GMTv1.02_BULLwithSniper

August 30, 2024

[1575]:
```
# Import necessary libraries for the simulation, data manipulation, and␣
↪visualization.

import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import numba
import json
from datetime import datetime


# Configuration for consistent visualizations across the notebook.
plt.rcParams['figure.figsize'] = (14, 7)
np.random.seed(42) # Set a seed for reproducibility of random number generation.
```

[1576]:
```
# Load and validate JSON configuration files
"""
    Loads a JSON file from the specified path and returns the parsed data.
    If the file is not found or the content is not valid JSON, returns None.

    Parameters:
    file_path (str): The path to the JSON configuration file.

    Returns:
    dict or None: Parsed JSON data if successful, None otherwise.
    """

def load_json(file_path):
    try:
        with open(file_path, 'r') as file:
            data = json.load(file)
        return data
    except (FileNotFoundError, json.JSONDecodeError):
        return None
```

[1577]:
```
# ==================================
# TradingPool Class
"""
This class models a basic liquidity pool in a decentralized exchange, similar to␣
↪those used in Uniswap.
It uses the constant product formula (x * y = k) to manage the relationship␣
↪between two assets (ETH and TKN),
ensuring that their product remains constant during swaps.

Attributes:
    eth_reserves (float): Current amount of ETH in the pool.
    tkn_reserves (float): Current amount of TKN in the pool.
    k (float): The constant product of the reserves (eth_reserves *␣
↪tkn_reserves).
    lp_fees_eth (float): Accumulated ETH fees for liquidity providers.
    lp_fees_tkn (float): Accumulated TKN fees for liquidity providers.
    tkn_hard_cap (float): Maximum allowable TKN reserves in the pool to prevent␣
↪overflow.
"""
class TradingPool:
    def __init__(self, initial_eth, initial_tkn, tkn_hard_cap=10000):
        """
        Initializes the trading pool with initial reserves of ETH and TKN tokens.
        The product of the initial reserves (k) is calculated to enforce the␣
↪constant product formula.

        Parameters:
        initial_eth (float): Initial amount of ETH in the pool.
        initial_tkn (float): Initial amount of TKN in the pool.
        tkn_hard_cap (float): Maximum allowed TKN in the pool, default is 10000.
        """
        self.eth_reserves = initial_eth
        self.tkn_reserves = initial_tkn
        self.k = self.eth_reserves * self.tkn_reserves  # Constant product to␣
↪ensure balanced liquidity
        self.lp_fees_eth = 0  # Initial liquidity provider fee balance in ETH
        self.lp_fees_tkn = 0  # Initial liquidity provider fee balance in TKN
        self.tkn_hard_cap = tkn_hard_cap  # Cap to prevent excessive TKN reserves

    # ==================================
    # SWAP
    """
    This method handles swaps between ETH and TKN, adhering to the constant␣
↪product formula (x * y = k).
    It allows users to swap one asset for another while maintaining the balance␣
↪of the pool.
```

```
    Parameters:
    amount_eth (float): Amount of ETH to swap, default is 0.
    amount_tkn (float): Amount of TKN to swap, default is 0.

    Returns:
    float: Amount of the opposite asset received in the swap, adjusted for␣
↪slippage.
    """

    def swap(self, amount_eth=0, amount_tkn=0):
        """
        Executes a swap operation based on the constant product formula, with␣
↪slippage and fee considerations.

        Parameters:
        amount_eth (float): The amount of ETH being swapped for TKN. Defaults to␣
↪0 if TKN is provided.
        amount_tkn (float): The amount of TKN being swapped for ETH. Defaults to␣
↪0 if ETH is provided.

        Returns:
        float: The amount of the opposite asset obtained from the swap, after␣
↪slippage and fees.
        """
        if amount_eth > 0:
            trade_ratio = amount_eth / self.eth_reserves
        elif amount_tkn > 0:
            trade_ratio = amount_tkn / self.tkn_reserves
        else:
            trade_ratio = 0

    # Increase slippage factor for larger trades
        slippage_factor = 0.003 + (trade_ratio * 0.01)  # 1% additional slippage␣
↪for large trades

        if amount_eth > 0:
            tkn_out = self._eth_to_tkn(amount_eth) * (1 - slippage_factor)
            tkn_out = min(tkn_out, self.tkn_reserves)
            if tkn_out <= 0:
                return 0
            self.eth_reserves += (amount_eth * (1 - slippage_factor))
            self.tkn_reserves -= tkn_out
            self.lp_fees_eth += (amount_eth * slippage_factor)
            return tkn_out
```

```
        elif amount_tkn > 0:
            eth_out = self._tkn_to_eth(amount_tkn) * (1 - slippage_factor)
            eth_out = min(eth_out, self.eth_reserves)
            if eth_out <= 0:
                return 0
            self.tkn_reserves += (amount_tkn * (1 - slippage_factor))
            self.eth_reserves -= eth_out
            self.lp_fees_tkn += (amount_tkn * slippage_factor)
            return eth_out

        return 0

    # ===================================
    # ETH to TKN
    """
    This private method calculates the amount of TKN tokens received for a given␣
↪ETH input,
    using the constant product formula. It simulates the real-world scenario␣
↪where the price of TKN
    increases as more ETH is added to the pool (reflecting a typical␣
↪demand-supply relationship).

    Parameters:
    amount_eth (float): The amount of ETH being swapped.

    Returns:
    float: The amount of TKN tokens received.
    """

    def _eth_to_tkn(self, amount_eth):
        """
        Computes the TKN output for a given amount of ETH based on the constant␣
↪product formula.
        This method assumes a 0.3% fee (commonly used in AMMs like Uniswap).

        Parameters:
        amount_eth (float): The amount of ETH being input to the pool.

        Returns:
        float: The calculated amount of TKN to be returned.
        """
        effective_eth = amount_eth * 0.997  # Apply a 0.3% fee, leaving 99.7% of␣
↪the ETH for the swap
        new_eth_reserve = self.eth_reserves + effective_eth  # Hypothetical new␣
↪ETH reserve after the swap
```

```
        new_tkn_reserve = self.k / new_eth_reserve  # Calculate the new TKN reserve using the constant product formula
        tkn_out = self.tkn_reserves - new_tkn_reserve  # Determine how much TKN will be taken out
        return tkn_out

    # ==================================
    # TKN to ETH
    """
    This private method calculates the amount of ETH received for a given TKN input,
    mirroring the ETH to TKN swap logic but in reverse. It uses the same constant product formula
    to maintain market equilibrium.

    Parameters:
    amount_tkn (float): The amount of TKN being swapped.

    Returns:
    float: The amount of ETH received.
    """

    def _tkn_to_eth(self, amount_tkn):
        """
        Computes the ETH output for a given amount of TKN based on the constant product formula.
        This method also assumes a 0.3% fee, aligning with typical AMM fee structures.

        Parameters:
        amount_tkn (float): The amount of TKN being input to the pool.

        Returns:
        float: The calculated amount of ETH to be returned.
        """
        effective_tkn = amount_tkn * 0.997  # Apply a 0.3% fee, leaving 99.7% of the TKN for the swap
        new_tkn_reserve = self.tkn_reserves + effective_tkn  # Hypothetical new TKN reserve after the swap
        new_eth_reserve = self.k / new_tkn_reserve  # Calculate the new ETH reserve using the constant product formula
        eth_out = self.eth_reserves - new_eth_reserve  # Determine how much ETH will be taken out
        return eth_out

    # ==================================
```

```
    # Get Price Eth per TKN
    """
    This method returns the current price of ETH in terms of TKN.
    It reflects the ratio of ETH to TKN in the reserves, which is the simplest␣
↪way to determine the price
    in an AMM like Uniswap.

    Returns:
    float: The price of ETH per TKN.
    """

    def get_price_eth_per_tkn(self):
        """
        Returns the current ETH price in terms of TKN by dividing the ETH␣
↪reserves by the TKN reserves.

        Returns:
        float: The calculated price of 1 ETH in TKN.
        """
        return self.eth_reserves / self.tkn_reserves

    def get_price_tkn_per_eth(self):
        """
        Returns the current TKN price in terms of ETH by dividing the TKN␣
↪reserves by the ETH reserves.

        Returns:
        float: The calculated price of 1 TKN in ETH.
        """
        return self.tkn_reserves / self.eth_reserves

    # ===================================
    # Aggregate LP Fees
    """
    This method aggregates and returns the total liquidity provider fees␣
↪accumulated in the pool.
    LPs typically earn a share of the trading fees in decentralized exchanges,␣
↪so this function
    is crucial for tracking those earnings.

    Returns:
    tuple: Total ETH and TKN fees accumulated.
    """

    def aggregate_lp_fees(self):
        """
```

```
        Returns the cumulative liquidity provider fees in both ETH and TKN,␣
↪allowing LPs to understand
        their earnings over time.

        Returns:
        tuple: A tuple of (ETH fees, TKN fees) accumulated in the pool.
        """
        return self.lp_fees_eth, self.lp_fees_tkn

    # ==================================
    # Calculate Price Impact
    """
    This method calculates the price impact of a given trade size in either ETH␣
↪or TKN.
    Price impact refers to the change in the asset's price due to the trade, a␣
↪crucial factor
    in decentralized finance (DeFi) trading strategies.

    Parameters:
    trade_size_eth (float): Amount of ETH to trade, default is 0.
    trade_size_tkn (float): Amount of TKN to trade, default is 0.

    Returns:
    float: The price impact of the trade.
    """

    def calculate_price_impact(pool, trade_size_eth=0, trade_size_tkn=0):
        """
        Computes the price impact of a trade based on the current reserves and␣
↪the trade size,
        using a more aggressive slippage model to reflect real-world conditions.

        Parameters:
        trade_size_eth (float): The size of the ETH trade.
        trade_size_tkn (float): The size of the TKN trade.

        Returns:
        float: The calculated price impact as a proportion of the current␣
↪reserves.
        """
        if trade_size_eth > 0:
            new_eth_reserves = pool.eth_reserves + trade_size_eth
            new_tkn_reserves = pool.k / new_eth_reserves
            price_impact = abs((pool.tkn_reserves - new_tkn_reserves) / pool.
↪tkn_reserves)
        elif trade_size_tkn > 0:
```

```
            new_tkn_reserves = pool.tkn_reserves + trade_size_tkn
            new_eth_reserves = pool.k / new_tkn_reserves
            price_impact = abs((pool.eth_reserves - new_eth_reserves) / pool.
↪eth_reserves)
        else:
            price_impact = 0

    # Implement more aggressive slippage by increasing the impact factor.
        return price_impact * 1.5  # Increase slippage to better reflect large␣
↪trade impacts
```

[1578]:
```
class Participant:

    def __init__(self, eth_balance, tkn_balance, initial_eth_price,␣
↪initial_tkn_price):
        """
        Initializes a Participant with initial balances and the prices at the␣
↪start of the experiment.

        Args:
        eth_balance (float): Initial ETH balance.
        tkn_balance (float): Initial TKN balance.
        initial_eth_price (float): The initial price of ETH in USD.
        initial_tkn_price (float): The initial price of TKN in USD.
        """
        self.eth_balance = eth_balance
        self.tkn_balance = tkn_balance

        # Initialize initial balances
        self.initial_eth_balance = eth_balance
        self.initial_tkn_balance = tkn_balance

        # Initialize prices
        self.initial_eth_price = initial_eth_price
        self.initial_tkn_price = initial_tkn_price

        # Calculate the initial USD balance
        self.initial_usd_balance = (self.initial_eth_balance * self.
↪initial_eth_price) + \
                                   (self.initial_tkn_balance * self.
↪initial_tkn_price)

        # Initialize trade volumes
        self.total_eth_swapped = 0  # Total ETH swapped during the simulation
        self.total_tkn_swapped = 0  # Total TKN swapped during the simulation

        # Initialize trade history
```

```
        self.trade_history = []  # List to keep a record of all trades made by␣
↪the participant


    def buy(self, pool, amount_eth):
        """
        Executes a trade where the participant uses ETH to buy TKN from the pool.
        The trade is limited by the participant's current ETH balance.

        Parameters:
        pool (TradingPool): The pool that the participant will trade with.
        amount_eth (float): The amount of ETH the participant wants to trade for␣
↪TKN.
        """
        amount_eth = min(amount_eth, self.eth_balance)  # Ensure participant␣
↪doesn't spend more ETH than they have
        tkn_received = pool.swap(amount_eth=amount_eth)  # Perform the swap on␣
↪the pool and receive TKN
        if tkn_received > 0:
            self.eth_balance -= amount_eth  # Deduct the spent ETH from the␣
↪balance
            self.tkn_balance += tkn_received  # Add the received TKN to the␣
↪balance
            self._log_trade("buy", amount_eth, tkn_received)  # Log the trade␣
↪for record-keeping
            self.total_eth_swapped += amount_eth  # Track total ETH used for␣
↪swaps

    def sell(self, pool, amount_tkn):
        """
        Executes a trade where the participant sells TKN to receive ETH from the␣
↪pool.
        The trade is limited by the participant's current TKN balance.

        Parameters:
        pool (TradingPool): The pool that the participant will trade with.
        amount_tkn (float): The amount of TKN the participant wants to trade for␣
↪ETH.
        """
        amount_tkn = min(amount_tkn, self.tkn_balance)  # Ensure participant␣
↪doesn't sell more TKN than they have
        eth_received = pool.swap(amount_tkn=amount_tkn)  # Perform the swap on␣
↪the pool and receive ETH
        if eth_received > 0:
            self.tkn_balance -= amount_tkn  # Deduct the spent TKN from the␣
↪balance
```

```
            self.eth_balance += eth_received  # Add the received ETH to the␣
↪balance
            self._log_trade("sell", eth_received, amount_tkn)  # Log the trade␣
↪for record-keeping
            self.total_tkn_swapped += amount_tkn  # Track total TKN used for␣
↪swaps

    def calculate_roi(self):
        """
        Computes the participant's return on investment (ROI) in percentage␣
↪terms.
        This is done by comparing the current USD balance of the participant␣
↪(ETH and TKN holdings converted to USD)
        with their initial USD balance.

        Returns:
        float: The ROI as a percentage. Positive values indicate profit,␣
↪negative values indicate loss.
        """
        # Calculate the initial value of ETH and TKN in USD
        initial_eth_value_usd = self.initial_eth_balance * self.initial_eth_price
        initial_tkn_value_usd = self.initial_tkn_balance * self.initial_tkn_price

        # Total initial USD balance
        initial_usd_balance = initial_eth_value_usd + initial_tkn_value_usd

        # Calculate the current value of ETH and TKN in USD
        current_eth_value_usd = self.eth_balance * self.get_current_eth_price()
        current_tkn_value_usd = self.tkn_balance * self.get_current_tkn_price()

        # Total current USD balance
        current_usd_balance = current_eth_value_usd + current_tkn_value_usd

        # Calculate ROI as the percentage change from the initial USD balance
        roi = ((current_usd_balance - initial_usd_balance) /␣
↪initial_usd_balance) * 100

        return roi

    def _log_trade(self, trade_type, eth_amount, tkn_amount):
        """
        Records the details of each trade made by the participant, including the␣
↪time of trade,
        the type of trade, the amount of ETH and TKN involved, and the␣
↪participant's balance after the trade.
```

```
        Parameters:
        trade_type (str): The type of trade ('buy' or 'sell').
        eth_amount (float): The amount of ETH involved in the trade.
        tkn_amount (float): The amount of TKN involved in the trade.
        """
        self.trade_history.append({
            'time': datetime.now(),  # Timestamp of the trade
            'type': trade_type,  # Type of trade ('buy' or 'sell')
            'eth_amount': eth_amount,  # Amount of ETH involved in the trade
            'tkn_amount': tkn_amount,  # Amount of TKN involved in the trade
            'eth_balance': self.eth_balance,  # Participant's ETH balance after␣
↪the trade
            'tkn_balance': self.tkn_balance  # Participant's TKN balance after␣
↪the trade
        })

    def get_current_eth_price(self):
        """
        Provides the current price of ETH in USD. In this simulation, the price␣
↪is fixed at $2500,
        but in a real-world scenario, this value would be dynamically fetched␣
↪from a market API.

        Returns:
        float: The fixed price of ETH in USD for the simulation.
        """
        return 2500  # Placeholder value, should be dynamically updated in a␣
↪real application

    def get_current_tkn_price(self):
        """
        Provides the current price of TKN in USD. In this simulation, the price␣
↪is fixed at $2,
        but in a real-world scenario, this value would be dynamically fetched␣
↪from a market API.

        Returns:
        float: The fixed price of TKN in USD for the simulation.
        """
        return 2  # Placeholder value, should be dynamically updated in a real␣
↪application
```

[1579]:
```
class Strategy:
    """
    The `Strategy` class serves as a base class for implementing various trading␣
↪strategies.
```

```
    Each strategy is associated with a participant, who holds balances of ETH␣
↪and TKN, and can execute trades based on specific triggers.
    This class is designed to be extended by subclasses that define specific␣
↪trading logic.
    """

    def __init__(self, name, initial_eth, initial_tkn, initial_eth_price,␣
↪initial_tkn_price):
        """
        Constructor for the Strategy class.

        This initializes the strategy with a unique name and sets up a␣
↪participant
        with initial balances of ETH and TKN.

        Parameters:
        name (str): The name of the strategy.
        initial_eth (float): The initial ETH balance allocated to the␣
↪participant.
        initial_tkn (float): The initial TKN balance allocated to the␣
↪participant.
        """
        self.name = name  # Name of the strategy for identification purposes
        self.participant = Participant(initial_eth, initial_tkn,␣
↪initial_eth_price, initial_tkn_price)

    def execute(self, pool, current_price, market_conditions):
        """
        Main execution method for the strategy.

        This method checks if the strategy's trigger conditions are met using
        the `calculate_trigger` method. If the trigger is activated, it proceeds
        to execute a trade via the `perform_trade` method.

        Parameters:
        pool (TradingPool): The trading pool with which to interact for␣
↪executing trades.
        current_price (float): The current price of ETH in the market.
        market_conditions (dict): A dictionary containing relevant market␣
↪conditions.
        """
        if self.calculate_trigger(current_price, market_conditions):
            self.perform_trade(pool, current_price, market_conditions)

    def perform_trade(self, pool, current_price, market_conditions):
        """
```

```
        Abstract method for performing trades.

        This method must be overridden by any subclass of `Strategy`. It defines␣
↪the
        specific actions to be taken when the strategy decides to execute a␣
↪trade.

        Parameters:
        pool (TradingPool): The trading pool to interact with for executing␣
↪trades.
        current_price (float): The current price of ETH in the market.
        market_conditions (dict): A dictionary containing relevant market␣
↪conditions.

        Raises:
        NotImplementedError: If the subclass does not implement this method.
        """
        raise NotImplementedError("Each strategy must implement the␣
↪perform_trade method.")

    def calculate_trigger(self, current_price, market_conditions):
        """
        Abstract method for determining if a trade should be executed.

        This method must be overridden by any subclass of `Strategy`. It defines
        the logic to decide when the strategy should perform a trade based on
        current market conditions.

        Parameters:
        current_price (float): The current price of ETH in the market.
        market_conditions (dict): A dictionary containing relevant market␣
↪conditions.

        Returns:
        bool: True if the trigger condition is met, otherwise False.

        Raises:
        NotImplementedError: If the subclass does not implement this method.
        """
        raise NotImplementedError("Each strategy must implement the␣
↪calculate_trigger method.")

    def evaluate_performance(self):
        """
        Evaluates the strategy's success by calculating the participant's Return␣
↪on Investment (ROI).
```

```
        This method provides a way to assess how well the strategy has performed
        by returning the ROI, which is computed based on the participant's␣
↪initial
        and current balances converted to USD.

        Returns:
        float: The ROI percentage for the strategy.
        """
        return self.participant.calculate_roi()
```

[1580]:
```
# Strategy Implementations
"""
This section defines various trading strategies that participants can use in a␣
↪game-theory-based trading environment.
Each strategy extends the `Strategy` base class and implements specific logic to␣
↪decide when to buy or sell TKN and ETH
based on market conditions. The goal for each participant is to maximize their␣
↪USD holdings.
"""
class BullStrategy(Strategy):
    """
    The BullStrategy is an active trading strategy that buys TKN during␣
↪perceived bullish conditions.
    The strategy buys TKN when the current price is below the EMA but also looks␣
↪for opportunities to sell during upward trends.
    """

    MIN_ETH_THRESHOLD = 0.001  # Minimum ETH balance threshold to continue␣
↪trading

    def __init__(self, name, initial_eth, initial_tkn, initial_eth_price,␣
↪initial_tkn_price):
        super().__init__(name, initial_eth, initial_tkn, initial_eth_price,␣
↪initial_tkn_price)

    def perform_trade(self, pool, current_price, market_conditions):
        if self.calculate_trigger(current_price, market_conditions):
            if current_price < market_conditions['ema'] and self.participant.
↪eth_balance > self.MIN_ETH_THRESHOLD:
                amount_to_trade = round(self.participant.eth_balance * 0.5, 8)
                print(f"{self.name}: Actively buying TKN with {amount_to_trade}␣
↪ETH at price {current_price}")
                self.participant.buy(pool, amount_to_trade)
            elif current_price > market_conditions['ema'] * 1.02 and self.
↪participant.tkn_balance > 0:
```

```
                tkn_to_sell = round(self.participant.tkn_balance * 0.5, 8)
                print(f"{self.name}: Selling {tkn_to_sell} TKN at price␣
↪{current_price} to realize profits")
                self.participant.sell(pool, tkn_to_sell)
            self.update_balance()

    def calculate_trigger(self, current_price, market_conditions):
        return current_price < market_conditions['ema'] or current_price >␣
↪market_conditions['ema'] * 1.02

    def update_balance(self):
        if self.participant.eth_balance < self.MIN_ETH_THRESHOLD:
            self.participant.eth_balance = 0
        print(f"{self.name}: Updated ETH balance is {self.participant.
↪eth_balance}")

class BearStrategy(Strategy):
    """
    The BearStrategy is a defensive trading strategy that primarily sells TKN␣
↪during perceived bearish conditions.
    The strategy sells TKN when the current price is above the SMA and avoids␣
↪aggressive buying.
    It only buys during extreme market dips when the price is significantly␣
↪below the SMA.
    """

    MIN_TKN_THRESHOLD = 0.001  # Minimum TKN balance threshold to continue␣
↪trading
    DIP_THRESHOLD = 0.95  # Buy only if the price drops below 95% of the SMA,␣
↪indicating a market dip

    def __init__(self, name, initial_eth, initial_tkn, initial_eth_price,␣
↪initial_tkn_price):
        super().__init__(name, initial_eth, initial_tkn, initial_eth_price,␣
↪initial_tkn_price)

    def perform_trade(self, pool, current_price, market_conditions):
        if self.calculate_trigger(current_price, market_conditions):
            if current_price > market_conditions['sma'] and self.participant.
↪tkn_balance > self.MIN_TKN_THRESHOLD:
                amount_to_trade = round(self.participant.tkn_balance * 0.5, 8)
                print(f"{self.name}: Actively selling {amount_to_trade} TKN at␣
↪price {current_price}")
                self.participant.sell(pool, amount_to_trade)
            elif current_price < market_conditions['sma'] * self.DIP_THRESHOLD␣
↪and self.participant.eth_balance > 0:
```

```
                amount_to_trade = round(self.participant.eth_balance * 0.1, 8) ␣
↪# Buy very conservatively
                print(f"{self.name}: Cautiously buying TKN with␣
↪{amount_to_trade} ETH at price {current_price} during a major dip")
                self.participant.buy(pool, amount_to_trade)
            self.update_balance()

    def calculate_trigger(self, current_price, market_conditions):
        # Trigger on significant price deviations from the SMA
        return current_price > market_conditions['sma'] or current_price <␣
↪market_conditions['sma'] * self.DIP_THRESHOLD

    def update_balance(self):
        if self.participant.tkn_balance < self.MIN_TKN_THRESHOLD:
            self.participant.tkn_balance = 0
        print(f"{self.name}: Updated TKN balance is {self.participant.
↪tkn_balance}")


class DcaStrategy(Strategy):
    """
    The Dollar-Cost Averaging (DCA) Strategy involves consistently investing a␣
↪fixed amount of ETH daily,
    regardless of the current market price, to reduce the impact of volatility␣
↪over time.
    The strategy should end with 0 ETH by fully investing each day.
    """

    def __init__(self, name, initial_eth, initial_tkn, initial_eth_price,␣
↪initial_tkn_price, daily_investment):
        super().__init__(name, initial_eth, initial_tkn, initial_eth_price,␣
↪initial_tkn_price)
        self.daily_investment = daily_investment

    def perform_trade(self, pool, current_price, market_conditions):
        while self.participant.eth_balance > 0:
            if self.participant.eth_balance >= self.daily_investment:
                self.participant.buy(pool, self.daily_investment)
            else:
                self.participant.buy(pool, self.participant.eth_balance)  # Use␣
↪all remaining ETH if it's less than the daily investment

    def calculate_trigger(self, current_price, market_conditions):
        return True
```

```
class HarvesterStrategy(Strategy):
    """
    The HarvesterStrategy focuses on periodically taking profits by selling a␣
↪portion of the participant's TKN holdings.
    This strategy sells 20% of the TKN balance whenever the participant holds␣
↪any TKN, effectively harvesting profits over time.
    """

    MIN_TKN_THRESHOLD = 0.001  # Minimum TKN balance threshold to continue␣
↪trading

    def __init__(self, name, initial_eth, initial_tkn, initial_eth_price,␣
↪initial_tkn_price):
        super().__init__(name, initial_eth, initial_tkn, initial_eth_price,␣
↪initial_tkn_price)

    def perform_trade(self, pool, current_price, market_conditions):
        tkn_to_sell = round(self.participant.tkn_balance * 0.2, 8)
        if tkn_to_sell > self.MIN_TKN_THRESHOLD:
            print(f"{self.name}: Selling {tkn_to_sell} TKN at price␣
↪{current_price}")
            self.participant.sell(pool, tkn_to_sell)
            self.update_balance()

    def calculate_trigger(self, current_price, market_conditions):
        return self.participant.tkn_balance > self.MIN_TKN_THRESHOLD

    def update_balance(self):
        if self.participant.tkn_balance < self.MIN_TKN_THRESHOLD:
            self.participant.tkn_balance = 0
        print(f"{self.name}: Updated TKN balance is {self.participant.
↪tkn_balance}")

class DoveStrategy(Strategy):
    """
    The DoveStrategy is a conservative strategy that buys TKN when both the␣
↪simple moving average (SMA) and
    the exponential moving average (EMA) indicate the price is low. This dual␣
↪confirmation aims to reduce the risk of buying too early.
    The strategy is designed to be cautious, executing small trades only under␣
↪favorable conditions.
    """

    def __init__(self, name, initial_eth, initial_tkn, initial_eth_price,␣
↪initial_tkn_price):
```

```
        super().__init__(name, initial_eth, initial_tkn, initial_eth_price,␣
↪initial_tkn_price)

    def perform_trade(self, pool, current_price, market_conditions):
        max_trade_eth = self.participant.eth_balance * 0.02  # Using 2% of␣
↪available ETH per trade
        if max_trade_eth > 0 and self.calculate_trigger(current_price,␣
↪market_conditions):
            print(f"{self.name}: Cautiously buying TKN with {max_trade_eth} ETH␣
↪at price {current_price}")
            self.participant.buy(pool, max_trade_eth)

    def calculate_trigger(self, current_price, market_conditions):
        conservative_factor = 0.97  # 3% below both the SMA and EMA
        return (current_price < market_conditions['sma'] * conservative_factor␣
↪and
                current_price < market_conditions['ema'] * conservative_factor)

class HawkStrategy(Strategy):
    """
    The HawkStrategy is an aggressive, profit-targeting strategy. The␣
↪participant buys TKN when the strategy is initiated and
    sells when the price reaches a specified profit target. The entry price is␣
↪recorded to calculate when the profit target is met.
    """

    def __init__(self, name, initial_eth, initial_tkn, initial_eth_price,␣
↪initial_tkn_price, target_profit):
        super().__init__(name, initial_eth, initial_tkn, initial_eth_price,␣
↪initial_tkn_price)
        self.target_profit = target_profit
        self.entry_price = None

    def perform_trade(self, pool, current_price, market_conditions):
        if self.entry_price is None:
            self.entry_price = current_price
            self.participant.buy(pool, self.participant.eth_balance)
        elif current_price >= self.entry_price * (1 + self.target_profit):
            print(f"{self.name}: Selling all TKN at price {current_price} after␣
↪reaching profit target")
            self.participant.sell(pool, self.participant.tkn_balance)

    def calculate_trigger(self, current_price, market_conditions):
        return self.entry_price is None or current_price >= self.entry_price *␣
↪(1 + self.target_profit)
```

```python
class Sniper1Strategy(Strategy):
    def __init__(self, name, initial_eth, initial_tkn, initial_eth_price,␣
↪initial_tkn_price, target_profit):
        super().__init__(name, initial_eth, initial_tkn, initial_eth_price,␣
↪initial_tkn_price)
        self.target_profit = target_profit
        self.entry_price = None

    def perform_trade(self, pool, current_price, market_conditions):
        if self.entry_price is None:
            self.entry_price = current_price
            # Limit the purchase to 10-20% of the pool's TKN reserves
            tkn_to_purchase = min(self.participant.eth_balance * current_price,␣
↪pool.tkn_reserves * 0.2)
            self.participant.buy(pool, tkn_to_purchase)
            print(f"{self.name}: Purchased {tkn_to_purchase} TKN at price␣
↪{current_price}")
        elif current_price >= self.entry_price * (1 + self.target_profit):
            print(f"{self.name}: Selling all TKN at price {current_price} for␣
↪profit")
            self.participant.sell(pool, self.participant.tkn_balance)
        elif current_price < self.entry_price * 0.95:  # Stop-loss at a 5% drop
            print(f"{self.name}: Selling all TKN at price {current_price} due to␣
↪stop-loss")
            self.participant.sell(pool, self.participant.tkn_balance)

    def calculate_trigger(self, current_price, market_conditions):
        # Always trigger a trade if no entry price, otherwise check profit␣
↪target or stop-loss
        return self.entry_price is None or current_price >= self.entry_price *␣
↪(1 + self.target_profit) or current_price < self.entry_price * 0.95


class Sniper2Strategy(Strategy):
    def __init__(self, name, initial_eth, initial_tkn, initial_eth_price,␣
↪initial_tkn_price, target_profit, delay=1):
        super().__init__(name, initial_eth, initial_tkn, initial_eth_price,␣
↪initial_tkn_price)
        self.target_profit = target_profit
        self.entry_price = None
        self.delay = delay

    def perform_trade(self, pool, current_price, market_conditions):
        current_day = market_conditions['current_day']
        if self.entry_price is None and current_day > self.delay:
            self.entry_price = current_price
```

```
            # Limit the purchase to 10-20% of the pool's TKN reserves
            tkn_to_purchase = min(self.participant.eth_balance * current_price,␣
↪pool.tkn_reserves * 0.2)
            self.participant.buy(pool, tkn_to_purchase)
            print(f"{self.name}: Purchased {tkn_to_purchase} TKN at price␣
↪{current_price}")
        elif current_price >= self.entry_price * (1 + self.target_profit):
            print(f"{self.name}: Selling all TKN at price {current_price} for␣
↪profit")
            self.participant.sell(pool, self.participant.tkn_balance)
        elif current_price < self.entry_price * 0.95:  # Stop-loss at a 5% drop
            print(f"{self.name}: Selling all TKN at price {current_price} due to␣
↪stop-loss")
            self.participant.sell(pool, self.participant.tkn_balance)

    def calculate_trigger(self, current_price, market_conditions):
        current_day = market_conditions['current_day']
        # Buy if after delay period and no entry price set, otherwise check␣
↪profit target or stop-loss
        if self.entry_price is None:
            return current_day > self.delay
        else:
            return current_price >= self.entry_price * (1 + self.target_profit)␣
↪or current_price < self.entry_price * 0.95
```

