Fintech Group Project 01

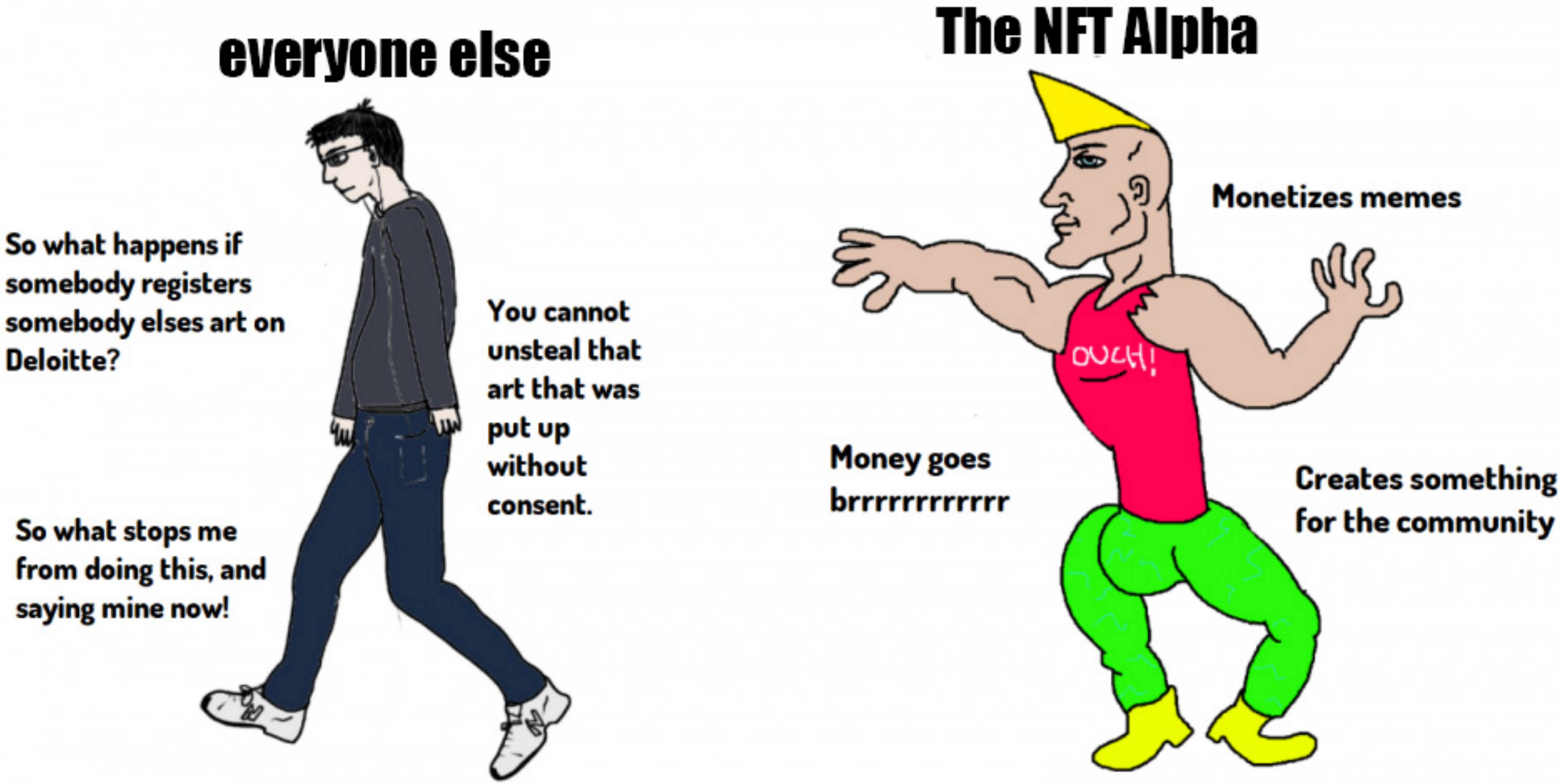
Azuki, BAYC and Crypto Punks NFTs.

by Github users /@dockingbay24 /@angel-estrada7 and /@mmsaki, **2022.**

Table of Contents

- Hypothesis of Project
- Data Collection
- Data Cleanup & Exploration
- Data Analysis
- Discussion
- Postmortem
- Questions?

Azuki Token ID: 7558 I **Current Owner:** ron1n.eth I **Sale:** 27ETH

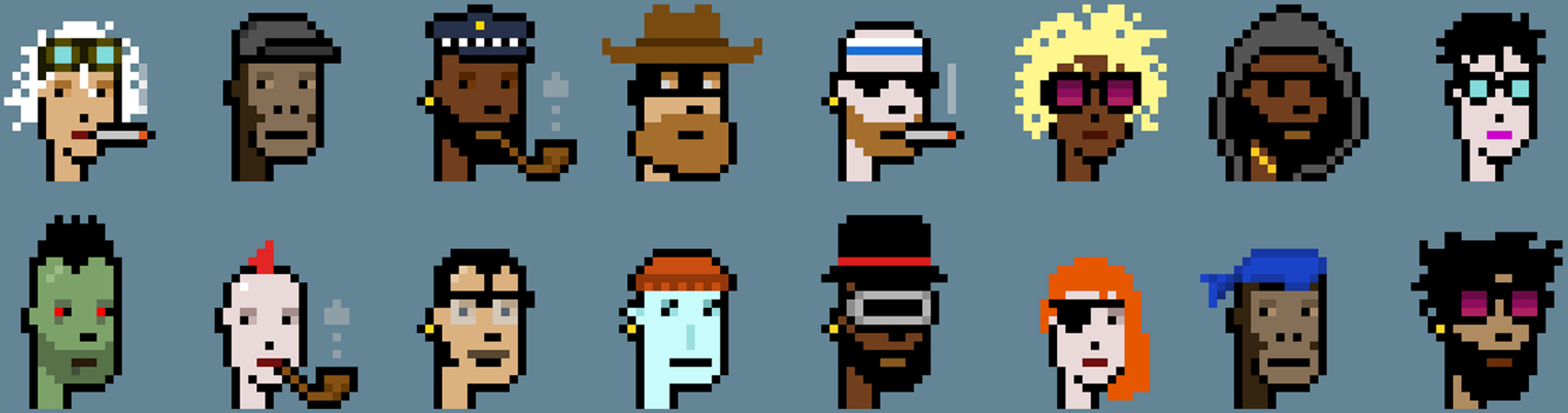

everyone else
So what happens if somebody registers somebody elses art on Deloitte?
You cannot unsteal that art that was put up without consent.
So what stops me from doing this, and saying mine now!
The NFT Alpha
Monetizes memes
OUCH!
Money goes brrrrrrrrrrrr
Creates something for the community

1. Hypothesis

Hypothesis of Project

Our motivation and summary…

- Should you invest in Azuki, BAYC or Crypto Punks?
- How can you tell which collection is performing well?



Data Collection

Describing what kinds of data we needed and where to find it.

2. Data Collection

Collecting data for NFT Collections

- Covalent API
- Etherscan Python Dependacy

group_project_01

May 5, 2022

0.1 Import dependancies

[1]:
```
# Import dependancies
import os
import requests
import pandas as pd
import json
from dotenv import load_dotenv
from etherscan_py import etherscan_py
import plotly.express as px
```

[2]:
```
# Loading .env containing our keys
load_dotenv()
```

[2]:
```
True
```

[3]:
```
# create variable for api key
api_key = os.getenv('COVALENT_API_KEY')
type(api_key)
```

[3]:
```
str
```

0.2 Current value of ETH

[4]:
```
# import dependancy
from etherscan_py import etherscan_py
etherscan_api = etherscan_py.Client(os.getenv('ETHERSCAN_API'))

# Print current eth price and latest block height
eth_value = etherscan_api.get_eth_price()
eth_value
```

[4]:
```
2738.51
```

3. Data Cleanup & Exploration

Exploring our collection data through APIs preparing it for analysis

project_analysis_2.0

May 6, 2022

0.1 Part One: Bored Ape Yatch Club

@angel-estrada7

[1]:

```
# Import dependancies
import os
import requests
import pandas as pd
import json
from dotenv import load_dotenv
from etherscan_py import etherscan_py
import plotly.express as px

import matplotlib.pyplot as plt
import hvplot.pandas
import numpy as np
import datetime as dt
import seaborn as sns
from pathlib import Path

%matplotlib inline
```

[2]:

```
# Loading .env containing our keys
load_dotenv()
```

[2]:

```
True
```

[3]:

```
# create variable for api key
api_key = os.getenv('COVALENT_API_KEY')
type(api_key)
```

[3]:

```
str
```

[4]:

```
# import dependancy
from etherscan_py import etherscan_py
etherscan_api = etherscan_py.Client(os.getenv('ETHERSCAN_API'))

# Print current eth price
```

4. Data Analysis

Analyzing our data and developing figures to answer our questions

@angel-estrada7

Part 1: Bored Apes

```
eth_value = etherscan_api.get_eth_price()
eth_value
```

```
[4]: 2728.75
```

