:00+00:00

1440 rows × 6 columns

BTC Open Interest by Exchange (Exchange-Asset Endpoint)

In [12]:
```
oi_catalog = client.catalog_exchange_assets().to_dataframe()
oi_catalog = oi_catalog.loc[oi_catalog['metric']=='open_interest_reported_future_usd']
oi_catalog = oi_catalog.loc[oi_catalog['frequency']=='1d']
oi_catalog = oi_catalog[oi_catalog['exchange_asset'].str.split('-').str[1] == 'btc']
oi_catalog
```

Out[12]:

	exchange_asset	metric	frequency	min_time	max_time
3351	binance-btc	open_interest_reported_future_usd	1d	2020-07-27 18:00:00+00:00	2024-01-26 17:00:00+00:00
16121	bitfinex-btc	open_interest_reported_future_usd	1d	2020-07-27 18:00:00+00:00	2024-01-26 17:00:00+00:00
20243	bitmex-btc	open_interest_reported_future_usd	1d	2020-07-27 18:00:00+00:00	2024-01-26 17:00:00+00:00
27422	bybit-btc	open_interest_reported_future_usd	1d	2021-05-01 20:00:00+00:00	2024-01-26 17:00:00+00:00
40920	cme-btc	open_interest_reported_future_usd	1d	2017-12-19 22:00:00+00:00	2024-01-26 17:00:00+00:00
43001	deribit-btc	open_interest_reported_future_usd	1d	2020-07-27 18:00:00+00:00	2024-01-26 17:00:00+00:00
44669	ftx-btc	open_interest_reported_future_usd	1d	2020-07-27 18:00:00+00:00	2022-11-19 03:00:00+00:00
56384	huobi-btc	open_interest_reported_future_usd	1d	2020-07-27 18:00:00+00:00	2024-01-26 17:00:00+00:00
65344	kraken-btc	open_interest_reported_future_usd	1d	2020-10-09 09:00:00+00:00	2024-01-26 17:00:00+00:00
76060	okex-btc	open_interest_reported_future_usd	1d	2020-07-27 18:00:00+00:00	2024-01-26 17:00:00+00:00

In [13]:
```
exchange_assets = oi_catalog['exchange_asset'].to_list()
```

Use the **get_exchange_asset_metrics** client function to pull the all BTC exchange-asset pairs at daily frequency:

In [14]:
```
btc_oi = client.get_exchange_asset_metrics(
    exchange_assets = exchange_assets,
    metrics = 'open_interest_reported_future_usd',
    start_time = datetime.utcnow() - timedelta(days=365),
    frequency = '1d'
).to_dataframe()
```

In [15]:
```
# Convert 'open_interest_reported_future_usd' to numeric
btc_oi['open_interest_reported_future_usd'] = btc_oi['open_interest_reported_future_usd'].astype(np.float64)

# Convert 'time' to datetime
btc_oi['time'] = btc_oi['time'].dt.tz_localize(None).astype('datetime64[ns]')
btc_oi
```

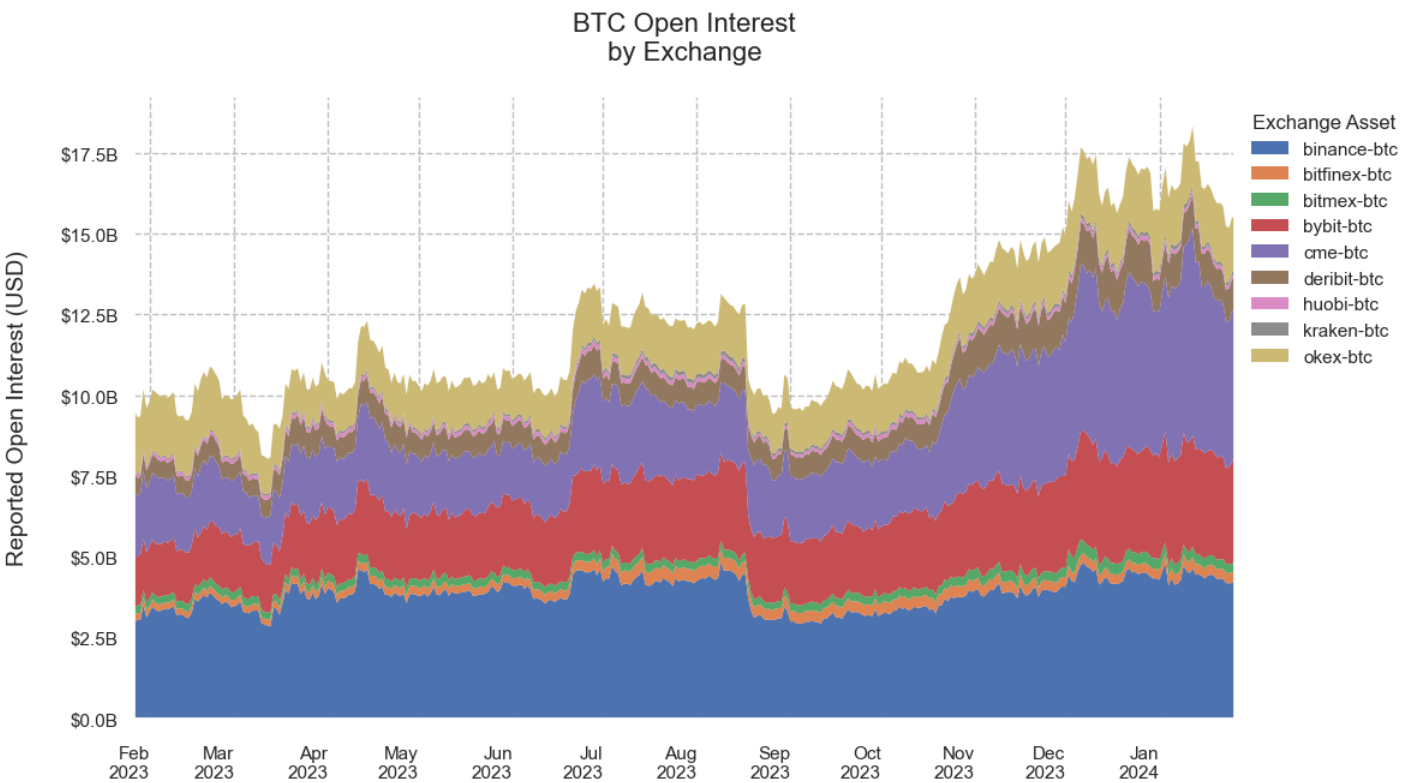
Out[15]:

	exchange_asset	time	open_interest_reported_future_usd
0	binance-btc	2023-01-27	2.982349e+09
1	binance-btc	2023-01-28	3.034175e+09
2	binance-btc	2023-01-29	3.038160e+09
3	binance-btc	2023-01-30	3.502465e+09
4	binance-btc	2023-01-31	3.117494e+09
...	...	...	...
3280	okex-btc	2024-01-22	1.805142e+09
3281	okex-btc	2024-01-23	1.762006e+09
3282	okex-btc	2024-01-24	1.659433e+09
3283	okex-btc	2024-01-25	1.682376e+09
3284	okex-btc	2024-01-26	1.670104e+09

3285 rows × 3 columns

In [16]:
```
# Drop rows with missing data
btc_oi.dropna(inplace=True)
```

In [17]:
```
exchanges = btc_oi['exchange_asset'].unique()
dates = btc_oi['time'].unique()
stacked_data = [btc_oi[btc_oi['exchange_asset'] == exchange]['open_interest_reported_future_usd'].values for exchange in exchanges]

fig, ax = plt.subplots(figsize=(12, 7))
ax.stackplot(dates, stacked_data, labels=exchanges, edgecolor='none')

ax.set_title('\nBTC Open Interest\nby Exchange\n', fontsize=16)
ax.set_xlabel('', fontsize=14)
ax.set_ylabel('Reported Open Interest (USD)\n', fontsize=14)
ax.legend(loc='upper left', title='Exchange Asset', bbox_to_anchor=(1,1), frameon=False)
ax.grid(True, linestyle='--', alpha=0.5, color='gray')
ax.set_facecolor('white')

# Format y-axis in billions of dollars
def billions(x, pos):
    return f'${x * 1e-9:.1f}B'

ax.yaxis.set_major_formatter(FuncFormatter(billions))

ax.set_xlim([btc_oi['time'].min(), btc_oi['time'].max()])
ax.xaxis.set_major_locator(mdates.MonthLocator())
ax.xaxis.set_major_formatter(mdates.DateFormatter('%b\n%Y'))
fig.autofmt_xdate()
ax.tick_params(axis='x', which='major', pad=10)
_ = plt.xticks(rotation=0)
plt.tight_layout()
plt.show()
```



Liquidations

Next, we'll take a look at liquidations data. As a reminder, exchanges which offer futures markets utilize a risk management system that will attempt to close a user's position before the point at which the user begins to owe more than what is in the user's account. The trade or order that closes the user's position is referred to as a liquidation.