[1581]:
```
# Simulation Execution - Function to simulate ETH price using Geometric Brownian␣
↪Motion (GBM).
"""
This section defines functions to simulate the price path of ETH over time using␣
↪Geometric Brownian Motion (GBM).
GBM is a mathematical model commonly used in finance to model stock prices and␣
↪other financial instruments.
It assumes that the logarithm of the price follows a Brownian motion with drift.
"""

# ========
# Simulate Eth price using GBM
"""
Simulates ETH prices using a Geometric Brownian Motion (GBM) model.

Parameters:
initial_price (float): Starting price of ETH.
days (int): Number of days to simulate.
mu (float): Expected return (drift) of the ETH price.
sigma (float): Volatility (standard deviation) of the ETH price.
```

```
mean_reversion_level (float, optional): Level to which price may revert, default␣
↪is None.

Returns:
list: Simulated ETH prices over the given number of days.
"""

def simulate_eth_price_gbm(initial_price, days, mu, sigma,␣
↪mean_reversion_level=None):
    """
    Simulates the price of ETH over a specified number of days using the GBM␣
↪model with optional mean reversion.

    Parameters:
    initial_price (float): The initial price of ETH at the start of the␣
↪simulation.
    days (int): The number of days to simulate.
    mu (float): The drift coefficient, representing the expected daily return of␣
↪ETH.
    sigma (float): The volatility coefficient, representing the standard␣
↪deviation of returns.
    mean_reversion_level (float, optional): The price level to which the price␣
↪reverts over time, if provided.

    Returns:
    list: A list of simulated ETH prices over the specified number of days.
    """
    dt = 1 / 365  # Time increment (1 day)
    prices = [initial_price]  # List to store simulated prices, starting with␣
↪the initial price

    for _ in range(days):
        # Generate a random shock based on the normal distribution, representing␣
↪daily returns
        shock = np.random.normal(loc=mu * dt, scale=sigma * np.sqrt(dt))
        # Calculate the new price using the GBM formula: S(t+1) = S(t) *␣
↪exp(shock)
        price = prices[-1] * np.exp(shock)

        # If mean reversion is specified, adjust the price towards the mean␣
↪reversion level
        if mean_reversion_level is not None:
            reversion_adjustment = (mean_reversion_level - price) * 0.1  # 10%␣
↪adjustment towards mean level
            price += reversion_adjustment
```

```
        prices.append(price)  # Append the new price to the list

    return prices  # Return the list of simulated prices

# ========
# Function to simulate market conditions based on GBM.
"""
Simulates market conditions, particularly the price path of ETH.

Parameters:
market_conditions (str or dict): JSON file path or dict containing simulation␣
 ↪parameters.
days (int): Number of days to simulate.

Returns:
list: Simulated ETH prices over the given number of days.
"""

def simulate_market_conditions(market_conditions, days):
    """
    Simulates the price of ETH over a specified number of days using a Geometric␣
 ↪Brownian Motion model.
    This function allows for the configuration of market conditions either via a␣
 ↪JSON file or directly as a dictionary.

    Parameters:
    market_conditions (str or dict): If a string is provided, it is assumed to␣
 ↪be a file path to a JSON file
                                     containing the simulation parameters ('mu'␣
 ↪and 'volatility').
                                     If a dictionary is provided, it directly␣
 ↪contains these parameters.
    days (int): The number of days to simulate.

    Returns:
    list: A list of simulated ETH prices over the specified number of days.
    """
    # Load parameters from a JSON file if a file path is provided
    if isinstance(market_conditions, str):  # Check if market_conditions is a␣
 ↪file path
        with open(market_conditions, 'r') as file:
            config = json.load(file)  # Load the JSON file into a dictionary
        print("Loaded JSON config:", config)  # Debugging statement to confirm␣
 ↪the configuration was loaded
        mu = config['market_conditions']['mu']  # Expected return (drift) from␣
 ↪the JSON config
```

```
        sigma = config['market_conditions']['volatility']  # Volatility from the␣
↪JSON config
    else:
        # Default parameters if no file is provided or if market_conditions is␣
↪directly passed as a dictionary
        mu = market_conditions.get('mu', 0.1)  # Drift factor, default to 10%␣
↪annual return
        sigma = market_conditions.get('volatility', 0.3)  # Volatility, default␣
↪to 30% annual volatility

    dt = 1 / 365  # Daily time step for the simulation
    eth_prices = [2500]  # Starting ETH price, for example purposes

    for _ in range(1, days):
        # Calculate the daily price change using the GBM formula:
        # Price change = S(t) * (μ * dt + σ * ϵ * sqrt(dt))
        # Where ϵ is a random variable from a standard normal distribution
        price_change = eth_prices[-1] * (mu * dt + sigma * np.random.normal() *␣
↪np.sqrt(dt))
        eth_prices.append(eth_prices[-1] + price_change)  # Update the ETH price␣
↪list with the new price

    return eth_prices  # Return the list of simulated ETH prices
```

[1582]:
```
# ========
# Function to execute the simulation and return relevant data.
"""
Runs the trading simulation over the specified number of days using the given␣
↪strategies.

Parameters:
strategy_instances (list): List of strategy objects to execute.
pool (TradingPool): The trading pool to interact with.
market_conditions (dict): The market conditions for the simulation.
days (int): Number of days to simulate.

Returns:
tuple: Various statistics from the simulation including price history, trade␣
↪counts, and more.
"""

def run_simulation(strategy_instances, pool, market_conditions, days):
    """
    Executes a simulation of ETH trading strategies over a specified number of␣
↪days.
```

```
    The simulation tracks ETH prices, strategy executions, and the state of the␣
↪trading pool.
    It calculates important metrics such as price impacts, impermanent loss, and␣
↪liquidity provider fees.

    Parameters:
    strategy_instances (list): A list of instantiated strategy objects.
    pool (TradingPool): The trading pool that strategies will interact with.
    market_conditions (dict): Market conditions, possibly containing parameters␣
↪like drift (mu) and volatility (sigma).
    days (int): The number of days to run the simulation.

    Returns:
    tuple: Contains the following:
        - eth_price_history (list): The daily ETH prices over the simulation␣
↪period.
        - tkn_price_history (list): The daily TKN prices over the simulation␣
↪period.
        - trade_counts (dict): The count of trades executed by each strategy.
        - eth_reserves_over_time (list): The ETH reserves in the pool over time.
        - tkn_reserves_over_time (list): The TKN reserves in the pool over time.
        - lp_fees_over_time (list): The accumulated liquidity provider fees over␣
↪time.
        - impermanent_losses (list): The calculated impermanent losses over time.
        - price_impacts (list): The price impacts caused by trades over time.
        - detailed_logs (list): Detailed logs of trades including day, strategy,␣
↪action, and slippage.
    """

    # Simulate ETH prices over the given number of days using the specified␣
↪market conditions.
    eth_prices = simulate_market_conditions(market_conditions, days)

    # Initialize data structures to store simulation results
    eth_price_history = []
    tkn_price_history = []
    trade_counts = {strategy.name: 0 for strategy in strategy_instances}
    eth_reserves_over_time = []
    tkn_reserves_over_time = []
    lp_fees_over_time = []
    impermanent_losses = []
    price_impacts = []
    detailed_logs = []  # Initialize detailed logs for recording trade details

    # Calculate initial pool values for impermanent loss
    initial_eth = pool.eth_reserves
```

```
    initial_tkn = pool.tkn_reserves

    # Loop over each day to execute the simulation
    for day in range(days):
        current_eth_price = eth_prices[day]
        current_tkn_price = 1 / current_eth_price  # Assuming inverse␣
↪relationship for TKN price

        # Store the prices in their respective lists
        eth_price_history.append(current_eth_price)
        tkn_price_history.append(current_tkn_price)

        market_state = {
            'ema': np.mean(eth_prices[max(0, day - 10):day+1]),
            'sma': np.mean(eth_prices[max(0, day - 30):day+1]),
            'last_price': eth_prices[day-1] if day > 0 else current_eth_price,
            'current_day': day
        }

        # Force Sniper1 and Sniper2 strategies to execute first on Day 0
        if day == 0:
            sniper_strategies = [strategy for strategy in strategy_instances if␣
↪isinstance(strategy, (Sniper1Strategy, Sniper2Strategy))]
            for strategy in sniper_strategies:
                print(f"Day {day}: {strategy.name} is executing forced trade")
                price_impact = pool.
↪calculate_price_impact(trade_size_eth=strategy.participant.eth_balance)
                strategy.execute(pool, current_eth_price, market_state)
                trade_counts[strategy.name] += 1
                price_impacts.append(price_impact)
                print(f"{strategy.name} ETH Balance: {strategy.participant.
↪eth_balance}, TKN Balance: {strategy.participant.tkn_balance}")
                detailed_logs.append({
                    "Day": day,
                    "Strategy": strategy.name,
                    "Action": "Forced Execution",
                    "ETH_Swapped": strategy.participant.eth_balance,
                    "TKN_Received": pool.get_price_tkn_per_eth() * strategy.
↪participant.eth_balance,
                    "USD_Value": strategy.participant.eth_balance *␣
↪current_eth_price,
                    "Slippage": price_impact,
                })

        # Execute remaining strategies
        for strategy in strategy_instances:
```

```
            if strategy in sniper_strategies and day == 0:
                continue

            print(f"Day {day}: {strategy.name} is checking triggers")
            last_trade_day = strategy.participant.trade_history[-1]['time'].day␣
↪if strategy.participant.trade_history else -1
            if last_trade_day == day:
                print(f"Skipping {strategy.name} - trade already made today")
                continue

            # Calculate price impact before executing the trade
            price_impact = pool.calculate_price_impact(trade_size_eth=strategy.
↪participant.eth_balance)
            strategy.execute(pool, current_eth_price, market_state)  # Changed␣
↪from current_price to current_eth_price
            trade_counts[strategy.name] += 1
            price_impacts.append(price_impact)

            # Log detailed information about the trade
            if strategy.participant.trade_history:
                last_trade = strategy.participant.trade_history[-1]
                detailed_logs.append({
                    "Day": day,
                    "Strategy": strategy.name,
                    "Action": last_trade['type'],
                    "ETH_Swapped": last_trade['eth_amount'] if␣
↪last_trade['type'] == 'buy' else 0,
                    "TKN_Swapped": last_trade['tkn_amount'] if␣
↪last_trade['type'] == 'sell' else 0,
                    "USD_Value": last_trade['eth_amount'] * current_eth_price if␣
↪last_trade['type'] == 'buy' else last_trade['tkn_amount'] * (1/
↪current_eth_price),
                    "Slippage": price_impact,
                })

            print(f"{strategy.name} Post-Execution ETH Balance: {strategy.
↪participant.eth_balance}, TKN Balance: {strategy.participant.tkn_balance}")

        # Track daily simulation data
        eth_reserves_over_time.append(pool.eth_reserves)
        tkn_reserves_over_time.append(pool.tkn_reserves)
        lp_fees_over_time.append(pool.aggregate_lp_fees())

        # Calculate impermanent loss for the day
        impermanent_loss = calculate_impermanent_loss(
            initial_eth=initial_eth,
```

```
            initial_tkn=initial_tkn,
            current_eth=pool.eth_reserves,
            current_tkn=pool.tkn_reserves,
            eth_price=current_eth_price,
            tkn_price=current_tkn_price
        )
        impermanent_losses.append(impermanent_loss)

    # Return all gathered simulation data
    return eth_price_history, tkn_price_history, trade_counts,␣
↪eth_reserves_over_time, tkn_reserves_over_time, lp_fees_over_time,␣
↪impermanent_losses, price_impacts, detailed_logs
```

[1583]:
```
# Performance Evaluation and Table Creation
"""
This section includes functions to evaluate the performance of each trading␣
↪strategy and to summarize the trades made during the simulation.
"""

# ========
# Function to evaluate the performance of each strategy.
"""
Evaluates the performance of each strategy after the simulation.

Parameters:
strategy_instances (list): List of strategy objects.

Returns:
list: List of tuples containing strategy names and their corresponding ROI.
"""

def evaluate_simulation(strategy_instances):
    """
    Evaluates the Return on Investment (ROI) for each strategy based on the␣
↪final state of their associated participants.

    Parameters:
    strategy_instances (list): A list of strategy objects that were used in the␣
↪simulation.

    Returns:
    list: A list of tuples, where each tuple contains the strategy name and its␣
↪corresponding ROI.
    """
    results = []
    for strategy in strategy_instances:
        roi = strategy.evaluate_performance()  # Calculate the ROI
```

```
        results.append((strategy.name, roi))  # Store the strategy name and ROI␣
↪as a tuple in the results list
    return results


# ========
# Function to summarize trades made by each strategy.
"""
Summarizes the trades made by each strategy during the simulation.

Parameters:
strategy_instances (list): List of strategy objects.

Returns:
dict: Summary of ETH and TKN swapped by each strategy.
"""

def summarize_trades(strategy_instances):
    """
    Creates a summary of the trades made by each strategy during the simulation.

    The summary includes the total amount of ETH and TKN swapped by each␣
↪strategy.

    Parameters:
    strategy_instances (list): A list of strategy objects that were used in the␣
↪simulation.

    Returns:
    dict: A dictionary where each key is a strategy name, and the value is␣
↪another dictionary
          with the total amount of ETH and TKN swapped by that strategy.
    """
    summary = {}
    for strategy in strategy_instances:
        eth_swapped = sum(trade['eth_amount'] for trade in strategy.participant.
↪trade_history)
        tkn_swapped = sum(trade['tkn_amount'] for trade in strategy.participant.
↪trade_history)
        summary[strategy.name] = {'ETH Swapped': eth_swapped, 'TKN Swapped':␣
↪tkn_swapped}
    return summary
```

[1584]:
```
# Create final balances table for all strategies.
"""
```

```
Creates a DataFrame summarizing the final balances of ETH, TKN, and ROI for each␣
↪strategy.

Parameters:
strategy_instances (list): List of strategy objects.

Returns:
pd.DataFrame: DataFrame containing final balances and ROI for each strategy.
"""

def create_final_balances_table(strategy_instances, pool):
    """
    Generates a summary table containing the final ETH and TKN balances,
    as well as the calculated USD balance for each strategy after the simulation.

    Parameters:
    strategy_instances (list): A list of strategy objects used in the simulation.
    pool (TradingPool): The trading pool used in the simulation, needed to␣
↪calculate TKN price.

    Returns:
    pd.DataFrame: A DataFrame summarizing the final ETH and TKN balances,
                  final USD balance, and placeholder for Sharpe Ratio for each␣
↪strategy.
    """
    data = []
    for strategy in strategy_instances:
        # Get the current ETH price and calculate the TKN price
        current_eth_price = strategy.participant.get_current_eth_price()
        current_tkn_price = (pool.eth_reserves * current_eth_price) / pool.
↪tkn_reserves

        # Calculate the final USD balance
        final_usd_balance = (strategy.participant.eth_balance *␣
↪current_eth_price) + (strategy.participant.tkn_balance * current_tkn_price)

        # Ensure every strategy is included, regardless of balance
        data.append({
            'Strategy': strategy.name,
            'Final_ETH_Balance': strategy.participant.eth_balance,
            'Final_TKN_Balance': strategy.participant.tkn_balance,
            'Final_USD_Balance': final_usd_balance,
            'Sharpe_Ratio': np.nan  # Placeholder; Sharpe Ratio can be added␣
↪later if needed
        })
```

```
    # Convert the list of dictionaries to a DataFrame
    df = pd.DataFrame(data)

    return df




# Create a table of total volume swapped per strategy.
"""
Creates a DataFrame summarizing the total volume of ETH and TKN swapped by each␣
 ↪strategy.

Parameters:
trade_summary (dict): Dictionary summarizing trade volumes for each strategy.

Returns:
pd.DataFrame: DataFrame summarizing ETH and TKN swapped by each strategy.
"""

def create_total_volume_swapped_table(trade_summary):
    """
    Generates a table summarizing the total volume of ETH and TKN swapped by␣
 ↪each strategy.

    Parameters:
    trade_summary (dict): A dictionary where keys are strategy names, and values␣
 ↪are dictionaries
                          containing 'ETH Swapped' and 'TKN Swapped' volumes.

    Returns:
    pd.DataFrame: A DataFrame summarizing the total volume of ETH and TKN␣
 ↪swapped by each strategy.
    """
    # Convert the trade summary dictionary to a DataFrame
    df = pd.DataFrame.from_dict(trade_summary, orient='index').fillna(0)
    df.index.name = 'Strategy'  # Set the index name for the DataFrame
    return df  # Return the DataFrame summarizing the total volume swapped

def update_total_volume_swapped(trade_summary, strategy_name, participant):
    """
    Updates the trade summary dictionary with the total ETH and TKN swapped by a␣
 ↪participant.

    Parameters:
    trade_summary (dict): The dictionary to update, keyed by strategy name.
    strategy_name (str): The name of the strategy being updated.
```

```
    participant (Participant): The participant whose swaps are being tracked.
    """
    if strategy_name not in trade_summary:
        trade_summary[strategy_name] = {'ETH Swapped': 0, 'TKN Swapped': 0}

    # Update the totals with the values from the participant
    trade_summary[strategy_name]['ETH Swapped'] += participant.total_eth_swapped
    trade_summary[strategy_name]['TKN Swapped'] += participant.total_tkn_swapped


# Create a table of ROI per strategy.
"""
Creates a DataFrame summarizing the ROI of each strategy.

Parameters:
results (list): List of tuples containing strategy names and their corresponding␣
 ↪ROI.

Returns:
pd.DataFrame: DataFrame summarizing ROI for each strategy.
"""

def create_roi_per_strategy_table(results):
    """
    Generates a table summarizing the Return on Investment (ROI) for each␣
 ↪strategy.

    Parameters:
    results (list): A list of tuples, where each tuple contains a strategy name␣
 ↪and its corresponding ROI.

    Returns:
    pd.DataFrame: A DataFrame summarizing the ROI for each strategy.
    """
    # Create a DataFrame from the list of results
    df = pd.DataFrame(results, columns=['Strategy', 'ROI'])

    # Ensure that all strategies are included, even if they have no ROI data
    if df.empty:
        df = pd.DataFrame({'Strategy': [], 'ROI': []})

    return df  # Return the DataFrame summarizing the ROI per strategy

# Impermanent Loss Calculation Function
"""
Calculates impermanent loss based on initial and current reserves of ETH and TKN.
```

```
Parameters:
initial_eth (float): Initial ETH reserve.
initial_tkn (float): Initial TKN reserve.
current_eth (float): Current ETH reserve.
current_tkn (float): Current TKN reserve.
eth_price (float): Current ETH price.
tkn_price (float): Current TKN price.

Returns:
float: Impermanent loss percentage.
"""

def calculate_impermanent_loss(initial_eth, initial_tkn, current_eth,␣
↪current_tkn, eth_price, tkn_price):
    """
    Calculates the impermanent loss experienced by a liquidity provider.

    Parameters:
    initial_eth (float): Initial amount of ETH in the pool.
    initial_tkn (float): Initial amount of TKN in the pool.
    current_eth (float): Current amount of ETH in the pool.
    current_tkn (float): Current amount of TKN in the pool.
    eth_price (float): Current market price of ETH.
    tkn_price (float): Current market price of TKN.

    Returns:
    float: The percentage of impermanent loss.
    """
    # Calculate the initial value of the pool's reserves
    initial_value = initial_eth * eth_price + initial_tkn * tkn_price

    # Calculate the value if the liquidity provider had held assets outside of␣
↪the pool
    held_value = (initial_eth + initial_tkn / tkn_price) * eth_price

    # Calculate the current value of the pool's reserves
    current_value = current_eth * eth_price + current_tkn * tkn_price

    # If there are no reserves or values are zero, return zero impermanent loss
    if initial_value == 0 or current_value == 0:
        return 0

    # Calculate the impermanent loss as a percentage
    impermanent_loss = (held_value - current_value) / held_value * 100

    return impermanent_loss
```

```
# Create a table of price history.
"""
Creates a DataFrame summarizing the daily price history of ETH during the␣
 ↪simulation.

Parameters:
price_history (list): List of ETH prices over time.

Returns:
pd.DataFrame: DataFrame containing the day and corresponding ETH price.
"""

def create_price_history_table(price_history):
    """
    Generates a table that records the daily price of ETH over the simulation␣
 ↪period.

    Parameters:
    price_history (list): A list of ETH prices recorded for each day of the␣
 ↪simulation.

    Returns:
    pd.DataFrame: A DataFrame with columns 'Day' and 'Price', summarizing the␣
 ↪ETH price history.
    """
    # Create a DataFrame from the price history data
    df = pd.DataFrame({
        'Day': range(len(price_history)),
        'Price': price_history
    })
    return df  # Return the DataFrame containing the price history

# Create a table tracking volatility over time.
"""
Creates a DataFrame tracking the volatility of ETH prices during the simulation.

Parameters:
price_history (list): List of ETH prices over time.

Returns:
pd.DataFrame: DataFrame containing the day and the corresponding annualized␣
 ↪volatility.
"""

def create_volatility_tracking_table(price_history, window_size=30):
    """
```

```
    Generates a table that tracks the volatility of ETH prices over the␣
↪simulation period.

    Volatility is calculated based on the daily percentage changes in ETH price,
    using a rolling window to smooth the results.

    Parameters:
    price_history (list): A list of ETH prices recorded for each day of the␣
↪simulation.
    window_size (int): The window size for calculating rolling volatility.␣
↪Defaults to 30.

    Returns:
    pd.DataFrame: A DataFrame with columns 'Day' and 'Volatility', showing the␣
↪annualized volatility.
    """
    # Calculate daily returns
    returns = pd.Series(price_history).pct_change()

    # Calculate volatility using a rolling window; initial values where the␣
↪window isn't full will have fewer data points
    volatility = returns.rolling(window=window_size, min_periods=1).std() * np.
↪sqrt(365)  # Annualized volatility

    # Replace NaN values with 0.0 for the first few days where there isn't␣
↪enough data
    volatility.fillna(0.0, inplace=True)

    # Create a DataFrame to store the volatility data
    df = pd.DataFrame({
        'Day': range(len(volatility)),
        'Volatility': volatility
    })

    return df  # Return the DataFrame containing volatility over time


# Create a table showing deviations from the mean reversion level.
"""
Creates a DataFrame tracking deviations of ETH prices from a mean reversion␣
↪level during the simulation.

Parameters:
price_history (list): List of ETH prices over time.
mean_reversion_level (float): The mean reversion level to compare prices against.
```

```
Returns:
pd.DataFrame: DataFrame containing the day, ETH price, and deviation from the␣
 ↪mean reversion level.
"""

def create_mean_reversion_table(price_history, mean_reversion_level):
    """
    Generates a table that tracks the deviation of ETH prices from a specified␣
 ↪mean reversion level.

    Parameters:
    price_history (list): A list of ETH prices recorded for each day of the␣
 ↪simulation.
    mean_reversion_level (float): The level to which prices are expected to␣
 ↪revert.

    Returns:
    pd.DataFrame: A DataFrame with columns 'Day', 'Price', and␣
 ↪'Deviation_from_Mean',
                  showing the daily deviations from the mean reversion level.
    """
    # Calculate the deviation of each price from the mean reversion level
    deviations = [price - mean_reversion_level for price in price_history]

    # Create a DataFrame to store the deviations and price history
    df = pd.DataFrame({
        'Day': range(len(price_history)),
        'Price': price_history,
        'Deviation_from_Mean': deviations
    })
    return df  # Return the DataFrame containing deviations from the mean␣
 ↪reversion level


# Create a table of impermanent losses over time.
"""
Creates a DataFrame summarizing impermanent losses for each day of the␣
 ↪simulation.

Parameters:
impermanent_losses (list): List of impermanent loss values over time.

Returns:
pd.DataFrame: DataFrame containing the day and corresponding impermanent loss.
"""
```

```
def create_impermanent_loss_table(impermanent_losses):
    """
    Generates a table that tracks the impermanent loss experienced by liquidity␣
 ↪providers over time.

    Impermanent loss is calculated for each day of the simulation based on␣
 ↪changes in the pool's reserves.

    Parameters:
    impermanent_losses (list): A list of impermanent loss percentages recorded␣
 ↪for each day of the simulation.

    Returns:
    pd.DataFrame: A DataFrame with columns 'Day' and 'Impermanent_Loss',
                  showing the daily impermanent loss.
    """
    # Create a DataFrame from the impermanent loss data
    df = pd.DataFrame({
        'Day': range(len(impermanent_losses)),
        'Impermanent_Loss': impermanent_losses
    })
    return df  # Return the DataFrame containing impermanent loss over time

# Create a table of total liquidity provider (LP) fees over time.
"""
Creates a DataFrame summarizing the accumulated LP fees (in ETH and TKN) for␣
 ↪each day of the simulation.

Parameters:
lp_fees_over_time (list): List of LP fees accumulated over time.

Returns:
pd.DataFrame: DataFrame containing the day and corresponding LP fees in ETH and␣
 ↪TKN.
"""

def create_total_lp_fees_table(lp_fees_over_time):
    """
    Generates a table that tracks the total liquidity provider fees accumulated␣
 ↪over the simulation period.

    The fees are tracked separately for ETH and TKN, and recorded for each day.

    Parameters:
    lp_fees_over_time (list): A list of tuples, each containing the accumulated␣
 ↪ETH and TKN fees for each day.
```

```
    Returns:
    pd.DataFrame: A DataFrame with columns 'Day', 'ETH_Fees', and 'TKN_Fees',
                  showing the daily accumulated fees for liquidity providers.
    """
    # Create a DataFrame from the LP fees data
    df = pd.DataFrame(lp_fees_over_time, columns=['ETH_Fees', 'TKN_Fees'])
    df.index.name = 'Day'  # Set the index name for the DataFrame
    return df  # Return the DataFrame containing total LP fees over time

def calculate_sharpe_ratio(daily_returns, risk_free_rate=0.05):
    """
    Calculates the Sharpe Ratio for a series of daily returns.

    Parameters:
    daily_returns (list): A list of daily returns for the strategy.
    risk_free_rate (float): The annual risk-free rate, default is 0.05 (5%).

    Returns:
    float: The Sharpe Ratio for the strategy.
    """
    # Convert risk-free rate to a daily rate
    daily_risk_free_rate = risk_free_rate / 365

    # Calculate the excess returns
    excess_returns = [r - daily_risk_free_rate for r in daily_returns]

    # Calculate the mean of excess returns
    mean_excess_return = np.mean(excess_returns)

    # Calculate the standard deviation of excess returns
    std_dev_excess_return = np.std(excess_returns)

    # If the standard deviation is zero, return NaN for the Sharpe Ratio
    if std_dev_excess_return == 0:
        return np.nan

    # Calculate the Sharpe Ratio
    sharpe_ratio = mean_excess_return / std_dev_excess_return
    return sharpe_ratio

def calculate_daily_returns(strategy, price_history):
    """
    Calculates daily returns for a strategy based on its USD balance over time.

    Parameters:
    strategy (Strategy): The strategy instance.
```

```
    price_history (list): The list of ETH prices over the simulation period.

    Returns:
    list: A list of daily returns as percentages.
    """
    daily_returns = []
    initial_balance = strategy.participant.initial_usd_balance

    for day in range(1, len(price_history)):
        previous_day_balance = (strategy.participant.eth_balance *␣
↪price_history[day-1]) + \
                               (strategy.participant.tkn_balance * (1/
↪price_history[day-1]))
        current_day_balance = (strategy.participant.eth_balance *␣
↪price_history[day]) + \
                              (strategy.participant.tkn_balance * (1/
↪price_history[day]))

        # Calculate the daily return
        daily_return = (current_day_balance - previous_day_balance) /␣
↪previous_day_balance
        daily_returns.append(daily_return)

    return daily_returns
```

[1585]:
```
# Function to create a corrected final balances table for all strategies.
def create_corrected_final_balances_table(strategy_instances, price_history):
    """
    Generates a corrected summary table containing the final ETH and TKN␣
↪balances,
    as well as the calculated ROI for each strategy after the simulation.

    Parameters:
    strategy_instances (list): A list of strategy objects used in the simulation.

    Returns:
    pd.DataFrame: A DataFrame summarizing the final ETH and TKN balances,
                  final USD balance, and placeholder for Sharpe Ratio for each␣
↪strategy.
    """
    data = []  # Initialize an empty list to collect strategy data
    for strategy in strategy_instances:
        # Retrieve current prices
        current_eth_price = strategy.participant.get_current_eth_price()
        current_tkn_price = strategy.participant.get_current_tkn_price()
```

```
        # Calculate final USD balance
        final_usd_balance = (strategy.participant.eth_balance *␣
↪current_eth_price) + (strategy.participant.tkn_balance * current_tkn_price)
        roi = strategy.participant.calculate_roi()

        # Calculate daily returns and Sharpe Ratio
        daily_returns = calculate_daily_returns(strategy, price_history)
        sharpe_ratio = calculate_sharpe_ratio(daily_returns)

        # Debugging: Print strategy details
        print(f"Strategy: {strategy.name}")
        print(f"  Final ETH Balance: {strategy.participant.eth_balance}")
        print(f"  Final TKN Balance: {strategy.participant.tkn_balance}")
        print(f"  Final USD Balance: {final_usd_balance}")
        print(f"  ROI: {roi}")

        # Append strategy data to the list
        data.append({
            'Strategy': strategy.name,  # Name of the strategy
            'Final_ETH_Balance': strategy.participant.eth_balance,  # Final ETH␣
↪balance after simulation
            'Final_TKN_Balance': strategy.participant.tkn_balance,  # Final TKN␣
↪balance after simulation
            'Final_USD_Balance': final_usd_balance,  # Final USD balance after␣
↪simulation
            'ROI': roi,  # ROI percentage
            'Sharpe_Ratio': sharpe_ratio  # Sharpe Ratio
        })

    df = pd.DataFrame(data)  # Convert the list of data into a pandas DataFrame
    print("Final Balances DataFrame:\n", df)  # Debugging: Print the final␣
↪DataFrame
    return df  # Return the DataFrame summarizing the final balances

# Function to create a corrected table of total volume swapped per strategy.
def create_corrected_total_volume_swapped_table(trade_summary):
    """
    Generates a corrected table summarizing the total volume of ETH and TKN␣
↪swapped by each strategy.

    Parameters:
    trade_summary (dict): A dictionary where keys are strategy names, and values␣
↪are dictionaries
                          containing 'ETH Swapped' and 'TKN Swapped' volumes.

    Returns:
```

```
    pd.DataFrame: A DataFrame summarizing the total volume of ETH and TKN␣
 ↪swapped by each strategy.
    """
    # Convert the trade summary dictionary to a DataFrame
    df = pd.DataFrame.from_dict(trade_summary, orient='index')
    df.index.name = 'Strategy'  # Set the index name for the DataFrame
    return df  # Return the DataFrame summarizing the total volume swapped

# Function to create a corrected table of ROI per strategy.
def create_corrected_roi_per_strategy_table(results):
    """
    Generates a corrected table summarizing the Return on Investment (ROI) for␣
 ↪each strategy.

    Parameters:
    results (list): A list of tuples, where each tuple contains a strategy name␣
 ↪and its corresponding ROI.

    Returns:
    pd.DataFrame: A DataFrame summarizing the ROI for each strategy.
    """
    # Create a DataFrame from the list of results
    df = pd.DataFrame(results, columns=['Strategy', 'ROI'])
    return df  # Return the DataFrame summarizing the ROI per strategy

# Function to display the corrected tables.
def display_corrected_tables(strategies, trade_summary, results):
    """
    Displays the corrected final balances, total volume swapped, and ROI per␣
 ↪strategy tables.

    This function generates and prints out the three corrected tables that␣
 ↪summarize
    the final results of the simulation.

    Parameters:
    strategies (list): A list of strategy objects used in the simulation.
    trade_summary (dict): A dictionary summarizing trade volumes for each␣
 ↪strategy.
    results (list): A list of tuples containing strategy names and their␣
 ↪corresponding ROI.

    Returns:
    None
    """
    # Generate the corrected final balances table
```

```
    final_balances_table_corrected =␣
↪create_corrected_final_balances_table(strategies)

    # Generate the corrected total volume swapped table
    total_volume_swapped_table_corrected =␣
↪create_corrected_total_volume_swapped_table(trade_summary)

    # Generate the corrected ROI per strategy table
    roi_per_strategy_table_corrected =␣
↪create_corrected_roi_per_strategy_table(results)

    # Display the corrected final balances table
    print("Corrected Final Balances Table")
    print(final_balances_table_corrected)

    # Display the corrected total volume swapped table
    print("\nCorrected Total Volume Swapped Table")
    print(total_volume_swapped_table_corrected)

    # Display the corrected ROI per strategy table
    print("\nCorrected ROI per Strategy Table")
    print(roi_per_strategy_table_corrected)
```

