Week 1

NGUYEN, Le Hung
{andrwngyn,junhao}@postech.ac.kr
Blockchain and Cryptocurrency
DPNM. Lab
Pohang University of Science and Technology

Topic and Objective :

- **Understanding the Project and Setting Up Basics**
- **Acquire an overview of the project and basic knowledge.**

Activities **:**

1. **Explain project overview and set goals.**
2. **Learn basic concepts of cryptocurrencies and trading strategies.**
3. **Set up the required development environment (Python, Jupyter Notebook, relevant libraries).**
4. **Perform simple cryptocurrency data retrieval and analysis.**

Abstract - ***The cryptocurrency market has grown exponentially over the past decade, characterized by its volatility and the rapid pace of innovation. This report aims to provide an overview of the key trading strategies employed in the cryptocurrency market, analyzing their methodologies, advantages, risks, and suitability for different types of investors.***

I. Basic concepts of cryptocurrency and trading strategies

1.1 Definition

Cryptocurrency is a digital or virtual currency that uses cryptography for security. Cryptography is a process that translates legible information into codes that cannot be broken at all. Blockchain records individual transactions and ownership of all cryptocurrencies that are in circulation.[1]

Bitcoin was the first cryptocurrency that emerged in 2009, created by an anonymous person, Satoshi Nakamoto. It didn't get the attention that it deserved back then, but now it is impossible to get around it. Some people sold their houses and properties just to buy Bitcoins with the expectation of waking up rich the next morning. There are more than thousand different cryptocurrencies that can be bought, but here are some of the most valuable that have the biggest capital on the market.[2]

1. Bitcoin
2. Ethereum
3. Ripple
4. Litecoin

The fact about these cryptocurrencies is that it is very difficult for the government and the law enforcement to control them, especially the Bitcoin is that it can't be controlled by no server or any authority, it is completely safe and there is a bigger chance for the humanity to be wiped from the face of the Earth, than for a transaction or a user on this platform can be revealed.[2]

The idea was to create a digital cash system, that will work on the principle of peer-to-peer network and files would be shared like this. And that is how it became the cryptocurrency. So, there is no server that controls this platform, but every peer has to own a list that has all the transactions so it can see if some transactions in the future are valid. It is possible to simplify the definition of cryptocurrencies as *limited entries in a database no one can change without fulfilling specific conditions*.[3]

1.2 Trading strategies

Cryptocurrency trading involves buying and selling digital assets such as Bitcoin, Ethereum, and other altcoins with the goal of generating profit. The market operates 24/7, providing ample opportunities for traders. However, its high volatility also means increased risk. Strategies in this market range from long-term investments to high-frequency trading, each with its unique approach and risk profile.

No.	Name	Methodology	Advantages	Risks	Suitability
1	HOLD	-HOLDing is a long-term investment strategy where investors buy cryptocurrencies and hold onto them regardless of market fluctuations.	-Simplicity and low maintenance. -Avoids the stress and complexity of market timing. -Potential for significant returns if the asset appreciates over time.	-Exposure to long-term market downturns. -Potentially missing out on profit-taking opportunities during market highs.	-Ideal for investors who believe in the long-term success of specific cryptocurrencies. -Suitable for those with a lower risk

					tolerance and less time to actively manage investments.
2	Day Trading	-Involves buying and selling cryptocurrencies within the same day to take advantage of short-term price movements. -Relies on technical analysis, chart patterns, and market news	-Potential for high returns due to frequent trades. -Capitalizes on daily market volatility.	-Requires constant monitoring of the market. -High transaction fees can erode profits. -High risk due to market unpredictability.	-Suitable for experienced traders with a deep understanding of the market. -Best for those who can dedicate significant time to trading activities.
3	Swing Trading	-Holding positions for several days to weeks to capitalize on expected upward or downward market trends. -Utilizes both technical and fundamental analysis to identify entry and exit points.	-Flexibility in holding periods allows traders to avoid daily market noise. -Opportunities for significant gains without the need for constant market monitoring.	-Exposure to overnight and weekend market movements. -Requires a good understanding of market trends and indicators.	-Suitable for traders who cannot dedicate full time to trading but still want to capitalize on medium-term trends. -Ideal for those with moderate risk tolerance.
4	Scalping	-Involves making numerous small trades throughout the day to profit from tiny price movements. -Requires use of automated trading bots and high-frequency trading techniques.	-Potential for steady, small profits. -Low exposure to market risk due to very short holding periods.	-Requires significant time and attention. -High transaction costs can negate small profits. -Risk of substantial loss if a large movement occurs unexpectedly.	-Best for highly experienced traders with access to advanced trading tools and platforms. -Suitable for those who can dedicate full time to trading and have a high risk tolerance.
5	Dollar-Cost Averaging (DCA)	-Investing a fixed amount of money at regular intervals, regardless of the cryptocurrency's price. -Reduces the impact of volatility by spreading out investments over time.	-Simple and low maintenance. -Reduces the risk of making poor investment decisions based on market timing. -Suitable for long-term investment goals.	-Potentially lower returns compared to lump-sum investing in a rising market. -Continued investment in a declining market can result in losses.	-Ideal for novice investors and those with a long-term perspective. -Suitable for individuals with a lower risk tolerance and limited time for active management.