0.2 Set Variables

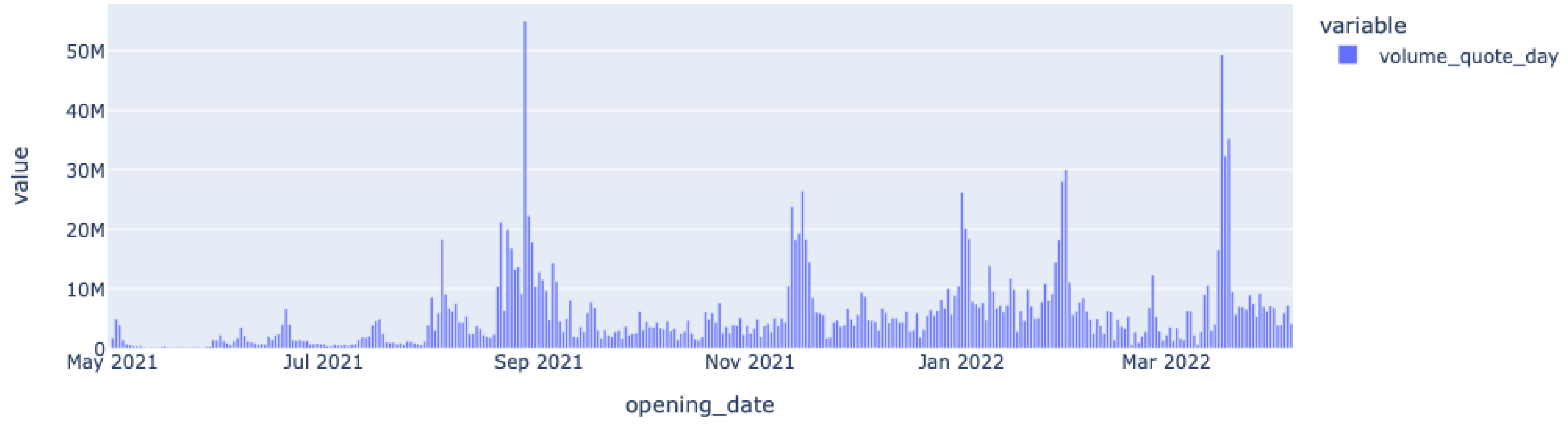
```
[5]: # Append url for our api
url = "https://api.covalenthq.com/v1"
chain_id = "/1"
azuki_address = "/0xED5AF388653567Af2F388E6224dC7C4b3241C544"
cryptopunks_address = "/0xb47e3cd837dDF8e4c57F05d70Ab865de6e193BBB"
BAYC_address = "/0xBC4CA0EdA7647A8aB7C2061c2E118A18a936f13D"
date_option = '/?quote-currency=USD&format=JSON&from=2017-01-01&to=2022-05-01'
page_option = '/transactions_v2/?
 ↪quote-currency=USD&format=JSON&block-signed-at-asc=false&no-logs=false&page-number=0&page-s
api_option = "&key=" + api_key
api_no_option = '/?key=' + api_key
```

0.3 a. Daily Volume

```
[6]: # Create variables needed for owner data and add to url
BAYC_historical_url = url + chain_id + "/nft_market/collection" + BAYC_address␣
 ↪+ api_no_option

# Get request
BAYC_historical_json = requests.get(BAYC_historical_url).json()

# Convert historical json data to a dataframe and view data
BAYC_df = pd.DataFrame(BAYC_historical_json['data']['items'])

# Set index to date
BAYC_df = BAYC_df.set_index('opening_date')

# Create Volume dataframe
BAYC_vol_df = pd.DataFrame(BAYC_df, columns = ['volume_quote_day',␣
 ↪'unique_token_ids_sold_count_day']).sort_index()
BAYC_vol_df.head()
```

```
[6]:               volume_quote_day  unique_token_ids_sold_count_day
opening_date
2021-04-30        8.241964e+02                                1
2021-05-01        1.737182e+06                             1635
2021-05-02        4.950946e+06                             1534
2021-05-03        3.948996e+06                              996
2021-05-04        1.388962e+06                              336
```

[7]:
```
# Plot Volume quote per day
BAYC_volume = BAYC_vol_df['volume_quote_day'].astype(int)

# BAYC_volume.plot.bar(figsize = (20,4))
px.bar(BAYC_volume)
```



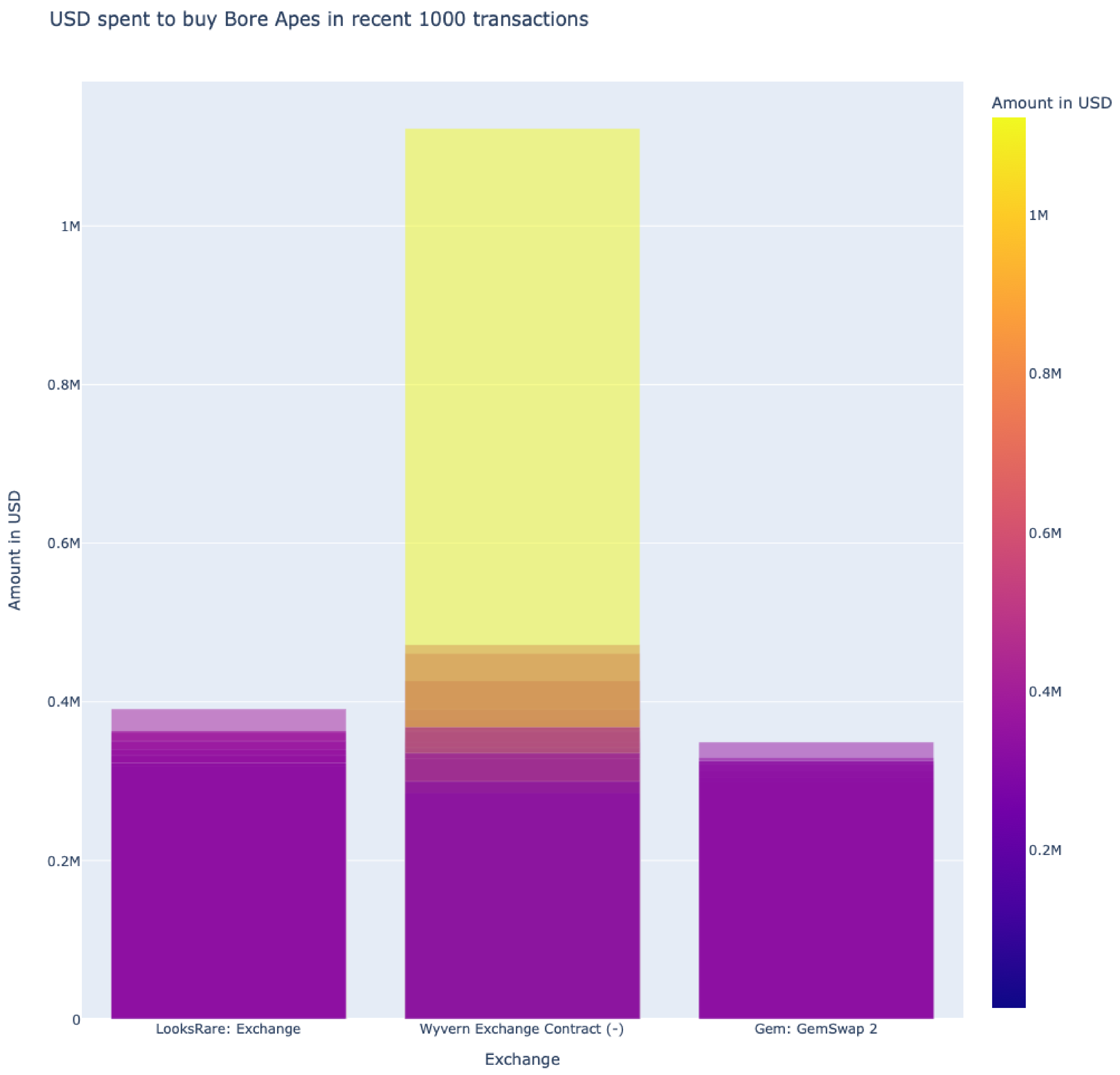
0.4 b. Recent 1000 transactions

[8]:
```
# Quering the API for transaction data
BAYC_tx_url = url + chain_id + "/address" + BAYC_address + page_option +␣
 ↪api_option
BAYC_tx = requests.get(BAYC_tx_url).json()

# Convert transactions data to dataframe
BAYC_tx_df = pd.DataFrame(BAYC_tx['data']['items'], columns =␣
 ↪['to_address_label','fees_paid', 'value_quote','block_signed_at']).
 ↪set_index('block_signed_at').sort_index()

BAYC_tx_df.head()
```

[8]:
```
                     to_address_label          fees_paid  value_quote
block_signed_at
2022-05-03T01:17:55Z             None  12168548098847650          0.0
2022-05-03T01:17:59Z             None   2259301753432880          0.0
2022-05-03T01:27:56Z             None   6401478612864081          0.0
2022-05-03T01:27:56Z             None   9922248312316456          0.0
2022-05-03T01:30:16Z             None   7634578388514804          0.0
```

[9]:
```
# Filter Through data for non null transactions
BAYC_sales_df = BAYC_tx_df[BAYC_tx_df['value_quote'] != 0]
BAYC_sales = BAYC_sales_df[BAYC_sales_df['to_address_label'].notnull()].dropna()

# Creating the plot using plotly express
```