[1586]:
```
# Function to debug and verify the data structures.
def debug_and_verify_data(strategies, trade_summary, results):
    """
    Debugs and verifies the content of the data structures used in the␣
↪simulation.

    This function prints the details of the strategies, trade summary, and ROI␣
↪results
    to help in identifying any discrepancies or issues in the data.

    Parameters:
    strategies (list): A list of strategy objects used in the simulation.
    trade_summary (dict): A dictionary summarizing trade volumes for each␣
↪strategy.
    results (list): A list of tuples containing strategy names and their␣
↪corresponding ROI.

    Returns:
    None
    """
    print("Debugging Data Structures:")

    # Print details of each strategy in the strategies list
```

```
    for strategy in strategies:
        print(f"Strategy: {strategy.name}")
        print(f"  ETH Balance: {strategy.participant.eth_balance}")  # Print the␣
↪ETH balance of the strategy
        print(f"  TKN Balance: {strategy.participant.tkn_balance}")  # Print the␣
↪TKN balance of the strategy
        print(f"  ROI: {strategy.participant.calculate_roi()}")  # Print the ROI␣
↪calculated for the strategy

    # Print the trade summary data structure
    print("\nTrade Summary:")
    for key, value in trade_summary.items():
        print(f"Strategy: {key}, Data: {value}")  # Print the strategy name and␣
↪its corresponding trade data

    # Print the results data structure (ROI results)
    print("\nROI Results:")
    for result in results:
        print(f"Strategy: {result[0]}, ROI: {result[1]}")  # Access the tuple by␣
↪index and print the strategy name and ROI

# Update the display function to include debugging
def display_corrected_tables_with_debugging(strategies, trade_summary, results):
    """
    Displays the corrected tables and includes debugging information.

    This function first calls the debugging function to print the content of the␣
↪data structures,
    and then generates and prints the corrected tables for final balances, total␣
↪volume swapped,
    and ROI per strategy.

    Parameters:
    strategies (list): A list of strategy objects used in the simulation.
    trade_summary (dict): A dictionary summarizing trade volumes for each␣
↪strategy.
    results (list): A list of tuples containing strategy names and their␣
↪corresponding ROI.

    Returns:
    None
    """
    # Call the debug function to print and verify the data structures
    debug_and_verify_data(strategies, trade_summary, results)

    # Generate the corrected final balances table
```

```
    final_balances_table_corrected =␣
↪create_corrected_final_balances_table(strategies)

    # Generate the corrected total volume swapped table
    total_volume_swapped_table_corrected =␣
↪create_corrected_total_volume_swapped_table(trade_summary)

    # Generate the corrected ROI per strategy table
    roi_per_strategy_table_corrected =␣
↪create_corrected_roi_per_strategy_table(results)

    # Display the corrected final balances table
    print("\nCorrected Final Balances Table")
    print(final_balances_table_corrected)

    # Display the corrected total volume swapped table
    print("\nCorrected Total Volume Swapped Table")
    print(total_volume_swapped_table_corrected)

    # Display the corrected ROI per strategy table
    print("\nCorrected ROI per Strategy Table")
    print(roi_per_strategy_table_corrected)
```

[1587]:
```
# Function to plot the ROI by strategy.
def plot_roi_by_strategy(results):
    """
    Plots a bar chart showing the Return on Investment (ROI) for each strategy.

    Parameters:
    results (list): A list of tuples where each tuple contains a strategy name␣
↪and its corresponding ROI.
    """
    # Extract strategy names and their corresponding ROIs from the results
    strategy_names = [name for name, roi in results]
    rois = [roi for name, roi in results]

    # Plot the ROI for each strategy using a barplot
    sns.barplot(x=strategy_names, y=rois)
    plt.title('ROI by Strategy')  # Title of the plot
    plt.ylabel('ROI (%)')  # Label for the y-axis
    plt.xlabel('Strategy')  # Label for the x-axis
    plt.xticks(rotation=45)  # Rotate the strategy names for better readability
    plt.show()  # Display the plot

# Function to plot the ETH price path over time.
def plot_price_path(price_history):
    """
```

```
    Plots the price path of ETH over the simulation period.

    Parameters:
    price_history (list): A list of ETH prices recorded for each day of the␣
 ↪simulation.
    """
    plt.plot(price_history)  # Plot the ETH price over time
    plt.title('ETH Price Path Over Time')  # Title of the plot
    plt.xlabel('Day')  # Label for the x-axis
    plt.ylabel('Price')  # Label for the y-axis
    plt.show()  # Display the plot

# Function to plot the volatility of ETH over time.
def plot_volatility_over_time(volatility_table):
    """
    Plots the volatility of ETH over time, calculated using a rolling window.

    Parameters:
    price_history (list): A list of ETH prices recorded for each day of the␣
 ↪simulation.
    """
    # Calculate daily returns and then compute volatility using a rolling window
    plt.figure(figsize=(10, 6))
    plt.plot(volatility_table['Day'], volatility_table['Volatility'],␣
 ↪color='blue', linewidth=2)
    plt.title('Volatility Over Time')
    plt.xlabel('Day')
    plt.ylabel('Volatility (Annualized)')
    plt.grid(True)
    plt.show()

def simulate_market_conditions(market_conditions, days):
    """
    Simulates ETH price over a specified number of days using a Geometric␣
 ↪Brownian Motion model.
    """
    # Load or assign simulation parameters for drift and volatility
    mu = market_conditions.get('mu', 0.1)
    sigma = market_conditions.get('volatility', 0.3)
    initial_price = 2500
    eth_prices = simulate_eth_price_gbm(initial_price, days, mu, sigma)

    return eth_prices

def simulate_and_track_volatility(days, market_conditions):
    """
```

```
    Simulates the ETH market conditions and tracks the volatility over the␣
↪simulation period.

    Parameters:
    days (int): The number of days to simulate.
    market_conditions (dict): Parameters for the simulation.

    Returns:
    pd.DataFrame: A DataFrame tracking the volatility over the simulated period.
    """
    # Simulate ETH prices based on the given market conditions
    eth_prices = simulate_market_conditions(market_conditions, days)

    # Create a volatility tracking table
    volatility_table = create_volatility_tracking_table(eth_prices)

    # Plot the volatility over time for visual confirmation
    plot_volatility_over_time(volatility_table)

    return volatility_table

# Function to plot the mean reversion dynamics of ETH prices.
def plot_mean_reversion(price_history, mean_reversion_level):
    """
    Plots the deviation of ETH prices from a specified mean reversion level over␣
↪time.

    Parameters:
    price_history (list): A list of ETH prices recorded for each day of the␣
↪simulation.
    mean_reversion_level (float): The level to which prices are expected to␣
↪revert.
    """
    # Calculate the deviation of each price from the mean reversion level
    deviations = [price - mean_reversion_level for price in price_history]
    plt.figure(figsize=(10, 6))
    plt.plot(deviations, color='purple')  # Plot the deviations over time
    plt.title('Mean Reversion Dynamics')  # Title of the plot
    plt.xlabel('Day')  # Label for the x-axis
    plt.ylabel('Deviation from Mean')  # Label for the y-axis
    plt.grid(True)
    plt.show()  # Display the plot

# Function to plot impermanent loss over time.
def plot_impermanent_loss_over_time(impermanent_losses):
    """
    Plots the impermanent loss experienced by liquidity providers over time.
```

```
    Parameters:
    impermanent_losses (list): A list of impermanent loss percentages recorded␣
 ↪for each day of the simulation.
    """
    plt.plot(impermanent_losses)  # Plot the impermanent loss over time
    plt.title('Impermanent Loss Over Time')  # Title of the plot
    plt.xlabel('Day')  # Label for the x-axis
    plt.ylabel('Impermanent Loss (%)')  # Label for the y-axis
    plt.show()  # Display the plot

# Function to plot ETH reserves over time.
def plot_eth_reserves_over_time(eth_reserves_over_time):
    """
    Plots the ETH reserves in the trading pool over time.

    Parameters:
    eth_reserves_over_time (list): A list of ETH reserves recorded for each day␣
 ↪of the simulation.
    """
    plt.plot(eth_reserves_over_time)  # Plot the ETH reserves over time
    plt.title('ETH Reserves Over Time')  # Title of the plot
    plt.xlabel('Day')  # Label for the x-axis
    plt.ylabel('ETH Reserves')  # Label for the y-axis
    plt.show()  # Display the plot

# Function to plot TKN reserves over time.
def plot_tkn_reserves_over_time(tkn_reserves_over_time):
    """
    Plots the TKN reserves in the trading pool over time.

    Parameters:
    tkn_reserves_over_time (list): A list of TKN reserves recorded for each day␣
 ↪of the simulation.
    """
    plt.plot(tkn_reserves_over_time)  # Plot the TKN reserves over time
    plt.title('TKN Reserves Over Time')  # Title of the plot
    plt.xlabel('Day')  # Label for the x-axis
    plt.ylabel('TKN Reserves')  # Label for the y-axis
    plt.show()  # Display the plot
```

[1588]:
```
# Function to plot a histogram of price changes.

def plot_tkn_price_path(tkn_price_history):
    """
    Plots the price path of TKN over the simulation period.
```

```
    Parameters:
    tkn_price_history (list): A list of TKN prices recorded for each day of the␣
 ↪simulation.
    """
    plt.plot(tkn_price_history)  # Plot the TKN price over time
    plt.title('TKN Price Path Over Time')  # Title of the plot
    plt.xlabel('Day')  # Label for the x-axis
    plt.ylabel('Price')  # Label for the y-axis
    plt.show()  # Display the plot


def plot_histogram_of_price_changes(price_history):
    """
    Plots a histogram of percentage changes in ETH prices during the simulation.

    Parameters:
    price_history (list): A list of ETH prices recorded for each day of the␣
 ↪simulation.
    """
    # Calculate daily percentage changes in price and drop any missing values (e.
 ↪g., first day has no change)
    price_changes = pd.Series(price_history).pct_change().dropna()

    # Plot the histogram of price changes with a kernel density estimate (KDE)␣
 ↪overlay
    sns.histplot(price_changes, kde=True, bins=50)
    plt.title('Histogram of Price Changes')  # Title of the plot
    plt.xlabel('Percentage Change')  # Label for the x-axis
    plt.ylabel('Frequency')  # Label for the y-axis
    plt.show()  # Display the plot

# Function to plot the distribution of slippage.
def plot_slippage_distribution(price_impacts):
    """
    Plots a histogram of slippage experienced during trades in the simulation.

    Parameters:
    price_impacts (list): A list of slippage values recorded for each trade in␣
 ↪the simulation.
    """
    # Plot the histogram of slippage with a KDE overlay
    sns.histplot(price_impacts, kde=True, bins=50)
    plt.title('Slippage Distribution')  # Title of the plot
    plt.xlabel('Slippage')  # Label for the x-axis
    plt.ylabel('Frequency')  # Label for the y-axis
    plt.show()  # Display the plot
```

```
# Function to plot the number of trades made by each strategy.
def plot_trade_count_by_strategy(trade_counts):
    """
    Plots a bar chart showing the number of trades executed by each strategy.

    Parameters:
    trade_counts (dict): A dictionary where keys are strategy names and values␣
 ↪are the number of trades executed.
    """
    # Extract strategy names and their corresponding trade counts
    strategies = list(trade_counts.keys())
    counts = list(trade_counts.values())

    # Plot the number of trades for each strategy using a barplot
    sns.barplot(x=strategies, y=counts)
    plt.title('Trade Count by Strategy')  # Title of the plot
    plt.ylabel('Number of Trades')  # Label for the y-axis
    plt.xlabel('Strategy')  # Label for the x-axis
    plt.xticks(rotation=45)  # Rotate the strategy names for better readability
    plt.show()  # Display the plot

# Function to plot ETH price history with SMA and EMA overlays.
def plot_price_history_with_sma_ema(price_history):
    """
    Plots the ETH price history with Simple Moving Average (SMA) and Exponential␣
 ↪Moving Average (EMA) overlays.

    Parameters:
    price_history (list): A list of ETH prices recorded for each day of the␣
 ↪simulation.
    """
    # Create a DataFrame to hold the price data
    df = pd.DataFrame({
        'Price': price_history
    })

    # Calculate the 30-day SMA and EMA
    df['SMA'] = df['Price'].rolling(window=30).mean()
    df['EMA'] = df['Price'].ewm(span=30, adjust=False).mean()

    # Plot the price, SMA, and EMA
    plt.plot(df['Price'], label='Price')
    plt.plot(df['SMA'], label='SMA (30)')
    plt.plot(df['EMA'], label='EMA (30)')
    plt.title('Price History with SMA/EMA')  # Title of the plot
    plt.xlabel('Day')  # Label for the x-axis
    plt.ylabel('Price')  # Label for the y-axis
```

```
    plt.legend()  # Add a legend to distinguish between Price, SMA, and EMA
    plt.show()  # Display the plot

# Function to plot the final ETH balances by strategy.
def plot_final_eth_balances_by_strategy(final_balances_table):
    """
    Plots a bar chart showing the final ETH balances held by each strategy at␣
 ↪the end of the simulation.

    Parameters:
    final_balances_table (pd.DataFrame): A DataFrame containing the final ETH␣
 ↪balances for each strategy.
    """
    # Plot the final ETH balances for each strategy using a barplot
    sns.barplot(x='Strategy', y='Final_ETH_Balance', data=final_balances_table)
    plt.title('Final ETH Balances by Strategy')  # Title of the plot
    plt.ylabel('ETH Balance')  # Label for the y-axis
    plt.xlabel('Strategy')  # Label for the x-axis
    plt.xticks(rotation=45)  # Rotate the strategy names for better readability
    plt.show()  # Display the plot

# Function to plot the final TKN balances by strategy.
def plot_final_tkn_balances_by_strategy(final_balances_table):
    """
    Plots a bar chart showing the final TKN balances held by each strategy at␣
 ↪the end of the simulation.

    Parameters:
    final_balances_table (pd.DataFrame): A DataFrame containing the final TKN␣
 ↪balances for each strategy.
    """
    # Plot the final TKN balances for each strategy using a barplot
    sns.barplot(x='Strategy', y='Final_TKN_Balance', data=final_balances_table)
    plt.title('Final TKN Balances by Strategy')  # Title of the plot
    plt.ylabel('TKN Balance')  # Label for the y-axis
    plt.xlabel('Strategy')  # Label for the x-axis
    plt.xticks(rotation=45)  # Rotate the strategy names for better readability
    plt.show()  # Display the plot

# Function to plot the final USD balances by strategy.
def plot_final_usd_balances_by_strategy(final_balances_table):
    """
    Plots a bar chart showing the final USD balances for each strategy, based on␣
 ↪their ETH and TKN holdings.

    Parameters:
```

```
    final_balances_table (pd.DataFrame): A DataFrame containing the final USD␣
↪balances for each strategy.
    """
    # Plot the final USD balances for each strategy using a barplot
    sns.barplot(x='Strategy', y='Final_USD_Balance', data=final_balances_table)
    plt.title('Final USD Balances by Strategy')  # Title of the plot
    plt.ylabel('USD Balance')  # Label for the y-axis
    plt.xlabel('Strategy')  # Label for the x-axis
    plt.xticks(rotation=45)  # Rotate the strategy names for better readability
    plt.show()  # Display the plot

# Function to plot the total ETH swapped per strategy.
def plot_total_eth_swapped_per_strategy(total_volume_swapped_table):
    """
    Plots a bar chart showing the total volume of ETH swapped by each strategy␣
↪during the simulation.

    Parameters:
    total_volume_swapped_table (pd.DataFrame): A DataFrame summarizing the total␣
↪ETH swapped by each strategy.
    """
    # Plot the total ETH swapped by each strategy using a barplot
    sns.barplot(x=total_volume_swapped_table.index, y='ETH Swapped',␣
↪data=total_volume_swapped_table)
    plt.title('Total ETH Swapped per Strategy')  # Title of the plot
    plt.ylabel('Total ETH Swapped')  # Label for the y-axis
    plt.xlabel('Strategy')  # Label for the x-axis
    plt.xticks(rotation=45)  # Rotate the strategy names for better readability
    plt.show()  # Display the plot

# Function to plot the total TKN swapped per strategy.
def plot_total_tkn_swapped_per_strategy(total_volume_swapped_table):
    """
    Plots a bar chart showing the total volume of TKN swapped by each strategy␣
↪during the simulation.

    Parameters:
    total_volume_swapped_table (pd.DataFrame): A DataFrame summarizing the total␣
↪TKN swapped by each strategy.
    """
    # Plot the total TKN swapped by each strategy using a barplot
    sns.barplot(x=total_volume_swapped_table.index, y='TKN Swapped',␣
↪data=total_volume_swapped_table)
    plt.title('Total TKN Swapped per Strategy')  # Title of the plot
    plt.ylabel('Total TKN Swapped')  # Label for the y-axis
    plt.xlabel('Strategy')  # Label for the x-axis
```

```
    plt.xticks(rotation=45)  # Rotate the strategy names for better readability
    plt.show()  # Display the plot

# Function to plot the performance of strategies over time in terms of ETH and␣
↪TKN holdings.
def plot_strategy_performance_over_time(strategy_instances, price_history):
    """
    Plots the performance of each strategy over time, showing their ETH and TKN␣
↪holdings throughout the simulation.

    Parameters:
    strategy_instances (list): A list of strategy objects used in the simulation.
    price_history (list): A list of ETH prices recorded for each day of the␣
↪simulation.
    """
    days = list(range(len(price_history)))

    plt.figure(figsize=(14, 7))

    for strategy in strategy_instances:
        eth_holdings = [entry['eth_balance'] for entry in strategy.participant.
↪trade_history]

        # If eth_holdings is shorter than the number of days, pad it with the␣
↪last known value
        if len(eth_holdings) == 0:
            # Handle the case where eth_holdings is empty, you can initialize it␣
↪with a default value if appropriate
            eth_holdings = [0] * len(days)  # or any other default value you see␣
↪fit
        elif len(eth_holdings) < len(days):
            eth_holdings += [eth_holdings[-1]] * (len(days) - len(eth_holdings))
        else:
            eth_holdings = eth_holdings[:len(days)]

        plt.plot(days, eth_holdings, label=f'{strategy.name} ETH Holdings')

    plt.title('Strategy Performance Over Time (ETH Holdings)')
    plt.xlabel('Day')
    plt.ylabel('ETH Holdings')
    plt.legend()
    plt.grid(True)
    plt.show()

    # Plot TKN Holdings over time
    plt.figure(figsize=(12, 6))  # Set the figure size
```

```
    for strategy in strategy_instances:
        # Check if the strategy has made trades and extract TKN holdings after␣
↪each trade
        if strategy.participant.trade_history:
            tkn_holdings = [trade['tkn_balance'] for trade in strategy.
↪participant.trade_history]
        else:
            # If no trades were made, assume initial balances throughout the␣
↪period
            tkn_holdings = [strategy.participant.tkn_balance] * len(days)

        # Pad TKN holdings to match the number of days, if necessary
        if len(tkn_holdings) != len(days):
            if len(tkn_holdings) < len(days):
                tkn_holdings += [tkn_holdings[-1]] * (len(days) -␣
↪len(tkn_holdings))
            else:
                tkn_holdings = tkn_holdings[:len(days)]

        plt.plot(days, tkn_holdings, label=f'{strategy.name} TKN Holdings')  #␣
↪Plot TKN holdings over time

    plt.title('Strategy Performance Over Time (TKN Holdings)')  # Title of the␣
↪plot
    plt.xlabel('Day')  # Label for the x-axis
    plt.ylabel('TKN Holdings')  # Label for the y-axis
    plt.legend()  # Add a legend to distinguish between strategies
    plt.show()  # Display the plot
```