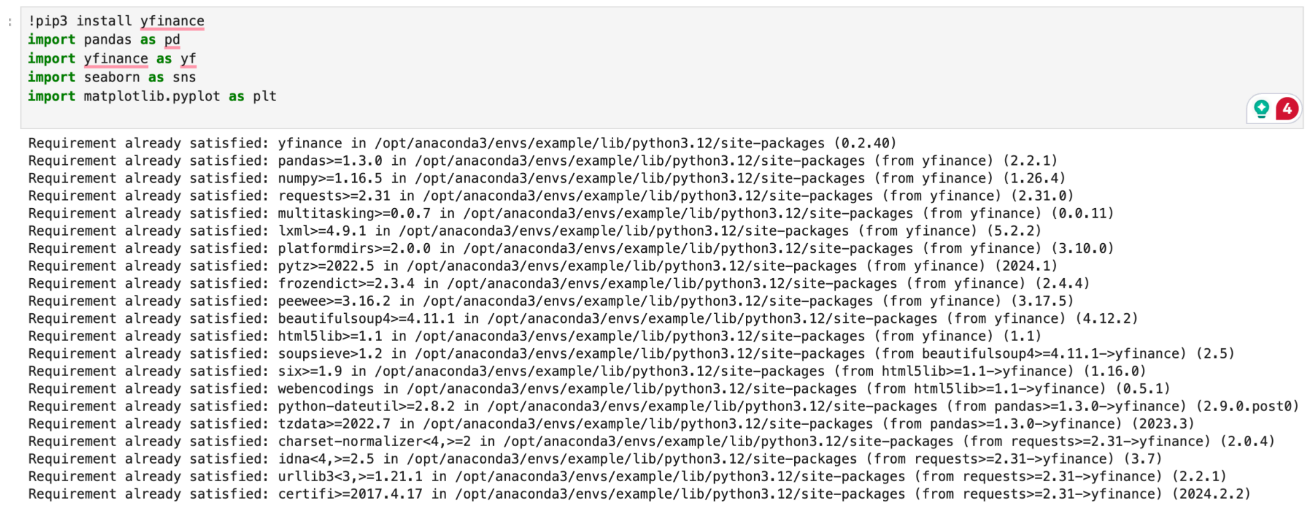
*Conclusion

The cryptocurrency market offers a wide range of trading strategies, each with its own set of advantages and risks. From long-term holding to high-frequency trading, investors can choose a strategy that aligns with their risk tolerance, market knowledge, and investment goals. Successful trading in the cryptocurrency market requires a thorough understanding of the chosen strategy, diligent market analysis, and disciplined risk management.

By diversifying strategies and continuously learning about market trends and technologies, traders can better navigate the volatile landscape of cryptocurrencies and potentially achieve their financial objectives.

II. Setting up the required development environment

Tools/Libraries	Role
Jupyter notbook	IDE for coding
Python version >= 3.1	Programming Language
pandas	Powerful and flexible data manipulation and analysis library for Python. It is particularly well-suited for working with structured data (like spreadsheets or SQL tables).
yfinance	A library that allows easy access to financial data from Yahoo Finance. It can be used to download historical market data, real-time tick data, and financial statements.
seaborn	A statistical data visualization library based on Matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.
matplotlib.pyplot	A plotting library used for creating static, interactive, and animated visualizations in Python. It is the core library upon which Seaborn is built.



```python
!pip3 install yfinance
import pandas as pd
import yfinance as yf
import seaborn as sns
import matplotlib.pyplot as plt
```

```
Requirement already satisfied: yfinance in /opt/anaconda3/envs/example/lib/python3.12/site-packages (0.2.40)
Requirement already satisfied: pandas>=1.3.0 in /opt/anaconda3/envs/example/lib/python3.12/site-packages (from yfinance) (2.2.1)
Requirement already satisfied: numpy>=1.16.5 in /opt/anaconda3/envs/example/lib/python3.12/site-packages (from yfinance) (1.26.4)
Requirement already satisfied: requests>=2.31 in /opt/anaconda3/envs/example/lib/python3.12/site-packages (from yfinance) (2.31.0)
Requirement already satisfied: multitasking>=0.0.7 in /opt/anaconda3/envs/example/lib/python3.12/site-packages (from yfinance) (0.0.11)
Requirement already satisfied: lxml>=4.9.1 in /opt/anaconda3/envs/example/lib/python3.12/site-packages (from yfinance) (5.2.2)
Requirement already satisfied: platformdirs>=2.0.0 in /opt/anaconda3/envs/example/lib/python3.12/site-packages (from yfinance) (3.10.0)
Requirement already satisfied: pytz>=2022.5 in /opt/anaconda3/envs/example/lib/python3.12/site-packages (from yfinance) (2024.1)
Requirement already satisfied: frozendict>=2.3.4 in /opt/anaconda3/envs/example/lib/python3.12/site-packages (from yfinance) (2.4.4)
Requirement already satisfied: peewee>=3.16.2 in /opt/anaconda3/envs/example/lib/python3.12/site-packages (from yfinance) (3.17.5)
Requirement already satisfied: beautifulsoup4>=4.11.1 in /opt/anaconda3/envs/example/lib/python3.12/site-packages (from yfinance) (4.12.2)
Requirement already satisfied: html5lib>=1.1 in /opt/anaconda3/envs/example/lib/python3.12/site-packages (from yfinance) (1.1)
Requirement already satisfied: soupsieve>1.2 in /opt/anaconda3/envs/example/lib/python3.12/site-packages (from beautifulsoup4>=4.11.1->yfinance) (2.5)
Requirement already satisfied: six>=1.9 in /opt/anaconda3/envs/example/lib/python3.12/site-packages (from html5lib>=1.1->yfinance) (1.16.0)
Requirement already satisfied: webencodings in /opt/anaconda3/envs/example/lib/python3.12/site-packages (from html5lib>=1.1->yfinance) (0.5.1)
Requirement already satisfied: python-dateutil>=2.8.2 in /opt/anaconda3/envs/example/lib/python3.12/site-packages (from pandas>=1.3.0->yfinance) (2.9.0.post0)
Requirement already satisfied: tzdata>=2022.7 in /opt/anaconda3/envs/example/lib/python3.12/site-packages (from pandas>=1.3.0->yfinance) (2023.3)
Requirement already satisfied: charset-normalizer<4,>=2 in /opt/anaconda3/envs/example/lib/python3.12/site-packages (from requests>=2.31->yfinance) (2.0.4)
Requirement already satisfied: idna<4,>=2.5 in /opt/anaconda3/envs/example/lib/python3.12/site-packages (from requests>=2.31->yfinance) (3.7)
Requirement already satisfied: urllib3<3,>=1.21.1 in /opt/anaconda3/envs/example/lib/python3.12/site-packages (from requests>=2.31->yfinance) (2.2.1)
Requirement already satisfied: certifi>=2017.4.17 in /opt/anaconda3/envs/example/lib/python3.12/site-packages (from requests>=2.31->yfinance) (2024.2.2)
```