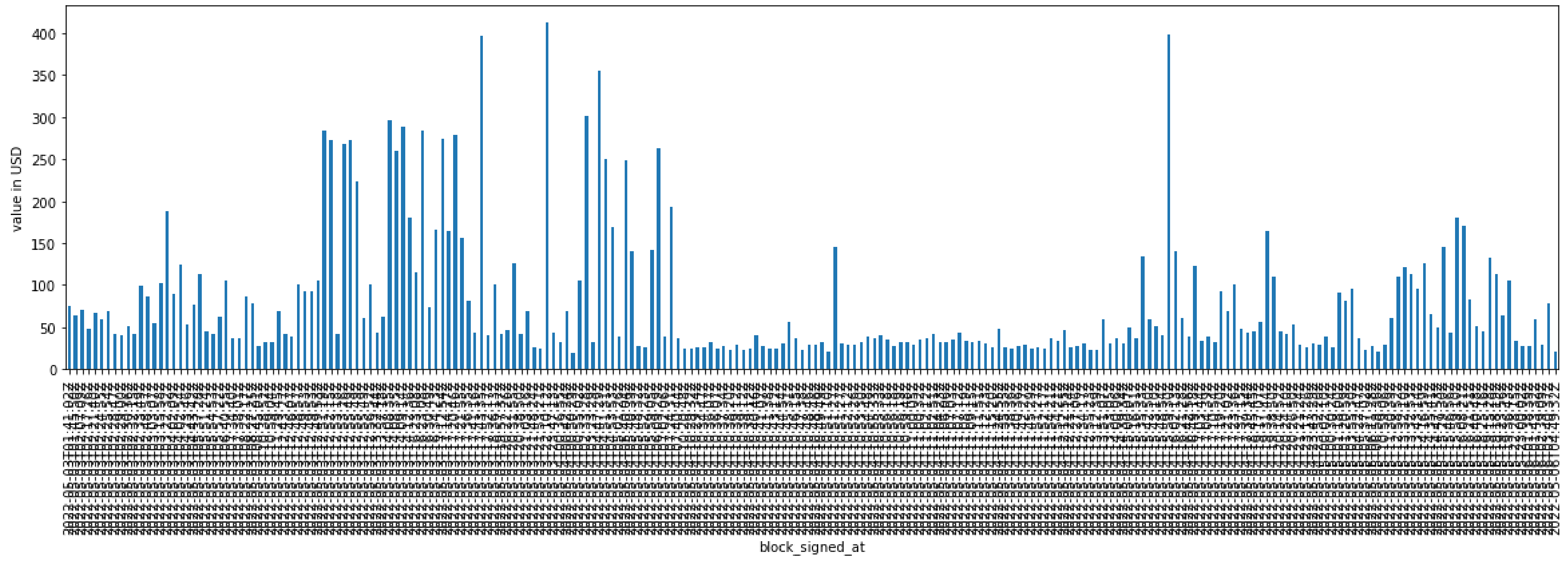
```
BAYC_fig = px.bar(BAYC_sales,
                  x='to_address_label',
                  y= 'value_quote',
                  color='value_quote',
                  height=1020,
                  width = 1000,
                  barmode = 'overlay',
                  labels={'value_quote':'Amount in USD', 'to_address_label':␣
 ↪'Exchange'},
                  title='USD spent to buy Bore Apes in recent 1000 transactions'
                 )
BAYC_fig.show()
```



0.5 c. Fees Spend

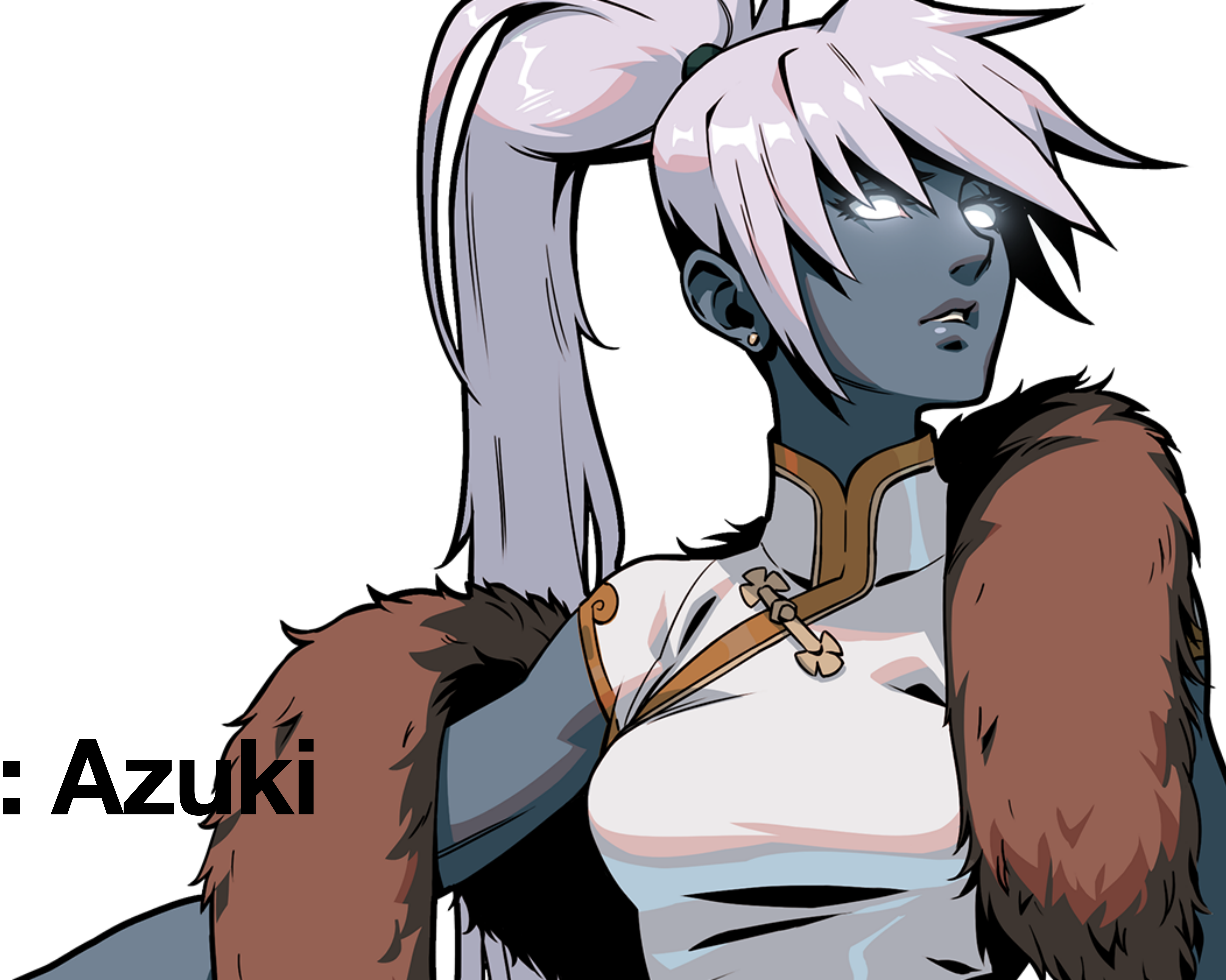
```
[10]: # Filter Through data for non null transactions
      BAYC_fees = BAYC_sales_df['fees_paid'].astype(int)/10**18*eth_value

      BAYC_fees.plot.bar(rot = 90, figsize = (20,5), ylabel = 'value in USD')
```

```
[10]: <AxesSubplot:xlabel='block_signed_at', ylabel='value in USD'>
```



Part Two: Azuki

@mmsaki



0.6 Part TWO: Azuki

@mmsaki

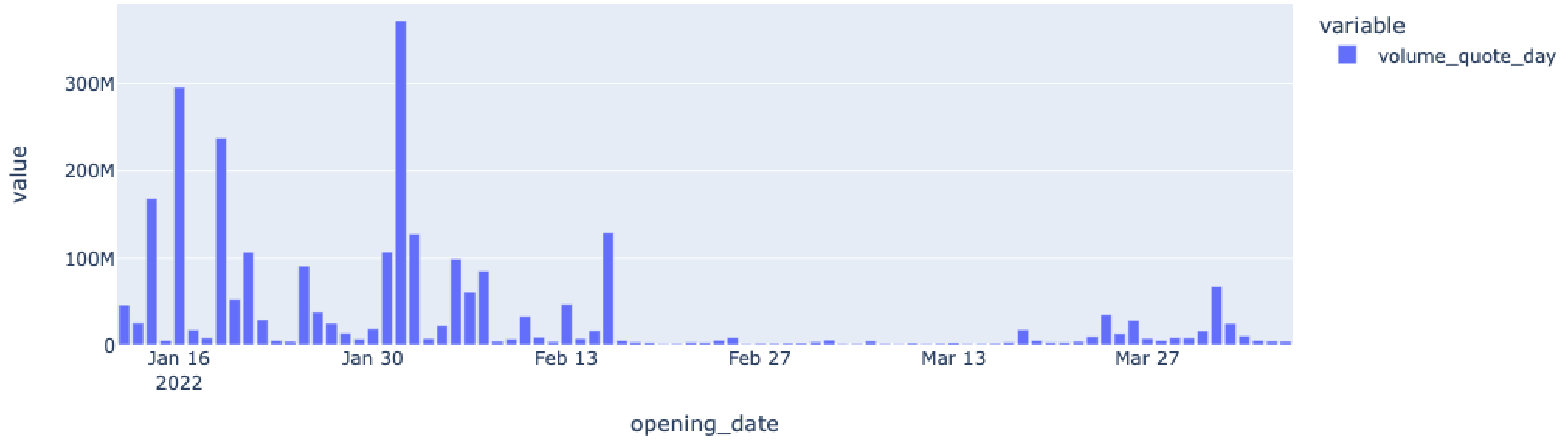
0.7 a. Daily Volume

[11]:

```
# Create variables needed for owner data and add to url
azuki_url = url + chain_id + "/nft_market/collection" + azuki_address +␣
 ↪api_no_option

# Get request
azuki_historical_json = requests.get(azuki_url).json()

# Convert historical json data to a dataframe and view data
azuki_df = pd.DataFrame(azuki_historical_json['data']['items'])

# Set index to date
azuki_df = azuki_df.set_index('opening_date')

# Create Volume dataframe
azuki_vol_df = pd.DataFrame(azuki_df, columns = ['volume_quote_day',␣
 ↪'unique_token_ids_sold_count_day']).sort_index()
azuki_vol_df.head()
```