This time, we'll use the **get_market_liquidations** client function to pull all BTCUSDT liquidations over the last 24 hours on Binance:

In [18]:
```
market = 'binance-BTCUSDT-future'
```

In [19]:
```
liquidations_df = client.get_market_liquidations(
    markets = market,
    start_time = datetime.utcnow() - timedelta(days=1),
).to_dataframe()
liquidations_df['amount'] = liquidations_df['amount'].astype(np.float64)
liquidations_df['price'] = liquidations_df['price'].astype(np.float64)
```

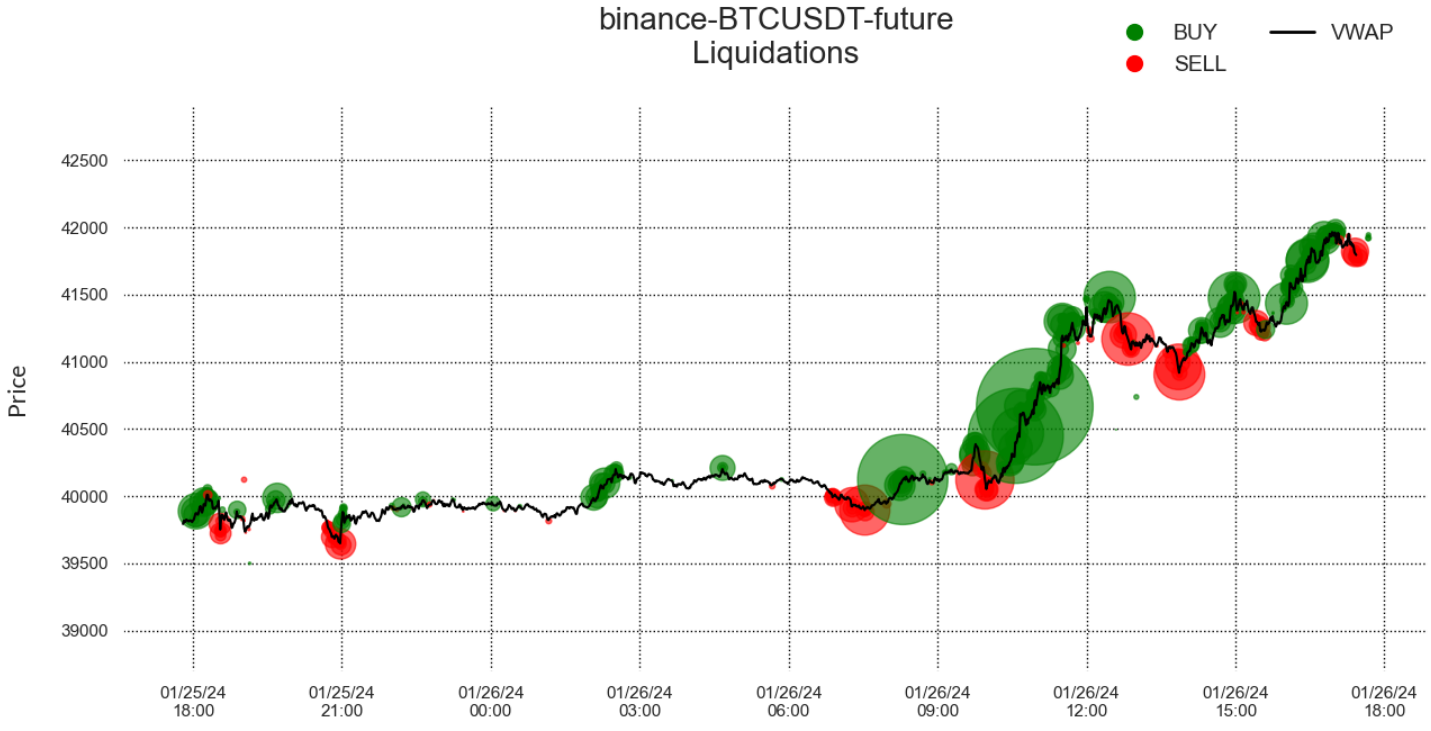
In [20]:
```
liquidations_df.head()
```

Out[20]:

	market	time	coin_metrics_id	amount	price	type	database_time	side
0	binance-BTCUSDT-future	2024-01-25 18:01:41.414000+00:00	1706205701414000000	0.111	39824.0	trade	2024-01-25 18:01:41.931890+00:00	buy
1	binance-BTCUSDT-future	2024-01-25 18:02:59.322000+00:00	1706205779322000000	0.023	39858.6	trade	2024-01-25 18:02:59.756148+00:00	buy
2	binance-BTCUSDT-future	2024-01-25 18:03:00.767000+00:00	1706205780767000000	0.066	39855.4	trade	2024-01-25 18:03:01.432103+00:00	buy
3	binance-BTCUSDT-future	2024-01-25 18:03:02.636000+00:00	1706205782636000000	0.069	39855.4	trade	2024-01-25 18:03:03.200242+00:00	buy
4	binance-BTCUSDT-future	2024-01-25 18:03:25.502000+00:00	1706205805502000000	0.794	39859.3	trade	2024-01-25 18:03:25.664413+00:00	buy

In [21]:
```
# Get volume-weighted average price of the futures contract from the market-candles endpoint
price = client.get_market_candles(
        markets = market,
        start_time = datetime.utcnow() - timedelta(days=1),
        end_time = datetime.utcnow(),
        frequency='1m'
).to_dataframe()
price['vwap'] = price['vwap'].astype(np.float64)
```