III. Perform simple cryptocurrency data retrieval and analysis

3.1 Simple data collection and cleaning

This report will perform simple cryptocurrency data retrieval and analysis using mentioned tools and libraries above.

My analysis will select 4 popular cryptocurrencies (cryptocoins) on the crypto market which are:

1. Bitcoin
2. Ethereum
3. Dogecoin
4. Polkadot

```
cryptocurrencies = ['DOGE-USD', 'BTC-USD', 'DOT-USD', 'ETH-USD']
```

Then the data will be collected by using finance library with below syntax in Python

```
data = yf.download(cryptocurrencies,start= '2022-01-01', end ='2024-06-26')
```

```
[*********************100%***********************]  4 of 4 completed
```

```
data.head()
```

The raw data will look like this

Price	Adj Close				Close				High		...	Low		
Ticker	BTC-USD	DOGE-USD	DOT-USD	ETH-USD	BTC-USD	DOGE-USD	DOT-USD	ETH-USD	BTC-USD	DOGE-USD	...	DOT-USD	ETH-USD	BTC-USD
Date														
2022-01-01	47686.812500	0.173035	28.583582	3769.697021	47686.812500	0.173035	28.583582	3769.697021	47827.312500	0.173423	...	26.705992	3682.286621	46311.746094
2022-01-02	47345.218750	0.174403	29.731167	3829.564941	47345.218750	0.174403	29.731167	3829.564941	47881.406250	0.175989	...	27.922352	3727.357422	47680.925781
2022-01-03	46458.117188	0.170088	30.105101	3761.380371	46458.117188	0.170088	30.105101	3761.380371	47510.726562	0.174406	...	28.751802	3698.047607	47343.542969
2022-01-04	45897.574219	0.168803	28.777731	3794.056641	45897.574219	0.168803	28.777731	3794.056641	47406.546875	0.172339	...	28.777731	3723.349854	46458.851562
2022-01-05	43569.003906	0.159420	26.796141	3550.386963	43569.003906	0.159420	26.796141	3550.386963	46929.046875	0.170747	...	25.521626	3456.745361	45899.359375

5 rows × 24 columns

For this analysis, I only chose the 'Adj Close' price for these 4 cryptocoins. This amount of data will help me clean and analyze better since this is the first-time try

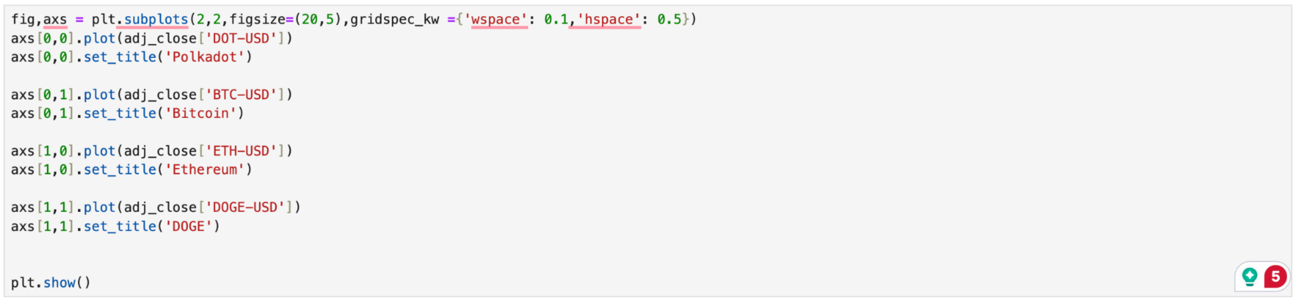
```
adj_close = data['Adj Close']
adj_close.head()
```