[11]:
```
                 volume_quote_day  unique_token_ids_sold_count_day
opening_date
2022-01-12             45941404.0                             2402
2022-01-13             25129178.0                             1318
2022-01-14            168151840.0                              470
2022-01-15              4408686.0                              499
2022-01-16            295638336.0                              368
```

[12]:
```python
# Plot Volume quote per day
azuki_volume = azuki_vol_df['volume_quote_day'].astype(int)

# Plot Historical daily volume
px.bar(azuki_volume)
```



0.8 b. Recent 1000 transactions

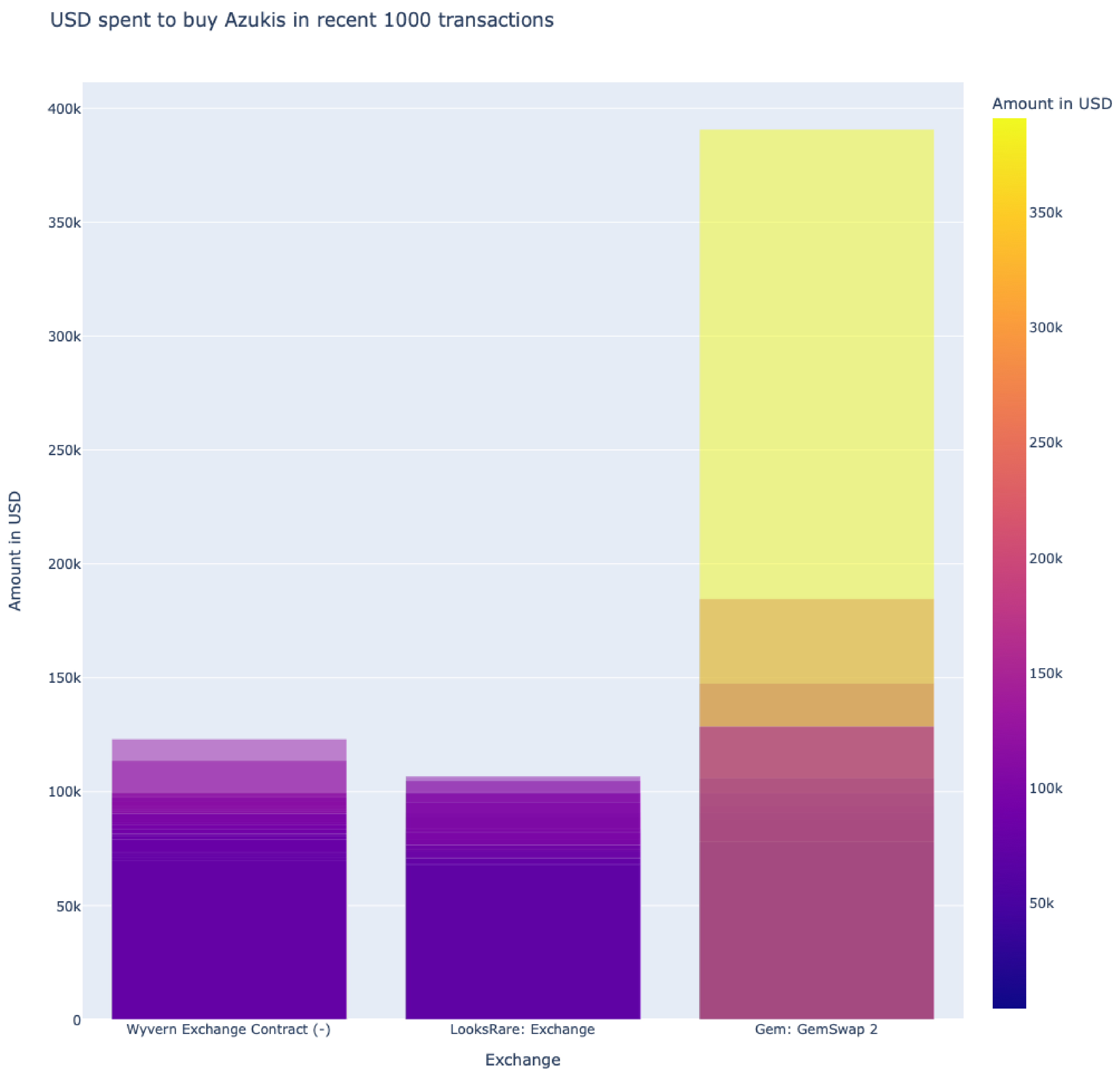
[13]:
```python
# Quering the API for transaction data
azuki_tx_url = url + chain_id + "/address" + azuki_address + page_option +␣
 ↪api_option
azuki_tx = requests.get(azuki_tx_url).json()

# Convert transactions data to dataframe
azuki_tx_df = pd.DataFrame(azuki_tx['data']['items'], columns =␣
 ↪['to_address_label','fees_paid', 'value_quote','block_signed_at']).
 ↪set_index('block_signed_at').sort_index()

azuki_tx_df.head()
```

[13]:
```
                                   to_address_label          fees_paid  \
block_signed_at
2022-05-03T01:14:10Z                           None   2723068601452056
```

```
2022-05-03T01:18:07Z                          None   2057488554010604
2022-05-03T01:18:52Z                          None   2253722629285699

                       value_quote
block_signed_at
2022-05-03T01:14:10Z      0.000000
2022-05-03T01:17:28Z  85294.050293
2022-05-03T01:17:28Z      0.000000
2022-05-03T01:18:07Z      0.000000
2022-05-03T01:18:52Z      0.000000
```

[14]:

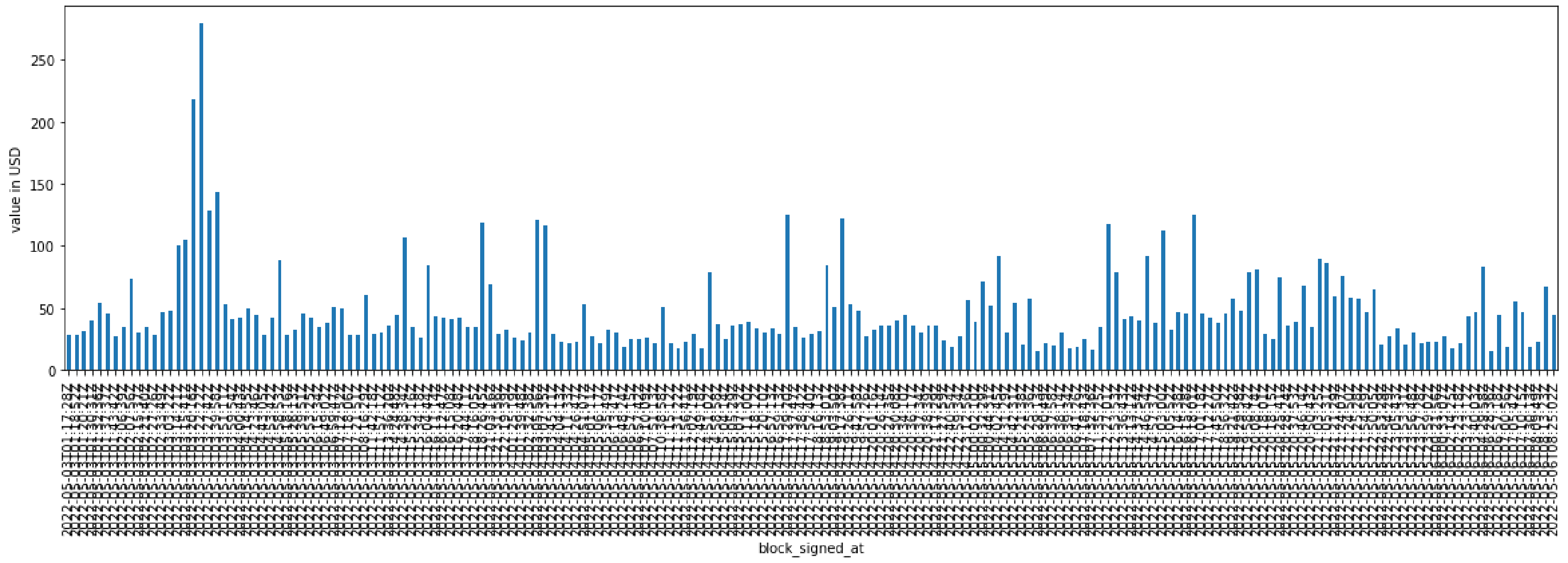
```
# Filter Through data for non null transactions
azuki_sales_df = azuki_tx_df[azuki_tx_df['value_quote'] != 0]
azuki_sales = azuki_sales_df[azuki_sales_df['to_address_label'].notnull()]

# Creating the plot using plotly express
azuki_fig = px.bar(azuki_sales,
                   x='to_address_label',
                   y= 'value_quote',
                   color='value_quote',
                   height=1020,
                   width = 1000,
                   barmode='overlay',
                   labels={'value_quote':'Amount in USD', 'to_address_label':␣
 ↪'Exchange'},
                   title='USD spent to buy Azukis in recent 1000 transactions'
                  )
azuki_fig.show()
```



0.9 c. fees paid

[15]:
```
# Filter Through data for non null transactions
azuki_fees = azuki_sales_df['fees_paid'].astype(int)/10**18*eth_value

azuki_fees.plot.bar(rot = 90, figsize = (20,5), ylabel = 'value in USD')
```