[1589]:
```
def main():
    market_conditions = load_json('bull_market.json')  # Load the correct JSON␣
↪configuration file
    days = market_conditions['simulation_parameters']['days_of_simulation']
    initial_eth = market_conditions['simulation_parameters']['initial_pool_eth']
    initial_tkn = market_conditions['simulation_parameters']['initial_pool_tkn']

    # Get the initial ETH and TKN prices from the correct section
    initial_eth_price =␣
↪market_conditions['market_conditions']['initial_eth_price']
    initial_tkn_price = 1  # Assuming initial TKN price is set to 1 as it's not␣
↪in the JSON

    pool = TradingPool(initial_eth, initial_tkn)

    strategies = [
        BullStrategy("Bull", 10, 0, initial_eth_price, initial_tkn_price),
```

```
        BearStrategy("Bear", 10, 0, initial_eth_price, initial_tkn_price),
        DcaStrategy("DCA", 10, 0, initial_eth_price, initial_tkn_price,␣
↪daily_investment=0.1),
        HarvesterStrategy("Harvester", 10, 100, initial_eth_price,␣
↪initial_tkn_price),
        DoveStrategy("Dove", 10, 0, initial_eth_price, initial_tkn_price),
        HawkStrategy("Hawk", 10, 0, initial_eth_price, initial_tkn_price,␣
↪target_profit=0.01),
        Sniper1Strategy("Sniper1", 10, 0, initial_eth_price, initial_tkn_price,␣
↪target_profit=0.1),
        Sniper2Strategy("Sniper2", 10, 0, initial_eth_price, initial_tkn_price,␣
↪delay=1, target_profit=0.1)
    ]

    # Initialize trade summary dictionary
    trade_summary = {}

    # Run the simulation
    eth_price_history, tkn_price_history, trade_counts, eth_reserves_over_time,␣
↪tkn_reserves_over_time, lp_fees_over_time, \
    impermanent_losses, price_impacts, detailed_logs =␣
↪run_simulation(strategies, pool, market_conditions, days)

    # Evaluate the results
    results = evaluate_simulation(strategies)

    # Update trade summary with actual volumes
    for strategy in strategies:
        update_total_volume_swapped(trade_summary, strategy.name, strategy.
↪participant)

    # Generate a summary of trades made by each strategy
    trade_summary = summarize_trades(strategies)

    # Create various tables to summarize the simulation results
    price_history_table = create_price_history_table(eth_price_history)
    volatility_table = create_volatility_tracking_table(eth_price_history)  #␣
↪Create the volatility table here
    mean_reversion_table = create_mean_reversion_table(eth_price_history,␣
↪mean_reversion_level=3000)
    final_balances_table = create_corrected_final_balances_table(strategies,␣
↪eth_price_history)  # Pass eth_price_history
    total_volume_swapped_table = create_total_volume_swapped_table(trade_summary)
    roi_per_strategy_table = create_roi_per_strategy_table(results)
    impermanent_loss_table = create_impermanent_loss_table(impermanent_losses)
    total_lp_fees_table = create_total_lp_fees_table(lp_fees_over_time)
```

```
    # Print the first few rows of each table for inspection
    print("Price History Table")
    print(price_history_table.head())
    print("\nVolatility Tracking Table")
    print(volatility_table.head())
    print("\nMean Reversion Table")
    print(mean_reversion_table.head())
    print("\nFinal Balances Table")
    print(final_balances_table.head())
    print("\nTotal Volume Swapped Table")
    print(total_volume_swapped_table.head())
    print("\nROI per Strategy Table")
    print(roi_per_strategy_table.head())
    print("\nImpermanent Loss Table")
    print(impermanent_loss_table.head())
    print("\nTotal LP Fees Table")
    print(total_lp_fees_table.head())

    # Return all relevant data, including strategies and the volatility table
    return strategies, eth_price_history, tkn_price_history,␣
↪final_balances_table, total_volume_swapped_table, results, price_impacts,␣
↪trade_counts, eth_reserves_over_time, tkn_reserves_over_time,␣
↪volatility_table, impermanent_losses

# Call main() and unpack the returned data
strategies, eth_price_history, tkn_price_history, final_balances_table,␣
↪total_volume_swapped_table, results, price_impacts, trade_counts,␣
↪eth_reserves_over_time, tkn_reserves_over_time, volatility_table,␣
↪impermanent_losses = main()

# Generate and display the plots
plot_roi_by_strategy(results)
plot_price_path(eth_price_history)  # ETH price plot
plot_tkn_price_path(tkn_price_history)  # New TKN price plot
plot_volatility_over_time(volatility_table)  # Now volatility_table should be␣
↪defined
plot_mean_reversion(eth_price_history, mean_reversion_level=3000)
plot_impermanent_loss_over_time(impermanent_losses)  # Now impermanent_losses␣
↪should be defined
plot_eth_reserves_over_time(eth_reserves_over_time)
plot_tkn_reserves_over_time(tkn_reserves_over_time)

# Additional Visualizations
plot_histogram_of_price_changes(eth_price_history)
plot_slippage_distribution(price_impacts)
plot_trade_count_by_strategy(trade_counts)
```

```
plot_price_history_with_sma_ema(eth_price_history)

# Added Visualizations
plot_final_eth_balances_by_strategy(final_balances_table)
plot_final_tkn_balances_by_strategy(final_balances_table)
plot_final_usd_balances_by_strategy(final_balances_table)
plot_total_eth_swapped_per_strategy(total_volume_swapped_table)
plot_total_tkn_swapped_per_strategy(total_volume_swapped_table)
plot_strategy_performance_over_time(strategies, eth_price_history)
```

```
Day 0: Sniper1 is executing forced trade
Sniper1: Purchased 2000.0 TKN at price 2500
Sniper1 ETH Balance: 0, TKN Balance: 4927.586379569353
Day 0: Sniper2 is executing forced trade
Sniper2 ETH Balance: 10, TKN Balance: 0
Day 0: Bull is checking triggers
Bull Post-Execution ETH Balance: 10, TKN Balance: 0
Day 0: Bear is checking triggers
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 0: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 0: Harvester is checking triggers
Harvester: Selling 20.0 TKN at price 2500
Harvester: Updated TKN balance is 80.0
Harvester Post-Execution ETH Balance: 10.175956717135797, TKN Balance: 80.0
Day 0: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 0: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 1: Bull is checking triggers
Bull Post-Execution ETH Balance: 10, TKN Balance: 0
Day 1: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 1: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 1: Harvester is checking triggers
Harvester: Selling 16.0 TKN at price 2520.266038005257
Harvester: Updated TKN balance is 64.0
Harvester Post-Execution ETH Balance: 10.473967754962954, TKN Balance: 64.0
Day 1: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 1: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 1: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 1: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 10, TKN Balance: 0
```

```
Day 2: Bull is checking triggers
Bull Post-Execution ETH Balance: 10, TKN Balance: 0
Day 2: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 2: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 2: Harvester is checking triggers
Harvester: Selling 12.8 TKN at price 2515.4892411174633
Harvester: Updated TKN balance is 51.2
Harvester Post-Execution ETH Balance: 10.670394936845566, TKN Balance: 51.2
Day 2: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 2: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 2: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 2: Sniper2 is checking triggers
Sniper2: Purchased 511.4449371325086 TKN at price 2515.4892411174633
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 516.6602019380239
Day 3: Bull is checking triggers
Bull Post-Execution ETH Balance: 10, TKN Balance: 0
Day 3: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 3: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 3: Harvester is checking triggers
Harvester: Selling 10.24 TKN at price 2541.8998064951365
Harvester: Updated TKN balance is 40.96
Harvester Post-Execution ETH Balance: 10.957347420954024, TKN Balance: 40.96
Day 3: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 3: Hawk is checking triggers
Hawk: Selling all TKN at price 2541.8998064951365 after reaching profit target
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 3: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 3: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 516.6602019380239
Day 4: Bull is checking triggers
Bull: Updated ETH balance is 10
Bull Post-Execution ETH Balance: 10, TKN Balance: 0
Day 4: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 4: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
```

```
Day 4: Harvester is checking triggers
Harvester: Selling 8.192 TKN at price 2604.1372679329706
Harvester: Updated TKN balance is 32.768
Harvester Post-Execution ETH Balance: 11.113824729646897, TKN Balance: 32.768
Day 4: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 4: Hawk is checking triggers
Hawk: Selling all TKN at price 2604.1372679329706 after reaching profit target
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 4: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 4: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 516.6602019380239
Day 5: Bull is checking triggers
Bull Post-Execution ETH Balance: 10, TKN Balance: 0
Day 5: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 5: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 5: Harvester is checking triggers
Harvester: Selling 6.5536 TKN at price 2595.2907830960385
Harvester: Updated TKN balance is 26.2144
Harvester Post-Execution ETH Balance: 11.19181265084174, TKN Balance: 26.2144
Day 5: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 5: Hawk is checking triggers
Hawk: Selling all TKN at price 2595.2907830960385 after reaching profit target
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 5: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 5: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 516.6602019380239
Day 6: Bull is checking triggers
Bull Post-Execution ETH Balance: 10, TKN Balance: 0
Day 6: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 6: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 6: Harvester is checking triggers
Harvester: Selling 5.24288 TKN at price 2586.4750173539264
Harvester: Updated TKN balance is 20.97152
Harvester Post-Execution ETH Balance: 11.253809503457648, TKN Balance: 20.97152
Day 6: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 6: Hawk is checking triggers
Hawk: Selling all TKN at price 2586.4750173539264 after reaching profit target
```

```
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 6: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 6: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 516.6602019380239
Day 7: Bull is checking triggers
Bull: Updated ETH balance is 10
Bull Post-Execution ETH Balance: 10, TKN Balance: 0
Day 7: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 7: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 7: Harvester is checking triggers
Harvester: Selling 4.194304 TKN at price 2652.142641197207
Harvester: Updated TKN balance is 16.777216000000003
Harvester Post-Execution ETH Balance: 11.30324631168401, TKN Balance:
16.777216000000003
Day 7: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 7: Hawk is checking triggers
Hawk: Selling all TKN at price 2652.142641197207 after reaching profit target
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 7: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 7: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 516.6602019380239
Day 8: Bull is checking triggers
Bull: Updated ETH balance is 10
Bull Post-Execution ETH Balance: 10, TKN Balance: 0
Day 8: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 8: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 8: Harvester is checking triggers
Harvester: Selling 3.3554432 TKN at price 2685.0319857376867
Harvester: Updated TKN balance is 13.421772800000003
Harvester Post-Execution ETH Balance: 11.342693648329835, TKN Balance:
13.421772800000003
Day 8: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 8: Hawk is checking triggers
Hawk: Selling all TKN at price 2685.0319857376867 after reaching profit target
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 8: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 8: Sniper2 is checking triggers
```

```
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 516.6602019380239
Day 9: Bull is checking triggers
Bull: Updated ETH balance is 10
Bull Post-Execution ETH Balance: 10, TKN Balance: 0
Day 9: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 9: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 9: Harvester is checking triggers
Harvester: Selling 2.68435456 TKN at price 2666.0409707318063
Harvester: Updated TKN balance is 10.737418240000004
Harvester Post-Execution ETH Balance: 11.37418639651347, TKN Balance:
10.737418240000004
Day 9: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 9: Hawk is checking triggers
Hawk: Selling all TKN at price 2666.0409707318063 after reaching profit target
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 9: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 9: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 516.6602019380239
Day 10: Bull is checking triggers
Bull: Updated ETH balance is 10
Bull Post-Execution ETH Balance: 10, TKN Balance: 0
Day 10: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 10: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 10: Harvester is checking triggers
Harvester: Selling 2.14748365 TKN at price 2689.588557451925
Harvester: Updated TKN balance is 8.589934590000004
Harvester Post-Execution ETH Balance: 11.399339034008747, TKN Balance:
8.589934590000004
Day 10: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 10: Hawk is checking triggers
Hawk: Selling all TKN at price 2689.588557451925 after reaching profit target
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 10: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 10: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 516.6602019380239
Day 11: Bull is checking triggers
Bull: Updated ETH balance is 10
Bull Post-Execution ETH Balance: 10, TKN Balance: 0
```

```
Day 11: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 11: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 11: Harvester is checking triggers
Harvester: Selling 1.71798692 TKN at price 2670.8193144922634
Harvester: Updated TKN balance is 6.871947670000004
Harvester Post-Execution ETH Balance: 11.419434604336821, TKN Balance:
6.871947670000004
Day 11: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 11: Hawk is checking triggers
Hawk: Selling all TKN at price 2670.8193144922634 after reaching profit target
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 11: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 11: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 516.6602019380239
Day 12: Bull is checking triggers
Bull Post-Execution ETH Balance: 10, TKN Balance: 0
Day 12: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 12: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 12: Harvester is checking triggers
Harvester: Selling 1.37438953 TKN at price 2652.084764964638
Harvester: Updated TKN balance is 5.497558140000004
Harvester Post-Execution ETH Balance: 11.435494105482546, TKN Balance:
5.497558140000004
Day 12: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 12: Hawk is checking triggers
Hawk: Selling all TKN at price 2652.084764964638 after reaching profit target
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 12: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 12: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 516.6602019380239
Day 13: Bull is checking triggers
Bull Post-Execution ETH Balance: 10, TKN Balance: 0
Day 13: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 13: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 13: Harvester is checking triggers
```

```
Harvester: Selling 1.09951163 TKN at price 2662.909900195288
Harvester: Updated TKN balance is 4.398046510000004
Harvester Post-Execution ETH Balance: 11.448330870739657, TKN Balance:
4.398046510000004
Day 13: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 13: Hawk is checking triggers
Hawk: Selling all TKN at price 2662.909900195288 after reaching profit target
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 13: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 13: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 516.6602019380239
Day 14: Bull is checking triggers
Bull: Actively buying TKN with 5.0 ETH at price 2584.8041611833373
Bull: Updated ETH balance is 5.0
Bull Post-Execution ETH Balance: 5.0, TKN Balance: 370.30726426156053
Day 14: Bear is checking triggers
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 14: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 14: Harvester is checking triggers
Harvester: Selling 0.8796093 TKN at price 2584.8041611833373
Harvester: Updated TKN balance is 3.5184372100000036
Harvester Post-Execution ETH Balance: 11.47983872657864, TKN Balance:
3.5184372100000036
Day 14: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 14: Hawk is checking triggers
Hawk: Selling all TKN at price 2584.8041611833373 after reaching profit target
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 14: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 14: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 516.6602019380239
Day 15: Bull is checking triggers
Bull: Actively buying TKN with 2.5 ETH at price 2516.421418533736
Bull: Updated ETH balance is 2.5
Bull Post-Execution ETH Balance: 2.5, TKN Balance: 522.6656369940378
Day 15: Bear is checking triggers
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 15: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 15: Harvester is checking triggers
Harvester: Selling 0.70368744 TKN at price 2516.421418533736
Harvester: Updated TKN balance is 2.8147497700000037
Harvester Post-Execution ETH Balance: 11.49998143084351, TKN Balance:
2.8147497700000037
```

```
Day 15: Dove is checking triggers
Dove: Cautiously buying TKN with 0.2 ETH at price 2516.421418533736
Dove Post-Execution ETH Balance: 9.8, TKN Balance: 11.426137856838182
Day 15: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 15: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 15: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 516.6602019380239
Day 16: Bull is checking triggers
Bull: Actively buying TKN with 1.25 ETH at price 2494.9840852664915
Bull: Updated ETH balance is 1.25
Bull Post-Execution ETH Balance: 1.25, TKN Balance: 591.6774919105383
Day 16: Bear is checking triggers
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 16: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 16: Harvester is checking triggers
Harvester: Selling 0.56294995 TKN at price 2494.9840852664915
Harvester: Updated TKN balance is 2.2517998200000036
Harvester Post-Execution ETH Balance: 11.514204207446737, TKN Balance:
2.2517998200000036
Day 16: Dove is checking triggers
Dove: Cautiously buying TKN with 0.196 ETH at price 2494.9840852664915
Dove Post-Execution ETH Balance: 9.604000000000001, TKN Balance:
21.893412421705527
Day 16: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 16: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 16: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 516.6602019380239
Day 17: Bull is checking triggers
Bull: Actively buying TKN with 0.625 ETH at price 2456.290099497565
Bull: Updated ETH balance is 0.625
Bull Post-Execution ETH Balance: 0.625, TKN Balance: 624.4534266271195
Day 17: Bear is checking triggers
Bear: Cautiously buying TKN with 1.0 ETH at price 2456.290099497565 during a
major dip
Bear: Updated TKN balance is 50.59367669125125
Bear Post-Execution ETH Balance: 9.0, TKN Balance: 50.59367669125125
Day 17: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 17: Harvester is checking triggers
Harvester: Selling 0.45035996 TKN at price 2456.290099497565
Harvester: Updated TKN balance is 1.8014398600000034
Harvester Post-Execution ETH Balance: 11.526271991597099, TKN Balance:
1.8014398600000034
```

```
Day 17: Dove is checking triggers
Dove: Cautiously buying TKN with 0.19208000000000003 ETH at price
2456.290099497565
Dove Post-Execution ETH Balance: 9.41192, TKN Balance: 31.345422568949047
Day 17: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 17: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 17: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 516.6602019380239
Day 18: Bull is checking triggers
Bull: Actively buying TKN with 0.3125 ETH at price 2469.1170861677865
Bull: Updated ETH balance is 0.3125
Bull Post-Execution ETH Balance: 0.3125, TKN Balance: 639.6856450927233
Day 18: Bear is checking triggers
Bear Post-Execution ETH Balance: 9.0, TKN Balance: 50.59367669125125
Day 18: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 18: Harvester is checking triggers
Harvester: Selling 0.36028797 TKN at price 2469.1170861677865
Harvester: Updated TKN balance is 1.4411518900000035
Harvester Post-Execution ETH Balance: 11.534577635145752, TKN Balance:
1.4411518900000035
Day 18: Dove is checking triggers
Dove: Cautiously buying TKN with 0.1882384 ETH at price 2469.1170861677865
Dove Post-Execution ETH Balance: 9.2236816, TKN Balance: 40.40805803908302
Day 18: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 18: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 18: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 516.6602019380239
Day 19: Bull is checking triggers
Bull: Actively buying TKN with 0.15625 ETH at price 2434.8281002071153
Bull: Updated ETH balance is 0.15625
Bull Post-Execution ETH Balance: 0.15625, TKN Balance: 647.1779156044278
Day 19: Bear is checking triggers
Bear: Cautiously buying TKN with 0.9 ETH at price 2434.8281002071153 during a
major dip
Bear: Updated TKN balance is 92.6362107015519
Bear Post-Execution ETH Balance: 8.1, TKN Balance: 92.6362107015519
Day 19: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 19: Harvester is checking triggers
Harvester: Selling 0.28823038 TKN at price 2434.8281002071153
Harvester: Updated TKN balance is 1.1529215100000036
Harvester Post-Execution ETH Balance: 11.543535078167022, TKN Balance:
1.1529215100000036
```

```
Day 19: Dove is checking triggers
Dove: Cautiously buying TKN with 0.18447363200000003 ETH at price
2434.8281002071153
Dove Post-Execution ETH Balance: 9.039207968000001, TKN Balance:
48.827695382838186
Day 19: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 19: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 19: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 516.6602019380239
Day 20: Bull is checking triggers
Bull: Actively buying TKN with 0.078125 ETH at price 2382.0777831395862
Bull: Updated ETH balance is 0.078125
Bull Post-Execution ETH Balance: 0.078125, TKN Balance: 650.7485412058859
Day 20: Bear is checking triggers
Bear: Cautiously buying TKN with 0.81 ETH at price 2382.0777831395862 during a
major dip
Bear: Updated TKN balance is 128.7165229332335
Bear Post-Execution ETH Balance: 7.289999999999999, TKN Balance:
128.7165229332335
Day 20: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 20: Harvester is checking triggers
Harvester: Selling 0.2305843 TKN at price 2382.0777831395862
Harvester: Updated TKN balance is 0.9223372100000037
Harvester Post-Execution ETH Balance: 11.551197211937351, TKN Balance:
0.9223372100000037
Day 20: Dove is checking triggers
Dove: Cautiously buying TKN with 0.18078415936000003 ETH at price
2382.0777831395862
Dove Post-Execution ETH Balance: 8.858423808640001, TKN Balance:
56.716471485766384
Day 20: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 20: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 20: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2382.0777831395862 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 21: Bull is checking triggers
Bull: Actively buying TKN with 0.0390625 ETH at price 2438.204155275336
Bull: Updated ETH balance is 0.0390625
Bull Post-Execution ETH Balance: 0.0390625, TKN Balance: 655.6007053504284
Day 21: Bear is checking triggers
Bear Post-Execution ETH Balance: 7.289999999999999, TKN Balance:
128.7165229332335
Day 21: DCA is checking triggers
```

```
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 21: Harvester is checking triggers
Harvester: Selling 0.18446744 TKN at price 2438.204155275336
Harvester: Updated TKN balance is 0.7378697700000036
Harvester Post-Execution ETH Balance: 11.55412474698769, TKN Balance:
0.7378697700000036
Day 21: Dove is checking triggers
Dove: Cautiously buying TKN with 0.17716847617280004 ETH at price
2438.204155275336
Dove Post-Execution ETH Balance: 8.6812553324672, TKN Balance: 68.57021771014234
Day 21: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 21: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 21: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 22: Bull is checking triggers
Bull: Actively buying TKN with 0.01953125 ETH at price 2430.241019672264
Bull: Updated ETH balance is 0.01953125
Bull Post-Execution ETH Balance: 0.01953125, TKN Balance: 656.9363978760542
Day 22: Bear is checking triggers
Bear Post-Execution ETH Balance: 7.289999999999999, TKN Balance:
128.7165229332335
Day 22: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 22: Harvester is checking triggers
Harvester: Selling 0.14757395 TKN at price 2430.241019672264
Harvester: Updated TKN balance is 0.5902958200000037
Harvester Post-Execution ETH Balance: 11.556376028669531, TKN Balance:
0.5902958200000037
Day 22: Dove is checking triggers
Dove Post-Execution ETH Balance: 8.6812553324672, TKN Balance: 68.57021771014234
Day 22: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 22: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 22: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 23: Bull is checking triggers
Bull: Actively buying TKN with 0.00976562 ETH at price 2433.485973660647
Bull: Updated ETH balance is 0.00976563
Bull Post-Execution ETH Balance: 0.00976563, TKN Balance: 657.5865003254966
Day 23: Bear is checking triggers
Bear Post-Execution ETH Balance: 7.289999999999999, TKN Balance:
128.7165229332335
Day 23: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 23: Harvester is checking triggers
```

```
Harvester: Selling 0.11805916 TKN at price 2433.485973660647
Harvester: Updated TKN balance is 0.4722366600000037
Harvester Post-Execution ETH Balance: 11.55815901017893, TKN Balance:
0.4722366600000037
Day 23: Dove is checking triggers
Dove Post-Execution ETH Balance: 8.6812553324672, TKN Balance: 68.57021771014234
Day 23: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 23: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 23: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 24: Bull is checking triggers
Bull: Actively buying TKN with 0.00488282 ETH at price 2380.299548536723
Bull: Updated ETH balance is 0.0048828100000000004
Bull Post-Execution ETH Balance: 0.0048828100000000004, TKN Balance:
657.911591748261
Day 24: Bear is checking triggers
Bear: Cautiously buying TKN with 0.729 ETH at price 2380.299548536723 during a
major dip
Bear: Updated TKN balance is 176.29035465023748
Bear Post-Execution ETH Balance: 6.560999999999999, TKN Balance:
176.29035465023748
Day 24: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 24: Harvester is checking triggers
Harvester: Selling 0.09444733 TKN at price 2380.299548536723
Harvester: Updated TKN balance is 0.3777893300000037
Harvester Post-Execution ETH Balance: 11.561832224237644, TKN Balance:
0.3777893300000037
Day 24: Dove is checking triggers
Dove: Cautiously buying TKN with 0.17362510664934402 ETH at price
2380.299548536723
Dove Post-Execution ETH Balance: 8.507630225817856, TKN Balance: 79.64731772291
Day 24: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 24: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 24: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2380.299548536723 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 25: Bull is checking triggers
Bull: Actively buying TKN with 0.00244141 ETH at price 2360.6854572765483
Bull: Updated ETH balance is 0.0024414000000000003
Bull Post-Execution ETH Balance: 0.0024414000000000003, TKN Balance:
658.0998903578668
Day 25: Bear is checking triggers
Bear: Cautiously buying TKN with 0.6561 ETH at price 2360.6854572765483 during a
```

```
major dip
Bear: Updated TKN balance is 217.27570572601365
Bear Post-Execution ETH Balance: 5.904899999999999, TKN Balance:
217.27570572601365
Day 25: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 25: Harvester is checking triggers
Harvester: Selling 0.07555787 TKN at price 2360.6854572765483
Harvester: Updated TKN balance is 0.30223146000000367
Harvester Post-Execution ETH Balance: 11.565036967333839, TKN Balance:
0.30223146000000367
Day 25: Dove is checking triggers
Dove: Cautiously buying TKN with 0.1701526045163571 ETH at price
2360.6854572765483
Dove Post-Execution ETH Balance: 8.337477621301499, TKN Balance:
90.06307762394107
Day 25: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 25: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2360.6854572765483 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 25: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2360.6854572765483 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 26: Bull is checking triggers
Bull: Actively buying TKN with 0.0012207 ETH at price 2365.448828846748
Bull: Updated ETH balance is 0.0012207000000000003
Bull Post-Execution ETH Balance: 0.0012207000000000003, TKN Balance:
881.0742184247151
Day 26: Bear is checking triggers
Bear: Cautiously buying TKN with 0.59049 ETH at price 2365.448828846748 during a
major dip
Bear: Updated TKN balance is 499.29549241107446
Bear Post-Execution ETH Balance: 5.314409999999999, TKN Balance:
499.29549241107446
Day 26: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 26: Harvester is checking triggers
Harvester: Selling 0.06044629 TKN at price 2365.448828846748
Harvester: Updated TKN balance is 0.24178517000000366
Harvester Post-Execution ETH Balance: 11.567015682042584, TKN Balance:
0.24178517000000366
Day 26: Dove is checking triggers
Dove Post-Execution ETH Balance: 8.337477621301499, TKN Balance:
90.06307762394107
Day 26: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 26: Sniper1 is checking triggers
```

```
Sniper1: Selling all TKN at price 2365.448828846748 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 26: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2365.448828846748 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 27: Bull is checking triggers
Bull: Actively buying TKN with 0.00061035 ETH at price 2323.7169307895133
Bull: Updated ETH balance is 0
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
Day 27: Bear is checking triggers
Bear: Cautiously buying TKN with 0.531441 ETH at price 2323.7169307895133 during
a major dip
Bear: Updated TKN balance is 734.024835300594
Bear Post-Execution ETH Balance: 4.782968999999999, TKN Balance:
734.024835300594
Day 27: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 27: Harvester is checking triggers
Harvester: Selling 0.04835703 TKN at price 2323.7169307895133
Harvester: Updated TKN balance is 0.19342814000000366
Harvester Post-Execution ETH Balance: 11.568775491327246, TKN Balance:
0.19342814000000366
Day 27: Dove is checking triggers
Dove: Cautiously buying TKN with 0.16674955242602998 ETH at price
2323.7169307895133
Dove Post-Execution ETH Balance: 8.17072806887547, TKN Balance:
160.42935783334113
Day 27: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 27: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2323.7169307895133 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 27: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2323.7169307895133 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 28: Bull is checking triggers
Bull: Updated ETH balance is 0
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
Day 28: Bear is checking triggers
Bear: Cautiously buying TKN with 0.4782969 ETH at price 2338.106658373268 during
a major dip
Bear: Updated TKN balance is 927.8667366489216
Bear Post-Execution ETH Balance: 4.304672099999999, TKN Balance:
927.8667366489216
Day 28: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 28: Harvester is checking triggers
Harvester: Selling 0.03868563 TKN at price 2338.106658373268
```

```
Harvester: Updated TKN balance is 0.15474251000000366
Harvester Post-Execution ETH Balance: 11.570348861611105, TKN Balance:
0.15474251000000366
Day 28: Dove is checking triggers
Dove Post-Execution ETH Balance: 8.17072806887547, TKN Balance:
160.42935783334113
Day 28: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 28: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2338.106658373268 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 28: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2338.106658373268 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 29: Bull is checking triggers
Bull: Updated ETH balance is 0
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
Day 29: Bear is checking triggers
Bear: Cautiously buying TKN with 0.43046721 ETH at price 2316.7927510247796
during a major dip
Bear: Updated TKN balance is 1092.5129446340259
Bear Post-Execution ETH Balance: 3.874204889999999, TKN Balance:
1092.5129446340259
Day 29: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 29: Harvester is checking triggers
Harvester: Selling 0.0309485 TKN at price 2316.7927510247796
Harvester: Updated TKN balance is 0.12379401000000366
Harvester Post-Execution ETH Balance: 11.57175189780573, TKN Balance:
0.12379401000000366
Day 29: Dove is checking triggers
Dove Post-Execution ETH Balance: 8.17072806887547, TKN Balance:
160.42935783334113
Day 29: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 29: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2316.7927510247796 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 29: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2316.7927510247796 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 30: Bull is checking triggers
Skipping Bull - trade already made today
Day 30: Bear is checking triggers
Skipping Bear - trade already made today
Day 30: DCA is checking triggers
Skipping DCA - trade already made today
Day 30: Harvester is checking triggers
```

```
Skipping Harvester - trade already made today
Day 30: Dove is checking triggers
Skipping Dove - trade already made today
Day 30: Hawk is checking triggers
Skipping Hawk - trade already made today
Day 30: Sniper1 is checking triggers
Skipping Sniper1 - trade already made today
Day 30: Sniper2 is checking triggers
Skipping Sniper2 - trade already made today
Day 31: Bull is checking triggers
Bull: Updated ETH balance is 0
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
Day 31: Bear is checking triggers
Bear: Cautiously buying TKN with 0.38742049 ETH at price 2285.7699496912696
during a major dip
Bear: Updated TKN balance is 1233.4988213595711
Bear Post-Execution ETH Balance: 3.4867843999999986, TKN Balance:
1233.4988213595711
Day 31: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 31: Harvester is checking triggers
Harvester: Selling 0.0247588 TKN at price 2285.7699496912696
Harvester: Updated TKN balance is 0.09903521000000366
Harvester Post-Execution ETH Balance: 11.57300510528492, TKN Balance:
0.09903521000000366
Day 31: Dove is checking triggers
Dove: Cautiously buying TKN with 0.1634145613775094 ETH at price
2285.7699496912696
Dove Post-Execution ETH Balance: 8.00731350749796, TKN Balance:
217.98090412523203
Day 31: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 31: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2285.7699496912696 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 31: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2285.7699496912696 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 32: Bull is checking triggers
Bull: Updated ETH balance is 0
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
Day 32: Bear is checking triggers
Bear: Cautiously buying TKN with 0.34867844 ETH at price 2353.87451403987 during
a major dip
Bear: Updated TKN balance is 1352.8149496301098
Bear Post-Execution ETH Balance: 3.1381059599999985, TKN Balance:
1352.8149496301098
Day 32: DCA is checking triggers
```

```
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 32: Harvester is checking triggers
Harvester: Selling 0.01980704 TKN at price 2353.87451403987
Harvester: Updated TKN balance is 0.07922817000000366
Harvester Post-Execution ETH Balance: 11.574127546302867, TKN Balance:
0.07922817000000366
Day 32: Dove is checking triggers
Dove Post-Execution ETH Balance: 8.00731350749796, TKN Balance:
217.98090412523203
Day 32: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 32: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2353.87451403987 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 32: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2353.87451403987 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 33: Bull is checking triggers
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
Day 33: Bear is checking triggers
Bear: Cautiously buying TKN with 0.3138106 ETH at price 2354.02052821169 during
a major dip
Bear: Updated TKN balance is 1456.0243634669466
Bear Post-Execution ETH Balance: 2.8242953599999985, TKN Balance:
1456.0243634669466
Day 33: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 33: Harvester is checking triggers
Harvester: Selling 0.01584563 TKN at price 2354.02052821169
Harvester: Updated TKN balance is 0.06338254000000366
Harvester Post-Execution ETH Balance: 11.575129600396274, TKN Balance:
0.06338254000000366
Day 33: Dove is checking triggers
Dove Post-Execution ETH Balance: 8.00731350749796, TKN Balance:
217.98090412523203
Day 33: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 33: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2354.02052821169 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 33: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2354.02052821169 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 34: Bull is checking triggers
Bull: Updated ETH balance is 0
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
Day 34: Bear is checking triggers
Bear: Cautiously buying TKN with 0.28242954 ETH at price 2315.880048563109
```

```
during a major dip
Bear: Updated TKN balance is 1545.827203538334
Bear Post-Execution ETH Balance: 2.5418658199999986, TKN Balance:
1545.827203538334
Day 34: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 34: Harvester is checking triggers
Harvester: Selling 0.01267651 TKN at price 2315.880048563109
Harvester: Updated TKN balance is 0.05070603000000366
Harvester Post-Execution ETH Balance: 11.576025901323312, TKN Balance:
0.05070603000000366
Day 34: Dove is checking triggers
Dove Post-Execution ETH Balance: 8.00731350749796, TKN Balance:
217.98090412523203
Day 34: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 34: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2315.880048563109 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 34: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2315.880048563109 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 35: Bull is checking triggers
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
Day 35: Bear is checking triggers
Bear Post-Execution ETH Balance: 2.5418658199999986, TKN Balance:
1545.827203538334
Day 35: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 35: Harvester is checking triggers
Harvester: Selling 0.01014121 TKN at price 2346.629231763812
Harvester: Updated TKN balance is 0.040564820000003665
Harvester Post-Execution ETH Balance: 11.576060600941947, TKN Balance:
0.040564820000003665
Day 35: Dove is checking triggers
Dove Post-Execution ETH Balance: 8.00731350749796, TKN Balance:
217.98090412523203
Day 35: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 35: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2346.629231763812 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 35: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2346.629231763812 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 36: Bull is checking triggers
Bull: Updated ETH balance is 0
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
```

```
Day 36: Bear is checking triggers
Bear: Cautiously buying TKN with 0.25418658 ETH at price 2302.702281881068
during a major dip
Bear: Updated TKN balance is 1624.266550460923
Bear Post-Execution ETH Balance: 2.287679239999999, TKN Balance:
1624.266550460923
Day 36: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 36: Harvester is checking triggers
Harvester: Selling 0.00811296 TKN at price 2302.702281881068
Harvester: Updated TKN balance is 0.03245186000000366
Harvester Post-Execution ETH Balance: 11.576856079917636, TKN Balance:
0.03245186000000366
Day 36: Dove is checking triggers
Dove Post-Execution ETH Balance: 8.00731350749796, TKN Balance:
217.98090412523203
Day 36: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 36: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2302.702281881068 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 36: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2302.702281881068 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 37: Bull is checking triggers
Bull: Updated ETH balance is 0
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
Day 37: Bear is checking triggers
Bear: Cautiously buying TKN with 0.22876792 ETH at price 2310.8999493062656
during a major dip
Bear: Updated TKN balance is 1693.012718228643
Bear Post-Execution ETH Balance: 2.0589113199999987, TKN Balance:
1693.012718228643
Day 37: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 37: Harvester is checking triggers
Harvester: Selling 0.00649037 TKN at price 2310.8999493062656
Harvester: Updated TKN balance is 0.025961490000003664
Harvester Post-Execution ETH Balance: 11.577568742595103, TKN Balance:
0.025961490000003664
Day 37: Dove is checking triggers
Dove Post-Execution ETH Balance: 8.00731350749796, TKN Balance:
217.98090412523203
Day 37: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 37: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2310.8999493062656 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
```

```
Day 37: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2310.8999493062656 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 38: Bull is checking triggers
Bull: Updated ETH balance is 0
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
Day 38: Bear is checking triggers
Bear: Cautiously buying TKN with 0.20589113 ETH at price 2241.4856060315246
during a major dip
Bear: Updated TKN balance is 1753.4411371002166
Bear Post-Execution ETH Balance: 1.8530201899999987, TKN Balance:
1753.4411371002166
Day 38: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 38: Harvester is checking triggers
Harvester: Selling 0.0051923 TKN at price 2241.4856060315246
Harvester: Updated TKN balance is 0.020769190000003664
Harvester Post-Execution ETH Balance: 11.578207448690637, TKN Balance:
0.020769190000003664
Day 38: Dove is checking triggers
Dove: Cautiously buying TKN with 0.1601462701499592 ETH at price
2241.4856060315246
Dove Post-Execution ETH Balance: 7.847167237348001, TKN Balance:
264.0688994925747
Day 38: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 38: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2241.4856060315246 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 38: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2241.4856060315246 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 39: Bull is checking triggers
Bull: Updated ETH balance is 0
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
Day 39: Bear is checking triggers
Bear: Cautiously buying TKN with 0.18530202 ETH at price 2195.822529570429
during a major dip
Bear: Updated TKN balance is 1805.931272630326
Bear Post-Execution ETH Balance: 1.6677181699999988, TKN Balance:
1805.931272630326
Day 39: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 39: Harvester is checking triggers
Harvester: Selling 0.00415384 TKN at price 2195.822529570429
Harvester: Updated TKN balance is 0.016615350000003665
Harvester Post-Execution ETH Balance: 11.578781793488654, TKN Balance:
0.016615350000003665
```

```
Day 39: Dove is checking triggers
Dove: Cautiously buying TKN with 0.15694334474696 ETH at price 2195.822529570429
Dove Post-Execution ETH Balance: 7.690223892601041, TKN Balance:
307.6166510292652
Day 39: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 39: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2195.822529570429 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 39: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2195.822529570429 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 40: Bull is checking triggers
Bull: Updated ETH balance is 0
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
Day 40: Bear is checking triggers
Bear: Cautiously buying TKN with 0.16677182 ETH at price 2203.2244227980777
during a major dip
Bear: Updated TKN balance is 1851.5571508397077
Bear Post-Execution ETH Balance: 1.5009463499999989, TKN Balance:
1851.5571508397077
Day 40: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 40: Harvester is checking triggers
Harvester: Selling 0.00332307 TKN at price 2203.2244227980777
Harvester: Updated TKN balance is 0.013292280000003664
Harvester Post-Execution ETH Balance: 11.579297154381978, TKN Balance:
0.013292280000003664
Day 40: Dove is checking triggers
Dove: Cautiously buying TKN with 0.1538044778520208 ETH at price
2203.2244227980777
Dove Post-Execution ETH Balance: 7.5364194147490196, TKN Balance:
348.8834347058033
Day 40: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 40: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2203.2244227980777 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 40: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2203.2244227980777 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 41: Bull is checking triggers
Bull: Updated ETH balance is 0
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
Day 41: Bear is checking triggers
Bear: Cautiously buying TKN with 0.15009463 ETH at price 2229.5323141975073
during a major dip
Bear: Updated TKN balance is 1891.3255415110893
```

```
Bear Post-Execution ETH Balance: 1.3508517199999988, TKN Balance:
1891.3255415110893
Day 41: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 41: Harvester is checking triggers
Harvester: Selling 0.00265846 TKN at price 2229.5323141975073
Harvester: Updated TKN balance is 0.010633820000003664
Harvester Post-Execution ETH Balance: 11.579759716158204, TKN Balance:
0.010633820000003664
Day 41: Dove is checking triggers
Dove Post-Execution ETH Balance: 7.5364194147490196, TKN Balance:
348.8834347058033
Day 41: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 41: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2229.5323141975073 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 41: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2229.5323141975073 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 42: Bull is checking triggers
Bull: Updated ETH balance is 0
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
Day 42: Bear is checking triggers
Bear: Cautiously buying TKN with 0.13508517 ETH at price 2236.1525092329434
during a major dip
Bear: Updated TKN balance is 1926.4906521407815
Bear Post-Execution ETH Balance: 1.2157665499999988, TKN Balance:
1926.4906521407815
Day 42: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 42: Harvester is checking triggers
Harvester: Selling 0.00212676 TKN at price 2236.1525092329434
Harvester: Updated TKN balance is 0.008507060000003664
Harvester Post-Execution ETH Balance: 11.580173488986034, TKN Balance:
0.008507060000003664
Day 42: Dove is checking triggers
Dove Post-Execution ETH Balance: 7.5364194147490196, TKN Balance:
348.8834347058033
Day 42: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 42: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2236.1525092329434 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 42: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2236.1525092329434 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 43: Bull is checking triggers
```

```
Bull: Updated ETH balance is 0
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
Day 43: Bear is checking triggers
Bear: Cautiously buying TKN with 0.12157665 ETH at price 2232.706975177926
during a major dip
Bear: Updated TKN balance is 1957.730328897527
Bear Post-Execution ETH Balance: 1.0941898999999988, TKN Balance:
1957.730328897527
Day 43: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 43: Harvester is checking triggers
Harvester: Selling 0.00170141 TKN at price 2232.706975177926
Harvester: Updated TKN balance is 0.006805650000003664
Harvester Post-Execution ETH Balance: 11.580544877854347, TKN Balance:
0.006805650000003664
Day 43: Dove is checking triggers
Dove Post-Execution ETH Balance: 7.5364194147490196, TKN Balance:
348.8834347058033
Day 43: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 43: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2232.706975177926 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 43: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2232.706975177926 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 44: Bull is checking triggers
Bull: Updated ETH balance is 0
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
Day 44: Bear is checking triggers
Bear: Cautiously buying TKN with 0.10941899 ETH at price 2222.7842256731824
during a major dip
Bear: Updated TKN balance is 1985.5208785064583
Bear Post-Execution ETH Balance: 0.9847709099999988, TKN Balance:
1985.5208785064583
Day 44: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 44: Harvester is checking triggers
Harvester: Selling 0.00136113 TKN at price 2222.7842256731824
Harvester: Updated TKN balance is 0.005444520000003663
Harvester Post-Execution ETH Balance: 11.580878309506168, TKN Balance:
0.005444520000003663
Day 44: Dove is checking triggers
Dove Post-Execution ETH Balance: 7.5364194147490196, TKN Balance:
348.8834347058033
Day 44: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 44: Sniper1 is checking triggers
```

```
Sniper1: Selling all TKN at price 2222.7842256731824 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 44: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2222.7842256731824 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 45: Bull is checking triggers
Bull: Updated ETH balance is 0
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
Day 45: Bear is checking triggers
Bear: Cautiously buying TKN with 0.09847709 ETH at price 2172.367800547104
during a major dip
Bear: Updated TKN balance is 2010.2733120673527
Bear Post-Execution ETH Balance: 0.8862938199999988, TKN Balance:
2010.2733120673527
Day 45: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 45: Harvester is checking triggers
Harvester: Selling 0.0010889 TKN at price 2172.367800547104
Harvester: Updated TKN balance is 0.004355620000003663
Harvester Post-Execution ETH Balance: 11.581177733592426, TKN Balance:
0.004355620000003663
Day 45: Dove is checking triggers
Dove: Cautiously buying TKN with 0.1507283882949804 ETH at price
2172.367800547104
Dove Post-Execution ETH Balance: 7.385691026454039, TKN Balance:
386.30069927002836
Day 45: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 45: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2172.367800547104 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 45: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2172.367800547104 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 46: Bull is checking triggers
Bull: Updated ETH balance is 0
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
Day 46: Bear is checking triggers
Bear: Cautiously buying TKN with 0.08862938 ETH at price 2148.539238408839
during a major dip
Bear: Updated TKN balance is 2032.1275149223773
Bear Post-Execution ETH Balance: 0.7976644399999988, TKN Balance:
2032.1275149223773
Day 46: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 46: Harvester is checking triggers
Harvester Post-Execution ETH Balance: 11.581177733592426, TKN Balance:
0.004355620000003663
```