The raw data after simple clean technique

Ticker	BTC-USD	DOGE-USD	DOT-USD	ETH-USD
Date				
2022-01-01	47686.812500	0.173035	28.583582	3769.697021
2022-01-02	47345.218750	0.174403	29.731167	3829.564941
2022-01-03	46458.117188	0.170088	30.105101	3761.380371
2022-01-04	45897.574219	0.168803	28.777731	3794.056641
2022-01-05	43569.003906	0.159420	26.796141	3550.386963

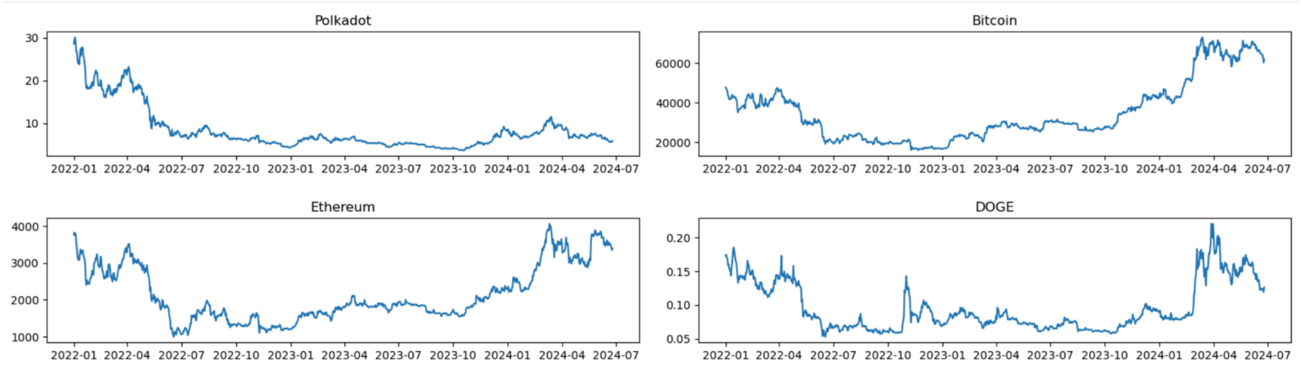
3.2 Data visualization

For better data visualization, we will represent line charts for processed data using the plot function in Matplotlib library, which connects data points with lines.

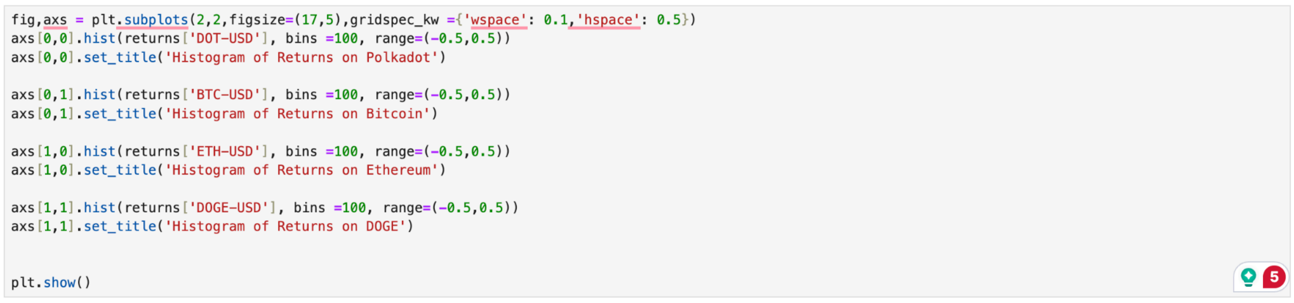


```
fig,axs = plt.subplots(2,2,figsize=(20,5),gridspec_kw ={'wspace': 0.1,'hspace': 0.5})
axs[0,0].plot(adj_close['DOT-USD'])
axs[0,0].set_title('Polkadot')

axs[0,1].plot(adj_close['BTC-USD'])
axs[0,1].set_title('Bitcoin')

axs[1,0].plot(adj_close['ETH-USD'])
axs[1,0].set_title('Ethereum')

axs[1,1].plot(adj_close['DOGE-USD'])
axs[1,1].set_title('DOGE')


plt.show()
```