[15]:
```
<AxesSubplot:xlabel='block_signed_at', ylabel='value in USD'>
```



0.10 d. comparison

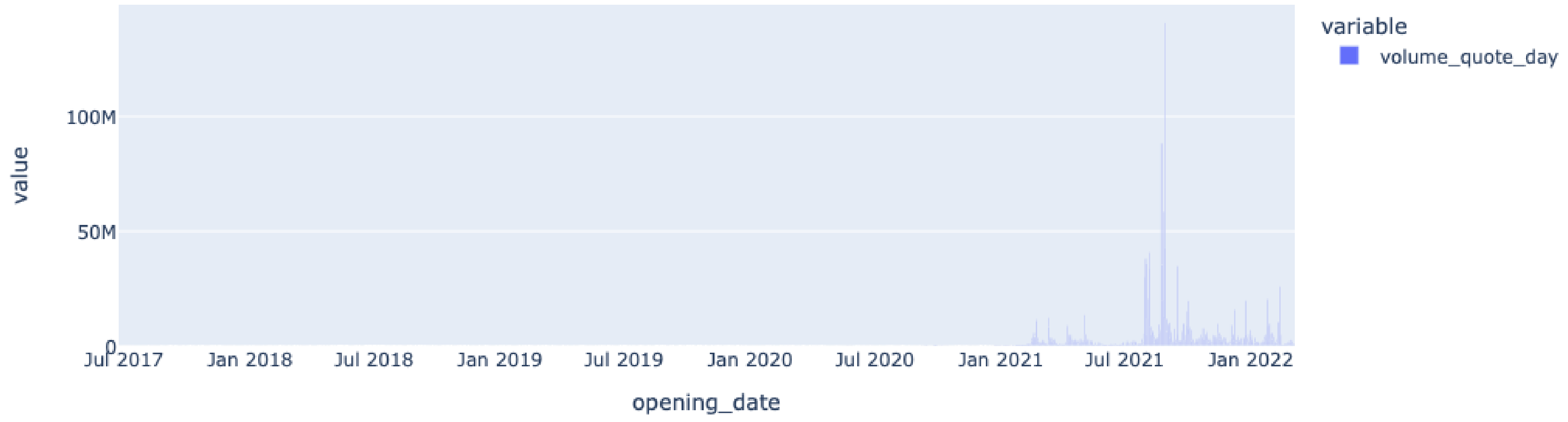
[16]:
```
# Create variables needed for owner data and append to url
cryptopunks_historical_url = url + chain_id + "/nft_market/collection" +␣
↪cryptopunks_address + api_no_option

# Get request
cryptopunks_historical_json = requests.get(cryptopunks_historical_url).json()

# Convert historical json data to a dataframe and view data
cryptopunks_df = pd.DataFrame(cryptopunks_historical_json['data']['items'])

# Set index to date
cryptopunks_df = cryptopunks_df.set_index('opening_date')

# Create Volume dataframe
cryptopunks_vol_df = pd.DataFrame(cryptopunks_df, columns =␣
↪['volume_quote_day', 'unique_token_ids_sold_count_day']).sort_index()
cryptopunks_vol_df.head()
```

[16]:
```
              volume_quote_day  unique_token_ids_sold_count_day
opening_date
2017-06-23                 0.0                               19
2017-06-24                 0.0                               22
2017-06-25                 0.0                               11
2017-06-26                 0.0                               18
2017-06-27                 0.0                               35
```

0.11 punk volume a.

[17]:

```
# Plot Volume quote per day
cryptopunks_volume = cryptopunks_vol_df['volume_quote_day'].astype(int)

# cryptopunks_volume.plot.line(figsize = (20,4))

px.bar(cryptopunks_volume)
```



0.12 punk sales b.

[18]:

```
# Quering the API for transaction data
cryptopunks_tx_url = url + chain_id + "/address" + cryptopunks_address +␣
 ↪page_option + api_option
cryptopunks_tx = requests.get(cryptopunks_tx_url).json()

# Convert transactions data to dataframe
cryptopunks_tx_df = pd.DataFrame(cryptopunks_tx['data']['items'], columns =␣
 ↪['to_address_label','fees_paid', 'value_quote','block_signed_at']).
 ↪set_index('block_signed_at').sort_index()

cryptopunks_tx_df.head()
```

[18]:

```
                         to_address_label          fees_paid  value_quote
block_signed_at
2022-04-27T21:12:57Z  CRYPTOPUNKS ( )  1421908968219564          0.0
2022-04-27T21:12:57Z  CRYPTOPUNKS ( )  1421908968219564          0.0
2022-04-27T21:12:57Z  CRYPTOPUNKS ( )  1421908968219564          0.0
2022-04-27T21:12:57Z  CRYPTOPUNKS ( )  1421908968219564          0.0
2022-04-27T21:14:42Z             None  6923227425050630          0.0
```

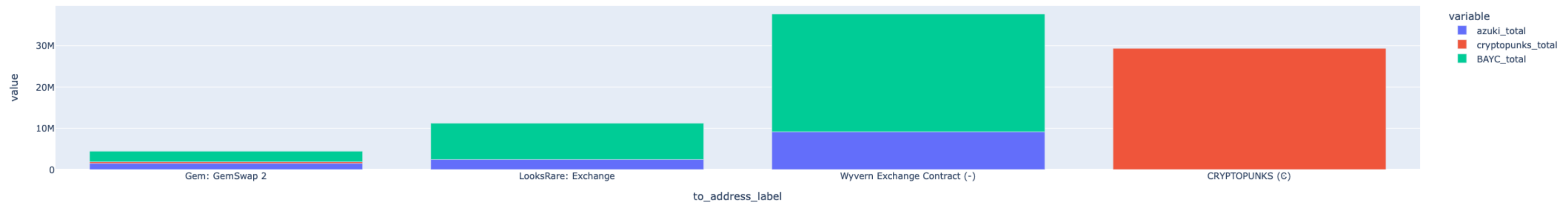
[19]:

```
# Create variables needed for owner data and append to url
```

```
BAYC_total = BAYC_sales.groupby('to_address_label').sum()
```

```
[22]: # Combine and rename columns for our total sales data
combined_totals = pd.concat([azuki_total,cryptopunks_total,BAYC_total], axis=1)
combined_totals.columns = ['azuki_total', 'cryptopunks_total','BAYC_total']
```

```
[23]: # Plot for combined figure
combined_total_fig = px.bar(combined_totals)


# Show Figure
combined_total_fig.show()
```

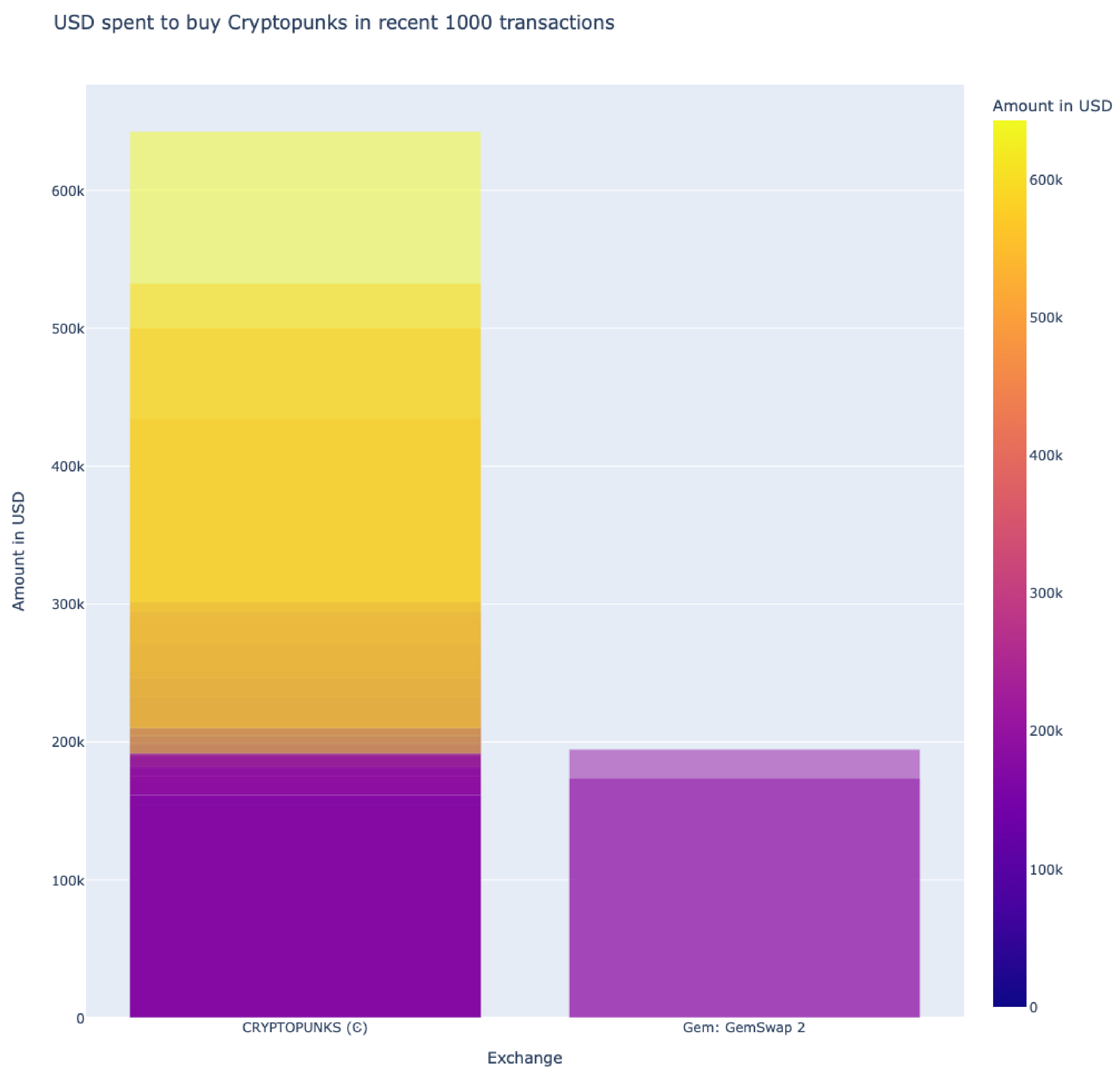


0.16 Combine Total Fees

```
[24]: # Group by address label and sum the value
combined_totals
```

```
[24]:                              azuki_total  cryptopunks_total    BAYC_total
      to_address_label
      Gem: GemSwap 2              1.494632e+06       3.683260e+05  2.568193e+06
      LooksRare: Exchange         2.452695e+06                NaN  8.755434e+06
      Wyvern Exchange Contract (-)  9.148483e+06              NaN  2.856156e+07
      CRYPTOPUNKS ( )                      NaN       2.937500e+07           NaN
```

