KILN Case Study

Introduction

Overview

This project aims to analyze and visualize monthly rewards for various cryptocurrencies. Leveraging Kiln API and data scraping techniques, we process rewards for cryptocurrencies such as Ethereum, Solana, Cardano, Near, Polygon, Cosmos, and Kusama.

Objectives

Retrieve reward data for each cryptocurrency. Calculate the total rewards in USD for each cryptocurrency. Analyze and compare rewards across different cryptocurrencies and pools. Methodology

I- Initial Setup: Establishing API keys and endpoints necessary to access reward data.

II- Cryptocurrency-Specific Functions: Creating functions to process rewards from each blockchain. Data Processing: Using Python to manipulate and prepare data for analysis.

III- Analysis and Visualization: Analyzing collected data and creating visualizations to represent the rewards. Key Features

API Requests: Utilizing API requests to fetch reward data from different cryptocurrencies. Kusama Data Handling: Specific processing of Kusama data using a downloaded CSV file.

IV- Cryptocurrency Identification: Determining the type of cryptocurrency based on wallet address.

V- Total Rewards Calculation: Calculating total rewards in USD for a specified period.

VI- Data Grouping and Analysis: Grouping data by cryptocurrency and pool (specifically for ADA) and analyzing total rewards.

VII- Results Visualization: Creating charts to illustrate the distribution of rewards among different cryptocurrencies.

Technologies Used

Python: Main language for data processing. (Jupyter) Pandas: For data manipulation. Matplotlib: For data visualization. Kiln API: For accessing reward data.

This project provides a comprehensive analysis of cryptocurrency rewards, offering insights into the performance and potential of these digital assets.

In [1]:
```
import requests
import pandas as pd
import csv
from datetime import datetime
```

I- Global API key and API endpoints

In [2]:
```
API_KEY = 
'kiln_UHppSzJVcUk5UWNtTjIxNEpOTk1xM0xKNXBOYk5kWmg6X3ZReHZfUEZSWmh2d252bGpWbk11UlpTMEh

API = {
    'eth': 'https://api.kiln.fi/v1/eth/rewards',
    'sol': 'https://api.kiln.fi/v1/sol/rewards',
    'ada': 'https://api.kiln.fi/v1/ada/rewards',
    'near': 'https://api.kiln.fi/v1/near/rewards',
    'polygon': 'https://api.kiln.fi/v1/matic/rewards',
    'cosmos': 'https://api.kiln.fi/v1/atom/rewards',
    'atom':'https://api.kiln.fi/v1/atom/rewards',
      }
```

II-Generic function for API requests

In [3]:

```
def send_api_request(crypto_symbol, params):
    api_url = API.get(crypto_symbol)  # Utilisez 'API' au lieu de 'API_URLS'
    if not api_url:
        print(f"No API URL found for {crypto_symbol}")
        return None

    headers = {'Authorization': f'Bearer {API_KEY}'}
    response = requests.get(api_url, headers=headers, params=params)
    if response.status_code == 200:
        return response.json()
    else:
        print(f"Error retrieving rewards: {response.status_code}")
        print(response.text)
        return None
```

III- Functions for specific blockchain APIs

Function for Cosmos

In [4]:

```
def get_cosmos_rewards(wallet, start_date, end_date, validators):
    params = {'wallets': wallet, 'start_date': start_date, 'end_date': end_date, 'include_usd': 1, 'validators': validators}
    return send_api_request('atom', params)
```

Function for Near

In [5]:

```
def get_near_rewards(wallet, start_date, end_date):
    params = {'wallets': wallet, 'start_date': start_date, 'end_date': end_date, 'include_usd': 1}
    return send_api_request('near', params)
```

Function for Solana (sol)

In [6]:

```
def get_solana_rewards(wallet, start_date, end_date):
    params = {'stake_accounts': wallet, 'start_date': start_date, 'end_date': end_date, 'include_usd': 1}
    return send_api_request('sol', params)
```

Function for Ethereum (eth)

In [7]:

```
def get_ethereum_rewards(wallet, start_date, end_date):
    params = {'wallets': wallet, 'start_date': start_date, 'end_date': end_date, 'include_usd': 1}
    return send_api_request('eth', params)
```

Function for Cardano (ada)

In [8]:

```
def get_cardano_rewards(wallet, start_date, end_date):
    params = {'stake_addresses': wallet, 'start_date': start_date, 'end_date': end_date, 'include_usd': 1}
    return send_api_request('ada', params)
```

Function for Polygon (MATIC)