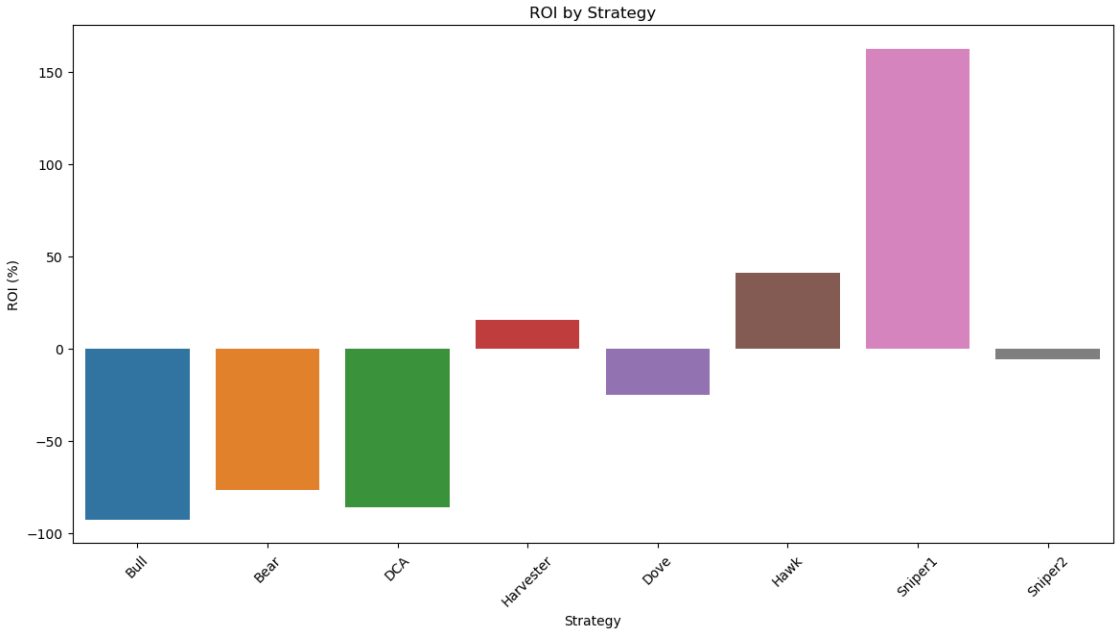
```
Day 46: Dove is checking triggers
Dove: Cautiously buying TKN with 0.1477138205290808 ETH at price
2148.539238408839
Dove Post-Execution ETH Balance: 7.237977205924959, TKN Balance:
422.18116131502796
Day 46: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 46: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2148.539238408839 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 46: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2148.539238408839 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 47: Bull is checking triggers
Bull: Updated ETH balance is 0
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
Day 47: Bear is checking triggers
Bear: Cautiously buying TKN with 0.07976644 ETH at price 2133.6388006707934
during a major dip
Bear: Updated TKN balance is 2051.3611135547385
Bear Post-Execution ETH Balance: 0.7178979999999988, TKN Balance:
2051.3611135547385
Day 47: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 47: Harvester is checking triggers
Harvester Post-Execution ETH Balance: 11.581177733592426, TKN Balance:
0.004355620000003663
Day 47: Dove is checking triggers
Dove: Cautiously buying TKN with 0.14475954411849917 ETH at price
2133.6388006707934
Dove Post-Execution ETH Balance: 7.0932176618064595, TKN Balance:
456.5691982207295
Day 47: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 47: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2133.6388006707934 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 47: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2133.6388006707934 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 48: Bull is checking triggers
Bull: Updated ETH balance is 0
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
Day 48: Bear is checking triggers
Bear: Cautiously buying TKN with 0.0717898 ETH at price 2169.946566970787 during
a major dip
Bear: Updated TKN balance is 2068.311877588073
Bear Post-Execution ETH Balance: 0.6461081999999988, TKN Balance:
```

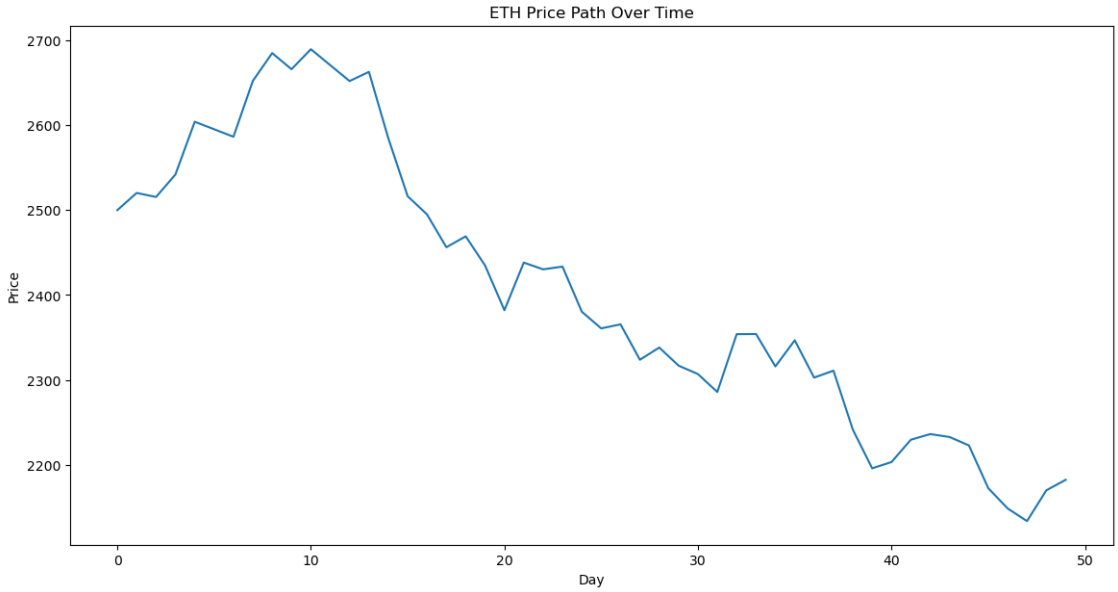
```
2068.311877588073
Day 48: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 48: Harvester is checking triggers
Harvester Post-Execution ETH Balance: 11.581177733592426, TKN Balance:
0.004355620000003663
Day 48: Dove is checking triggers
Dove Post-Execution ETH Balance: 7.0932176618064595, TKN Balance:
456.5691982207295
Day 48: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 48: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2169.946566970787 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 48: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2169.946566970787 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Day 49: Bull is checking triggers
Bull: Updated ETH balance is 0
Bull Post-Execution ETH Balance: 0, TKN Balance: 881.3563387908404
Day 49: Bear is checking triggers
Bear Post-Execution ETH Balance: 0.6461081999999988, TKN Balance:
2068.311877588073
Day 49: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 49: Harvester is checking triggers
Harvester Post-Execution ETH Balance: 11.581177733592426, TKN Balance:
0.004355620000003663
Day 49: Dove is checking triggers
Dove Post-Execution ETH Balance: 7.0932176618064595, TKN Balance:
456.5691982207295
Day 49: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 14.071191303556972, TKN Balance: 0.0
Day 49: Sniper1 is checking triggers
Sniper1: Selling all TKN at price 2182.284485698945 due to stop-loss
Sniper1 Post-Execution ETH Balance: 26.233791875519803, TKN Balance: 0.0
Day 49: Sniper2 is checking triggers
Sniper2: Selling all TKN at price 2182.284485698945 due to stop-loss
Sniper2 Post-Execution ETH Balance: 9.39988127362825, TKN Balance: 0.0
Strategy: Bull
  Final ETH Balance: 0
  Final TKN Balance: 881.3563387908404
  Final USD Balance: 1762.7126775816807
  ROI: -92.94914928967329
Strategy: Bear
  Final ETH Balance: 0.6461081999999988
  Final TKN Balance: 2068.311877588073
  Final USD Balance: 5751.894255176143
```

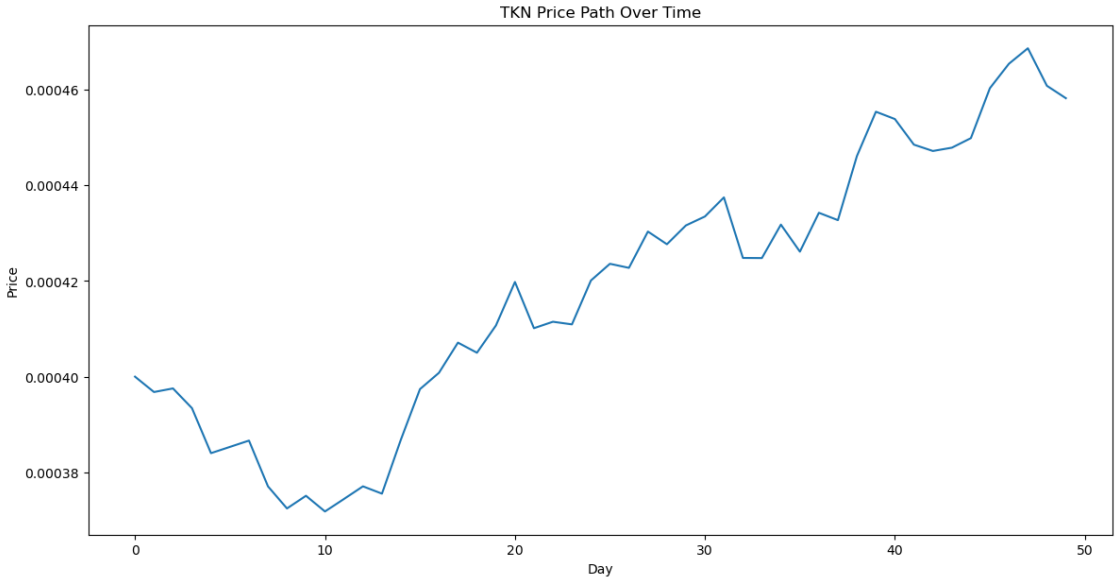
```
  ROI: -76.99242297929543
Strategy: DCA
  Final ETH Balance: 0.0
  Final TKN Balance: 1721.1611838303008
  Final USD Balance: 3442.3223676606017
  ROI: -86.2307105293576
Strategy: Harvester
  Final ETH Balance: 11.581177733592426
  Final TKN Balance: 0.004355620000003663
  Final USD Balance: 28952.953045221064
  ROI: 15.350410538729337
Strategy: Dove
  Final ETH Balance: 7.0932176618064595
  Final TKN Balance: 456.5691982207295
  Final USD Balance: 18646.18255095761
  ROI: -25.415269796169564
Strategy: Hawk
  Final ETH Balance: 14.071191303556972
  Final TKN Balance: 0.0
  Final USD Balance: 35177.97825889243
  ROI: 40.711913035569715
Strategy: Sniper1
  Final ETH Balance: 26.233791875519803
  Final TKN Balance: 0.0
  Final USD Balance: 65584.4796887995
  ROI: 162.33791875519802
Strategy: Sniper2
  Final ETH Balance: 9.39988127362825
  Final TKN Balance: 0.0
  Final USD Balance: 23499.703184070622
  ROI: -6.001187263717512
Final Balances DataFrame:
     Strategy  Final_ETH_Balance  Final_TKN_Balance  Final_USD_Balance  \
0       Bull           0.000000         881.356339        1762.712678
1       Bear           0.646108        2068.311878        5751.894255
2        DCA           0.000000        1721.161184        3442.322368
3  Harvester          11.581178           0.004356       28952.953045
4       Dove           7.093218         456.569198       18646.182551
5       Hawk          14.071191           0.000000       35177.978259
6    Sniper1          26.233792           0.000000       65584.479689
7    Sniper2           9.399881           0.000000       23499.703184

          ROI  Sharpe_Ratio
0  -92.949149      0.192201
1  -76.992423     -0.197348
2  -86.230711      0.192201
3   15.350411     -0.197342
4  -25.415270     -0.197342
```

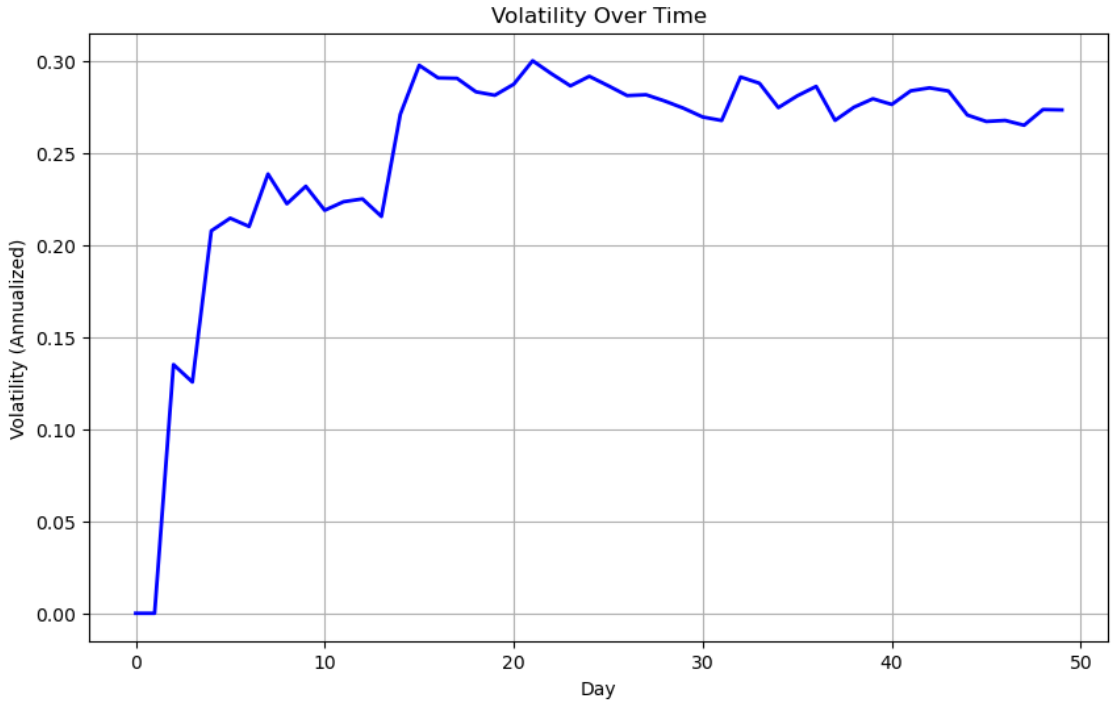
```
5   40.711913     -0.197342
6  162.337919     -0.197342
7   -6.001187     -0.197342
Price History Table
   Day        Price
0    0  2500.000000
1    1  2520.266038
2    2  2515.489241
3    3  2541.899806
4    4  2604.137268

Volatility Tracking Table
   Day  Volatility
0    0    0.000000
1    1    0.000000
2    2    0.135116
3    3    0.125616
4    4    0.207762

Mean Reversion Table
   Day        Price  Deviation_from_Mean
0    0  2500.000000          -500.000000
1    1  2520.266038          -479.733962
2    2  2515.489241          -484.510759
3    3  2541.899806          -458.100194
4    4  2604.137268          -395.862732

Final Balances Table
    Strategy  Final_ETH_Balance  Final_TKN_Balance  Final_USD_Balance  \
0       Bull           0.000000         881.356339        1762.712678
1       Bear           0.646108        2068.311878        5751.894255
2        DCA           0.000000        1721.161184        3442.322368
3  Harvester          11.581178           0.004356       28952.953045
4       Dove           7.093218         456.569198       18646.182551

         ROI  Sharpe_Ratio
0 -92.949149      0.192201
1 -76.992423     -0.197348
2 -86.230711      0.192201
3  15.350411     -0.197342
4 -25.415270     -0.197342

Total Volume Swapped Table
           ETH Swapped  TKN Swapped
Strategy
Bull          9.999390   881.356339
Bear          9.353892  2068.311878
DCA          10.000000  1721.161184
```

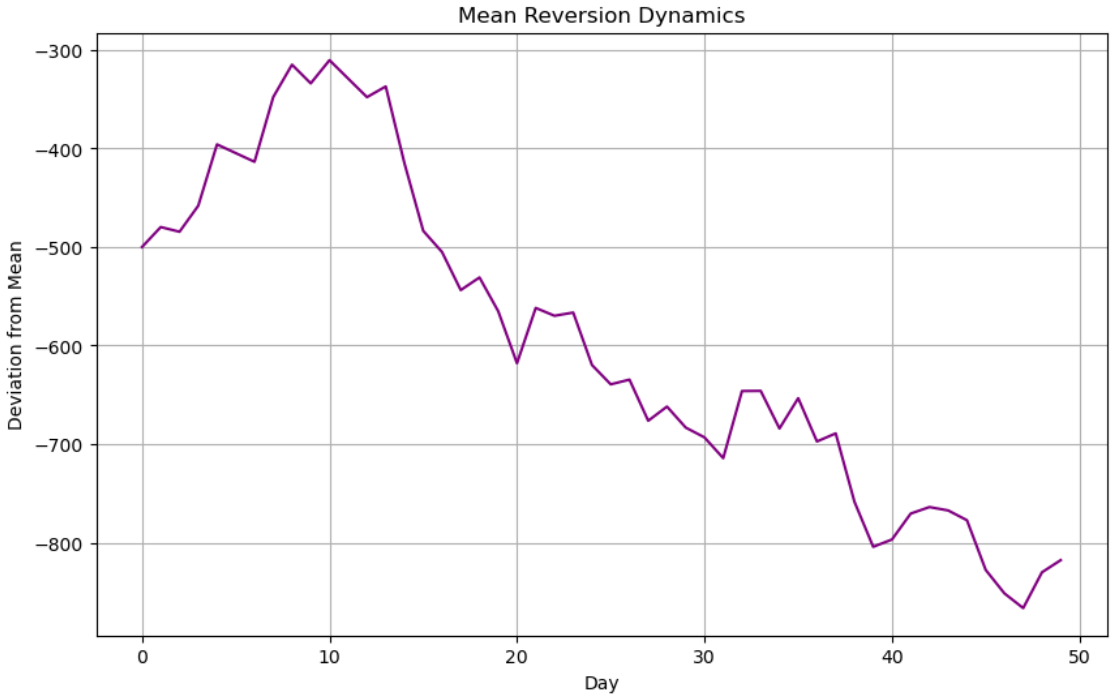
```
Harvester     1.581178    99.995644
Dove          2.906782   456.569198

ROI per Strategy Table
    Strategy        ROI
0       Bull -92.949149
1       Bear -76.992423
2        DCA -86.230711
3  Harvester  15.350411
4       Dove -25.415270

Impermanent Loss Table
   Day  Impermanent_Loss
0    0         99.999842
1    1         99.999844
2    2         99.999805
3    3         99.999864
4    4         99.999867

Total LP Fees Table
     ETH_Fees  TKN_Fees
Day
0    0.224120  0.061194
1    0.224120  0.110206
2    0.279692  0.149250
3    0.279692  6.171150
4    0.279692  6.195958

C:\Users\merca\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1765:
FutureWarning: unique with argument that is not not a Series, Index,
ExtensionArray, or np.ndarray is deprecated and will raise in a future version.
  order = pd.unique(vector)
```

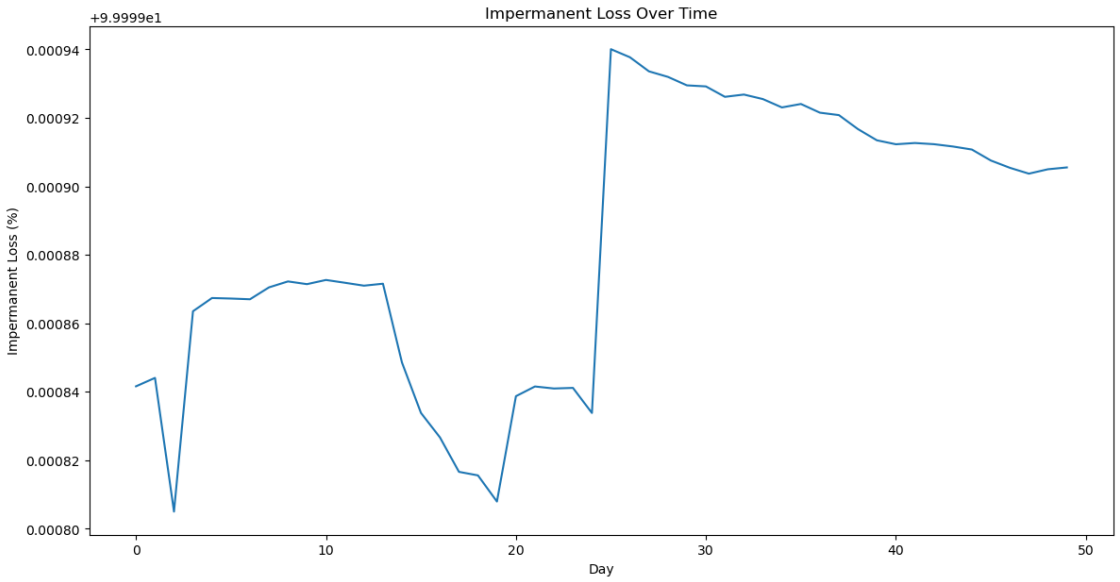

ROI by Strategy
ROI (%)
150
100
50
0
−50
−100
Bull
Bear
DCA
Harvester
Dove
Hawk
Sniper1
Sniper2
Strategy

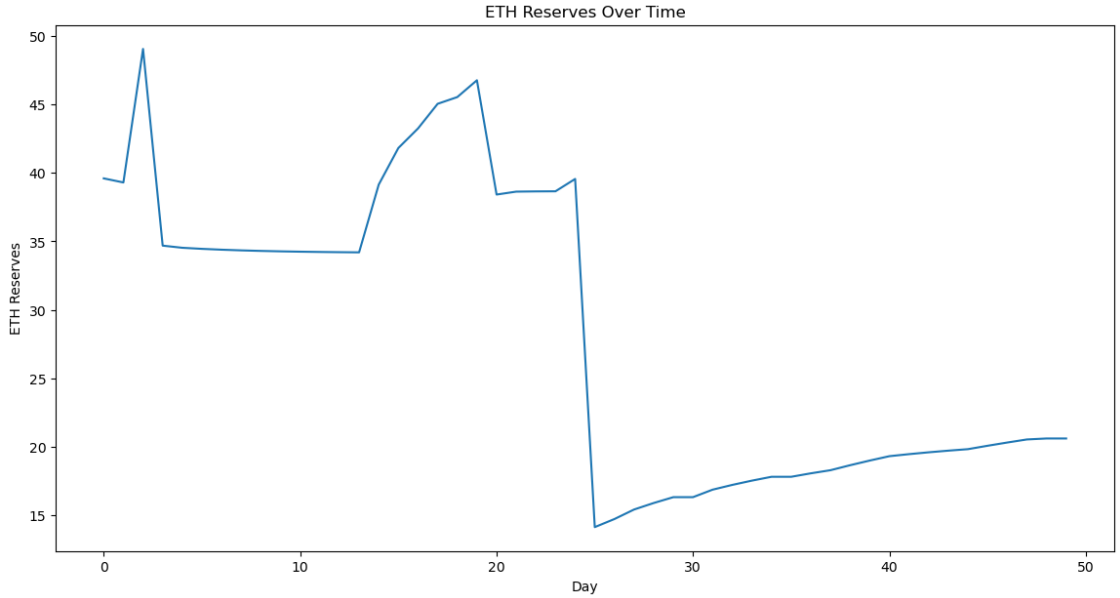

ETH Price Path Over Time
Price
2700
2600
2500
2400
2300
2200
0
10
20
30
40
50
Day

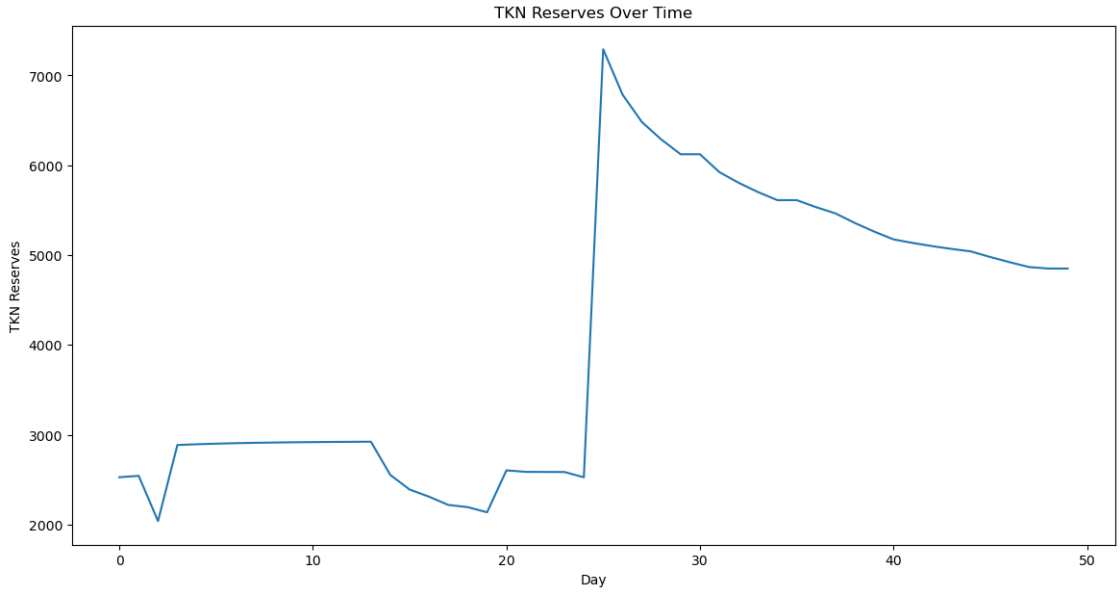

TKN Price Path Over Time
Price
0.00046
0.00044
0.00042
0.00040
0.00038
0
10
20
30
40
50
Day


Volatility Over Time
Volatility (Annualized)
0.30
0.25
0.20
0.15
0.10
0.05
0.00
0
10
20
30
40
50
Day

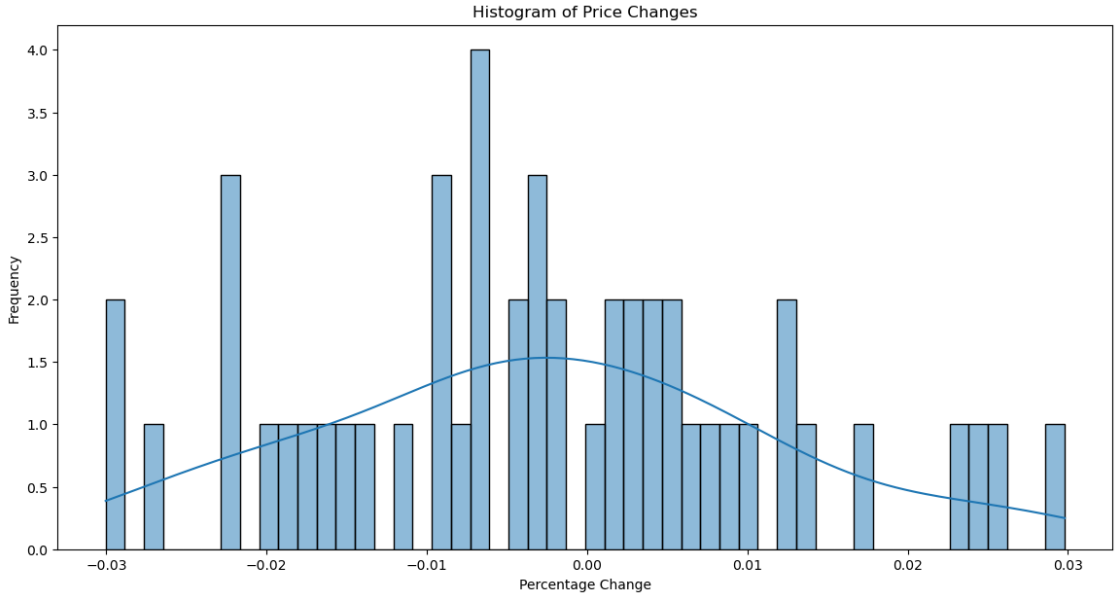

Mean Reversion Dynamics
Deviation from Mean
−300
−400
−500
−600
−700
−800
0
10
20
30
40
50
Day


Impermanent Loss Over Time
+9.9999e1
Impermanent Loss (%)
0.00094
0.00092
0.00090
0.00088
0.00086
0.00084
0.00082
0.00080
0
10
20
30
40
50
Day

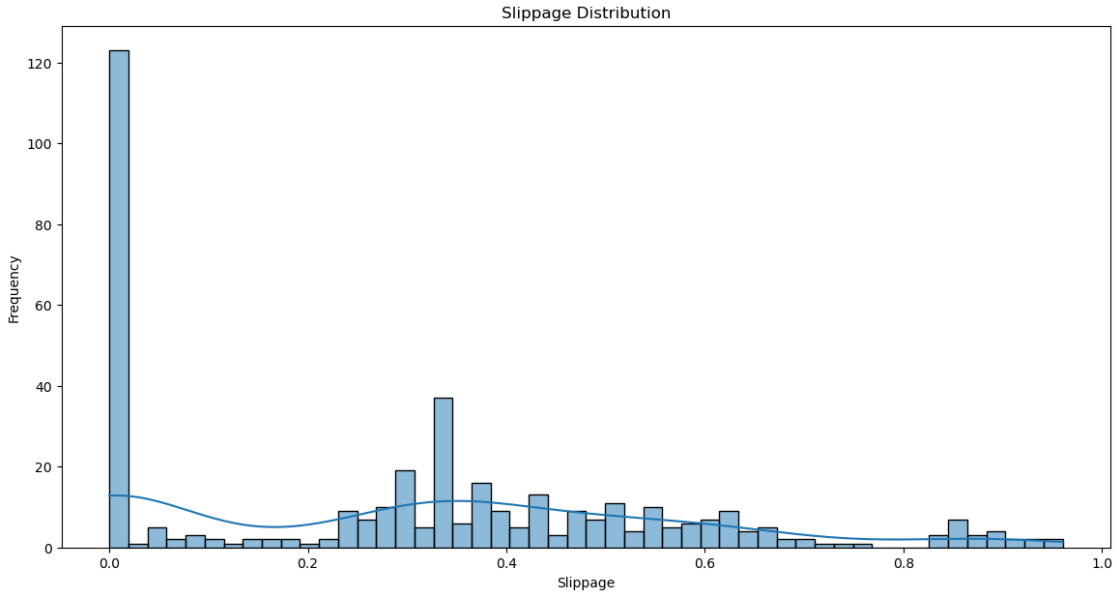





```
C:\Users\merca\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

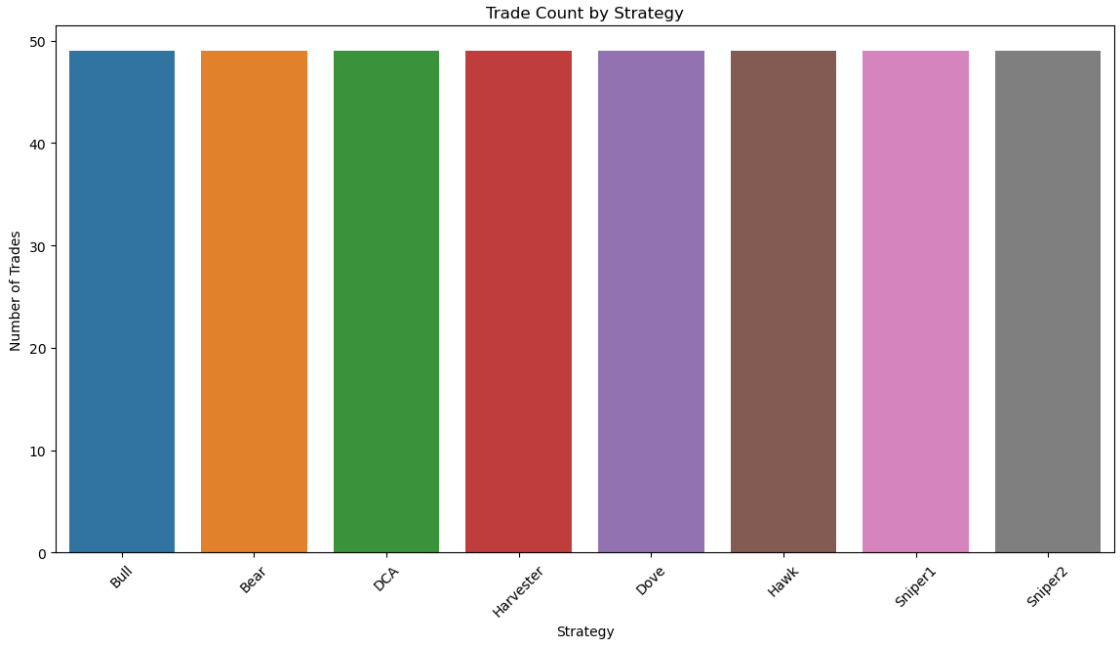


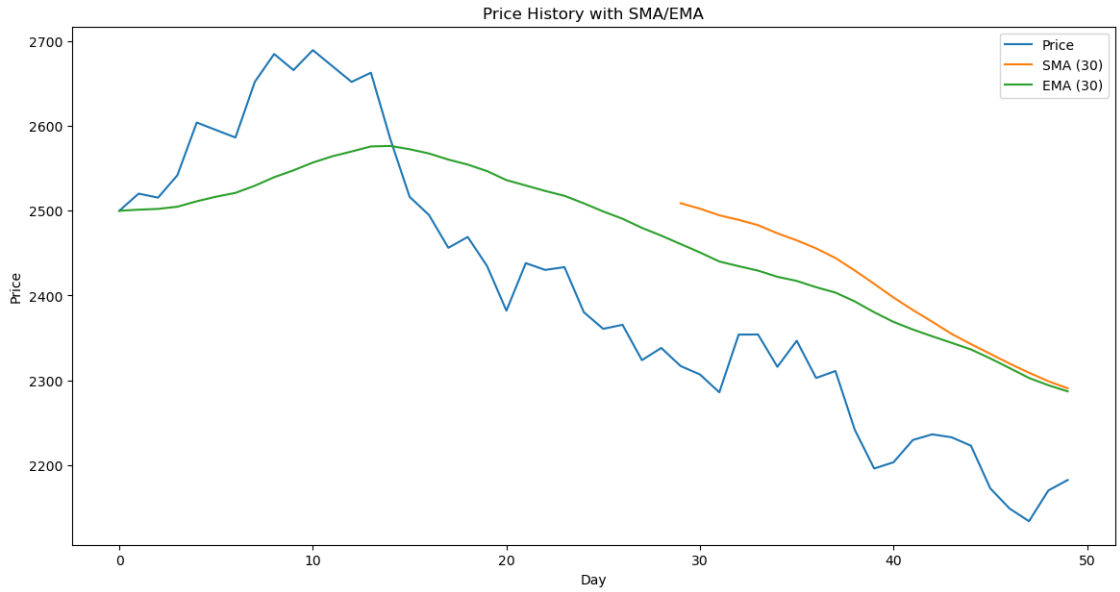