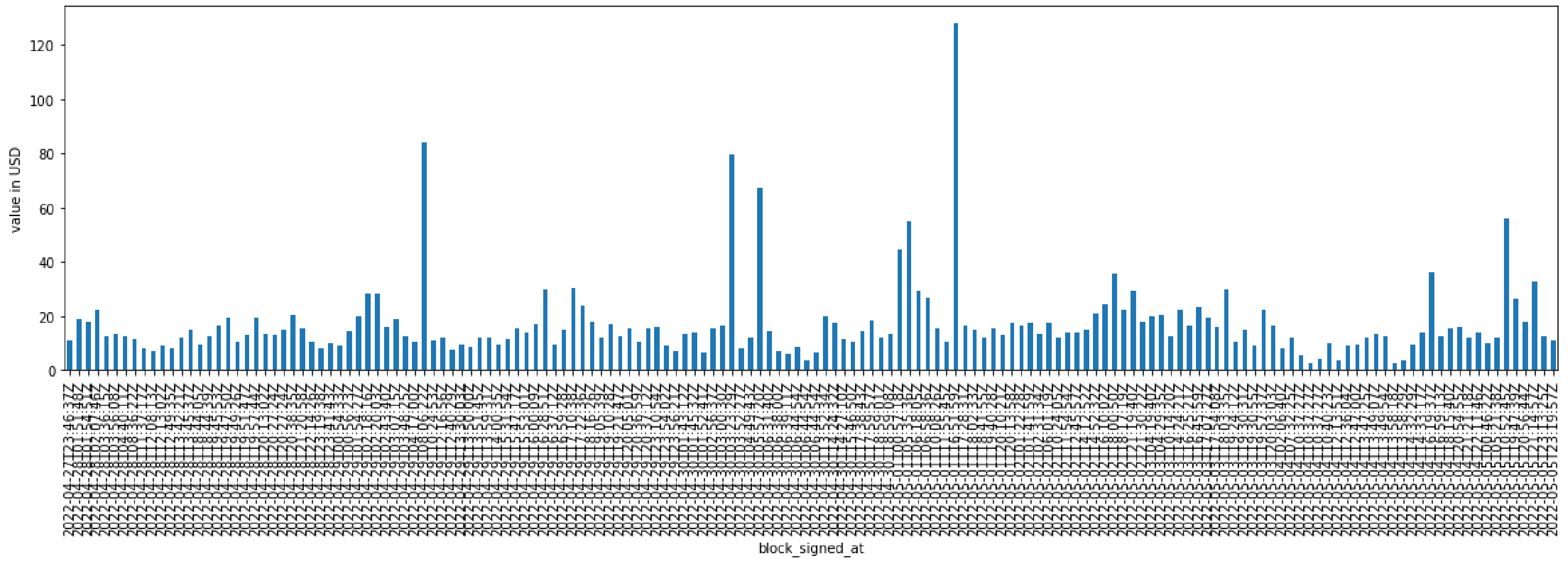
```
[25]: # Combine and rename columns for our total sales data
azuki_usd_fees = azuki_sales['fees_paid'].astype(int)/10**18*eth_value
cryptopunks_usd_fees = cryptopunks_sales['fees_paid'].astype(int)/
 ↪10**18*eth_value
BAYC_usd_fees = BAYC_sales['fees_paid'].astype(int)/10**18*eth_value

# Combine dataframe and drop nulls
combined_usd_fees = pd.concat([azuki_usd_fees.reset_index(drop=True),
                               cryptopunks_usd_fees.reset_index(drop=True),
                               BAYC_usd_fees.reset_index(drop=True)],
                              axis=1
                             ).dropna()
combined_usd_fees.columns = ['azuki_fees', 'cryptopunks_fees','BAYC_fees']
```



0.13 punk fees c.

```
[21]: # Filter Through data for non null transactions
      cryptopunks_fees = cryptopunks_sales_df['fees_paid'].astype(int)/
       ↪10**18*eth_value

      # plot
      cryptopunks_fees.plot.bar(rot = 90, figsize = (20,5), ylabel = 'value in USD')
```

```
[21]: <AxesSubplot:xlabel='block_signed_at', ylabel='value in USD'>
```



0.14 combined fees d.

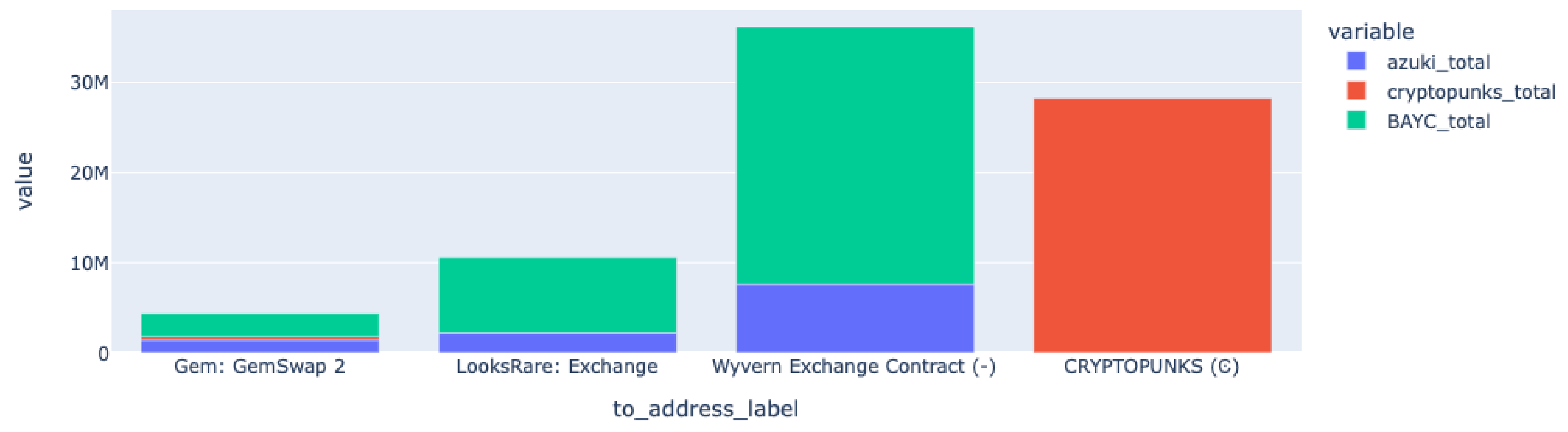
[22]:
```
# Group by address label and sum the value
azuki_total = azuki_sales.groupby('to_address_label').sum()
cryptopunks_total = cryptopunks_sales.groupby('to_address_label').sum()
BAYC_total = BAYC_sales.groupby('to_address_label').sum()
```

[23]:
```
# Combine and rename columns for our total sales data
combined_totals = pd.concat([azuki_total,cryptopunks_total,BAYC_total], axis=1)
combined_totals.columns = ['azuki_total', 'cryptopunks_total','BAYC_total']
```

[24]:
```
# Group by address label and sum the value
combined_totals
```

[24]:
```
                              azuki_total  cryptopunks_total    BAYC_total
to_address_label
Gem: GemSwap 2               1.408856e+06       3.683260e+05  2.568193e+06
LooksRare: Exchange          2.145862e+06                NaN  8.420908e+06
Wyvern Exchange Contract (-) 7.586556e+06                NaN  2.855636e+07
CRYPTOPUNKS ( )                       NaN       2.824688e+07           NaN
```

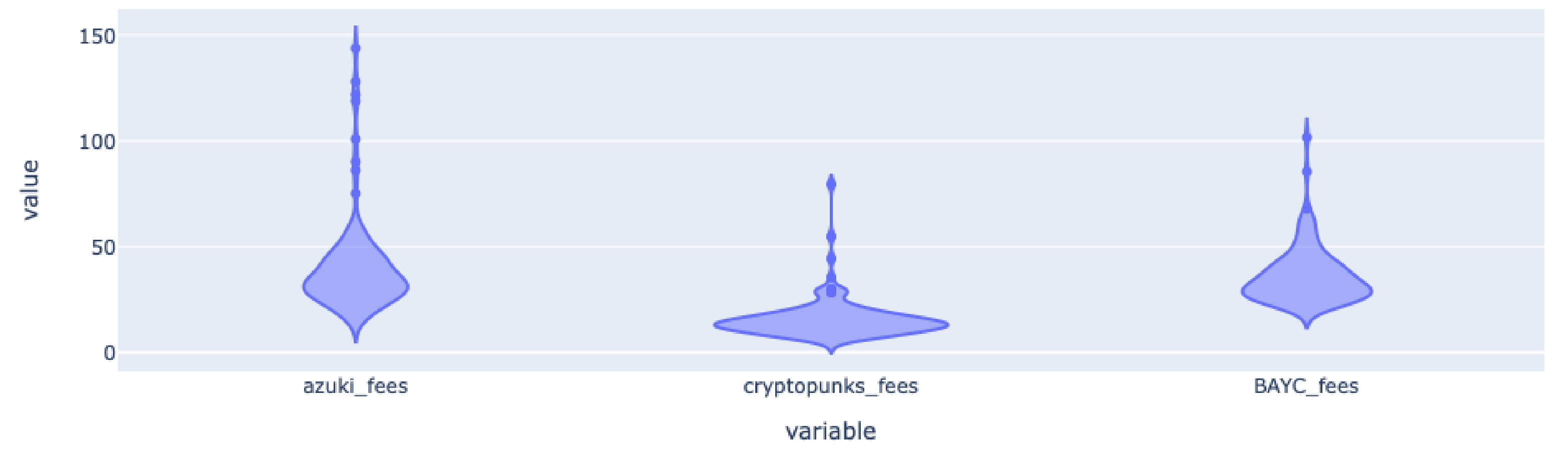
[25]:
```
# Plot for combined figure
combined_total_fig = px.bar(combined_totals)


# Show Figure
combined_total_fig.show()
```



0.15 combined fees e.