In [9]:

```
def get_polygon_rewards(wallet, start_date, end_date):
    params = {'wallets': wallet, 'start_date': start_date, 'end_date': end_date,
'include_usd': 1}
    return send_api_request('polygon', params)
```

Function For Kusama

We use the download of a file csv on this address on internet : https://kusama.subscan.io/account

In [10]:

```
def get_kusama_rewards(file_path, date_debut, date_fin):
    total_rewards = 0.0
    with open(file_path, newline='') as csvfile:
        reader = csv.DictReader(csvfile)
        for row in reader:
            date = datetime.strptime(row['Date'], '%Y-%m-%d %H:%M:%S')
            if date_debut <= date <= date_fin:
                total_rewards += float(row['Value'])
    return total_rewards
```

IV-Function to identify cryptocurrency symbol

In [11]:

```
def find_symbole_crypto(wallet_address):
    """Identify the cryptocurrency symbol based on the prefixed wallet address or
id."""
    # Split the wallet address into blockchain name and actual address
    parts = wallet_address.split('_')
    if len(parts) < 2:
        return 'unknown'  # Format non reconnu

    blockchain_name = parts[0].lower()  # Convertir le nom de la blockchain en
minuscules

    if blockchain_name == 'cosmos':
        return 'atom'
    elif blockchain_name == 'ethereum':
        return 'eth'
    elif blockchain_name == 'cardano':
        return 'ada'
    elif blockchain_name == 'solana':
        return 'sol'
    elif blockchain_name == 'matic' or blockchain_name == 'polygon':
        return 'polygon'
    elif blockchain_name == 'near':
        return 'near'
    elif blockchain_name == 'kusama':
        return 'kusama'
    else:
        return 'unknown'
```

V-This function allows us to calculate the amount of rewards considering the period

In [12]:

```
def calculate_total_rewards(rewards_data):
    """
    Calcule le gain total pour une crypto-monnaie sur la période spécifiée, en
utilisant le champ 'rewards'.

    :param rewards_data: Les données de récompense pour une crypto-monnaie
spécifique.
```

```
    :return: Le gain total en unités de 'rewards'.
    """
    total_reward = 0.0

    # Parcourir chaque entrée dans les données de récompense et additionner les
'rewards'
    for reward_entry in rewards_data['data']:
        reward_amount = int(reward_entry.get('rewards_usd', '0'))
        total_reward += reward_amount

    return total_reward
```

VI-Data Grouping and Analysis

In [13]:

```
def mainscraping(stake_accounts_df, start_date, end_date):
    """Main function to process the wallet addresses and fetch rewards."""

    rewards_list = []

    for index, row in stake_accounts_df.iterrows():
        wallet = row['address']
        identity = row['id']

        crypto_symbol = find_symbole_crypto(identity)

        # Handling Kusama separately
        if crypto_symbol == 'kusama':
            fichier_csv =
'C:\\Users\\khafif\\Desktop\\Kiln\\kusama\\kusamaCy9R9w9WFgwfs6s3bZy2tPc3KTU3MkmXmV6

            start_date1 = datetime.strptime(start_date, '%Y-%m-%d')
            end_date1 = datetime.strptime(end_date, '%Y-%m-%d')
            total_rewards = get_kusama_rewards(fichier_csv, start_date1,
end_date1)*50.12     # 50,12 it's the price of Kusama the 01/01/2024 we can also
use a dynamique price
        else:
            # Handling other cryptocurrencies
            if crypto_symbol == 'atom':
                validators = 'cosmosvaloper1uxlf7mvr8nep3gm7udf2u9remms2jyjqvwdul2'
                rewards = get_cosmos_rewards(wallet, start_date, end_date,
validators)
            elif crypto_symbol == 'eth':
                rewards = get_ethereum_rewards(wallet, start_date, end_date)
            elif crypto_symbol == 'sol':
                rewards = get_solana_rewards(wallet, start_date, end_date)
            elif crypto_symbol == 'ada':
                rewards = get_cardano_rewards(wallet, start_date, end_date)
            elif crypto_symbol == 'polygon':
                rewards = get_polygon_rewards(wallet, start_date, end_date)
            elif crypto_symbol == 'near':
                rewards = get_near_rewards(wallet, start_date, end_date)
            else:
                print(f"Unknown cryptocurrency for wallet: {wallet}")
                continue

            total_rewards = calculate_total_rewards(rewards) if rewards else 0

        # Add the information to the list
        rewards_list.append({
            'crypto': crypto_symbol,
            'address': wallet,
            'Monthly gross rewards in usd': total_rewards
        })

    # Convert the list to a DataFrame
    rewards_df = pd.DataFrame(rewards_list)
    print(rewards_df)

    return rewards_df
```