```
C:\Users\merca\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

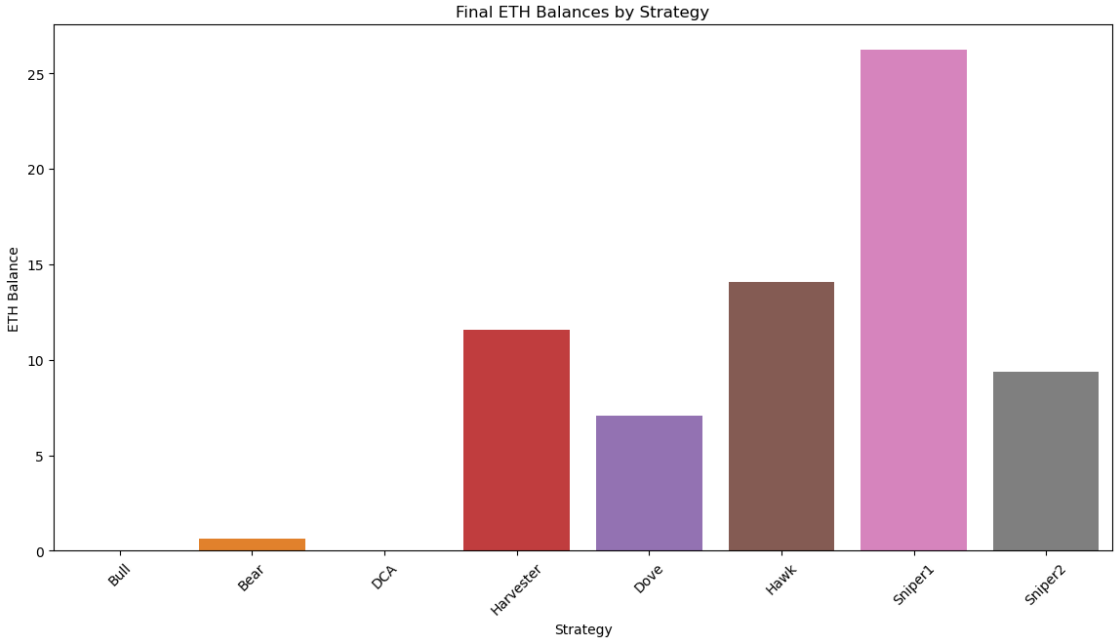


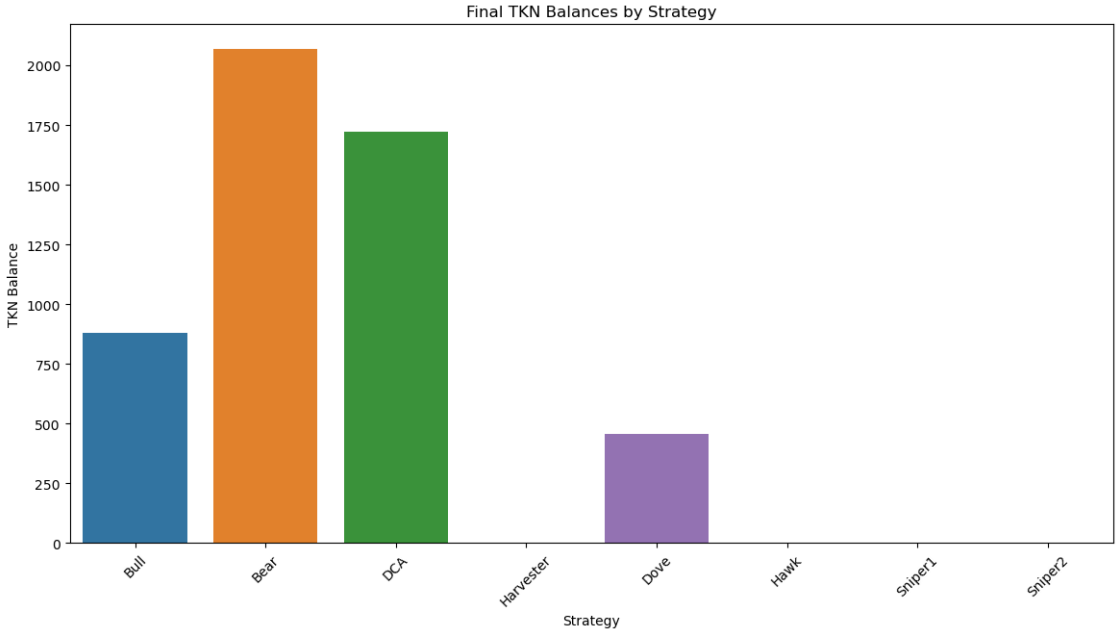

```
C:\Users\merca\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1765:
FutureWarning: unique with argument that is not not a Series, Index,
ExtensionArray, or np.ndarray is deprecated and will raise in a future version.
```

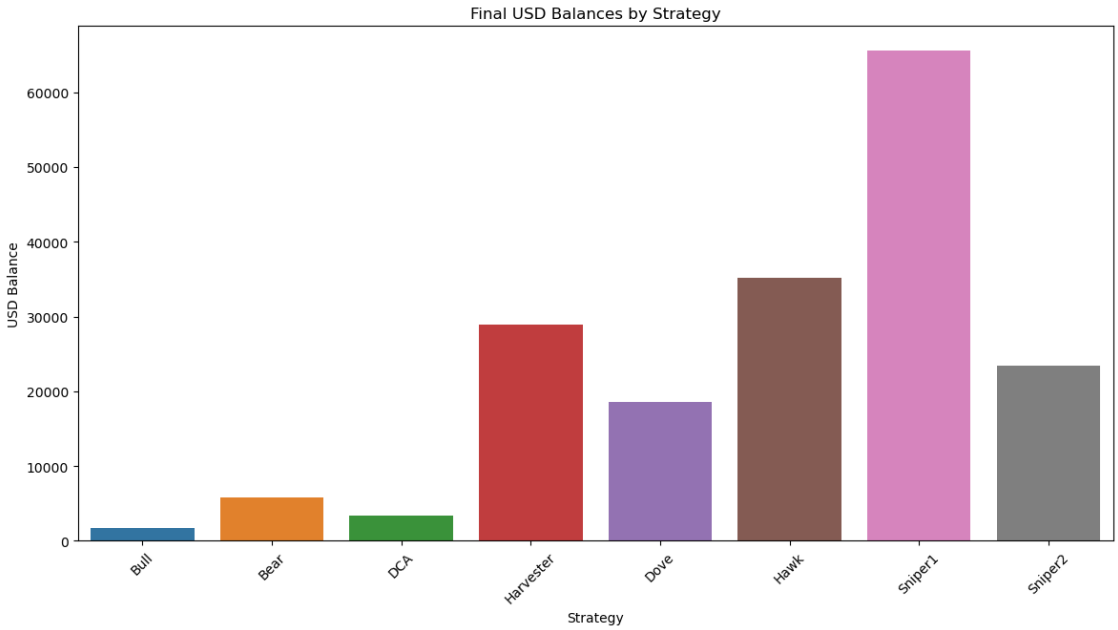
```
order = pd.unique(vector)
```

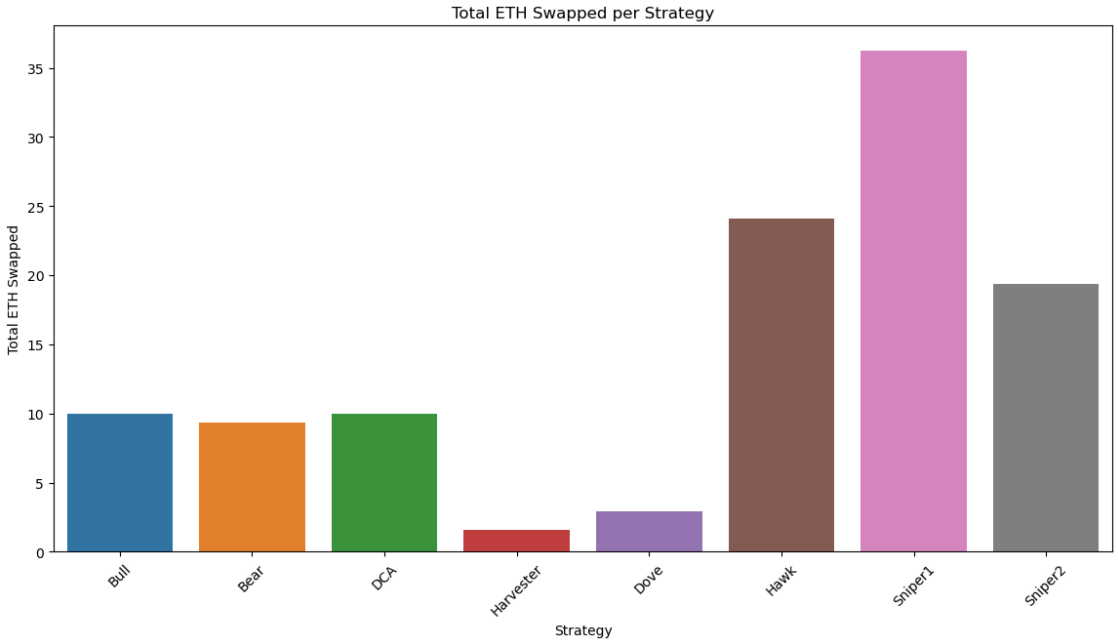


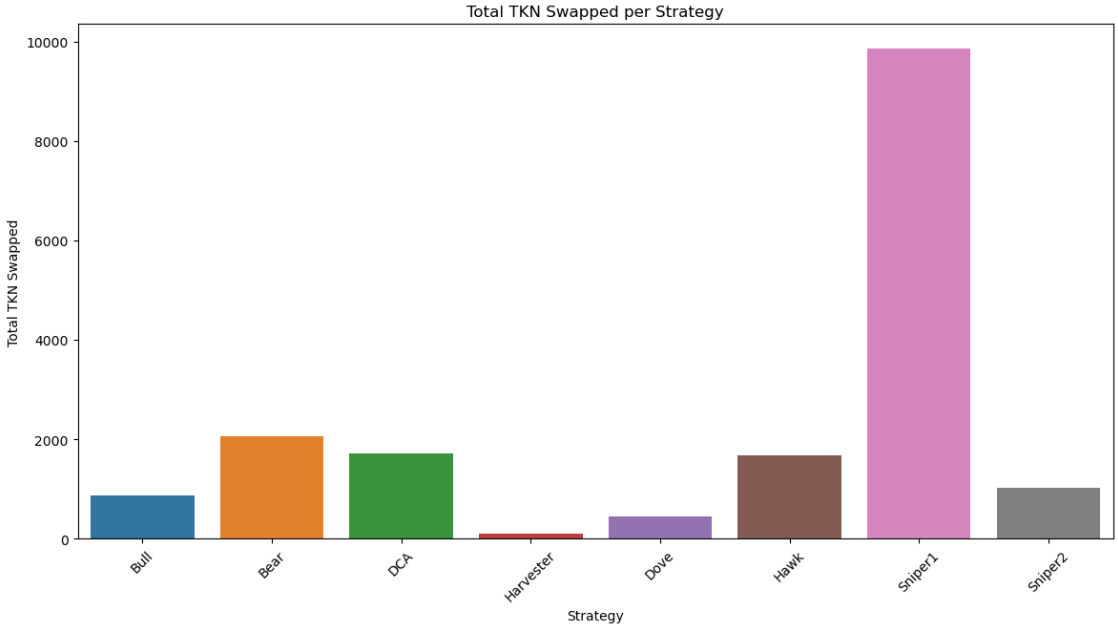


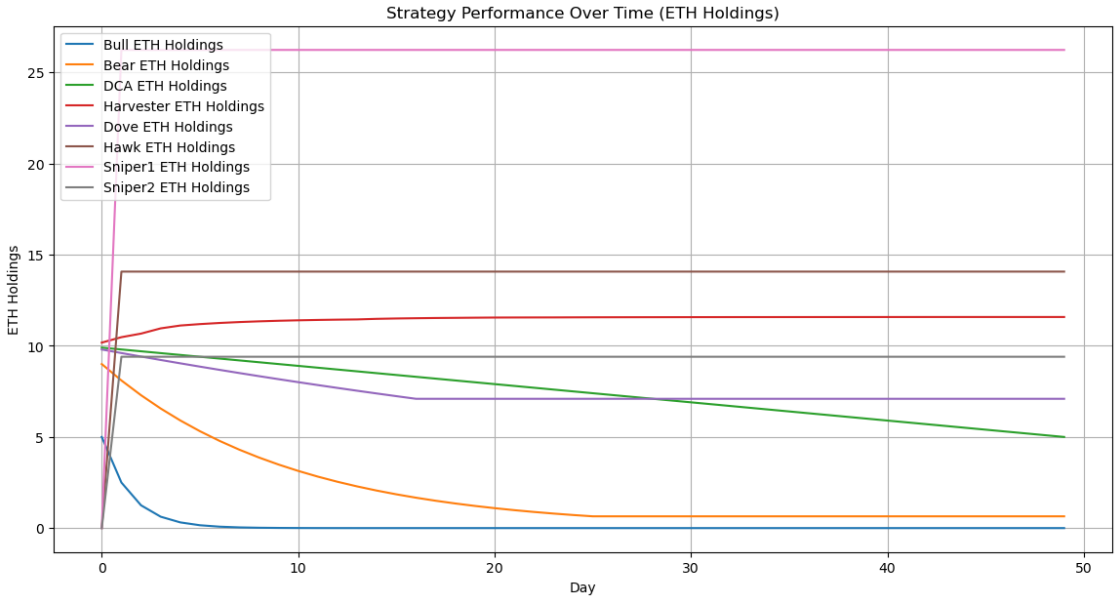

Final ETH Balances by Strategy
ETH Balance
0
5
10
15
20
25
Bull
Bear
DCA
Harvester
Dove
Hawk
Sniper1
Sniper2
Strategy

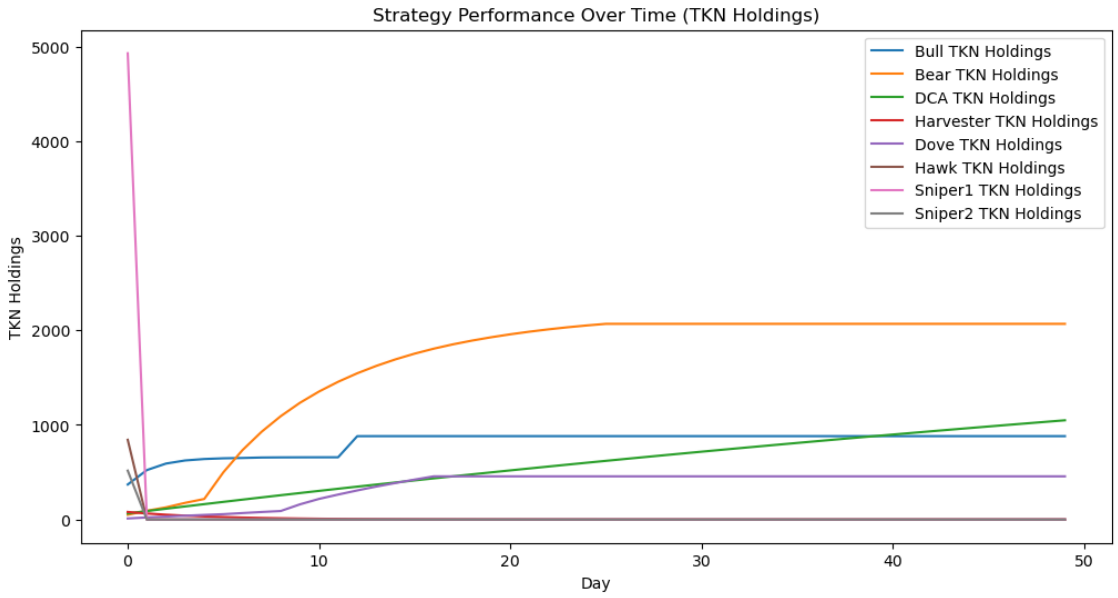

Final TKN Balances by Strategy
TKN Balance
0
250
500
750
1000
1250
1500
1750
2000
Bull
Bear
DCA
Harvester
Dove
Hawk
Sniper1
Sniper2
Strategy


Final USD Balances by Strategy
USD Balance
0
10000
20000
30000
40000
50000
60000
Bull
Bear
DCA
Harvester
Dove
Hawk
Sniper1
Sniper2
Strategy


Total ETH Swapped per Strategy
Total ETH Swapped
0
5
10
15
20
25
30
35
Bull
Bear
DCA
Harvester
Dove
Hawk
Sniper1
Sniper2
Strategy


Total TKN Swapped per Strategy
Total TKN Swapped
Strategy
Bull
Bear
DCA
Harvester
Dove
Hawk
Sniper1
Sniper2
0
2000
4000
6000
8000
10000


Strategy Performance Over Time (ETH Holdings)
Bull ETH Holdings
Bear ETH Holdings
DCA ETH Holdings
Harvester ETH Holdings
Dove ETH Holdings
Hawk ETH Holdings
Sniper1 ETH Holdings
Sniper2 ETH Holdings
ETH Holdings
Day
0
5
10
15
20
25
0
10
20
30
40
50