Line charts

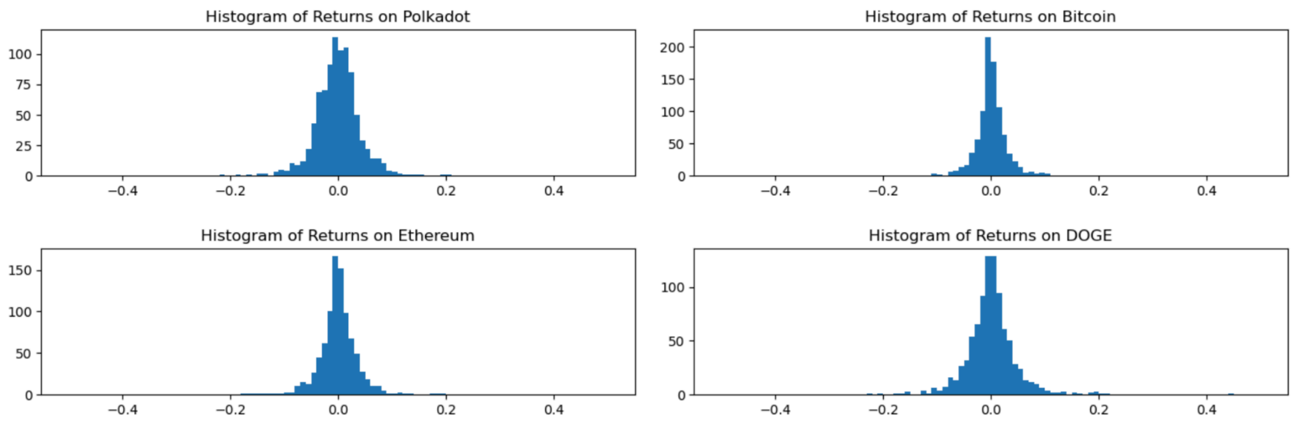


For barchart, we used the hist function



```
fig,axs = plt.subplots(2,2,figsize=(17,5),gridspec_kw ={'wspace': 0.1,'hspace': 0.5})
axs[0,0].hist(returns['DOT-USD'], bins =100, range=(-0.5,0.5))
axs[0,0].set_title('Histogram of Returns on Polkadot')

axs[0,1].hist(returns['BTC-USD'], bins =100, range=(-0.5,0.5))
axs[0,1].set_title('Histogram of Returns on Bitcoin')

axs[1,0].hist(returns['ETH-USD'], bins =100, range=(-0.5,0.5))
axs[1,0].set_title('Histogram of Returns on Ethereum')

axs[1,1].hist(returns['DOGE-USD'], bins =100, range=(-0.5,0.5))
axs[1,1].set_title('Histogram of Returns on DOGE')


plt.show()
```

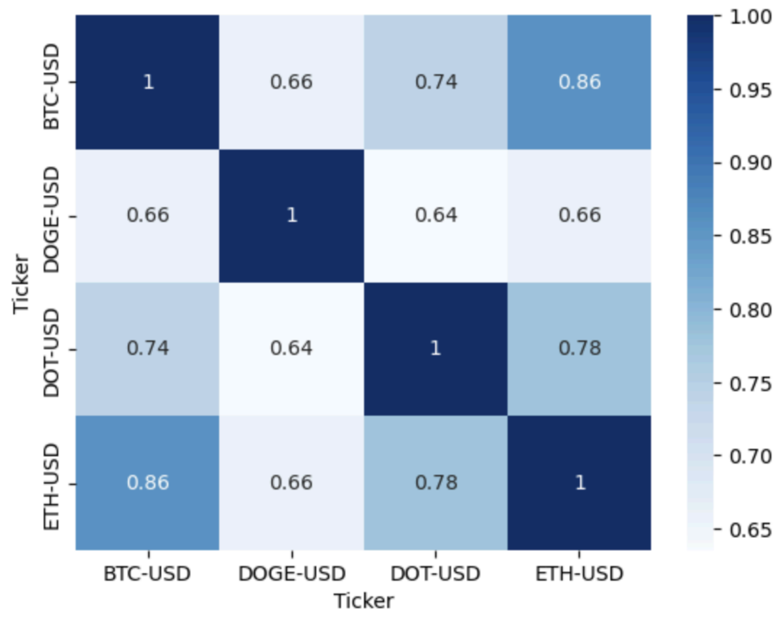
Histogram



For heatchart, we used the heatmap function

```
corr = returns.corr()
sns.heatmap(corr, annot = True, cmap='Blues')
```

```
<Axes: xlabel='Ticker', ylabel='Ticker'>
```



3.3 Data analysis

Market Behavior	All four cryptocurrencies showed significant declines during the first half of 2022, suggesting a broad market downturn or negative sentiment in the crypto market.
Recovery	Both Bitcoin and Ethereum showed signs of recovery starting from mid-2022 and peaking in early 2024, indicating investor confidence returning to major cryptocurrencies.
Volatility	Dogecoin exhibited the most volatility among the four, typical of its speculative nature and market sentiment-driven price changes.
Correlation	There appears to be a correlation between the trends of Bitcoin and Ethereum, reflecting their status as major market influencers in the cryptocurrency space.

Conclusion:

- Polkadot and Dogecoin show more sustained downward trends with occasional spikes, reflecting higher risk and possibly lower investor confidence.
- Bitcoin and Ethereum show more robust recovery patterns, making them potentially more stable investment options in the long term.