In [14]:
```
start_date = '2023-11-01'
end_date = '2023-11-30'
file_path = "C:/Users/khafif/Desktop/Kiln/[EXTERNAL] _ Minitel.wft Reporting - November 2023.xlsx"
stake_accounts_df = pd.read_excel(file_path, sheet_name='Stake accounts')
rewards_df = mainscraping(stake_accounts_df,start_date,end_date)
```

```
     crypto                                            address  \
0       ada  stake1u8j53lkzw5tv4p08am6uunwrjzrvpzfas8xzxcep...
1       ada  stake1uxshuuepjhaewd7ch8th96za0fj06t9plzjymjnr...
2      atom      cosmos1mfdn23y2ydnp6j3l3f8rw6r2gzazrmprgxn5xl
3       eth  0x807b7b004f582eb32ef767d3ea61a1992c1ce18c3034...
4       eth  0x809d9018817a7ebb25e3ec147631c799bf9d9fb73d6d...
..      ...                                                ...
141     sol       ABPPHUTB9vY2TuQZkLptKzkCDHUdaDAjXFWq8QeM8wob
142     sol       GqoH1myiruWecSFPXNXXMkCKuf1zTnXoZ4mei1xcKvun
143  kusama    Cy9R9w9WFgwfs6s3bZy2tPc3KTU3MkmXmV6VcSH8J2zmcav
144    near  bc2b5d51963545ec8ca28605fd013c65646d761e15b897...
145    near  e11503055fb40ccbd70423078b62d31f1f205d14c2cc52...

     Monthly gross rewards in usd
0                    62032.000000
1                    11461.000000
2                   144474.000000
3                        0.000000
4                        0.000000
..                            ...
141                  10021.000000
142                  92548.000000
143                  25096.688222
144                  15560.000000
145                   3616.000000

[146 rows x 3 columns]
```

In [15]:
```
rewards_df.tail()
```

Out[15]:

	crypto	address	Monthly gross rewards in usd
141	sol	ABPPHUTB9vY2TuQZkLptKzkCDHUdaDAjXFWq8QeM8wob	10021.000000
142	sol	GqoH1myiruWecSFPXNXXMkCKuf1zTnXoZ4mei1xcKvun	92548.000000
143	kusama	Cy9R9w9WFgwfs6s3bZy2tPc3KTU3MkmXmV6VcSH8J2zmcav	25096.688222
144	near	bc2b5d51963545ec8ca28605fd013c65646d761e15b897...	15560.000000
145	near	e11503055fb40ccbd70423078b62d31f1f205d14c2cc52...	3616.000000

Now we want the total amount by crypto in the Portfolio of Minitel

In [16]:
```
total_rewards_by_crypto = rewards_df.groupby('crypto')['Monthly gross rewards in usd'].sum().reset_index()
```

```
total_rewards_by_crypto.columns = ['Crypto', 'Total Monthly Gross Rewards in USD']
total_rewards_by_crypto.head()
```

Out[16]:

	Crypto	Total Monthly Gross Rewards in USD
0	ada	73493.000000
1	atom	144474.000000
2	eth	0.000000
3	kusama	25096.688222
4	near	19176.000000

We want seperate for ada Kiln0 and Kiln1 for the pool

In [17]:

```
rewards_df['Pool_id'] = stake_accounts_df['Pool_id']

# Ajouter une nouvelle colonne pour le regroupement
rewards_df['Crypto_Group'] = rewards_df.apply(
    lambda row: row['crypto'] + ' ' + row['Pool_id'] if row['crypto'] == 'ada' else 
row['crypto'], axis=1
)

# Regrouper par la nouvelle colonne et calculer la somme
total_rewards_by_group = rewards_df.groupby('Crypto_Group')['Monthly gross rewards 
in usd'].sum().reset_index()

# Renommer les colonnes pour plus de clarté
total_rewards_by_group.columns = ['Crypto_Group', 'Total Monthly Gross Rewards in 
USD']

print(total_rewards_by_group)
```

```
  Crypto_Group  Total Monthly Gross Rewards in USD
0    ada Kiln0                        11461.000000
1    ada Kiln1                        62032.000000
2         atom                       144474.000000
3          eth                            0.000000
4       kusama                        25096.688222
5         near                        19176.000000
6      polygon                        59423.000000
7          sol                       390241.000000
```