In [22]:
```
import matplotlib.pyplot as plt
from matplotlib.dates import DateFormatter

plt.figure(figsize=(13,7))
scaling_factor = 300  # Adjust this value to get the desired point size
color_map = {'buy': 'green', 'sell': 'red'}
liqs = plt.scatter(
    x=liquidations_df['time'],
    y=liquidations_df['price'],
    s=liquidations_df['amount'] * scaling_factor,  # Scale point sizes by the scaling factor
    c=liquidations_df['side'].map(color_map),
    alpha=0.6
)

plt.plot(price['time'], price['vwap'], color='black', linestyle='-', label='VWAP')

mean_price = liquidations_df['price'].mean()
std_price = liquidations_df['price'].std()
plt.ylim(mean_price - 3*std_price, mean_price + 3*std_price)
plt.xlabel("", fontsize=15)
plt.ylabel("Price\n", font='Lato',fontsize=15)
plt.title('\n' + str(market) + '\nLiquidations\n', size=20)

# Format the xtick labels
date_format = DateFormatter('%D\n%H:%M')
plt.gca().xaxis.set_major_formatter(date_format)

legend_labels = ['BUY', 'SELL', 'VWAP']
legend_handles = [
    plt.Line2D([0], [0], marker='o', color='w', markerfacecolor=color_map['buy'], markersize=12),
    plt.Line2D([0], [0], marker='o', color='w', markerfacecolor=color_map['sell'], markersize=12),
    plt.Line2D([0], [0], color='black', lw=2)  # Legend entry for VWAP
]
legend = plt.legend(legend_handles, legend_labels, loc='lower right', fontsize=14, ncol=2, framealpha=0, bbox_to_anchor=(0.99, 1.02))

plt.gca().set_facecolor('white')
plt.grid(color='black', linestyle='dotted')

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



Notice that this timeseries also includes the liquidation **type**. Some exchanges report "liquidations orders" in which they will report the creation of a liquidation **order** when a trader's position initially enters liquidation. When a trader's position enters liquidation, an exchange will typically enter a limit order at the price at which the trader will be bankruptcy price. The liquidation orders will show the amount of the position that is being liquidated and the liquidation price, but will not represent the matched trades that are executed as a result of the liquidation. Other exchanges will report "liquidation trades" which represent the actual matched **trade** as a result of a liquidation order but will not report liquidation orders. Some exchanges will report both liquidation orders and liquidation trades.

Aggregated Liquidation Metrics

In addition to examining individual liquidations, we can also leverage aggregated liquidations metrics. This allows us to quickly view the total amount of USD-denominated liquidations that have occurred over large timeframes, without needing to aggregate the amounts at the trade level.

In [23]:
```
exchange_asset_metrics = client.catalog_exchange_asset_metrics().to_dataframe()
exchange_asset_metrics.loc[exchange_asset_metrics['category'] == 'Liquidations']
```

Out[23]:

	metric	full_name	description	product	category	subcategory	unit	data_type	type	display_name	frequency	exchange_asset
50	liquidations_reported_future_buy_units_1d	Liquidations, reported, future, buys, native u...	The sum of all buy liquidations from perpetual...	Market Data	Liquidations	Futures	Native Units	decimal	Sum	Reported Futures Buy Liquidations, native units	1d	None
51	liquidations_reported_future_buy_units_1h	Liquidations, reported, future, buys, native u...	The sum of all buy liquidations from perpetual...	Market Data	Liquidations	Futures	Native Units	decimal	Sum	Reported Futures Buy Liquidations, native units	1h	None
52	liquidations_reported_future_buy_units_5m	Liquidations, reported, future, buys, native u...	The sum of all buy liquidations from perpetual...	Market Data	Liquidations	Futures	Native Units	decimal	Sum	Reported Futures Buy Liquidations, native units	5m	None
53	liquidations_reported_future_buy_usd_1d	Liquidations, reported, future, buys, USD, one...	The sum of all buy liquidations from perpetual...	Market Data	Liquidations	Futures	USD	decimal	Sum	Reported Futures Buy Liquidations, USD	1d	None
54	liquidations_reported_future_buy_usd_1h	Liquidations, reported, future, buys, USD, one...	The sum of all buy liquidations from perpetual...	Market Data	Liquidations	Futures	USD	decimal	Sum	Reported Futures Buy Liquidations, USD	1h	None
55	liquidations_reported_future_buy_usd_5m	Liquidations, reported, future, buys, USD, fiv...	The sum of all buy liquidations from perpetual...	Market Data	Liquidations	Futures	USD	decimal	Sum	Reported Futures Buy Liquidations, USD	5m	None
56	liquidations_reported_future_sell_units_1d	Liquidations, reported, future, sells, native ...	The sum of all sell liquidations from perpetua...	Market Data	Liquidations	Futures	Native Units	decimal	Sum	Reported Futures Sell Liquidations, native units	1d	None
57	liquidations_reported_future_sell_units_1h	Liquidations, reported, future, sells, native ...	The sum of all sell liquidations from perpetua...	Market Data	Liquidations	Futures	Native Units	decimal	Sum	Reported Futures Sell Liquidations, native units	1h	None
58	liquidations_reported_future_sell_units_5m	Liquidations, reported, future, sells, native ...	The sum of all sell liquidations from perpetua...	Market Data	Liquidations	Futures	Native Units	decimal	Sum	Reported Futures Sell Liquidations, native units	5m	None
59	liquidations_reported_future_sell_usd_1d	Liquidations, reported, future, sells, USD, on...	The sum of all sell liquidations from perpetua...	Market Data	Liquidations	Futures	USD	decimal	Sum	Reported Futures Sell Liquidations, USD	1d	None
60	liquidations_reported_future_sell_usd_1h	Liquidations, reported, future, sells, USD, on...	The sum of all sell liquidations from perpetua...	Market Data	Liquidations	Futures	USD	decimal	Sum	Reported Futures Sell Liquidations, USD	1h	None
61	liquidations_reported_future_sell_usd_5m	Liquidations, reported, future, sells, USD, fi...	The sum of all sell liquidations from perpetua...	Market Data	Liquidations	Futures	USD	decimal	Sum	Reported Futures Sell Liquidations, USD	5m	None

In [24]:
```
metric = ['liquidations_reported_future_buy_usd_1h', 'liquidations_reported_future_sell_usd_1h']
```

In [25]:
```
liq_catalog = client.catalog_exchange_assets().to_dataframe()
liq_catalog = liq_catalog.loc[liq_catalog['metric'] == metric[1]]
liq_catalog = liq_catalog[liq_catalog['exchange_asset'].str.split('-').str[1] == 'btc']
liq_catalog
```

Out[25]:

	exchange_asset	metric	frequency	min_time	max_time
3340	binance-btc	liquidations_reported_future_sell_usd_1h	1h	2019-09-10 19:00:00+00:00	2024-01-26 16:00:00+00:00
16114	bitfinex-btc	liquidations_reported_future_sell_usd_1h	1h	2019-10-03 09:00:00+00:00	2024-01-25 17:00:00+00:00
20232	bitmex-btc	liquidations_reported_future_sell_usd_1h	1h	2020-10-08 07:00:00+00:00	2024-01-26 12:00:00+00:00
27411	bybit-btc	liquidations_reported_future_sell_usd_1h	1h	2021-04-24 22:00:00+00:00	2024-01-26 16:00:00+00:00
42992	deribit-btc	liquidations_reported_future_sell_usd_1h	1h	2019-01-11 04:00:00+00:00	2024-01-18 20:00:00+00:00
44662	ftx-btc	liquidations_reported_future_sell_usd_1h	1h	2019-04-19 01:00:00+00:00	2022-11-12 03:00:00+00:00
56373	huobi-btc	liquidations_reported_future_sell_usd_1h	1h	2020-07-10 11:00:00+00:00	2024-01-26 16:00:00+00:00
65335	kraken-btc	liquidations_reported_future_sell_usd_1h	1h	2020-12-09 18:00:00+00:00	2024-01-26 13:00:00+00:00
76049	okex-btc	liquidations_reported_future_sell_usd_1h	1h	2020-10-01 16:00:00+00:00	2024-01-26 16:00:00+00:00

In [26]:
```
agg_liqs = client.get_exchange_asset_metrics(
    exchange_assets=liq_catalog['exchange_asset'].to_list(),
```