[1590]:
```
def run_simulation_and_prepare_data():
    market_conditions = load_json('bull_market.json')
    days = market_conditions['simulation_parameters']['days_of_simulation']
    initial_eth = market_conditions['simulation_parameters']['initial_pool_eth']
    initial_tkn = market_conditions['simulation_parameters']['initial_pool_tkn']
    initial_eth_price =␣
↪market_conditions['market_conditions']['initial_eth_price']
    initial_tkn_price = 1  # Assuming initial TKN price is set to 1 as it's not␣
↪in the JSON

    pool = TradingPool(initial_eth, initial_tkn)

    strategies = [
        BullStrategy("Bull", 10, 0, initial_eth_price, initial_tkn_price),
        BearStrategy("Bear", 10, 0, initial_eth_price, initial_tkn_price),
        DcaStrategy("DCA", 10, 0, initial_eth_price, initial_tkn_price,␣
↪daily_investment=0.1),
        HarvesterStrategy("Harvester", 10, 100, initial_eth_price,␣
↪initial_tkn_price),
        DoveStrategy("Dove", 10, 0, initial_eth_price, initial_tkn_price),
        HawkStrategy("Hawk", 10, 0, initial_eth_price, initial_tkn_price,␣
↪target_profit=0.05),
        Sniper1Strategy("Sniper1", 10, 0, initial_eth_price, initial_tkn_price,␣
↪target_profit=0.1),
        Sniper2Strategy("Sniper2", 10, 0, initial_eth_price, initial_tkn_price,␣
↪delay=1, target_profit=0.1)
```

```
    ]

    # Initialize trade summary dictionary
    trade_summary = {}

    # Run the simulation
    eth_price_history, tkn_price_history, trade_counts, eth_reserves_over_time,␣
↪tkn_reserves_over_time, \
    lp_fees_over_time, impermanent_losses, price_impacts, detailed_logs =␣
↪run_simulation(strategies, pool, market_conditions, days)

    # Evaluate the results
    results = evaluate_simulation(strategies)

    # Update trade summary with actual volumes
    for strategy in strategies:
        update_total_volume_swapped(trade_summary, strategy.name, strategy.
↪participant)

    # Generate a summary of trades made by each strategy
    trade_summary = summarize_trades(strategies)

    # Create various tables to summarize the simulation results
    price_history_table = create_price_history_table(eth_price_history)
    volatility_table = create_volatility_tracking_table(eth_price_history)
    mean_reversion_table = create_mean_reversion_table(eth_price_history,␣
↪mean_reversion_level=3000)
    final_balances_table = create_corrected_final_balances_table(strategies,␣
↪eth_price_history)
    total_volume_swapped_table = create_total_volume_swapped_table(trade_summary)
    roi_per_strategy_table = create_roi_per_strategy_table(results)
    impermanent_loss_table = create_impermanent_loss_table(impermanent_losses)
    total_lp_fees_table = create_total_lp_fees_table(lp_fees_over_time)

    return {
        "price_history_table": price_history_table,
        "volatility_table": volatility_table,
        "mean_reversion_table": mean_reversion_table,
        "final_balances_table": final_balances_table,
        "total_volume_swapped_table": total_volume_swapped_table,
        "roi_per_strategy_table": roi_per_strategy_table,
        "impermanent_loss_table": impermanent_loss_table,
        "total_lp_fees_table": total_lp_fees_table,
        "eth_price_history": eth_price_history,  # Note this change from␣
↪'price_history' to 'eth_price_history'
        "tkn_price_history": tkn_price_history,
```

```
        "results": results,
        "eth_reserves_over_time": eth_reserves_over_time,
        "tkn_reserves_over_time": tkn_reserves_over_time,
        "impermanent_losses": impermanent_losses,
        "price_impacts": price_impacts,
        "trade_counts": trade_counts,
        "strategies": strategies,
    }

# Run the simulation and store the results
simulation_data = run_simulation_and_prepare_data()
```

```
Day 0: Sniper1 is executing forced trade
Sniper1: Purchased 2000.0 TKN at price 2500
Sniper1 ETH Balance: 0, TKN Balance: 4927.586379569353
Day 0: Sniper2 is executing forced trade
Sniper2 ETH Balance: 10, TKN Balance: 0
Day 0: Bull is checking triggers
Bull Post-Execution ETH Balance: 10, TKN Balance: 0
Day 0: Bear is checking triggers
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 0: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 0: Harvester is checking triggers
Harvester: Selling 20.0 TKN at price 2500
Harvester: Updated TKN balance is 80.0
Harvester Post-Execution ETH Balance: 10.175956717135797, TKN Balance: 80.0
Day 0: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 0: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 1: Bull is checking triggers
Bull Post-Execution ETH Balance: 10, TKN Balance: 0
Day 1: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 1: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 1: Harvester is checking triggers
Harvester: Selling 16.0 TKN at price 2513.4434453823783
Harvester: Updated TKN balance is 64.0
Harvester Post-Execution ETH Balance: 10.473967754962954, TKN Balance: 64.0
Day 1: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 1: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 1: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
```

```
Day 1: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 10, TKN Balance: 0
Day 2: Bull is checking triggers
Bull: Actively buying TKN with 5.0 ETH at price 2498.9754748594537
Bull: Updated ETH balance is 5.0
Bull Post-Execution ETH Balance: 5.0, TKN Balance: 285.2364766351671
Day 2: Bear is checking triggers
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 2: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 2: Harvester is checking triggers
Harvester: Selling 12.8 TKN at price 2498.9754748594537
Harvester: Updated TKN balance is 51.2
Harvester Post-Execution ETH Balance: 10.739404677702224, TKN Balance: 51.2
Day 2: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 2: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 2: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 2: Sniper2 is checking triggers
Sniper2: Purchased 454.3976255462886 TKN at price 2498.9754748594537
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 3: Bull is checking triggers
Bull: Actively buying TKN with 2.5 ETH at price 2473.230758545399
Bull: Updated ETH balance is 2.5
Bull Post-Execution ETH Balance: 2.5, TKN Balance: 368.1972895897953
Day 3: Bear is checking triggers
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 3: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 3: Harvester is checking triggers
Harvester: Selling 10.24 TKN at price 2473.230758545399
Harvester: Updated TKN balance is 40.96
Harvester Post-Execution ETH Balance: 11.06983648230541, TKN Balance: 40.96
Day 3: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 3: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 3: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 3: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 4: Bull is checking triggers
Bull Post-Execution ETH Balance: 2.5, TKN Balance: 368.1972895897953
Day 4: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
```

```
Day 4: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 4: Harvester is checking triggers
Harvester: Selling 8.192 TKN at price 2497.784772445161
Harvester: Updated TKN balance is 32.768
Harvester Post-Execution ETH Balance: 11.326124979773422, TKN Balance: 32.768
Day 4: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 4: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 4: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 4: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 5: Bull is checking triggers
Bull Post-Execution ETH Balance: 2.5, TKN Balance: 368.1972895897953
Day 5: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 5: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 5: Harvester is checking triggers
Harvester: Selling 6.5536 TKN at price 2539.2473420017877
Harvester: Updated TKN balance is 26.2144
Harvester Post-Execution ETH Balance: 11.529461924737298, TKN Balance: 26.2144
Day 5: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 5: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 5: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 5: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 6: Bull is checking triggers
Bull: Selling 184.09864479 TKN at price 2577.3592196476784 to realize profits
Bull: Updated ETH balance is 7.6396371067496105
Bull Post-Execution ETH Balance: 7.6396371067496105, TKN Balance:
184.09864479979527
Day 6: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 6: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 6: Harvester is checking triggers
Harvester: Selling 5.24288 TKN at price 2577.3592196476784
Harvester: Updated TKN balance is 20.97152
Harvester Post-Execution ETH Balance: 11.677965191377082, TKN Balance: 20.97152
Day 6: Dove is checking triggers
```

```
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 6: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 6: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 6: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 7: Bull is checking triggers
Bull Post-Execution ETH Balance: 7.6396371067496105, TKN Balance:
184.09864479979527
Day 7: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 7: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 7: Harvester is checking triggers
Harvester: Selling 4.194304 TKN at price 2544.3145793278
Harvester: Updated TKN balance is 16.777216000000003
Harvester Post-Execution ETH Balance: 11.783985571197244, TKN Balance:
16.777216000000003
Day 7: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 7: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 7: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 7: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 8: Bull is checking triggers
Bull Post-Execution ETH Balance: 7.6396371067496105, TKN Balance:
184.09864479979527
Day 8: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 8: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 8: Harvester is checking triggers
Harvester: Selling 3.3554432 TKN at price 2532.684458951317
Harvester: Updated TKN balance is 13.421772800000003
Harvester Post-Execution ETH Balance: 11.868444523812753, TKN Balance:
13.421772800000003
Day 8: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 8: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 8: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 8: Sniper2 is checking triggers
```

```
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 9: Bull is checking triggers
Bull Post-Execution ETH Balance: 7.6396371067496105, TKN Balance:
184.09864479979527
Day 9: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 9: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 9: Harvester is checking triggers
Harvester: Selling 2.68435456 TKN at price 2546.5907400124656
Harvester: Updated TKN balance is 10.737418240000004
Harvester Post-Execution ETH Balance: 11.935807699627624, TKN Balance:
10.737418240000004
Day 9: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 9: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 9: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 9: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 10: Bull is checking triggers
Bull: Selling 92.0493224 TKN at price 2586.610114276139 to realize profits
Bull: Updated ETH balance is 9.835975129324478
Bull Post-Execution ETH Balance: 9.835975129324478, TKN Balance:
92.04932239979527
Day 10: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 10: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 10: Harvester is checking triggers
Harvester: Selling 2.14748365 TKN at price 2586.610114276139
Harvester: Updated TKN balance is 8.589934590000004
Harvester Post-Execution ETH Balance: 11.991397660518723, TKN Balance:
8.589934590000004
Day 10: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 10: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 10: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 10: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 11: Bull is checking triggers
Bull Post-Execution ETH Balance: 9.835975129324478, TKN Balance:
92.04932239979527
```

```
Day 11: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 11: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 11: Harvester is checking triggers
Harvester: Selling 1.71798692 TKN at price 2567.9240685729974
Harvester: Updated TKN balance is 6.871947670000004
Harvester Post-Execution ETH Balance: 12.030657381849457, TKN Balance:
6.871947670000004
Day 11: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 11: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 11: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 11: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 12: Bull is checking triggers
Bull Post-Execution ETH Balance: 9.835975129324478, TKN Balance:
92.04932239979527
Day 12: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 12: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 12: Harvester is checking triggers
Harvester: Selling 1.37438953 TKN at price 2561.150160872978
Harvester: Updated TKN balance is 5.497558140000004
Harvester Post-Execution ETH Balance: 12.062003436057088, TKN Balance:
5.497558140000004
Day 12: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 12: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 12: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 12: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 13: Bull is checking triggers
Bull: Actively buying TKN with 4.91798756 ETH at price 2517.730614629694
Bull: Updated ETH balance is 4.9179875693244774
Bull Post-Execution ETH Balance: 4.9179875693244774, TKN Balance:
285.94010214166923
Day 13: Bear is checking triggers
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 13: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
```

```
Day 13: Harvester is checking triggers
Harvester: Selling 1.09951163 TKN at price 2517.730614629694
Harvester: Updated TKN balance is 4.398046510000004
Harvester Post-Execution ETH Balance: 12.108974898614933, TKN Balance:
4.398046510000004
Day 13: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 13: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 13: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 13: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 14: Bull is checking triggers
Bull: Actively buying TKN with 2.45899378 ETH at price 2471.5567709338447
Bull: Updated ETH balance is 2.4589937893244773
Bull Post-Execution ETH Balance: 2.4589937893244773, TKN Balance:
370.1700222165385
Day 14: Bear is checking triggers
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 14: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 14: Harvester is checking triggers
Harvester: Selling 0.8796093 TKN at price 2471.5567709338447
Harvester: Updated TKN balance is 3.5184372100000036
Harvester Post-Execution ETH Balance: 12.143219530184435, TKN Balance:
3.5184372100000036
Day 14: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 14: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 14: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 14: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 15: Bull is checking triggers
Bull: Actively buying TKN with 1.22949689 ETH at price 2503.978983252638
Bull: Updated ETH balance is 1.2294968993244773
Bull Post-Execution ETH Balance: 1.2294968993244773, TKN Balance:
409.59598526046716
Day 15: Bear is checking triggers
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 15: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 15: Harvester is checking triggers
Harvester: Selling 0.70368744 TKN at price 2503.978983252638
Harvester: Updated TKN balance is 2.8147497700000037
Harvester Post-Execution ETH Balance: 12.169122770697243, TKN Balance:
```

```
2.8147497700000037
Day 15: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 15: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 15: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 15: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 16: Bull is checking triggers
Bull Post-Execution ETH Balance: 1.2294968993244773, TKN Balance:
409.59598526046716
Day 16: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 16: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 16: Harvester is checking triggers
Harvester: Selling 0.56294995 TKN at price 2558.578131500563
Harvester: Updated TKN balance is 2.2517998200000036
Harvester Post-Execution ETH Balance: 12.186907866974662, TKN Balance:
2.2517998200000036
Day 16: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 16: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 16: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 16: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 17: Bull is checking triggers
Bull Post-Execution ETH Balance: 1.2294968993244773, TKN Balance:
409.59598526046716
Day 17: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 17: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 17: Harvester is checking triggers
Harvester: Selling 0.45035996 TKN at price 2556.386926622091
Harvester: Updated TKN balance is 1.8014398600000034
Harvester Post-Execution ETH Balance: 12.20111908159869, TKN Balance:
1.8014398600000034
Day 17: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 17: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 17: Sniper1 is checking triggers
```

```
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 17: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 18: Bull is checking triggers
Bull: Selling 204.79799263 TKN at price 2597.7016444486167 to realize profits
Bull: Updated ETH balance is 6.996554521043906
Bull Post-Execution ETH Balance: 6.996554521043906, TKN Balance:
204.79799263046715
Day 18: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 18: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 18: Harvester is checking triggers
Harvester: Selling 0.36028797 TKN at price 2597.7016444486167
Harvester: Updated TKN balance is 1.4411518900000035
Harvester Post-Execution ETH Balance: 12.228218067568317, TKN Balance:
1.4411518900000035
Day 18: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 18: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 18: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 18: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 19: Bull is checking triggers
Bull: Selling 102.39899632 TKN at price 2613.21094463096 to realize profits
Bull: Updated ETH balance is 9.457776412558554
Bull Post-Execution ETH Balance: 9.457776412558554, TKN Balance:
102.39899631046715
Day 19: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 19: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 19: Harvester is checking triggers
Harvester: Selling 0.28823038 TKN at price 2613.21094463096
Harvester: Updated TKN balance is 1.1529215100000036
Harvester Post-Execution ETH Balance: 12.24225535296826, TKN Balance:
1.1529215100000036
Day 19: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 19: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 19: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 19: Sniper2 is checking triggers
```

```
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 20: Bull is checking triggers
Bull Post-Execution ETH Balance: 9.457776412558554, TKN Balance:
102.39899631046715
Day 20: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 20: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 20: Harvester is checking triggers
Harvester: Selling 0.2305843 TKN at price 2587.5812302026898
Harvester: Updated TKN balance is 0.9223372100000037
Harvester Post-Execution ETH Balance: 12.24756909243843, TKN Balance:
0.9223372100000037
Day 20: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 20: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 20: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 20: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 21: Bull is checking triggers
Bull Post-Execution ETH Balance: 9.457776412558554, TKN Balance:
102.39899631046715
Day 21: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 21: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 21: Harvester is checking triggers
Harvester: Selling 0.18446744 TKN at price 2603.0202804876176
Harvester: Updated TKN balance is 0.7378697700000036
Harvester Post-Execution ETH Balance: 12.251801491675545, TKN Balance:
0.7378697700000036
Day 21: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 21: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 0, TKN Balance: 842.6785009935214
Day 21: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 21: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 22: Bull is checking triggers
Bull: Selling 51.19949816 TKN at price 2667.382738130305 to realize profits
Bull: Updated ETH balance is 10.599737855524799
Bull Post-Execution ETH Balance: 10.599737855524799, TKN Balance:
51.19949815046716
```

```
Day 22: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 22: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 22: Harvester is checking triggers
Harvester: Selling 0.14757395 TKN at price 2667.382738130305
Harvester: Updated TKN balance is 0.5902958200000037
Harvester Post-Execution ETH Balance: 12.258447535341835, TKN Balance:
0.5902958200000037
Day 22: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 22: Hawk is checking triggers
Hawk: Selling all TKN at price 2667.382738130305 after reaching profit target
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 22: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 22: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 23: Bull is checking triggers
Bull: Selling 25.59974908 TKN at price 2666.6130596777134 to realize profits
Bull: Updated ETH balance is 10.939081345358394
Bull Post-Execution ETH Balance: 10.939081345358394, TKN Balance:
25.59974907046716
Day 23: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 23: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 23: Harvester is checking triggers
Harvester: Selling 0.11805916 TKN at price 2666.6130596777134
Harvester: Updated TKN balance is 0.4722366600000037
Harvester Post-Execution ETH Balance: 12.260775184770102, TKN Balance:
0.4722366600000037
Day 23: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 23: Hawk is checking triggers
Hawk: Selling all TKN at price 2666.6130596777134 after reaching profit target
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 23: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 23: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 24: Bull is checking triggers
Bull: Selling 12.79987454 TKN at price 2733.689823651925 to realize profits
Bull: Updated ETH balance is 11.079936839552142
Bull Post-Execution ETH Balance: 11.079936839552142, TKN Balance:
12.799874530467159
```

```
Day 24: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 24: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 24: Harvester is checking triggers
Harvester: Selling 0.09444733 TKN at price 2733.689823651925
Harvester: Updated TKN balance is 0.3777893300000037
Harvester Post-Execution ETH Balance: 12.262232674183004, TKN Balance:
0.3777893300000037
Day 24: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 24: Hawk is checking triggers
Hawk: Selling all TKN at price 2733.689823651925 after reaching profit target
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 24: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 24: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 25: Bull is checking triggers
Bull Post-Execution ETH Balance: 11.079936839552142, TKN Balance:
12.799874530467159
Day 25: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 25: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 25: Harvester is checking triggers
Harvester: Selling 0.07555787 TKN at price 2624.2342395809787
Harvester: Updated TKN balance is 0.30223146000000367
Harvester Post-Execution ETH Balance: 12.263064918085577, TKN Balance:
0.30223146000000367
Day 25: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 25: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 25: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 25: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 26: Bull is checking triggers
Bull Post-Execution ETH Balance: 11.079936839552142, TKN Balance:
12.799874530467159
Day 26: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 26: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
```

```
Day 26: Harvester is checking triggers
Harvester: Selling 0.06044629 TKN at price 2659.0507835489007
Harvester: Updated TKN balance is 0.24178517000000366
Harvester Post-Execution ETH Balance: 12.26372968201266, TKN Balance:
0.24178517000000366
Day 26: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 26: Hawk is checking triggers
Hawk: Selling all TKN at price 2659.0507835489007 after reaching profit target
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 26: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 26: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 27: Bull is checking triggers
Bull Post-Execution ETH Balance: 11.079936839552142, TKN Balance:
12.799874530467159
Day 27: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 27: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 27: Harvester is checking triggers
Harvester: Selling 0.04835703 TKN at price 2663.4174636308485
Harvester: Updated TKN balance is 0.19342814000000366
Harvester Post-Execution ETH Balance: 12.26426147097952, TKN Balance:
0.19342814000000366
Day 27: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 27: Hawk is checking triggers
Hawk: Selling all TKN at price 2663.4174636308485 after reaching profit target
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 27: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 27: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 28: Bull is checking triggers
Bull Post-Execution ETH Balance: 11.079936839552142, TKN Balance:
12.799874530467159
Day 28: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 28: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 28: Harvester is checking triggers
Harvester: Selling 0.03868563 TKN at price 2651.6678080273728
Harvester: Updated TKN balance is 0.15474251000000366
Harvester Post-Execution ETH Balance: 12.264686889955028, TKN Balance:
```

```
0.15474251000000366
Day 28: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 28: Hawk is checking triggers
Hawk: Selling all TKN at price 2651.6678080273728 after reaching profit target
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 28: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 28: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 29: Bull is checking triggers
Bull Post-Execution ETH Balance: 11.079936839552142, TKN Balance:
12.799874530467159
Day 29: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 29: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 29: Harvester is checking triggers
Harvester: Selling 0.0309485 TKN at price 2656.2189646438674
Harvester: Updated TKN balance is 0.12379401000000366
Harvester Post-Execution ETH Balance: 12.265027217249118, TKN Balance:
0.12379401000000366
Day 29: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 29: Hawk is checking triggers
Hawk: Selling all TKN at price 2656.2189646438674 after reaching profit target
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 29: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 29: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 30: Bull is checking triggers
Skipping Bull - trade already made today
Day 30: Bear is checking triggers
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 30: DCA is checking triggers
Skipping DCA - trade already made today
Day 30: Harvester is checking triggers
Skipping Harvester - trade already made today
Day 30: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 30: Hawk is checking triggers
Skipping Hawk - trade already made today
Day 30: Sniper1 is checking triggers
Skipping Sniper1 - trade already made today
Day 30: Sniper2 is checking triggers
Skipping Sniper2 - trade already made today
```

```
Day 31: Bull is checking triggers
Bull: Actively buying TKN with 5.53996842 ETH at price 2567.1387086984787
Bull: Updated ETH balance is 5.5399684195521415
Bull Post-Execution ETH Balance: 5.5399684195521415, TKN Balance:
440.46254443709233
Day 31: Bear is checking triggers
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 31: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 31: Harvester is checking triggers
Harvester: Selling 0.0247588 TKN at price 2567.1387086984787
Harvester: Updated TKN balance is 0.09903521000000366
Harvester Post-Execution ETH Balance: 12.286142510749826, TKN Balance:
0.09903521000000366
Day 31: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 31: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 31: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 31: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 32: Bull is checking triggers
Bull: Actively buying TKN with 2.76998421 ETH at price 2582.282108798289
Bull: Updated ETH balance is 2.7699842095521414
Bull Post-Execution ETH Balance: 2.7699842095521414, TKN Balance:
612.0155522634009
Day 32: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 32: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 32: Harvester is checking triggers
Harvester: Selling 0.01980704 TKN at price 2582.282108798289
Harvester: Updated TKN balance is 0.07922817000000366
Harvester Post-Execution ETH Balance: 12.295462528899225, TKN Balance:
0.07922817000000366
Day 32: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 32: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 32: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 32: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 33: Bull is checking triggers
Bull Post-Execution ETH Balance: 2.7699842095521414, TKN Balance:
612.0155522634009
```

```
Day 33: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 33: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 33: Harvester is checking triggers
Harvester: Selling 0.01584563 TKN at price 2643.633963035625
Harvester: Updated TKN balance is 0.06338254000000366
Harvester Post-Execution ETH Balance: 12.295760991338703, TKN Balance:
0.06338254000000366
Day 33: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 33: Hawk is checking triggers
Hawk: Selling all TKN at price 2643.633963035625 after reaching profit target
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 33: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 33: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 34: Bull is checking triggers
Bull: Actively buying TKN with 1.3849921 ETH at price 2622.9252261517922
Bull: Updated ETH balance is 1.3849921095521414
Bull Post-Execution ETH Balance: 1.3849921095521414, TKN Balance:
689.567025153935
Day 34: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 34: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 34: Harvester is checking triggers
Harvester: Selling 0.01267651 TKN at price 2622.9252261517922
Harvester: Updated TKN balance is 0.05070603000000366
Harvester Post-Execution ETH Balance: 12.300273121740698, TKN Balance:
0.05070603000000366
Day 34: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 34: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 34: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 34: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 35: Bull is checking triggers
Bull: Actively buying TKN with 0.69249605 ETH at price 2590.5458834015403
Bull: Updated ETH balance is 0.6924960595521413
Bull Post-Execution ETH Balance: 0.6924960595521413, TKN Balance:
726.512605848451
Day 35: Bear is checking triggers
```

```
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 35: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 35: Harvester is checking triggers
Harvester: Selling 0.01014121 TKN at price 2590.5458834015403
Harvester: Updated TKN balance is 0.040564820000003665
Harvester Post-Execution ETH Balance: 12.302563647365895, TKN Balance:
0.040564820000003665
Day 35: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 35: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 35: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 35: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 36: Bull is checking triggers
Bull: Actively buying TKN with 0.34624803 ETH at price 2570.9195660226196
Bull: Updated ETH balance is 0.34624802955214135
Bull Post-Execution ETH Balance: 0.34624802955214135, TKN Balance:
744.5526164938229
Day 36: Bear is checking triggers
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 36: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 36: Harvester is checking triggers
Harvester: Selling 0.00811296 TKN at price 2570.9195660226196
Harvester: Updated TKN balance is 0.03245186000000366
Harvester Post-Execution ETH Balance: 12.303759549601319, TKN Balance:
0.03245186000000366
Day 36: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 36: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 36: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 36: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 37: Bull is checking triggers
Bull: Actively buying TKN with 0.17312401 ETH at price 2608.856274577046
Bull: Updated ETH balance is 0.17312401955214135
Bull Post-Execution ETH Balance: 0.17312401955214135, TKN Balance:
753.4673874152032
Day 37: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 37: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
```

```
Day 37: Harvester is checking triggers
Harvester: Selling 0.00649037 TKN at price 2608.856274577046
Harvester: Updated TKN balance is 0.025961490000003664
Harvester Post-Execution ETH Balance: 12.30440287442374, TKN Balance:
0.025961490000003664
Day 37: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 37: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 37: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 37: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 38: Bull is checking triggers
Bull Post-Execution ETH Balance: 0.17312401955214135, TKN Balance:
753.4673874152032
Day 38: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 38: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 38: Harvester is checking triggers
Harvester: Selling 0.0051923 TKN at price 2623.0773111735193
Harvester: Updated TKN balance is 0.020769190000003664
Harvester Post-Execution ETH Balance: 12.304504820928242, TKN Balance:
0.020769190000003664
Day 38: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 38: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 38: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 38: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 39: Bull is checking triggers
Bull: Actively buying TKN with 0.08656201 ETH at price 2602.0600870624376
Bull: Updated ETH balance is 0.08656200955214136
Bull Post-Execution ETH Balance: 0.08656200955214136, TKN Balance:
757.8987714329903
Day 39: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 39: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 39: Harvester is checking triggers
Harvester: Selling 0.00415384 TKN at price 2602.0600870624376
Harvester: Updated TKN balance is 0.016615350000003665
Harvester Post-Execution ETH Balance: 12.30484378081407, TKN Balance:
```

```
0.016615350000003665
Day 39: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 39: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 39: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 39: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 40: Bull is checking triggers
Bull Post-Execution ETH Balance: 0.08656200955214136, TKN Balance:
757.8987714329903
Day 40: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 40: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 40: Harvester is checking triggers
Harvester: Selling 0.00332307 TKN at price 2623.835395306057
Harvester: Updated TKN balance is 0.013292280000003664
Harvester Post-Execution ETH Balance: 12.304909058223751, TKN Balance:
0.013292280000003664
Day 40: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 40: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 40: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 40: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 41: Bull is checking triggers
Bull Post-Execution ETH Balance: 0.08656200955214136, TKN Balance:
757.8987714329903
Day 41: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 41: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 41: Harvester is checking triggers
Harvester: Selling 0.00265846 TKN at price 2628.5582254485107
Harvester: Updated TKN balance is 0.010633820000003664
Harvester Post-Execution ETH Balance: 12.304960662418507, TKN Balance:
0.010633820000003664
Day 41: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 41: Hawk is checking triggers
Hawk: Selling all TKN at price 2628.5582254485107 after reaching profit target
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
```

```
Day 41: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 41: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 42: Bull is checking triggers
Bull: Selling 378.94938572 TKN at price 2669.576442442426 to realize profits
Bull: Updated ETH balance is 6.357294923220674
Bull Post-Execution ETH Balance: 6.357294923220674, TKN Balance:
378.9493857129903
Day 42: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 42: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 42: Harvester is checking triggers
Harvester: Selling 0.00212676 TKN at price 2669.576442442426
Harvester: Updated TKN balance is 0.008507060000003664
Harvester Post-Execution ETH Balance: 12.325305058644865, TKN Balance:
0.008507060000003664
Day 42: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 42: Hawk is checking triggers
Hawk: Selling all TKN at price 2669.576442442426 after reaching profit target
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 42: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 42: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 43: Bull is checking triggers
Bull Post-Execution ETH Balance: 6.357294923220674, TKN Balance:
378.9493857129903
Day 43: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 43: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 43: Harvester is checking triggers
Harvester: Selling 0.00170141 TKN at price 2641.031749761066
Harvester: Updated TKN balance is 0.006805650000003664
Harvester Post-Execution ETH Balance: 12.32539027690446, TKN Balance:
0.006805650000003664
Day 43: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 43: Hawk is checking triggers
Hawk: Selling all TKN at price 2641.031749761066 after reaching profit target
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 43: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
```

```
Day 43: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 44: Bull is checking triggers
Bull Post-Execution ETH Balance: 6.357294923220674, TKN Balance:
378.9493857129903
Day 44: Bear is checking triggers
Bear: Updated TKN balance is 0
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 44: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 44: Harvester is checking triggers
Harvester: Selling 0.00136113 TKN at price 2628.1980033763607
Harvester: Updated TKN balance is 0.005444520000003663
Harvester Post-Execution ETH Balance: 12.325409880491705, TKN Balance:
0.005444520000003663
Day 44: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 44: Hawk is checking triggers
Hawk: Selling all TKN at price 2628.1980033763607 after reaching profit target
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 44: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 44: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 45: Bull is checking triggers
Bull: Actively buying TKN with 3.17864746 ETH at price 2612.781206917272
Bull: Updated ETH balance is 3.1786474632206736
Bull Post-Execution ETH Balance: 3.1786474632206736, TKN Balance:
582.6400755224518
Day 45: Bear is checking triggers
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 45: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 45: Harvester is checking triggers
Harvester: Selling 0.0010889 TKN at price 2612.781206917272
Harvester: Updated TKN balance is 0.004355620000003663
Harvester Post-Execution ETH Balance: 12.335911574000727, TKN Balance:
0.004355620000003663
Day 45: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 45: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 45: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 45: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 46: Bull is checking triggers
Bull: Actively buying TKN with 1.58932373 ETH at price 2554.1208158801037
```

```
Bull: Updated ETH balance is 1.5893237332206736
Bull Post-Execution ETH Balance: 1.5893237332206736, TKN Balance:
673.2125968989349
Day 46: Bear is checking triggers
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 46: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 46: Harvester is checking triggers
Harvester Post-Execution ETH Balance: 12.335911574000727, TKN Balance:
0.004355620000003663
Day 46: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 46: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 46: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 46: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 47: Bull is checking triggers
Bull: Actively buying TKN with 0.79466187 ETH at price 2566.7279763013807
Bull: Updated ETH balance is 0.7946618632206736
Bull Post-Execution ETH Balance: 0.7946618632206736, TKN Balance:
716.2977356126174
Day 47: Bear is checking triggers
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 47: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 47: Harvester is checking triggers
Harvester Post-Execution ETH Balance: 12.335911574000727, TKN Balance:
0.004355620000003663
Day 47: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 47: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 47: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 47: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 48: Bull is checking triggers
Bull: Actively buying TKN with 0.39733093 ETH at price 2577.9774995719167
Bull: Updated ETH balance is 0.39733093322067353
Bull Post-Execution ETH Balance: 0.39733093322067353, TKN Balance:
737.2538339094189
Day 48: Bear is checking triggers
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 48: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 48: Harvester is checking triggers
```

```
Harvester Post-Execution ETH Balance: 12.335911574000727, TKN Balance:
0.004355620000003663
Day 48: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 48: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 48: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 48: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Day 49: Bull is checking triggers
Bull: Actively buying TKN with 0.19866547 ETH at price 2578.890955703066
Bull: Updated ETH balance is 0.19866546322067352
Bull Post-Execution ETH Balance: 0.19866546322067352, TKN Balance:
747.5897358487234
Day 49: Bear is checking triggers
Bear Post-Execution ETH Balance: 10, TKN Balance: 0
Day 49: DCA is checking triggers
DCA Post-Execution ETH Balance: 0.0, TKN Balance: 1721.1611838303008
Day 49: Harvester is checking triggers
Harvester Post-Execution ETH Balance: 12.335911574000727, TKN Balance:
0.004355620000003663
Day 49: Dove is checking triggers
Dove Post-Execution ETH Balance: 10, TKN Balance: 0
Day 49: Hawk is checking triggers
Hawk Post-Execution ETH Balance: 13.117609091690461, TKN Balance: 0.0
Day 49: Sniper1 is checking triggers
Sniper1 Post-Execution ETH Balance: 0, TKN Balance: 4927.586379569353
Day 49: Sniper2 is checking triggers
Sniper2 Post-Execution ETH Balance: 0, TKN Balance: 417.4134071304298
Strategy: Bull
  Final ETH Balance: 0.19866546322067352
  Final TKN Balance: 747.5897358487234
  Final USD Balance: 1991.8431297491306
  ROI: -92.03262748100349
Strategy: Bear
  Final ETH Balance: 10
  Final TKN Balance: 0
  Final USD Balance: 25000
  ROI: 0.0
Strategy: DCA
  Final ETH Balance: 0.0
  Final TKN Balance: 1721.1611838303008
  Final USD Balance: 3442.3223676606017
  ROI: -86.2307105293576
Strategy: Harvester
  Final ETH Balance: 12.335911574000727
  Final TKN Balance: 0.004355620000003663
```

```
  Final USD Balance: 30839.787646241817
  ROI: 22.86767986550525
Strategy: Dove
  Final ETH Balance: 10
  Final TKN Balance: 0
  Final USD Balance: 25000
  ROI: 0.0
Strategy: Hawk
  Final ETH Balance: 13.117609091690461
  Final TKN Balance: 0.0
  Final USD Balance: 32794.022729226155
  ROI: 31.17609091690462
Strategy: Sniper1
  Final ETH Balance: 0
  Final TKN Balance: 4927.586379569353
  Final USD Balance: 9855.172759138706
  ROI: -60.57930896344518
Strategy: Sniper2
  Final ETH Balance: 0
  Final TKN Balance: 417.4134071304298
  Final USD Balance: 834.8268142608596
  ROI: -96.66069274295656
Final Balances DataFrame:
     Strategy  Final_ETH_Balance  Final_TKN_Balance  Final_USD_Balance  \
0       Bull           0.198665         747.589736        1991.843130
1       Bear          10.000000           0.000000       25000.000000
2        DCA           0.000000        1721.161184        3442.322368
3  Harvester          12.335912           0.004356       30839.787646
4       Dove          10.000000           0.000000       25000.000000
5       Hawk          13.117609           0.000000       32794.022729
6    Sniper1           0.000000        4927.586380        9855.172759
7    Sniper2           0.000000         417.413407         834.826814

         ROI  Sharpe_Ratio
0 -92.032627      0.043196
1   0.000000      0.043218
2 -86.230711     -0.049172
3  22.867680      0.043218
4   0.000000      0.043218
5  31.176091      0.043218
6 -60.579309     -0.049172
7 -96.660693     -0.049172
```

[1591]:
```
# Display the Price History Table
display(simulation_data['price_history_table'].head())
```

```
   Day        Price
0    0  2500.000000
```

```
1    1  2513.443445
2    2  2498.975475
3    3  2473.230759
4    4  2497.784772
```

[1592]:
```
# Display the Volatility Tracking Table
display(simulation_data['volatility_table'].head())
```

```
   Day  Volatility
0    0    0.000000
1    1    0.000000
2    2    0.150407
3    3    0.154122
4    4    0.180103
```

[1593]:
```
# Display the Mean Reversion Table
display(simulation_data['mean_reversion_table'].head())
```

```
   Day        Price  Deviation_from_Mean
0    0  2500.000000          -500.000000
1    1  2513.443445          -486.556555
2    2  2498.975475          -501.024525
3    3  2473.230759          -526.769241
4    4  2497.784772          -502.215228
```

[1594]:
```
# Display the Final Balances Table
display(simulation_data['final_balances_table'].head())
```

```
    Strategy  Final_ETH_Balance  Final_TKN_Balance  Final_USD_Balance  \
0       Bull           0.198665         747.589736        1991.843130
1       Bear          10.000000           0.000000       25000.000000
2        DCA           0.000000        1721.161184        3442.322368
3  Harvester          12.335912           0.004356       30839.787646
4       Dove          10.000000           0.000000       25000.000000

         ROI  Sharpe_Ratio
0 -92.032627      0.043196
1   0.000000      0.043218
2 -86.230711     -0.049172
3  22.867680      0.043218
4   0.000000      0.043218
```

[1595]:
```
# Display the Total Volume Swapped Table
display(simulation_data['total_volume_swapped_table'].head())
```

```
           ETH Swapped  TKN Swapped
Strategy
Bull         56.715631  2851.376663
Bear          0.000000     0.000000
DCA          10.000000  1721.161184
```

```
Harvester     2.335912    99.995644
Dove          0.000000     0.000000
```

[1596]:
```
# Display the ROI per Strategy Table
display(simulation_data['roi_per_strategy_table'].head())
```

```
    Strategy        ROI
0       Bull -92.032627
1       Bear   0.000000
2        DCA -86.230711
3  Harvester  22.867680
4       Dove   0.000000
```

[1597]:
```
# Display the Impermanent Loss Table
display(simulation_data['impermanent_loss_table'].head())
```

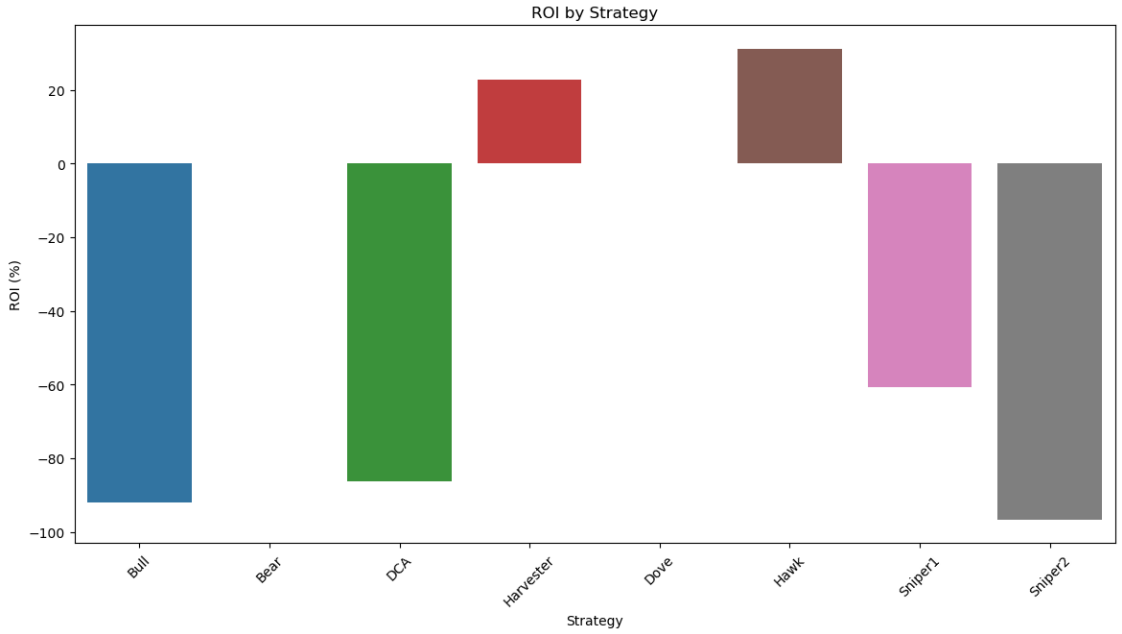
```
   Day  Impermanent_Loss
0    0         99.999842
1    1         99.999844
2    2         99.999784
3    3         99.999773
4    4         99.999776
```

[1598]:
```
# Display the Total LP Fees Table
display(simulation_data['total_lp_fees_table'].head())
```

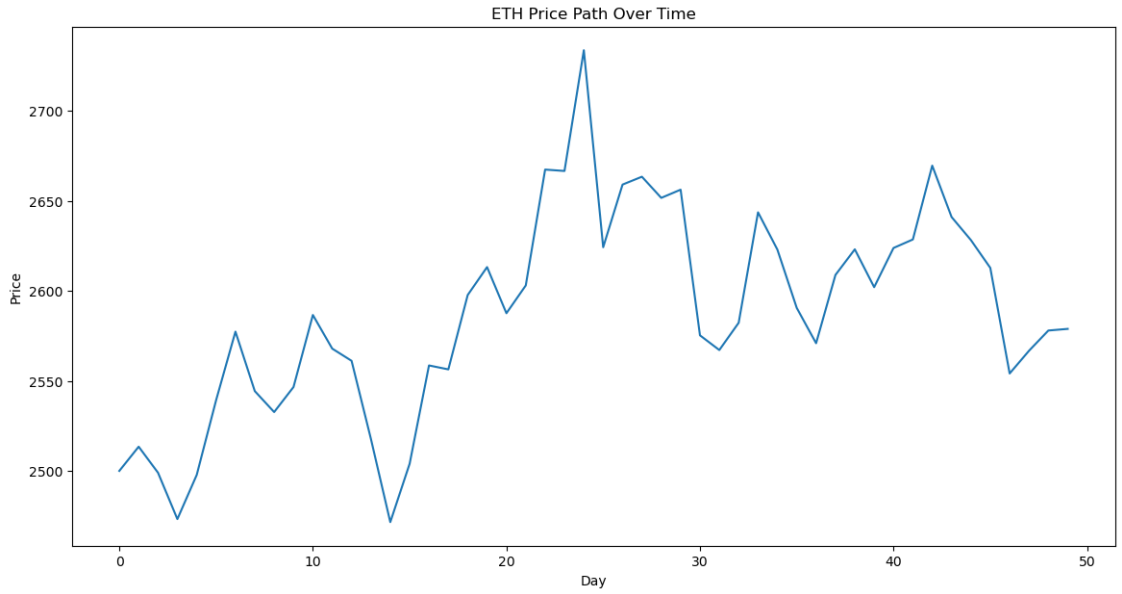
```
     ETH_Fees  TKN_Fees
Day
0    0.224120  0.061194
1    0.224120  0.110206
2    0.298201  0.149331
3    0.306859  0.180643
4    0.306859  0.205596
```

[1599]:
```
# Plot ROI by Strategy
plot_roi_by_strategy(simulation_data['results'])
```

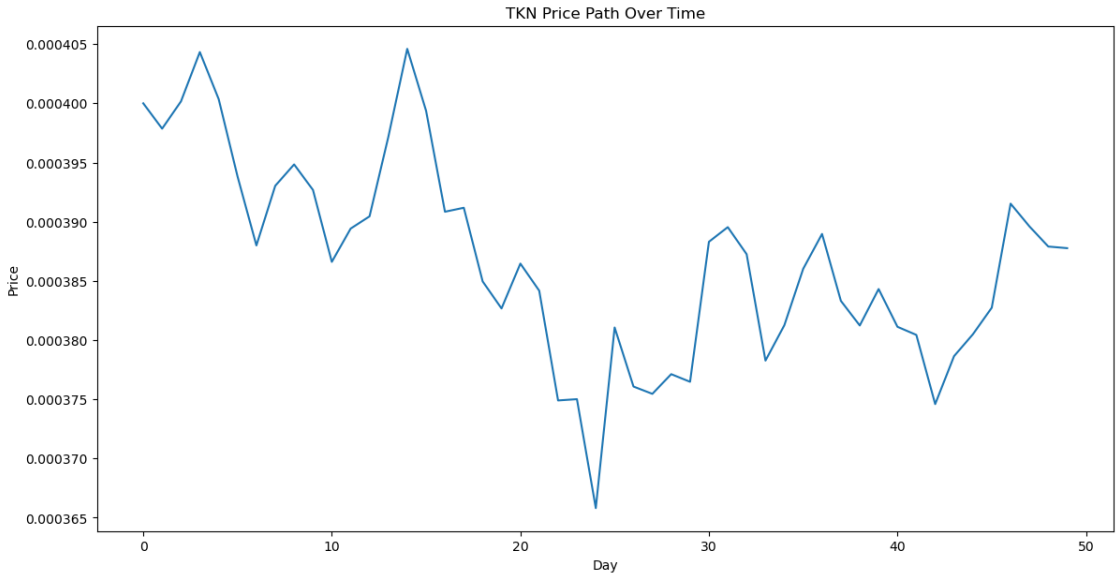
```
C:\Users\merca\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1765:
FutureWarning: unique with argument that is not not a Series, Index,
ExtensionArray, or np.ndarray is deprecated and will raise in a future version.
  order = pd.unique(vector)
```