[26]:
```
# Combine and rename columns for our total sales data
azuki_usd_fees = azuki_sales['fees_paid'].astype(int)/10**18*eth_value
cryptopunks_usd_fees = cryptopunks_sales['fees_paid'].astype(int)/
 ↪10**18*eth_value
BAYC_usd_fees = BAYC_sales['fees_paid'].astype(int)/10**18*eth_value

# Combine dataframe and drop nulls
combined_usd_fees = pd.concat([azuki_usd_fees.reset_index(drop=True),
                               cryptopunks_usd_fees.reset_index(drop=True),
                               BAYC_usd_fees.reset_index(drop=True)],
                              axis=1
                             ).dropna()
combined_usd_fees.columns = ['azuki_fees', 'cryptopunks_fees','BAYC_fees']
```

[27]:
```
# Plot for combined figure
combined_fees_fig = px.violin(combined_usd_fees)

# Show Figure
combined_fees_fig.show()
```



Part Three: Cryptopunks

@dockingbay

0.16 Part Three: Cryptopunks

@dockingbay24

[28]:
```
# create variable for api key  for etherscan
ETHERSCAN_API_KEY = os.getenv("ETHERSCAN_API")

#set api url variables for Etherscan call
cryptopunks_contract = "0xb47e3cd837dDF8e4c57F05d70Ab865de6e193BBB"
etherscan_url = "https://api.etherscan.io/api"
module = "?module=contract"
action = "&action=getsourcecode"
address ="&address=" + cryptopunks_contract
etherscan_key = "&apikey=" + ETHERSCAN_API_KEY
```

[29]:
```
# Set API call string
url_cryptopunks_contract_details =␣
 ↪etherscan_url+module+action+address+etherscan_key

# Get results from API call
cryptopunk_contract_details= requests.get(url_cryptopunks_contract_details).
 ↪json()
```

0.17 Etherscan Cryptopunks Transactions

[30]:
```
# Set api url variables for Etherscan call
transaction_hash =␣
 ↪"0x15f8e5ea1079d9a0bb04a4c58ae5fe7654b5b2b4463375ff7ffb490aa0032f3a"␣
 ↪#replace with trans_hash
etherscan_url = "https://api.etherscan.io/api"
module = "?module=transaction"
action = "&action=getstatus"
address ="&&txhash=" + transaction_hash
etherscan_key = "&apikey=" + ETHERSCAN_API_KEY
```

[31]:
```
# Set API call string
url_cryptopunks_transactions = etherscan_url+module+action+address+etherscan_key
```

[32]:
```
# Get results from API call
cryptopunk_transaction=requests.get(url_cryptopunks_transactions).json()
```

0.18 a. Wrapped Cryptopunks

[33]:
```
# Append url for our api
url = "https://api.covalenthq.com/v1"
chain_id = "/1"    #TEMP is it always chain1 for most part?
option = "/nft_market/collection"
# Add search queries to api url
contract_address = "/0xb7f7f6c52f2e2fdb1963eab30438024864c313f6"   #Do we want␣
 ↪to compare other contracts
currency = "/?quote-currency=USD"
format_output = "&format=JSON"
date_from ="&from=2022-01-25"
date_to = "&to=2022-04-25"
covalent_api_key = "&key=" + api_key

url_nft_market_cap_detail = url + chain_id + option + contract_address +␣
 ↪currency + format_output + date_from + date_to  + covalent_api_key
```

[34]:
```
#set API call string
url_nft_market_cap_detail = url + chain_id + option + contract_address +␣
 ↪currency + format_output + date_from + date_to  + covalent_api_key
```

[35]:
```
#get results from API call
nft_market_cap = requests.get(url_nft_market_cap_detail).json()
```

[36]:
```
#set data into a dataframe
nft_market_cap_df = pd.DataFrame(nft_market_cap['data']['items'])
```

[37]:
```
#display head and tail of df
display(nft_market_cap_df.head())
```

```
   chain_id      collection_name                          collection_address  \
0         1  Wrapped Cryptopunks  0xb7f7f6c52f2e2fdb1963eab30438024864c313f6
1         1  Wrapped Cryptopunks  0xb7f7f6c52f2e2fdb1963eab30438024864c313f6

  collection_ticker_symbol opening_date         volume_wei_day  \
0                   WPUNKS   2022-03-30  201500000000000000000
1                   WPUNKS   2022-02-03     250000000000000000

   volume_quote_day average_volume_wei_day  average_volume_quote_day  \
0       687428.9000  201500000000000000000               687428.9000
1          664.4058     250000000000000000                  664.4058

   unique_token_ids_sold_count_day  … fourth_nft_image_token_id  \
0                                1  …                        51
1                                1  …                        51

                                    fourth_nft_image  \
count                               2.0            2.000000
mean                                1.0       344046.652900
std                                 0.0       485615.830927
min                                 1.0          664.405800
25%                                 1.0       172355.529350
50%                                 1.0       344046.652900
75%                                 1.0       515737.776450
max                                 1.0       687428.900000

       gas_quote_rate_day
count            2.000000
mean          3034.590450
std            533.112056
min           2657.623300
25%           2846.106875
50%           3034.590450
75%           3223.074025
max           3411.557600
```

```
[39]: #TEMP list columns of df
      nft_market_cap_df.columns
```

```
[39]: Index(['chain_id', 'collection_name', 'collection_address',
             'collection_ticker_symbol', 'opening_date', 'volume_wei_day',
             'volume_quote_day', 'average_volume_wei_day',
             'average_volume_quote_day', 'unique_token_ids_sold_count_day',
             'floor_price_wei_7d', 'floor_price_quote_7d', 'gas_quote_rate_day',
             'quote_currency', 'first_nft_image_token_id', 'first_nft_image',
             'first_nft_image_256', 'first_nft_image_512', 'first_nft_image_1024',
             'second_nft_image_token_id', 'second_nft_image', 'second_nft_image_256',
             'second_nft_image_512', 'second_nft_image_1024',
             'third_nft_image_token_id', 'third_nft_image', 'third_nft_image_256',
             'third_nft_image_512', 'third_nft_image_1024',
             'fourth_nft_image_token_id', 'fourth_nft_image', 'fourth_nft_image_256',
             'fourth_nft_image_512', 'fourth_nft_image_1024',
             'fifth_nft_image_token_id', 'fifth_nft_image', 'fifth_nft_image_256',
             'fifth_nft_image_512', 'fifth_nft_image_1024'],
            dtype='object')
```

```
[40]: # Create a new data frame for graphing volume, drop un-needed columns
      market_cap_df_graph = nft_market_cap_df[["opening_date","volume_quote_day"]].
       ↪copy()
```

```
[41]: # TEMP display market_cap_df_graph
      market_cap_df_graph
```

```
[41]:   opening_date  volume_quote_day
      0   2022-03-30       687428.9000
      1   2022-02-03          664.4058
```

```
[42]: # Graph dataframe for analysis
      from bokeh.models.formatters import NumeralTickFormatter
      formatter = NumeralTickFormatter(format="0,0")

      market_cap_df_graph.hvplot.bar(
          x='opening_date',
          y='volume_quote_day',
          xlabel='Opening Date',
          ylabel='Volume',
          rot=90,
          title='Volume Quote Per Day - 0xb7f7f6c52f2e2fdb1963eab30438024864c313f6',
          height= 600,
          width = 2000
      ).opts(
        yformatter=formatter
      )
```

```
[42]: :Bars   [opening_date]   (volume_quote_day)
```

0.19 b. Punks not wrapped

```
[43]: # Set variables
      url = "https://api.covalenthq.com/v1"
      chain_id = "/1"    #TEMP is it always chain1 for most part?
      option = "/nft_market/collection"

      # Add search queries to api url
      contract_address2 = "/0xb47e3cd837dDF8e4c57F05d70Ab865de6e193BBB"
      currency = "/?quote-currency=USD"
      format_output = "&format=JSON"
      date_from ="&from=2022-01-25"
      date_to = "&to=2022-04-25"
      covalent_api_key = "&key=" + api_key

      # Append url for our api
      url_nft_market_cap_detail2 = url + chain_id + option + contract_address2 +␣
       ↪currency + format_output + date_from + date_to  + covalent_api_key
```

```
[44]: #set API call string
      url_nft_market_cap_detail2 = url + chain_id + option + contract_address2 +␣
       ↪currency + format_output + date_from + date_to  + covalent_api_key
      #get results from API call
      nft_market_cap2 = requests.get(url_nft_market_cap_detail2).json()
```