```
    metrics = metric,
    start_time = datetime.utcnow() - timedelta(days=1.5),
    frequency='1h'
).to_dataframe()
agg_liqs.replace('None', np.nan, inplace=True)
agg_liqs[metric[0]] = agg_liqs['liquidations_reported_future_buy_usd_1h'].astype(np.float64)
agg_liqs['liquidations_reported_future_sell_usd_1h'] = -1 * agg_liqs['liquidations_reported_future_sell_usd_1h'].astype(np.float64)
```

In [27]:
```
agg_liqs = agg_liqs.fillna(0)
agg_liqs
```

Out[27]:

	exchange_asset	time	liquidations_reported_future_buy_usd_1h	liquidations_reported_future_sell_usd_1h
0	binance-btc	2024-01-25 06:00:00+00:00	2.070071e+05	-1756.70528
1	binance-btc	2024-01-25 07:00:00+00:00	1.242980e+05	-480.85344
2	binance-btc	2024-01-25 08:00:00+00:00	4.719789e+04	-200.23460
3	binance-btc	2024-01-25 09:00:00+00:00	1.722529e+03	-80.10352
4	binance-btc	2024-01-25 10:00:00+00:00	1.361881e+04	-23326.69604
...	...	...	...	...
152	okex-btc	2024-01-26 12:00:00+00:00	4.558717e+05	-268553.39330
153	okex-btc	2024-01-26 13:00:00+00:00	0.000000e+00	-161005.32750
154	okex-btc	2024-01-26 14:00:00+00:00	1.383449e+06	0.00000
155	okex-btc	2024-01-26 15:00:00+00:00	1.406247e+04	-48698.35360
156	okex-btc	2024-01-26 16:00:00+00:00	2.353262e+06	-2071.03900

157 rows × 4 columns

In [28]:
```
btc_total_oi = client.get_asset_metrics(
    assets='btc',
    metrics='open_interest_reported_future_usd',
    frequency='1h',
    start_time = datetime.utcnow() - timedelta(days=2)
).to_dataframe()
btc_total_oi.head()
```

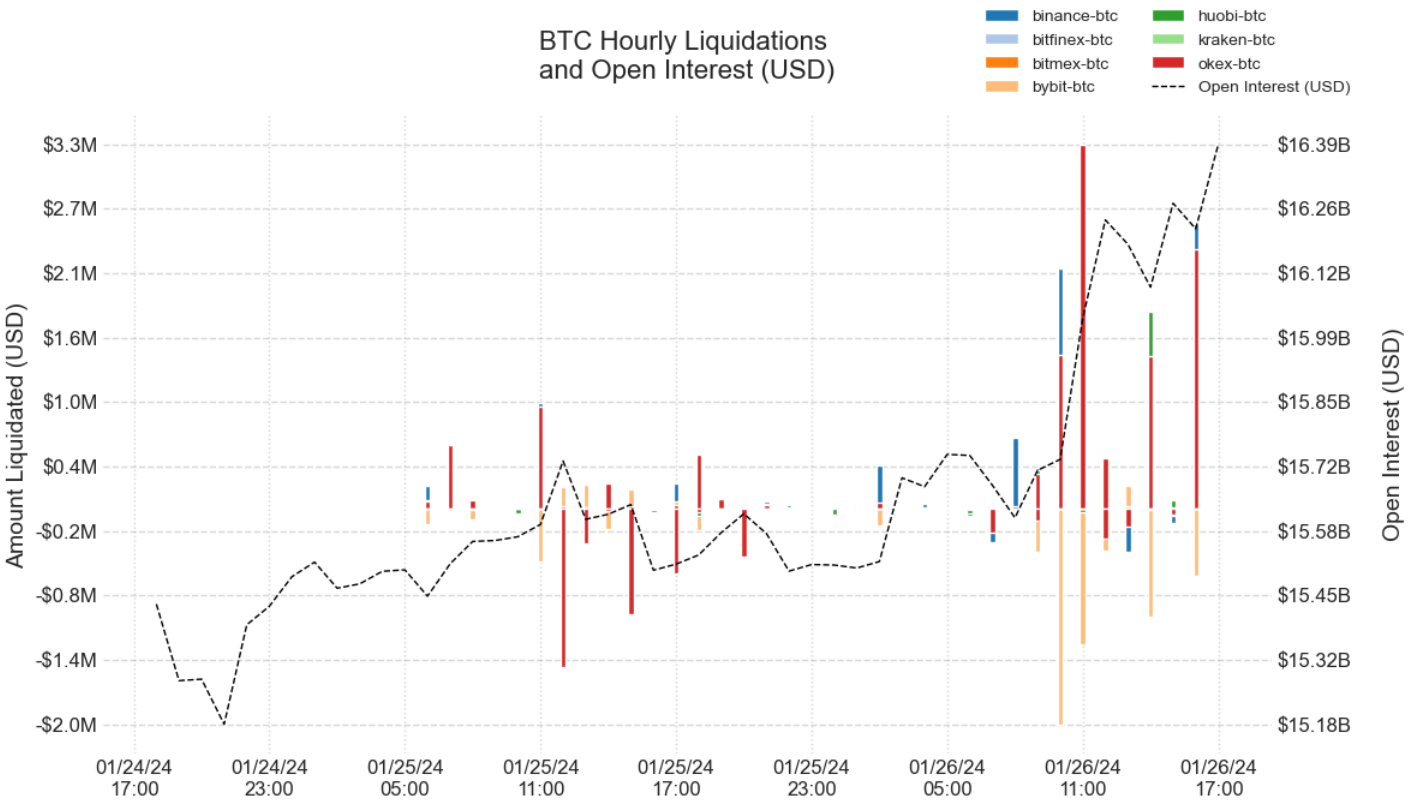
Out[28]:

	asset	time	open_interest_reported_future_usd
0	btc	2024-01-24 18:00:00+00:00	15432260406.7805
1	btc	2024-01-24 19:00:00+00:00	15272586146.772699
2	btc	2024-01-24 20:00:00+00:00	15275673126.749701
3	btc	2024-01-24 21:00:00+00:00	15181638278.4221
4	btc	2024-01-24 22:00:00+00:00	15388999601.331499

In [29]:
```
df = agg_liqs
```

In [30]:
```
melted_df = df.melt(id_vars=['time', 'exchange_asset'],
                    value_vars=['liquidations_reported_future_buy_usd_1h', 'liquidations_reported_future_sell_usd_1h'],
                    var_name='transaction_type', value_name='amount')
melted_df['amount'] /= 1e6
melted_df['time'] = melted_df['time'].dt.tz_localize(None)

fig, ax = plt.subplots(figsize=(12,7))
plt.gca().set_facecolor('white')
plt.grid(color='gray', linestyle='dotted',alpha=0.3)
ax2 = ax.twinx()

unique_assets = melted_df['exchange_asset'].unique()
colormap = plt.cm.tab20
colors = {asset: colormap(i) for i, asset in enumerate(unique_assets)}

for asset in unique_assets:
    subset = melted_df[melted_df['exchange_asset'] == asset]
    ax.bar(subset['time'], subset['amount'], width=0.01, label=asset, color=colors[asset])

# Plot open interest on secondary y-axis
btc_total_oi['time'] = btc_total_oi['time'].dt.tz_localize(None)
ax2.plot(btc_total_oi['time'], btc_total_oi['open_interest_reported_future_usd'], color='black', label='Open Interest (USD)', linewidth=1, linestyle='-

ax.set_xlabel('')
ax.set_ylabel('Amount Liquidated (USD)', fontsize=14)
ax2.set_ylabel('\nOpen Interest (USD)', fontsize=14)
ax.set_title('\nBTC Hourly Liquidations \nand Open Interest (USD)\n', fontsize=16)

locator = mdates.HourLocator(interval=6)
ax.xaxis.set_major_locator(locator)
ax.xaxis.set_major_formatter(mdates.DateFormatter('%D\n%H:%M'))

# Format y-axis ticks for liquidations
def y_formatter(x, pos):
    if x < 0:
        return f"-${abs(x):.1f}M"
    else:
        return f"${x:.1f}M"
```

```
ax.yaxis.set_major_formatter(mticker.FuncFormatter(y_formatter))
ax.yaxis.grid(True, linestyle='--', which='major')
ax2.yaxis.grid(False)
ax.yaxis.tick_left()
ax.tick_params(axis='both', length=0, labelsize=12)
ax2.tick_params(axis='both', length=0, labelsize=12)

# Format y-axis ticks for open interest in billions
def y_formatter_billion(x, pos):
    return f"${x*1e-9:.2f}B"
ax2.yaxis.set_major_formatter(mticker.FuncFormatter(y_formatter_billion))

num_ticks = 10
yticks = np.linspace(melted_df['amount'].min(), melted_df['amount'].max(), num_ticks)
ax.set_yticks(yticks)

# Set y-ticks based on a similar range
yticks2 = np.linspace(btc_total_oi['open_interest_reported_future_usd'].min(),
                      btc_total_oi['open_interest_reported_future_usd'].max(), num_ticks)
ax2.set_yticks(yticks2)