VII- Results Visualization

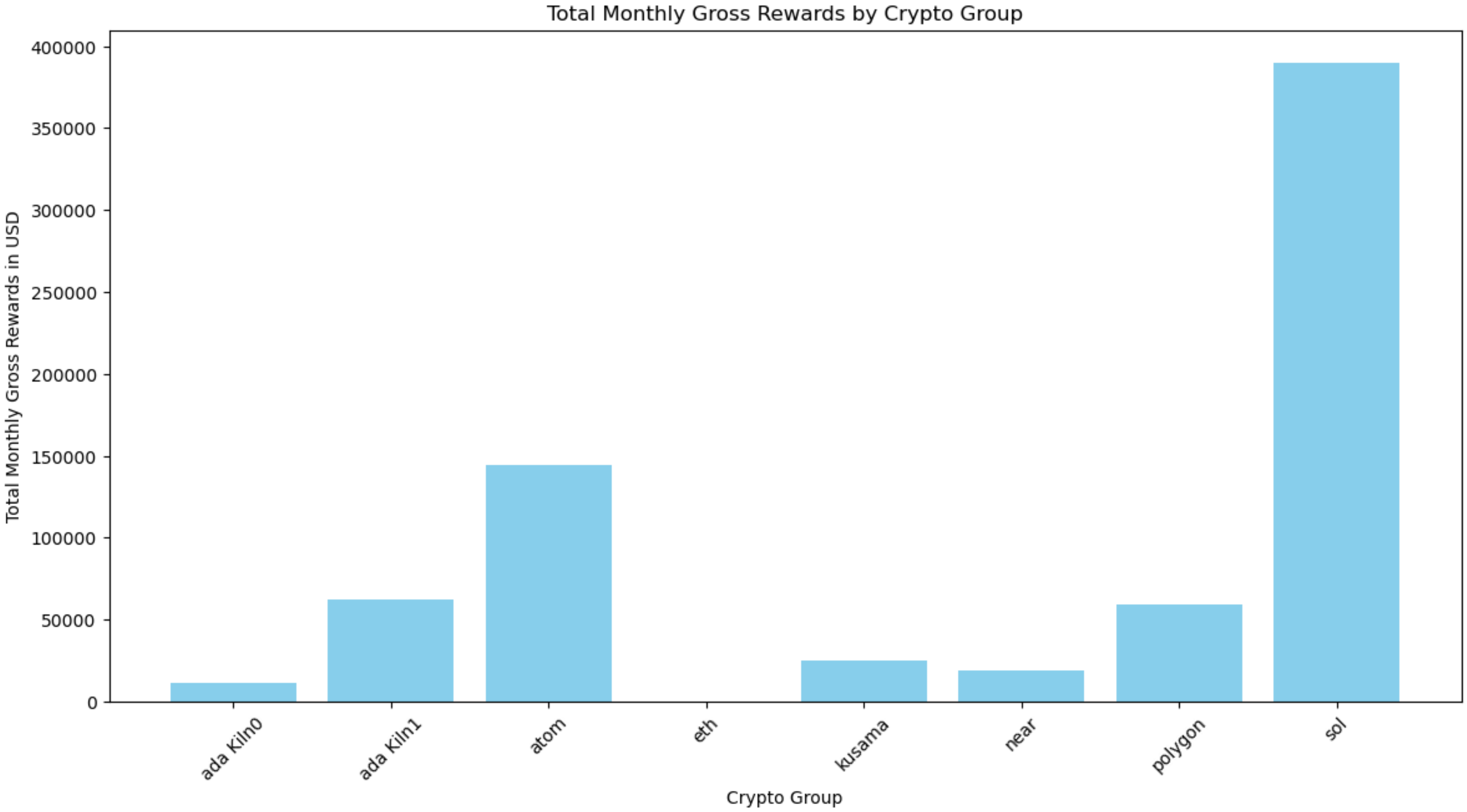
In [18]:

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

In [19]:

```
# Générer des couleurs aléatoires pour chaque barre
colors = plt.cm.viridis(np.linspace(0, 1, 
len(total_rewards_by_group['Crypto_Group'])))

plt.figure(figsize=(14, 7))
plt.bar(total_rewards_by_group['Crypto_Group'], total_rewards_by_group['Total 
Monthly Gross Rewards in USD'], color='skyblue')
plt.xlabel('Crypto Group')
plt.ylabel('Total Monthly Gross Rewards in USD')
plt.title('Total Monthly Gross Rewards by Crypto Group')
plt.xticks(rotation=45)
plt.show()
```


Total Monthly Gross Rewards by Crypto Group
Total Monthly Gross Rewards in USD
400000
350000
300000
250000
200000
150000
100000
50000
0
ada Kiln0
ada Kiln1
atom
eth
kusama
near
polygon
sol
Crypto Group