[45]:
```
#set data into a dataframe
nft_market_cap_df2 = pd.DataFrame(nft_market_cap2['data']['items'])
```

[46]:
```
#set data into a dataframe
nft_market_cap_df2 = pd.DataFrame(nft_market_cap2['data']['items'])
#create a new data frame for graphing volume, drop un-needed columns
market_cap_df_graph_2 = nft_market_cap_df2[["opening_date","volume_quote_day"]].
  ↪copy()
```

[47]:
```
#Graph dataframe for analysis
from bokeh.models.formatters import NumeralTickFormatter
formatter = NumeralTickFormatter(format="0,0")

market_cap_df_graph_2.hvplot.bar(
    x='opening_date',
    y='volume_quote_day',
    xlabel='Opening Date',
    ylabel='Volume',
    rot=90,
    title='Volume Quote Per Day - Cryptopunks',
    height= 600,
    width = 1000
).opts(
  yformatter=formatter
)
```

[47]:
```
:Bars   [opening_date]   (volume_quote_day)
```

[48]:
```
from bokeh.models.formatters import NumeralTickFormatter
formatter = NumeralTickFormatter(format="0,0")
graph2 = market_cap_df_graph_2.hvplot.bar(
    x='opening_date',
    y='volume_quote_day',
    xlabel='Opening Date',
    ylabel='Volume',
    rot=90,
    label='0xb47e3cd837dDF8e4c57F05d70Ab865de6e193BBB',
    height= 600,
    width = 1000
).opts(
  yformatter=formatter
)

graph1  = market_cap_df_graph.hvplot.bar(
    x='opening_date',
    y='volume_quote_day',
    xlabel='Opening Date',
    ylabel='Volume',
    rot=90,
    label='0xb7f7f6c52f2e2fdb1963eab30438024864c313f6',
    height= 600,
    width = 1000
).opts(
  yformatter=formatter
)
```

```
[49]: graph2 * graph1
```

```
[49]: :Overlay
         .Bars.A_0xb47e3cd837dDF8e4c57F05d70Ab865de6e193BBB :Bars   [opening_date]
      (volume_quote_day)
         .Bars.A_0xb7f7f6c52f2e2fdb1963eab30438024864c313f6 :Bars   [opening_date]
      (volume_quote_day)
```

```
[50]: market_cap_df_graph['Token'] = '0xb7f7f6c52f2e2fdb1963eab30438024864c313f6'
      market_cap_df_graph_2['Token'] = '0xb47e3cd837dDF8e4c57F05d70Ab865de6e193BBB'

      # combine dataframes into a single df
      combined_df = pd.concat([market_cap_df_graph, market_cap_df_graph_2],␣
       ↪join="outer", ignore_index=False)
```

0.20 c. Combined Token Graph

```
[51]: # Combined Token graph
      from bokeh.models.formatters import NumeralTickFormatter
      formatter = NumeralTickFormatter(format="0,0")
      combined_df.hvplot.scatter(
          x='opening_date',
          y='volume_quote_day',
          xlabel='Date',
          ylabel='Volume',
          rot=90,
          label='Combined Analysis',
          by='Token',
          attr_labels=False,
          height= 600,
          width = 1000
      ).opts(
        yformatter=formatter
      )
```

```
[51]: :NdOverlay   [Token]
         :Scatter   [opening_date]   (volume_quote_day)
```

0.21 d. import historical data

[52]:

```
# Read in all cryptopunkowners
cryptopunk_owners_path = Path("./Resources_punks/2022-05_all_cryptopunk_owners.
 ↪csv")

# Read in top20 sales, by ether value
top20_sales_path = Path("./Resources_punks/top20_sales_by_ether_value.csv")
```

[53]:

```
#import into dataframes
cryptopunk_owners_df = pd.read_csv(cryptopunk_owners_path, index_col="#",␣
 ↪parse_dates=True, infer_datetime_format=True)
top20_sales_df = pd.read_csv(top20_sales_path, index_col="Punk",␣
 ↪parse_dates=True, infer_datetime_format=True)

# Display tem values for dataframes
display(cryptopunk_owners_df.head())
display(top20_sales_df.head())
```

```
           Account    OpenSea / ENS  Number Owned    last Active
#
1  0xb7f7f6c52f2e2  WrappedCryptoPu           428    7 hours ago
2  0xa858ddc0445d8              NaN           423    1 month ago
3  0xa25803ab86a32       wilcox.eth           238    28 days ago
4  0xb88f61e6fbda8              NaN           215  11 months ago
5  0x577ebc5de943e              NaN           165     5 days ago
```

```
      Ether  EtherValueUSD_M      Date
Punk
5822   8000            23.70  02/12/22
7804   4200             7.57  03/11/21
3100   4200             7.58  03/11/22
5577   2500             7.70  02/09/22
4156   2500            10.26  12/09/21
```

[54]:

```
#plot top20 sales by Punk based on Ether
top20_sales_df.hvplot.scatter(
    x='EtherValueUSD_M',
    y='Ether',
    xlabel='Ether value in USD Millions',
    ylabel='Ether',
    rot=90,
    label='Top 20 Sales By Ether',
    by='Punk',
    height= 600,
    width = 1000
).opts(
    bgcolor='lightgray',
   #fontsize={'title': 16, 'labels': 14, 'xticks': 6, 'yticks': 12}
)
```

```
[54]: :NdOverlay   [Punk]
         :Scatter   [EtherValueUSD_M]   (Ether)
```

[55]:

```
# Plot top20 sales by Punk based on Ether
top20_sales_df.hvplot.table(
    x='EtherValueUSD_M',
    y='Ether',
    xlabel='Ether value in USD Millions',
    ylabel='Ether',
    rot=90,
    label='Top 20 Sales By Ether',
    by='Punk',
    height= 600,
    width = 1000
).opts(
    bgcolor='lightgray',
   #fontsize={'title': 16, 'labels': 14, 'xticks': 6, 'yticks': 12}
)
```

```
[55]: :Table   [Ether,EtherValueUSD_M,Date]
```

[56]:

```
#validate dataframe total owned is 10,000
cryptopunk_total_assets = cryptopunk_owners_df['Number Owned'].sum()
cryptopunk_total_assets
```

```
[56]: 10000
```

[57]:

```
#find mean number of NFTs owned per owner
cryptopunk_owners_mean = cryptopunk_owners_df['Number Owned'].mean()
cryptopunk_owners_mean
```

```
[57]: 2.914602156805596
```

[58]:

```
#top20 asset owners
top20_cryptopunk_owners = cryptopunk_owners_df.head(20)
top20_cryptopunk_owners
```

```
[58]:            Account    OpenSea / ENS  Number Owned    last Active
      #
      1    0xb7f7f6c52f2e2  WrappedCryptoPu           428    7 hours ago
      2    0xa858ddc0445d8              NaN           423    1 month ago
      3    0xa25803ab86a32       wilcox.eth           238    28 days ago
      4    0xb88f61e6fbda8              NaN           215  11 months ago
      5    0x577ebc5de943e              NaN           165     5 days ago
```

```
6   0x69021ae876958         sov.eth           146   6 months ago
7   0x26f744711ee9e             NaN           141    4 years ago
8   0x4084df8bf74ba             NaN            98            NaN
9   0x269616d549d7e             NaN            96     9 days ago
10  0x31a5ff62a1b2c             NaN            93    1 month ago
11  0x7174039818a41             NaN            89    3 years ago
12  0xcc7c335f3365a             NaN            87    13 days ago
13  0x51688cd36c188             NaN            79     6 days ago
14  0x810fdbc7e5cfe             NaN            77   13 hours ago
15  0xf5a4ba515dd36             NaN            75    1 month ago
16  0xcffc336e6d019             NaN            74   2 months ago
17  0x6f4a2d3a4f47f             NaN            70     9 days ago
18  0x062c5432107e3             NaN            68   3 months ago
19  0x7760e0243ca9b             NaN            66    3 years ago
20  0xdde8df9a7dc9f          Kenney            66   2 months ago
```

[59]:

```
#total amount of assets owned by the top20 owners
top20_owners_assets = cryptopunk_owners_df['Number Owned'].head(20).sum()
perc_top20_owners_assets = top20_owners_assets /cryptopunk_total_assets *100
print(f"The Top 20 owners own {top20_owners_assets} NFTs, which is␣
 ↪{perc_top20_owners_assets:.2f}% of total assets.")
```

```
The Top 20 owners own 2794 NFTs, which is 27.94% of total assets.
```

[60]:

```
#total amount of assets owned by the top100 owners
top100_owners_assets = cryptopunk_owners_df['Number Owned'].head(100).sum()
perc_top100_owners_assets = top100_owners_assets /cryptopunk_total_assets *100
print(f"The Top 100 owners own {top100_owners_assets} NFTs, which is␣
 ↪{perc_top100_owners_assets:.2f}% of total assets.")
```

```
The Top 100 owners own 4705 NFTs, which is 47.05% of total assets.
```

[61]:

```
#total amount of assets owned by the top20 owners
bot20_owners_assets = cryptopunk_owners_df['Number Owned'].tail(20).sum()
perc_bot20_owners_assets = bot20_owners_assets /cryptopunk_total_assets *100
print(f"The Bottom 20 owners own {bot20_owners_assets} NFTs, which is␣
 ↪{perc_bot20_owners_assets:.2f}% of total assets.")
```

```
The Bottom 20 owners own 20 NFTs, which is 0.20% of total assets.
```

5. Discussion

Combining our data and discussing our findings

6. Postmoterm

Did we find everything we expected to find?

- Our difficulties and how we dealt with them
- Additional questions that came up that we didn’t answer
- What would we research next if we had more time?

7. Questions

Open floor Q&A with the audience

thank you,