# Legend for both axes
lines, labels = ax.get_legend_handles_labels()
lines2, labels2 = ax2.get_legend_handles_labels()
ax2.legend(lines + lines2, labels + labels2, loc='upper right', fontsize=10, ncol=2, framealpha=0, bbox_to_anchor=(1.08, 1.19))

plt.tight_layout()

for spine in ax2.spines.values():
    spine.set_visible(False)
for spine in ax.spines.values():
    spine.set_visible(False)

plt.show()
```



Funding Rates

Funding rates are a mechanism that exchanges use to ensure that perpetual futures trade at a price that is close to the price of the underlying spot markets. The funding rate is used to calculate the funding fee which long position holders pay short position holders, or vice versa, as a way to incentivize market participants to take positions that keep perpetual futures prices close to the underlying.

Coin Metrics funding rate data from the *timeseries/market-funding-rates* endpoint includes the following fields:

- **market:** The id of the market. Market ids use the following naming convention: exchangeName-baseAsset-quoteAsset-spot for spot markets, exchangeName-futuresSymbol-future for futures markets, and exchangeName-optionsSymbol-option for options markets.
- **time:** The exchange-reported time in ISO 8601 date-time format. Always with nanoseconds precision.
- **rate:** The funding rate expressed as a percentage over the period. For example, if the funding rate is 0.10%, expressed as an 8 hour rate and calculated over the past 8 hours, the rate is 0.0010.
- **period:** The periodicity of the funding rate. If the rate is 0.0010then this rate would be applied every period defined by this field.

- **interval:** The interval of time over which the funding rate is calculated.
- **database_time:** The timestamp when the data was saved in the database in ISO 8601 date-time format with nanoseconds precision.

In [31]:
```
fr_catalog = client.catalog_market_funding_rates_v2(exchange='binance').to_dataframe()
```

In [32]:
```
fr_catalog
```

Out[32]:

	market	min_time	max_time
0	binance-1000BONKUSDT-future	2023-11-22 16:00:00+00:00	2024-01-26 16:00:00+00:00
1	binance-1000BTTCUSDT-future	2022-01-26 08:00:00.001000+00:00	2022-04-11 08:00:00+00:00
2	binance-1000FLOKIUSDT-future	2023-05-06 16:00:00+00:00	2024-01-26 16:00:00+00:00
3	binance-1000LUNCBUSD-future	2022-05-30 16:00:00.005000+00:00	2023-06-08 08:00:00+00:00
4	binance-1000LUNCUSDT-future	2022-09-09 16:00:00+00:00	2024-01-26 16:00:00+00:00
...	...	...	...
360	binance-ZECUSDT-future	2020-02-04 08:00:00.001000+00:00	2024-01-26 16:00:00+00:00
361	binance-ZENUSDT-future	2020-11-24 08:00:00+00:00	2024-01-26 16:00:00+00:00
362	binance-ZILUSDT-future	2020-06-17 08:00:00.007000+00:00	2024-01-26 16:00:00+00:00
363	binance-ZILUSD_PERP-future	2022-04-06 08:00:00.013000+00:00	2022-12-26 08:00:00.014000+00:00
364	binance-ZRXUSDT-future	2020-06-24 08:00:00+00:00	2024-01-26 16:00:00+00:00

365 rows × 3 columns

In [33]:
```
fr_markets = [
    'bitmex-XBTUSD-future',
    'bybit-BTCUSD-future',
    'okex-BTC-USD-SWAP-future'
]
```

In [34]:
```
fr_raw = client.get_market_funding_rates(
    markets = fr_markets,
    start_time=datetime.utcnow() - timedelta(days=7),
).to_dataframe()
fr_raw
```

Out[34]:

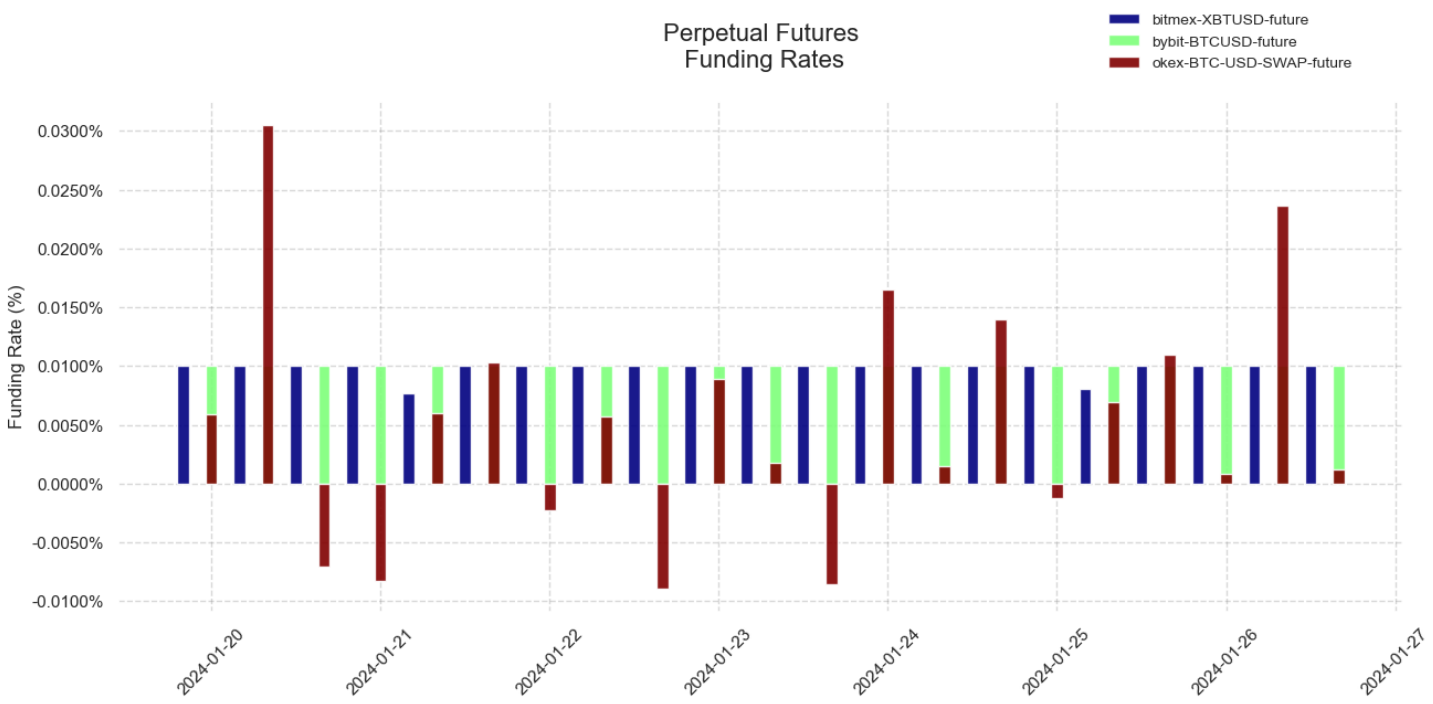
	market	time	database_time	rate	period	interval
0	bitmex-XBTUSD-future	2024-01-19 20:00:00+00:00	2024-01-19 20:00:10.051231+00:00	0.0001	08:00:00	08:00:00
1	bitmex-XBTUSD-future	2024-01-20 04:00:00+00:00	2024-01-20 04:00:24.437200+00:00	0.0001	08:00:00	08:00:00
2	bitmex-XBTUSD-future	2024-01-20 12:00:00+00:00	2024-01-20 12:00:09.707928+00:00	0.0001	08:00:00	08:00:00
3	bitmex-XBTUSD-future	2024-01-20 20:00:00+00:00	2024-01-20 20:00:30.796285+00:00	0.0001	08:00:00	08:00:00
4	bitmex-XBTUSD-future	2024-01-21 04:00:00+00:00	2024-01-21 04:00:44.402878+00:00	0.000077	08:00:00	08:00:00
...	...	...	...	...	...	...
58	okex-BTC-USD-SWAP-future	2024-01-25 08:00:00+00:00	2024-01-25 08:00:22.541996+00:00	0.000069	08:00:00	08:00:00
59	okex-BTC-USD-SWAP-future	2024-01-25 16:00:00+00:00	2024-01-25 16:00:26.739159+00:00	0.00011	08:00:00	08:00:00
60	okex-BTC-USD-SWAP-future	2024-01-26 00:00:00+00:00	2024-01-26 00:00:26.529946+00:00	0.000009	08:00:00	08:00:00
61	okex-BTC-USD-SWAP-future	2024-01-26 08:00:00+00:00	2024-01-26 08:00:27.724600+00:00	0.000236	08:00:00	08:00:00
62	okex-BTC-USD-SWAP-future	2024-01-26 16:00:00+00:00	2024-01-26 16:00:24.087122+00:00	0.000012	08:00:00	08:00:00

63 rows × 6 columns

In [35]:
```
# Convert 'time' to datetime for plotting, if not already in this format
fr_raw['time'] = pd.to_datetime(fr_raw['time'])
fr_raw = fr_raw.sort_values(by='time')
for column in fr_raw.columns:
    if column not in ['time', 'market']:
        fr_raw[column] = pd.to_numeric(fr_raw[column], errors='coerce') * 100
```

In [36]:
```
# Create a color map
markets = fr_raw['market'].unique()
colors = plt.cm.jet(np.linspace(0, 1, len(markets)))  # Generating a color for each market
color_map = dict(zip(markets, colors))