REFERENCES

[1] Monia Milutinović. "CRYPTOCURRENCY". Економика - Часопис за економску теорију и праксу и друштвена питања 1:105. (The Central and Eastern European Online Library)
[2][3]Monia Milutinović. "CRYPTOCURRENCY". Економика - Часопис за економску теорију и праксу и друштвена питања 1:106-108

[4]yfinance, Yahoo Finance crypto coin data market, Coin Market

Week 2

NGUYEN, Le Hung - LIM, Jun Hao

{andrwngyn,junhao}@postech.ac.kr
Blockchain and Cryptocurrency
DPNM. Lab
Pohang University of Science and Technology

Topic and Objective :

- **Data Collection and Database Building**
- **Collect data from crypto exchanges and design the database.**

***Activities* :**

5. **Learn how to use major crypto exchange APIs (e.g., Binance, Coinbase).**
6. **Write scripts for data collection.**
7. **Design and build a database to store the collected data (e.g., PostgreSQL, MongoDB).**
8. **Load initial data into the database.**

Abstract - *This report presents a comprehensive approach to data collection and database building for cryptocurrency market analysis. The primary objective is to gather data from major crypto exchanges and design a robust database to store and manage this data effectively.*

I. APIs of crypto exchange (Binance)

1.1 Definition

An API, which stands for **application programming interface**, is a set of protocols that enable different software components to communicate and transfer data. Developers use APIs to bridge the gaps between small, discrete chunks of code in order to create applications that are powerful, resilient, secure, and able to meet user needs. Even though you can't see them, APIs are everywhere—working continuously in the background to power the digital experiences that are essential to our modern lives. (For fully understanding, please go to url API article of Postman)

1.2 Binance API

Binance API is a set of programming interfaces provided by Binance, one of the world's largest cryptocurrency exchanges, which allows developers to interact with Binance's trading platform programmatically. The API enables users to access various functionalities of the Binance platform, such as retrieving market data, executing trades, managing accounts, and more.

Key Features of Binance API

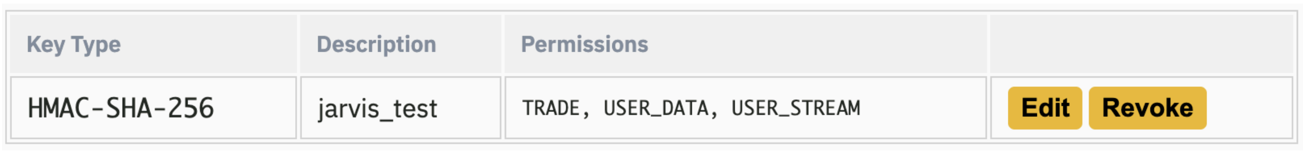
- Spot market data
- Trading
- Account management
- WebSocket stream
- User data stream

Types of APIs	**Function**
REST API	The REST API uses standard HTTP requests to interact with the Binance platform. It is used for fetching data, managing accounts, and executing trades. Common endpoints include `/api/v3/ticker/price`, `/api/v3/order`, `/api/v3/account`, etc.
WebSocket API	The WebSocket API provides real-time updates. It is ideal for applications that need live data feeds, such as trading bots and real-time dashboards. Common streams include `!ticker@arr`, `trade`, `depth`, etc.

Authentication

To access private endpoints (such as trading and account management), users need to authenticate using an API key and secret key, which can be generated from their Binance account.

In this report, I will use an alternative API to create a trading-bot using testnet



Key Type	Description	Permissions	
HMAC-SHA-256	jarvis_test	TRADE, USER_DATA, USER_STREAM	Edit Revoke

II. Write script to collect data

1.1 Setting up required environment

*main source code was written by Jun Hao

Tools/Libraries	**Role**

Jupyter notbook	IDE for coding
Python version >= 3.1	Programming Language
configparser	The `configparser` module in Python is used for working with configuration files. These files store settings in a simple, readable format, which helps to manage and organize different configurations, such as database credentials and API keys.
binance.client	The `binance.client` module is part of the official Binance API Python wrapper. It allows developers to interact with the Binance cryptocurrency exchange programmatically, enabling actions like fetching market data, placing orders, and retrieving account information.
pandas	`pandas` is a powerful data manipulation and analysis library in Python. It provides data structures like DataFrames and Series, which allow for efficient data processing and analysis.
psycopg2	`psycopg2` is a PostgreSQL adapter for Python. It allows Python code to execute PostgreSQL commands in a database session. It provides a connection to the database and a cursor to execute SQL queries.

1.2 Design database for storing data fetching from Binance

The purpose of this part is to explain the process of connecting to a PostgreSQL database, creating a table for storing cryptocurrency price data fetched from the Binance API, and ensuring data integrity by adding constraints.

1. Reading Configuration Settings

The script reads the database connection details from a configuration file and accessing PostgreSQL.

```
import configparser
import psycopg2
# Read configuration settings
config = configparser.ConfigParser()
config.read('config.ini')

# Access the PostgreSQL section
db_params = config['postgresql']
```

2. Connecting to PostgreSQL

   The script attempts to establish a connection to the PostgreSQL database using the extracted parameters.

```
# Connect to PostgreSQL
try:
    conn = psycopg2.connect(
        dbname=db_params['dbname'],
        user=db_params['user'],
        password=db_params['password'],
        host=db_params['host'],
        port=db_params['port']
    )
    cur = conn.cursor()
    print("Connected to PostgreSQL")
except Exception as e:
    print(f"Error connecting to PostgreSQL: {e}")
    raise
```