# Plotting
plt.figure(figsize=(15, 6))
plt.gca().set_facecolor('white')
# Plot bars for each market
for market in markets:
    market_data = fr_raw[fr_raw['market'] == market]
    plt.bar(market_data['time'], market_data['rate'], color=color_map[market], label=market, width=0.07, alpha=0.9)

formatter = mticker.FuncFormatter(lambda y, _: '{:.4f}%'.format(y))
plt.gca().yaxis.set_major_formatter(formatter)
plt.grid(True, linestyle='--', which='major', color='gray', alpha=0.3)
plt.xticks(rotation=45)
plt.xlabel('')
plt.ylabel('Funding Rate (%)')
```

```
plt.title('\nPerpetual Futures\n Funding Rates\n',fontsize=16)
plt.legend(loc='upper left', fontsize=10, bbox_to_anchor=(0.76,1.2), frameon=False)

plt.show()
```



Aggregated Funding Rates

Coin Metrics also calculates several aggregated funding rate metrics.

Aggregate Funding Rate is the average funding rate weighted by open interest, published once per hour and representing the average funding rate converted to 8 hour, 1 day, 30 day, and 1 year time periods.

- **futures_aggregate_funding_rate_usd_margin_*:** metrics represent the average funding rate weighted by open interest from perpetual futures markets where the margin asset is U.S. dollars or stablecoins converted to a specified time period.
- **futures_aggregate_funding_rate_coin_margin_*:** represent the average funding rate weighted by open interest from perpetual futures markets where the margin asset is equivalent to the underlying base asset converted to a specified period.
- **futures_aggregate_funding_rate_all_margin_*:** represent the average funding rate weighted by open interest from all perpetual futures markets, regardless of the margin asset, converted to a specified time period.

In [37]:
```
btc_fr = client.get_asset_metrics(
    assets='btc',
    start_time='2023-10-01',
    metrics = [
        'futures_aggregate_funding_rate_all_margin_1d_period',
        'futures_aggregate_funding_rate_all_margin_30d_period',
        'futures_aggregate_funding_rate_all_margin_1y_period'
        ]
).to_dataframe()
```

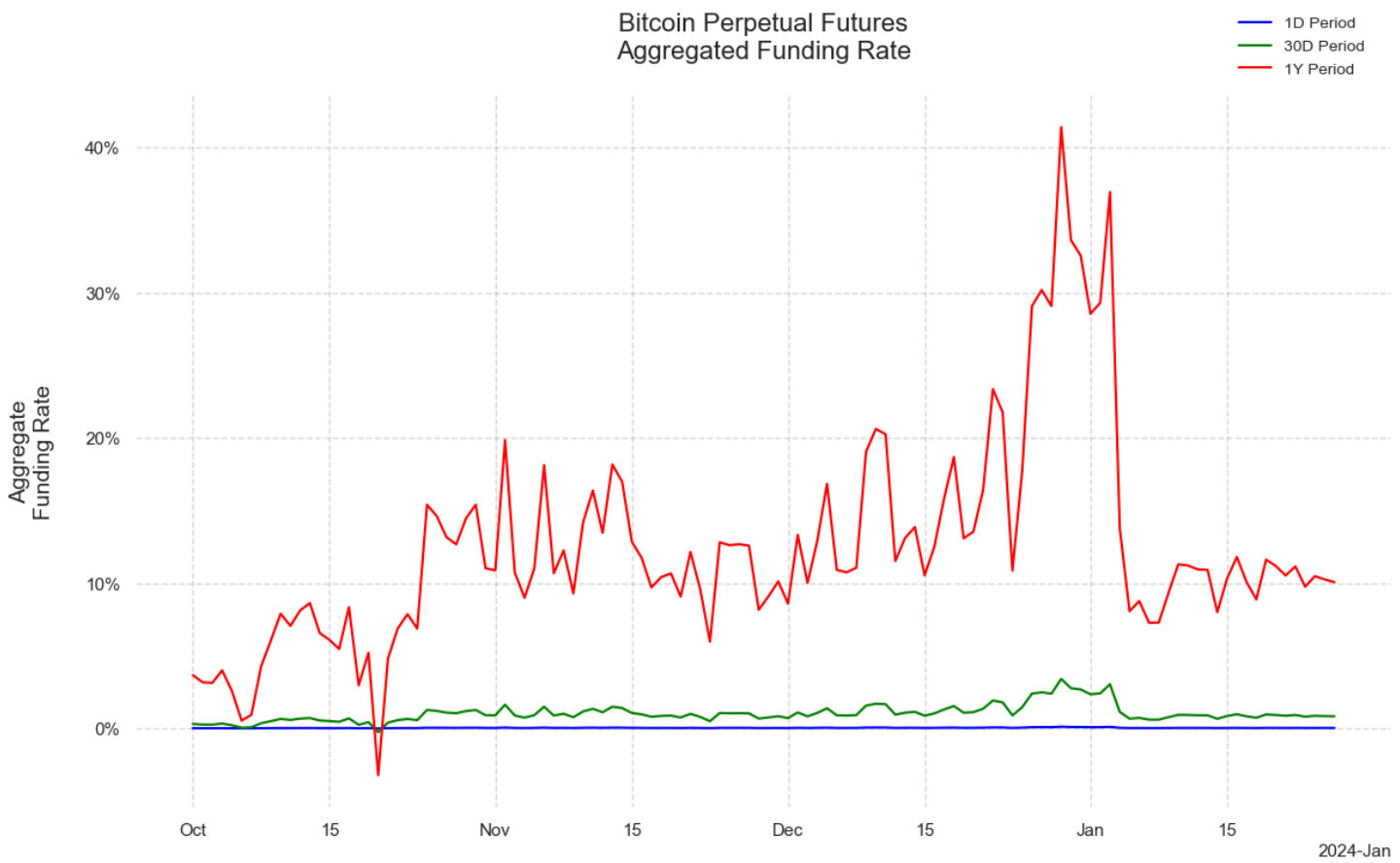
In [38]:
```
btc_fr.head()
```

Out[38]:

	asset	time	futures_aggregate_funding_rate_all_margin_1d_period	futures_aggregate_funding_rate_all_margin_1y_period	futures_aggregate_funding_rate_all_m
0	btc	2023-10-01 00:00:00+00:00	0.000101	0.036711	
1	btc	2023-10-02 00:00:00+00:00	0.000087	0.03172	
2	btc	2023-10-03 00:00:00+00:00	0.000086	0.031318	
3	btc	2023-10-04 00:00:00+00:00	0.000109	0.039945	
4	btc	2023-10-05 00:00:00+00:00	0.000071	0.025916	

In [39]:
```
plt.figure(figsize=(14, 8))

plt.plot(btc_fr['time'], btc_fr['futures_aggregate_funding_rate_all_margin_1d_period'] * 100, label='1D Period', color='blue')
plt.plot(btc_fr['time'], btc_fr['futures_aggregate_funding_rate_all_margin_30d_period'] * 100, label='30D Period', color='green')
plt.plot(btc_fr['time'], btc_fr['futures_aggregate_funding_rate_all_margin_1y_period'] * 100, label='1Y Period', color='red')

plt.gca().set_facecolor('white')
plt.grid(color='gray', linestyle='dotted', alpha=0.3)
```

```
plt.title('Bitcoin Perpetual Futures\nAggregated Funding Rate\n', fontsize=16)
plt.xlabel('')
plt.ylabel('Aggregate\nFunding Rate\n', fontsize=14)
plt.grid(True, alpha=0.3, linestyle='--')

# Set the formatter for the Y-axis to display percentages
formatter = mticker.FuncFormatter(lambda y, _: '{:.0f}%'.format(y))
plt.gca().yaxis.set_major_formatter(formatter)

plt.gca().xaxis.set_major_locator(mdates.AutoDateLocator())
plt.gca().xaxis.set_major_formatter(mdates.ConciseDateFormatter(mdates.AutoDateLocator()))
plt.legend(loc='upper right', fontsize=10, ncol=1, framealpha=0, bbox_to_anchor=(0.99, 1.13))
plt.show()
```



Plotting a heatmap of BTC funding rates across exchanges

In [40]:
```
btc_exch_fr_catalog = client.catalog_exchange_assets().to_dataframe()
btc_exch_fr_catalog = btc_exch_fr_catalog.loc[btc_exch_fr_catalog['metric']=='futures_aggregate_funding_rate_all_margin_1y_period']
btc_exch_fr_catalog = btc_exch_fr_catalog.loc[btc_exch_fr_catalog['frequency']=='1d']
btc_exch_fr_catalog = btc_exch_fr_catalog[btc_exch_fr_catalog['exchange_asset'].str.contains(r'-btc$', case=False)]
btc_exch_fr_catalog
```

Out[40]:

	exchange_asset	metric	frequency	min_time	max_time
3291	binance-btc	futures_aggregate_funding_rate_all_margin_1y_p...	1d	2023-09-09 16:00:00+00:00	2024-01-26 17:00:00+00:00
16065	bitfinex-btc	futures_aggregate_funding_rate_all_margin_1y_p...	1d	2023-09-09 16:00:00+00:00	2024-01-26 17:00:00+00:00
20183	bitmex-btc	futures_aggregate_funding_rate_all_margin_1y_p...	1d	2023-09-09 16:00:00+00:00	2024-01-26 17:00:00+00:00
27376	bybit-btc	futures_aggregate_funding_rate_all_margin_1y_p...	1d	2023-09-09 16:00:00+00:00	2024-01-26 17:00:00+00:00
42957	deribit-btc	futures_aggregate_funding_rate_all_margin_1y_p...	1d	2023-09-09 16:00:00+00:00	2024-01-26 17:00:00+00:00
56338	huobi-btc	futures_aggregate_funding_rate_all_margin_1y_p...	1d	2023-09-09 16:00:00+00:00	2024-01-26 17:00:00+00:00
65300	kraken-btc	futures_aggregate_funding_rate_all_margin_1y_p...	1d	2023-09-09 16:00:00+00:00	2024-01-26 17:00:00+00:00
76014	okex-btc	futures_aggregate_funding_rate_all_margin_1y_p...	1d	2023-09-09 16:00:00+00:00	2024-01-26 17:00:00+00:00

In [41]:
```
btc_fr_exchanges = client.get_exchange_asset_metrics(
    exchange_assets=btc_exch_fr_catalog['exchange_asset'].to_list(),
    start_time='2023-01-01',
    metrics = 'futures_aggregate_funding_rate_all_margin_1y_period',
    frequency='1d'
).to_dataframe()
btc_fr_exchanges
```

Out[41]:

	exchange_asset	time	futures_aggregate_funding_rate_all_margin_1y_period
0	binance-btc	2023-09-10 00:00:00+00:00	0.015781
1	binance-btc	2023-09-11 00:00:00+00:00	0.04394
2	binance-btc	2023-09-12 00:00:00+00:00	0.025475
3	binance-btc	2023-09-13 00:00:00+00:00	-0.037087
4	binance-btc	2023-09-14 00:00:00+00:00	0.024524
...	...	...	...
1107	okex-btc	2024-01-22 00:00:00+00:00	0.130537
1108	okex-btc	2024-01-23 00:00:00+00:00	0.039446
1109	okex-btc	2024-01-24 00:00:00+00:00	0.053005
1110	okex-btc	2024-01-25 00:00:00+00:00	0.078523
1111	okex-btc	2024-01-26 00:00:00+00:00	0.061474

1112 rows × 3 columns

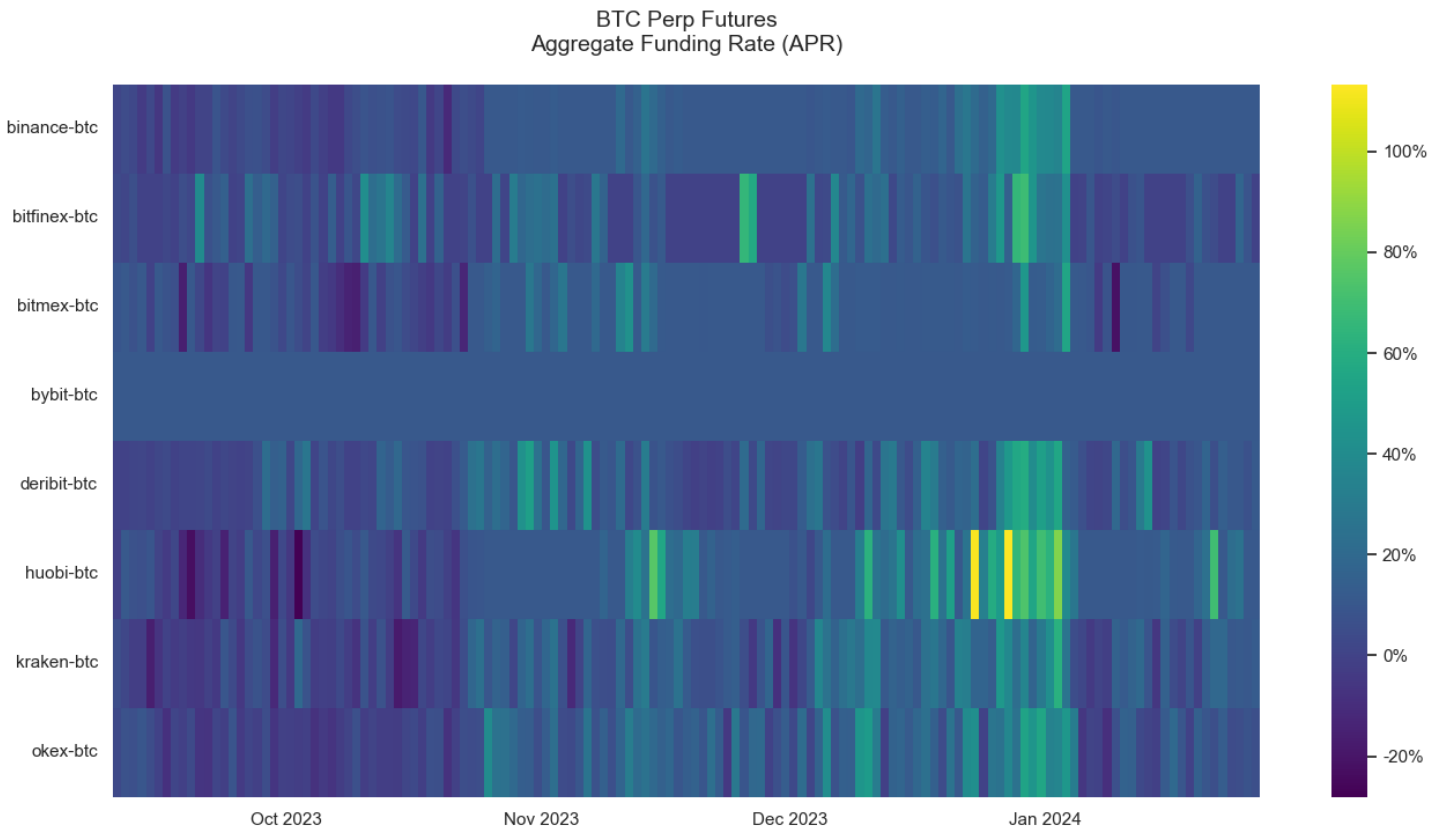
In [48]:
```
df = btc_fr_exchanges
# Pivot the DataFrame
pivot_df = df.pivot(index='exchange_asset', columns='time', values='futures_aggregate_funding_rate_all_margin_1y_period')
pivot_df = pivot_df.astype(float)
pivot_df
```

Out[48]:

time	2023-09-10 00:00:00+00:00	2023-09-11 00:00:00+00:00	2023-09-12 00:00:00+00:00	2023-09-13 00:00:00+00:00	2023-09-14 00:00:00+00:00	2023-09-15 00:00:00+00:00	2023-09-16 00:00:00+00:00	2023-09-17 00:00:00+00:00	2023-09 00:00:00+00
exchange_asset									
binance-btc	0.015781	0.043940	0.025475	-0.037087	0.024524	-0.054477	0.077938	-0.037573	-0.006
bitfinex-btc	0.034266	0.016949	0.059287	0.000000	0.000000	0.000000	0.020628	0.000000	0.051
bitmex-btc	0.090433	0.107158	0.075248	0.108107	-0.001438	0.107272	0.082111	0.064513	-0.159
bybit-btc	0.109500	0.109500	0.109500	0.109500	0.109500	0.109500	0.109500	0.109500	0.109
deribit-btc	-0.000354	0.000060	0.006619	0.012017	0.000652	0.021051	0.038959	0.000319	0.009
huobi-btc	-0.015292	0.109500	0.075166	0.065445	0.093933	0.002764	-0.037288	0.020860	-0.130
kraken-btc	0.051014	0.019125	-0.025588	-0.025686	-0.160542	-0.070392	-0.003192	-0.036900	-0.020
okex-btc	0.030967	0.077544	0.065015	0.099238	0.046852	-0.000690	-0.083597	0.007525	-0.012

8 rows × 139 columns

In [49]:
```
# Plotting the heatmap
plt.figure(figsize=(16, 8))
ax = sns.heatmap(pivot_df, cmap='viridis', annot=False)
plt.title('BTC Perp Futures\nAggregate Funding Rate (APR)\n', fontsize=14)