3. Design table to store data from Binance

   The script creates a table named binance_crypto_prices to store cryptocurrency price data, including an auto-incrementing id column and various columns for price data.

```
# Create table if it doesn't exist, including an auto-incrementing id column
create_table_query = """
CREATE TABLE IF NOT EXISTS binance_crypto_prices (
    id SERIAL PRIMARY KEY,
    crypto_id VARCHAR(10),
    date TIMESTAMP,
    open NUMERIC,
    high NUMERIC,
    low NUMERIC,
    close NUMERIC,
    volume NUMERIC,
    quote_asset_volume NUMERIC,
    number_of_trades INTEGER,
    taker_buy_base_asset_volume NUMERIC,
    taker_buy_quote_asset_volume NUMERIC
);
"""
add_constraint_query = """
ALTER TABLE binance_crypto_prices ADD CONSTRAINT unique_crypto_date UNIQUE (crypto_id, date);
"""
try:
    cur.execute(create_table_query)
    cur.execute(add_constraint_query)
    conn.commit()
    print("Table created successfully")
except Exception as e:
    print(f"Error creating table: {e}")
    conn.rollback()
```

1.3 Fetching data and load data into created database

This section explains how to collect historical cryptocurrency data using the Binance API and store it in a PostgreSQL database. This involves connecting to both the Binance API and PostgreSQL database, fetching the required data, and ensuring its correct insertion into the database.

1. Reading Configuration Settings

The script reads the API keys and database connection details from a configuration file and connect to PostgreSQL

```
# Read configuration settings
config = configparser.ConfigParser()
config.read('config.ini')

# Access the Binance section
api_key = config['binance']['api_key']
api_secret = config['binance']['api_secret']


# Access the PostgreSQL section
db_params = config['postgresql']

conn = psycopg2.connect(
    dbname=db_params['dbname'],
    user=db_params['user'],
    password=db_params['password'],
    host=db_params['host']
    )

cur = conn.cursor()
```

2. Initializing the Binance Client

The script initializes the Binance client with the API keys.

```
# Initialize the Binance client
client = Client(api_key, api_secret)
```

3. Fetching Historical Data

The script defines a function to fetch historical kline (candlestick) data from Binance.

```
# Function to get historical klines
def get_historical_klines(symbol, interval, start, end=None):
    klines = client.get_historical_klines(symbol, interval, start, end)
    data = pd.DataFrame(klines, columns=[
        'timestamp', 'open', 'high', 'low', 'close', 'volume',
        'close_time', 'quote_asset_volume', 'number_of_trades',
        'taker_buy_base_asset_volume', 'taker_buy_quote_asset_volume', 'ignore'
    ])
    data['timestamp'] = pd.to_datetime(data['timestamp'], unit='ms')
    data.set_index('timestamp', inplace=True)
    data = data.astype(float)
    return data
```

```
# Fetch historical data for BTC and ETH
btc_data = get_historical_klines('BTCUSDT', Client.KLINE_INTERVAL_1DAY, '1 Jan 2023')
eth_data = get_historical_klines('ETHUSDT', Client.KLINE_INTERVAL_1DAY, '1 Jan 2023')
```

4. Preparing data for inserting

The script combines the BTC and ETH data into a single DataFrame and prepares it for database insertion.

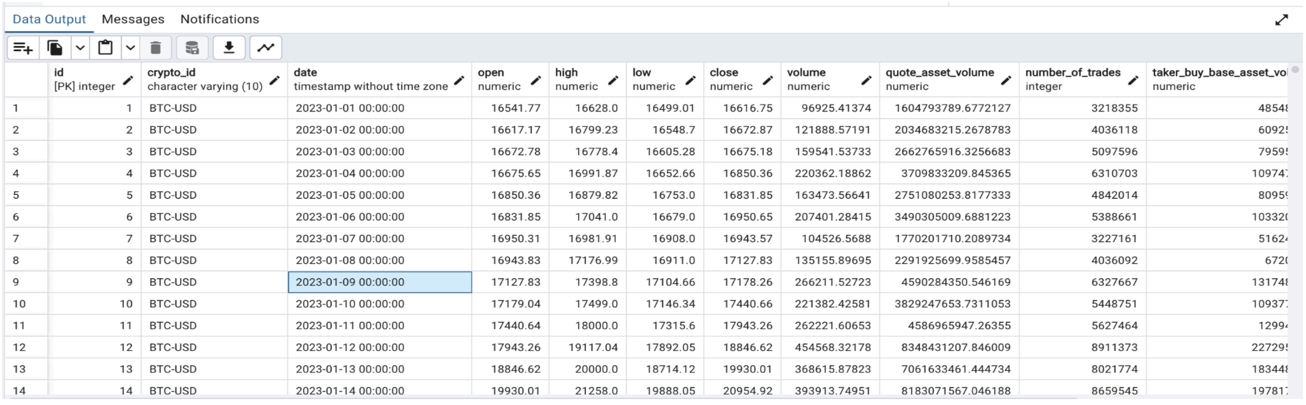
```
# Combine data into a single DataFrame with a 'crypto_id' column
btc_data['crypto_id'] = 'BTC-USD'
eth_data['crypto_id'] = 'ETH-USD'
data = pd.concat([btc_data, eth_data])

data.head()

print(data.head())


# Reset the index to flatten the DataFrame
data.reset_index(inplace=True)

# Rename columns to match SQL table
data = data[['crypto_id', 'timestamp', 'open', 'high', 'low', 'close', 'volume', 'quote_asset_volume', 'number_of_trades', 'taker_buy_base_asset_volume', 'take

# Rename timestamp column to date
data.rename(columns={'timestamp': 'date'}, inplace=True)
```