# Manually setting x-ticks for monthly intervals
date_labels = [pd.to_datetime(label).strftime('%b %Y') for label in pivot_df.columns]
monthly_intervals = [i for i, label in enumerate(pivot_df.columns) if pd.to_datetime(label).day == 1]

ax.set_xticks(monthly_intervals)
ax.set_xticklabels([date_labels[i] for i in monthly_intervals], rotation=0)
ax.set_yticklabels(ax.get_yticklabels(), rotation=0)
# Formatting colorbar labels as percentages
colorbar = ax.collections[0].colorbar
colorbar.ax.yaxis.set_major_formatter(mticker.FuncFormatter(lambda x, _: f'{x:.0%}'))

plt.xlabel('')
plt.ylabel('')
plt.show()
```



Cumulative Funding Rate

Cumulative Funding Rate is the cumulative average funding rate that would be accumulated by contract holders over a specified time period. Published once per hour, representing the cumulative realized funding rate over the previous 1 day, 7 day, and 30 day time periods.

- **futures_cumulative_funding_rate_usd_margin_*:** metrics represent the cumulative average funding rate weighted by open interest from futures markets where the margin asset is U.S. dollars or stablecoins over the previous specified time period.
- **futures_cumulative_funding_rate_coin_margin_*:** metrics represent the cumulative average funding rate weighted by open interest from futures markets where the margin asset is equivalent to the underlying base asset over the previous specified time period.
- **futures_cumulative_funding_rate_all_margin_*:** metrics represent the cumulative average funding rate weighted by open interest from all futures markets, regardless of the margin asset, over the previous specified time period.

In [44]:
```
btc_cumulative_fr = client.get_asset_metrics(
    assets='btc',
    start_time='2023-10-01',
    metrics = [
        'futures_cumulative_funding_rate_all_margin_1d',
        'futures_cumulative_funding_rate_all_margin_7d',
        'futures_cumulative_funding_rate_all_margin_30d'
        ]
).to_dataframe()
btc_cumulative_fr.head()
```

Out[44]:

	asset	time	futures_cumulative_funding_rate_all_margin_1d	futures_cumulative_funding_rate_all_margin_30d	futures_cumulative_funding_rate_all_margin_7d
0	btc	2023-10-01 00:00:00+00:00	0.000069	0.005681	0.000734
1	btc	2023-10-02 00:00:00+00:00	0.000136	0.005922	0.000786
2	btc	2023-10-03 00:00:00+00:00	0.000171	0.00616	0.000825
3	btc	2023-10-04 00:00:00+00:00	0.000142	0.006442	0.000845
4	btc	2023-10-05 00:00:00+00:00	0.000047	0.006555	0.000906

In [45]:
```
for column in btc_cumulative_fr.columns:
    if column != 'time':
        btc_cumulative_fr[column] = pd.to_numeric(btc_cumulative_fr[column], errors='coerce')
```

In [46]:
```
# Convert the 'time' column to datetime format if it's not already
btc_cumulative_fr['time'] = pd.to_datetime(btc_cumulative_fr['time'], errors='coerce')

# Ensure 'futures_cumulative_funding_rate_all_margin_7d' is numeric, replacing non-numeric values with numpy.nan
btc_cumulative_fr['futures_cumulative_funding_rate_all_margin_7d'] = pd.to_numeric(btc_cumulative_fr['futures_cumulative_funding_rate_all_margin_7d'],

# Check if there are still any non-numeric values
```

```
print(btc_cumulative_fr['futures_cumulative_funding_rate_all_margin_7d'].dtype)

# Plotting with safe checking
valid_7d_indices = ~btc_cumulative_fr['futures_cumulative_funding_rate_all_margin_7d'].isna()
valid_30d_indices = ~btc_cumulative_fr['futures_cumulative_funding_rate_all_margin_30d'].isna()

plt.plot(btc_cumulative_fr['time'], btc_cumulative_fr['futures_cumulative_funding_rate_all_margin_1d'] * 100, label='1D Period', color='blue')
plt.plot(btc_cumulative_fr['time'][valid_7d_indices], btc_cumulative_fr['futures_cumulative_funding_rate_all_margin_7d'][valid_7d_indices] * 100, label=
plt.plot(btc_cumulative_fr['time'][valid_30d_indices], btc_cumulative_fr['futures_cumulative_funding_rate_all_margin_30d'][valid_30d_indices] * 100, lab

plt.gca().set_facecolor('white')
plt.grid(color='gray', linestyle='dotted', alpha=0.3)

plt.title('Bitcoin Perpetual Futures\nCumulative Funding Rate\n', fontsize=16)
plt.xlabel('')
plt.ylabel('Cumulative\nFunding Rate\n', fontsize=14)
plt.grid(True, alpha=0.3, linestyle='--')

formatter = mticker.FuncFormatter(lambda y, _: '{:.2f}%'.format(y))
plt.gca().yaxis.set_major_formatter(formatter)
plt.gca().xaxis.set_major_locator(mdates.AutoDateLocator())
plt.gca().xaxis.set_major_formatter(mdates.ConciseDateFormatter(mdates.AutoDateLocator()))
plt.legend(loc='upper right', fontsize=10, ncol=1, framealpha=0, bbox_to_anchor=(0.99, 1.13))
plt.show()
```

```
Float64
```