5. Inserting Data into PostgreSQL

The script inserts the prepared data into the PostgreSQL table.

```
# Insert data into PostgreSQL
insert_query = """
    INSERT INTO binance_crypto_prices (crypto_id, date, open, high, low, close, volume, quote_asset_volume, number_of_trades, taker_buy_base_asset_volume, tak
    VALUES (%s, %s, %s, %s, %s, %s, %s, %s, %s, %s, %s)
    ON CONFLICT (crypto_id, date) DO NOTHING;
"""

# Use a try-except block to catch any errors
try:
    for index, row in data.iterrows():
        cur.execute(insert_query, (
            row['crypto_id'], row['date'], row['open'], row['high'], row['low'], row['close'],
            row['volume'], row['quote_asset_volume'], row['number_of_trades'],
            row['taker_buy_base_asset_volume'], row['taker_buy_quote_asset_volume']
        ))
    # Commit changes
    cur.execute("COMMIT")
    print("Data loaded into database successfully")
except Exception as e:
    print("Error loading data into database:", e)


cur.close()
```

***Final result**



Data Output | Messages | Notifications

	id [PK] integer	crypto_id character varying (10)	date timestamp without time zone	open numeric	high numeric	low numeric	close numeric	volume numeric	quote_asset_volume numeric	number_of_trades integer	taker_buy_base_asset_vol numeric
1	1	BTC-USD	2023-01-01 00:00:00	16541.77	16628.0	16499.01	16616.75	96925.41374	1604793789.6772127	3218355	4854
2	2	BTC-USD	2023-01-02 00:00:00	16617.17	16799.23	16548.7	16672.87	121888.57191	2034683215.2678783	4036118	6092
3	3	BTC-USD	2023-01-03 00:00:00	16672.78	16778.4	16605.28	16675.18	159541.53733	2662765916.3256683	5097596	7959
4	4	BTC-USD	2023-01-04 00:00:00	16675.65	16991.87	16652.66	16850.36	220362.18862	3709833209.845365	6310703	10974
5	5	BTC-USD	2023-01-05 00:00:00	16850.36	16879.82	16753.0	16831.85	163473.56641	2751080253.8177333	4842014	8095
6	6	BTC-USD	2023-01-06 00:00:00	16831.85	17041.0	16679.0	16950.65	207401.28415	3490305009.6881223	5388661	10332
7	7	BTC-USD	2023-01-07 00:00:00	16950.31	16981.91	16908.0	16943.57	104526.5688	1770201710.2089734	3227161	5162
8	8	BTC-USD	2023-01-08 00:00:00	16943.83	17176.99	16911.0	17127.83	135155.89695	2291925699.9585457	4036092	672
9	9	BTC-USD	2023-01-09 00:00:00	17127.83	17398.8	17104.66	17178.26	266211.52723	4590284350.546169	6327667	13174
10	10	BTC-USD	2023-01-10 00:00:00	17179.04	17499.0	17146.34	17440.66	221382.42581	3829247653.7311053	5448751	10937
11	11	BTC-USD	2023-01-11 00:00:00	17440.64	18000.0	17315.6	17943.26	262221.60653	4586965947.26355	5627464	1299
12	12	BTC-USD	2023-01-12 00:00:00	17943.26	19117.04	17892.05	18846.62	454568.32178	8348431207.846009	8911373	22729
13	13	BTC-USD	2023-01-13 00:00:00	18846.62	20000.0	18714.12	19930.01	368615.87823	7061633461.444734	8021774	18344
14	14	BTC-USD	2023-01-14 00:00:00	19930.01	21258.0	19888.05	20954.92	393913.74951	8183071567.046188	8659545	19781

Week 3

NGUYEN, Le Hung - LIM, Jun Hao
{andrwngyn,junhao}@postech.ac.kr
Blockchain and Cryptocurrency
DPNM. Lab

Pohang University of Science and Technology

Topic and Objective :

- **Designing and Implementing the Backtesting Platform**

Activities **:**

1. **- Understand the concept and importance of backtesting.**
2. **- Design the main components of the backtesting platform (data input, strategy application, result analysis).**
3. **- Implement basic backtesting functionalities.**
4. **- Apply and test simple trading strategies (e.g., moving average crossover strategy).**

Abstract ***-A key difference between a traditional investment management pro- cess and a quantitative investment process is the possibility of back- testing a quantitative investment strategy to see how it would have performed in the past.***

The difficulty is not the lack of ideas. The difficulty is to develop a taste for which strategy is suitable for your personal circum- stances and goals, and which ones look viable even before you de- vote the time to diligently backtest them

I. The importance and concept of backtesting

1.1 Definition

An API, which stands for **application programming interface**, is a set of protocols that enable different software components to communicate and transfer data. Developers use APIs to bridge the gaps between small, discrete chunks of code in order to create applications that are powerful, resilient, secure, and able to meet user needs. Even though you can't see them, APIs are everywhere—working continuously in the background to power the digital experiences that are essential to our modern lives. (For fully understanding, please go to url API article of Postman)

1.2 Binance API

Binance API is a set of programming interfaces provided by Binance, one of the world's largest cryptocurrency exchanges, which allows developers to interact with Binance's trading platform programmatically. The API enables users to access various functionalities of the Binance platform, such as retrieving market data, executing trades, managing accounts, and more.