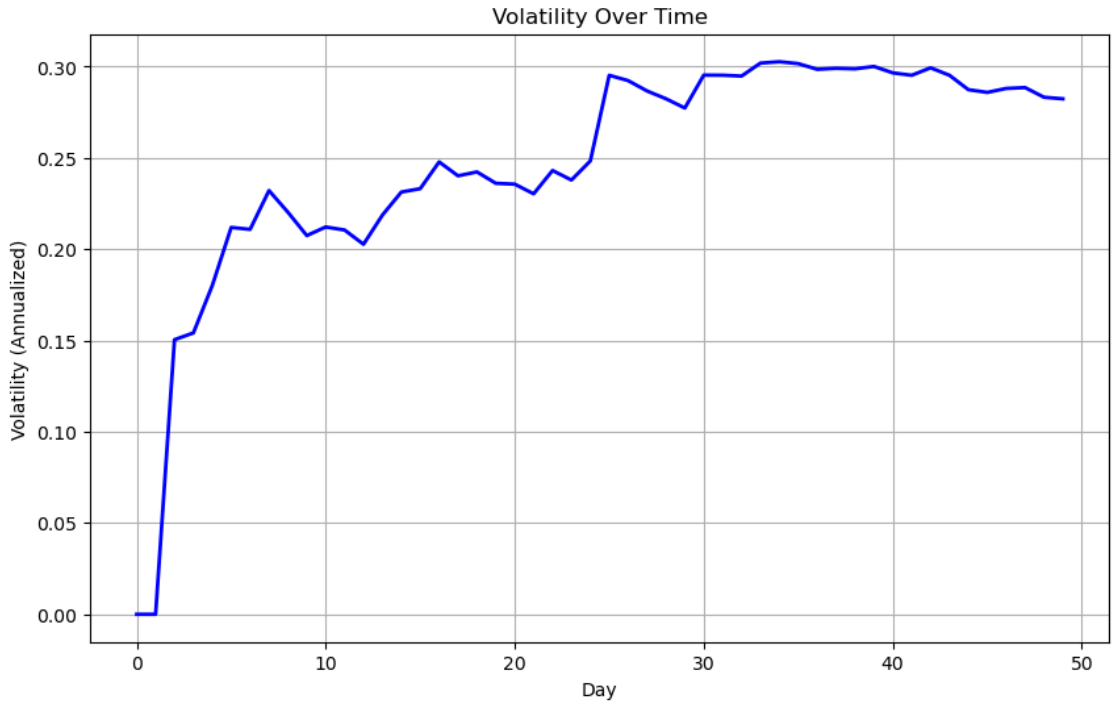
[1600]:
```
plot_price_path(simulation_data['eth_price_history'])
```



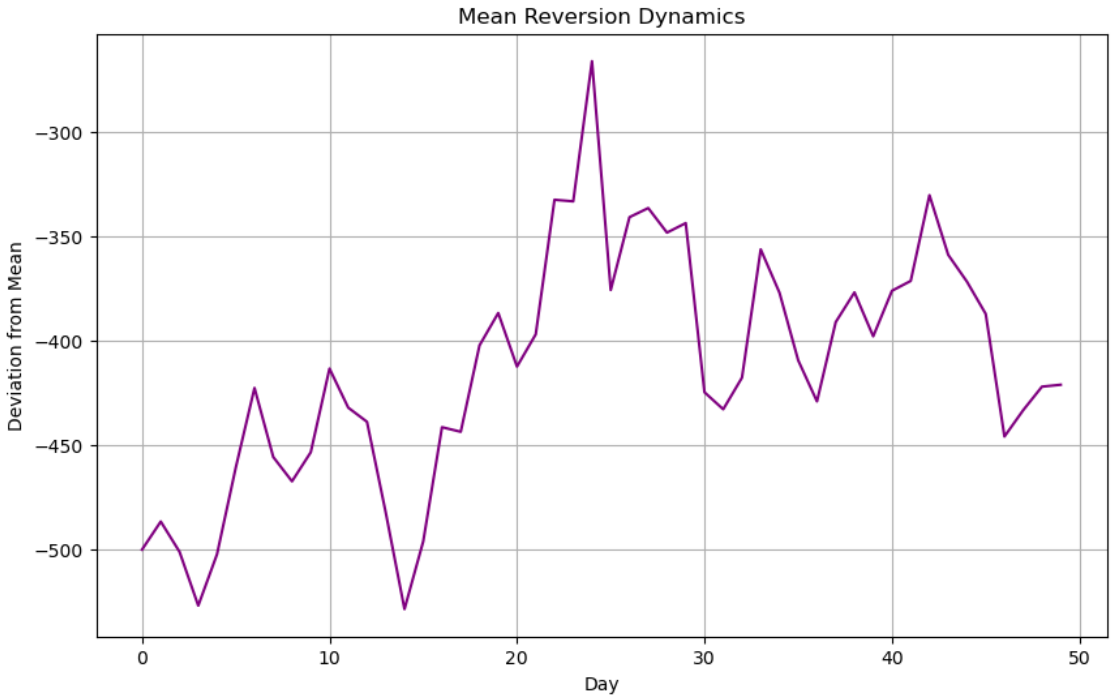
[1601]:
```
plot_tkn_price_path(simulation_data['tkn_price_history'])
```



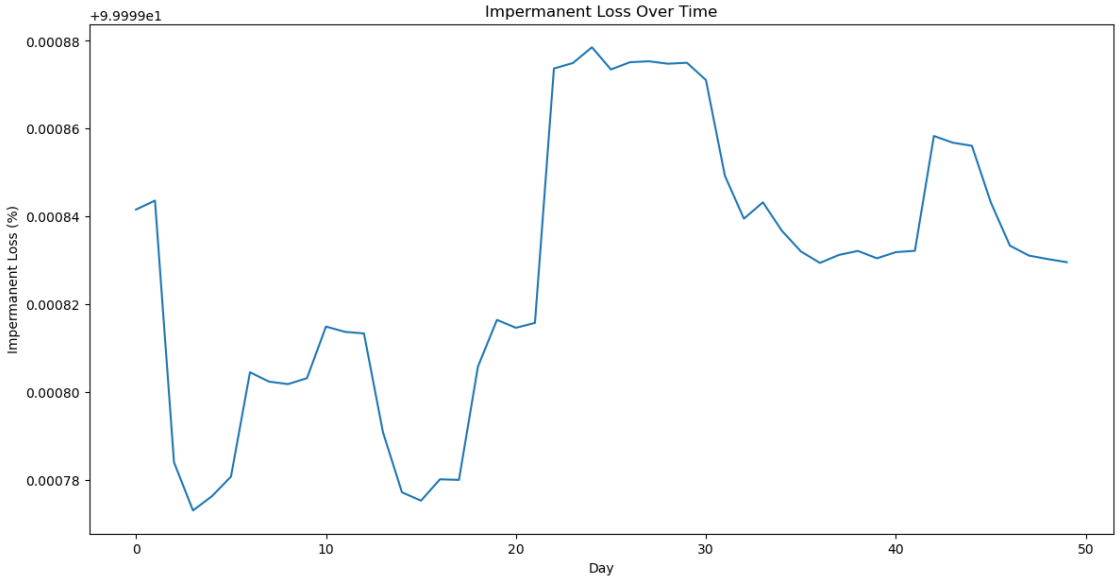
[1602]:
```
# Plot Volatility Over Time
plot_volatility_over_time(simulation_data['volatility_table'])
```



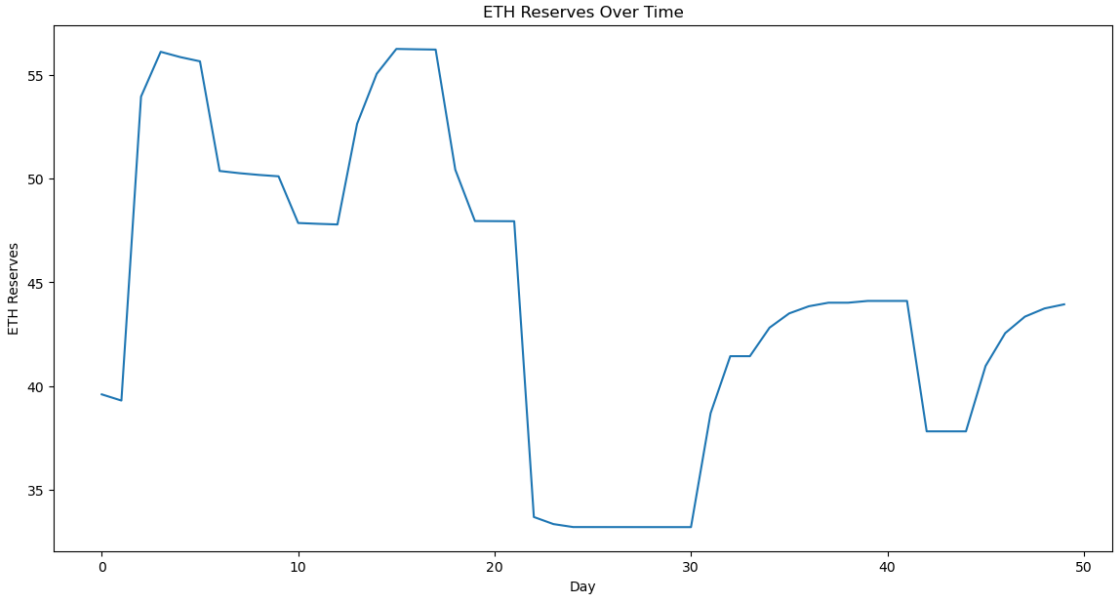
[1603]:
```
# Plot Mean Reversion Dynamics
plot_mean_reversion(simulation_data['eth_price_history'],␣
↪mean_reversion_level=3000)
```



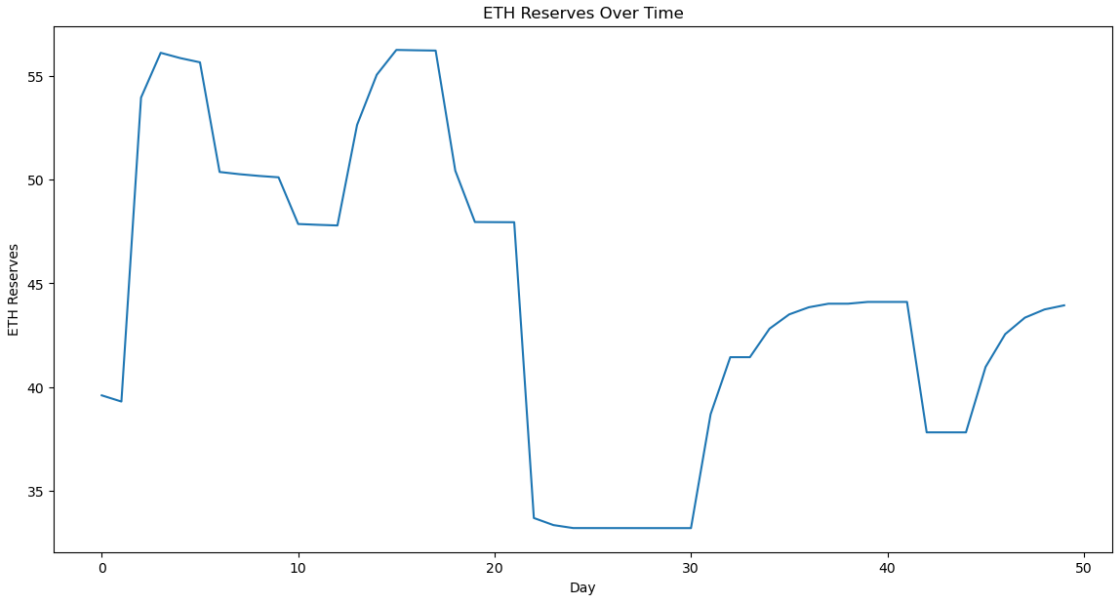
[1604]:
```
# Plot Impermanent Loss Over Time
plot_impermanent_loss_over_time(simulation_data['impermanent_losses'])
```



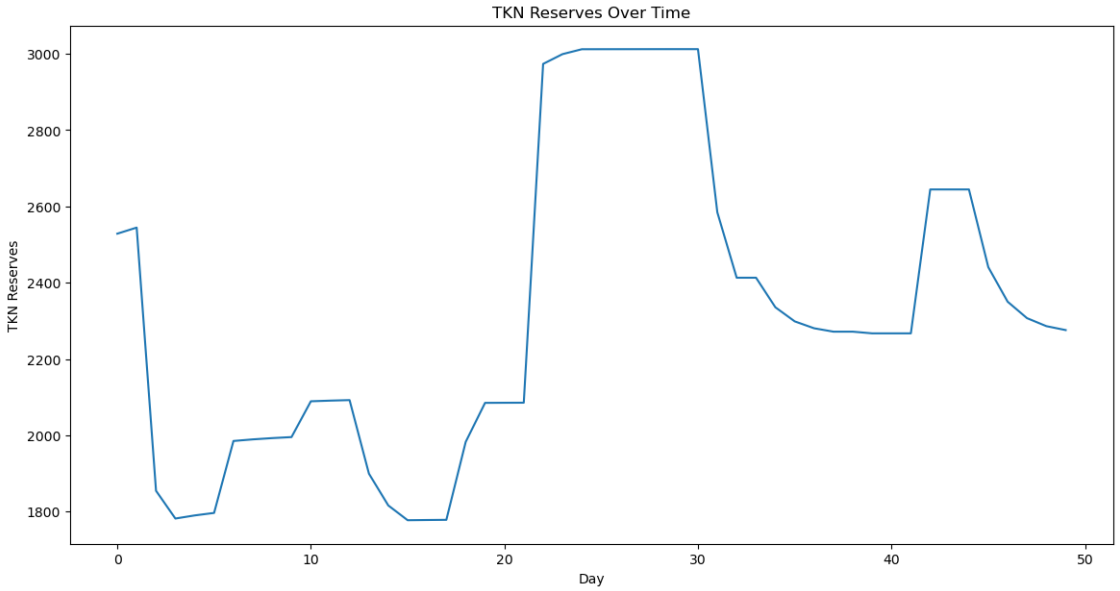
[1605]:
```
# Plot ETH Reserves Over Time
plot_eth_reserves_over_time(simulation_data['eth_reserves_over_time'])
```



[1606]:
```
# Plot ETH Reserves Over Time
plot_eth_reserves_over_time(simulation_data['eth_reserves_over_time'])
```

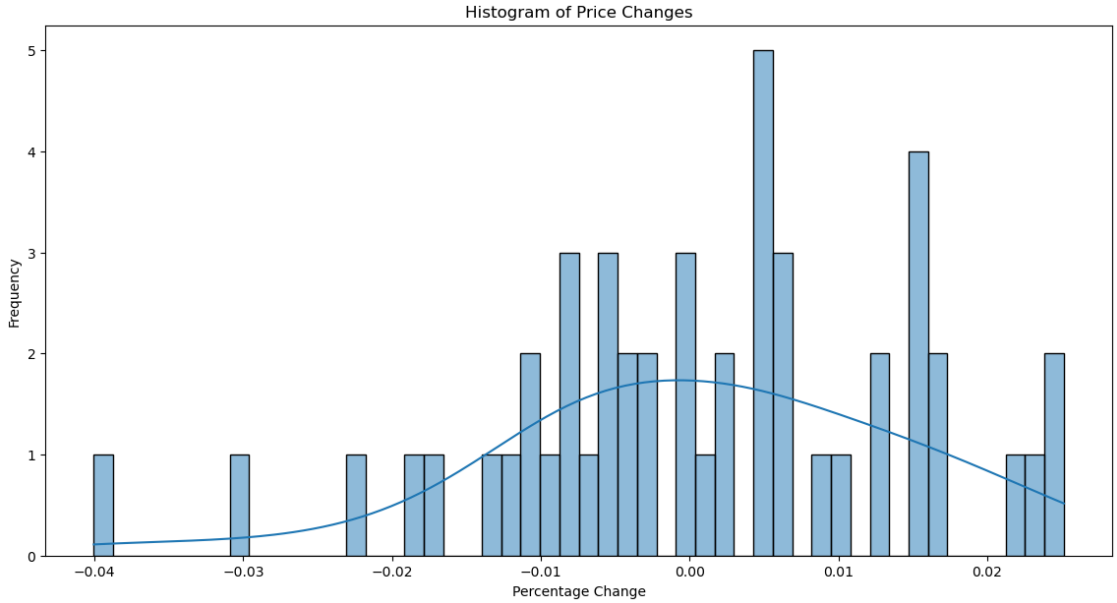


[1607]:
```
# Plot TKN Reserves Over Time
plot_tkn_reserves_over_time(simulation_data['tkn_reserves_over_time'])
```



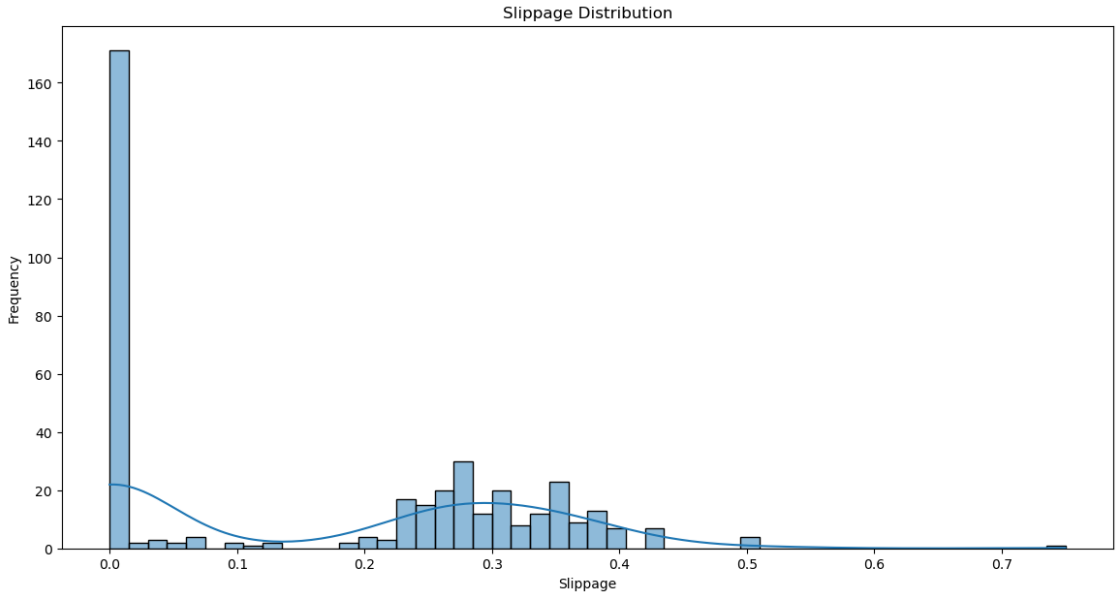
[1608]:
```
# Additional Visualizations
# Additional Visualizations
plot_histogram_of_price_changes(simulation_data['eth_price_history'])
```

```
C:\Users\merca\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```



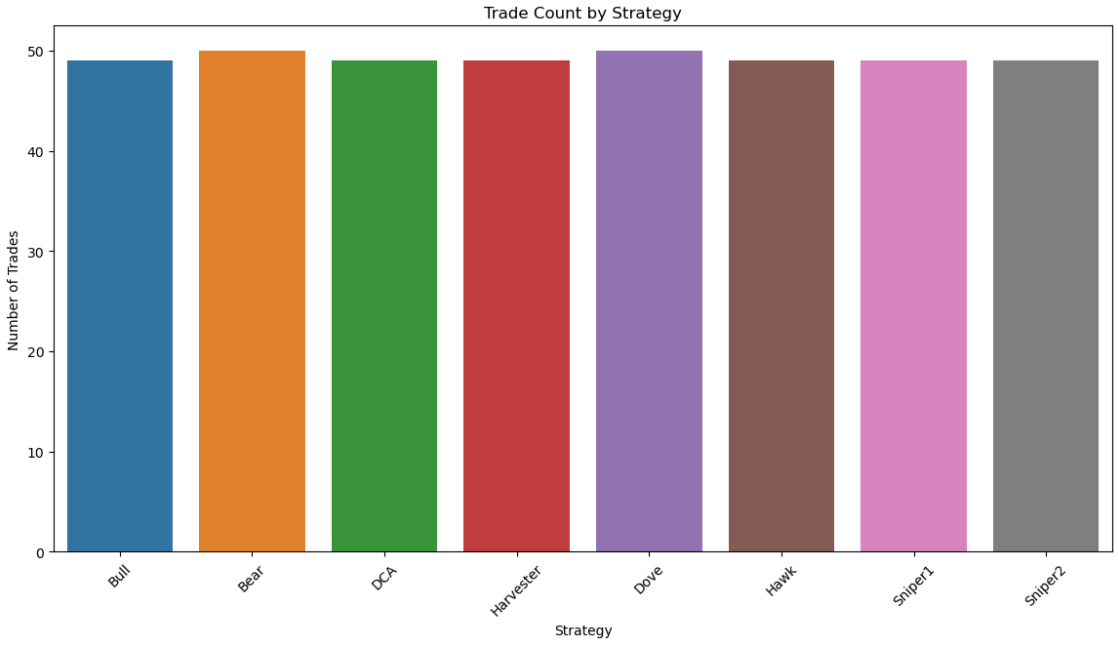
[1609]:
```
plot_slippage_distribution(simulation_data['price_impacts'])
```

```
C:\Users\merca\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1119:
FutureWarning: use_inf_as_na option is deprecated and will be removed in a
future version. Convert inf values to NaN before operating instead.
  with pd.option_context('mode.use_inf_as_na', True):
```

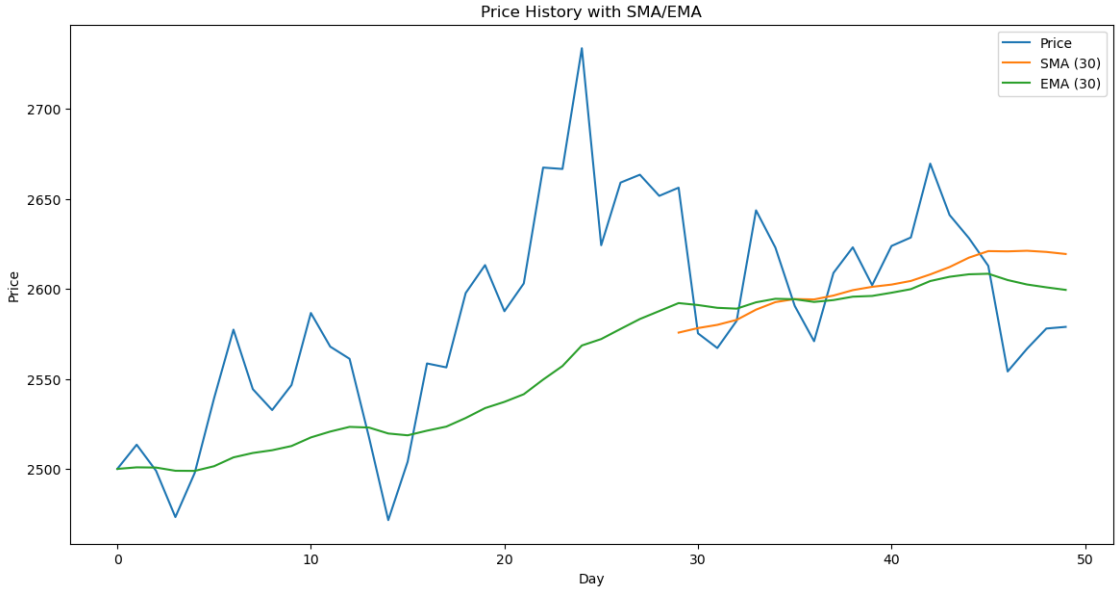


[1610]:
```
plot_trade_count_by_strategy(simulation_data['trade_counts'])
```

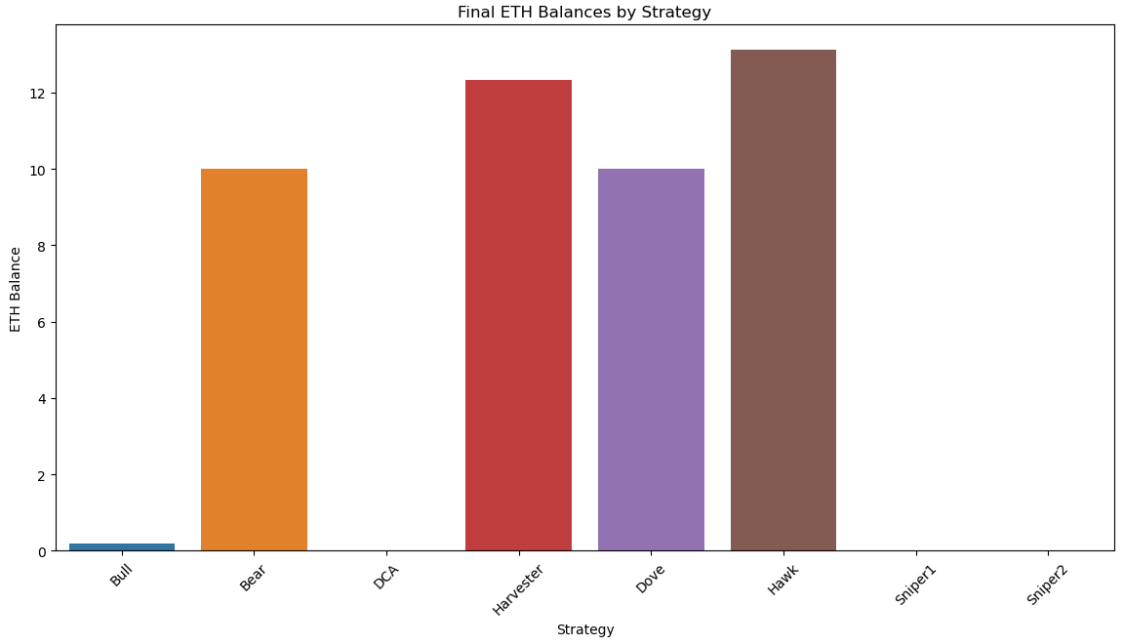
```
C:\Users\merca\anaconda3\Lib\site-packages\seaborn\_oldcore.py:1765:
FutureWarning: unique with argument that is not not a Series, Index,
ExtensionArray, or np.ndarray is deprecated and will raise in a future version.
  order = pd.unique(vector)
```



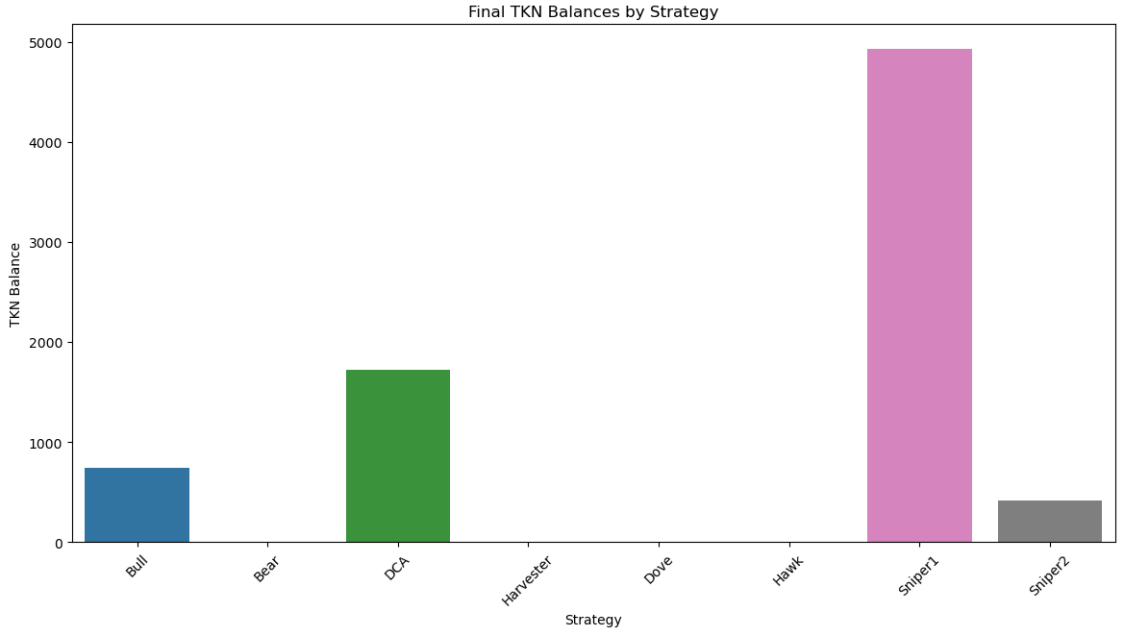
[1611]:
```
# Plot Price History with SMA and EMA
plot_price_history_with_sma_ema(simulation_data['eth_price_history'])
```



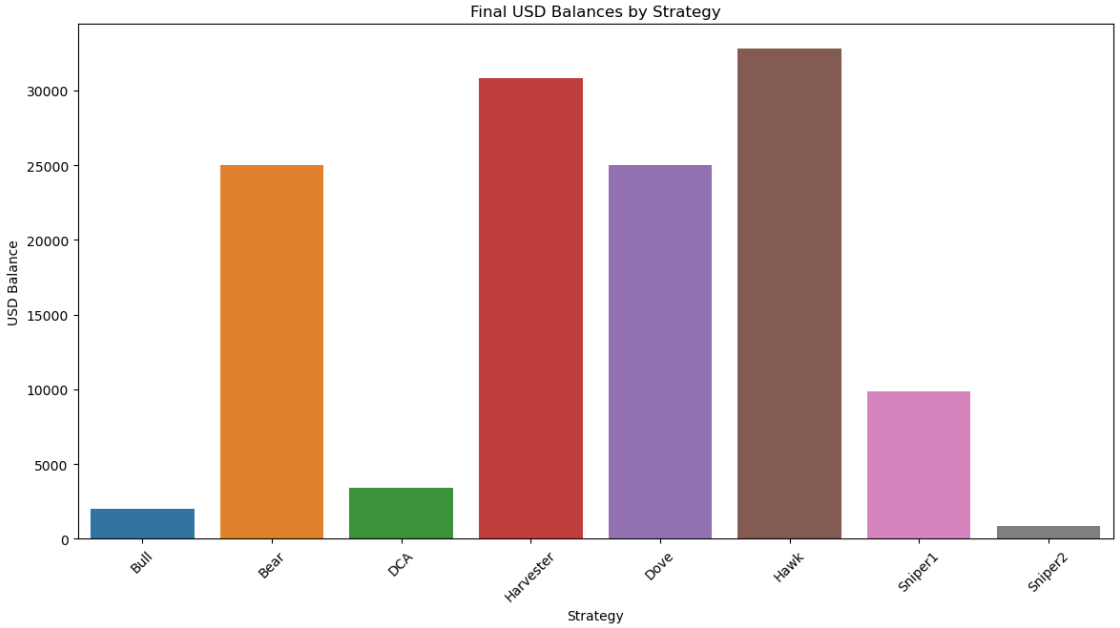
[1612]:
```
# Added Visualizations
plot_final_eth_balances_by_strategy(simulation_data['final_balances_table'])
```



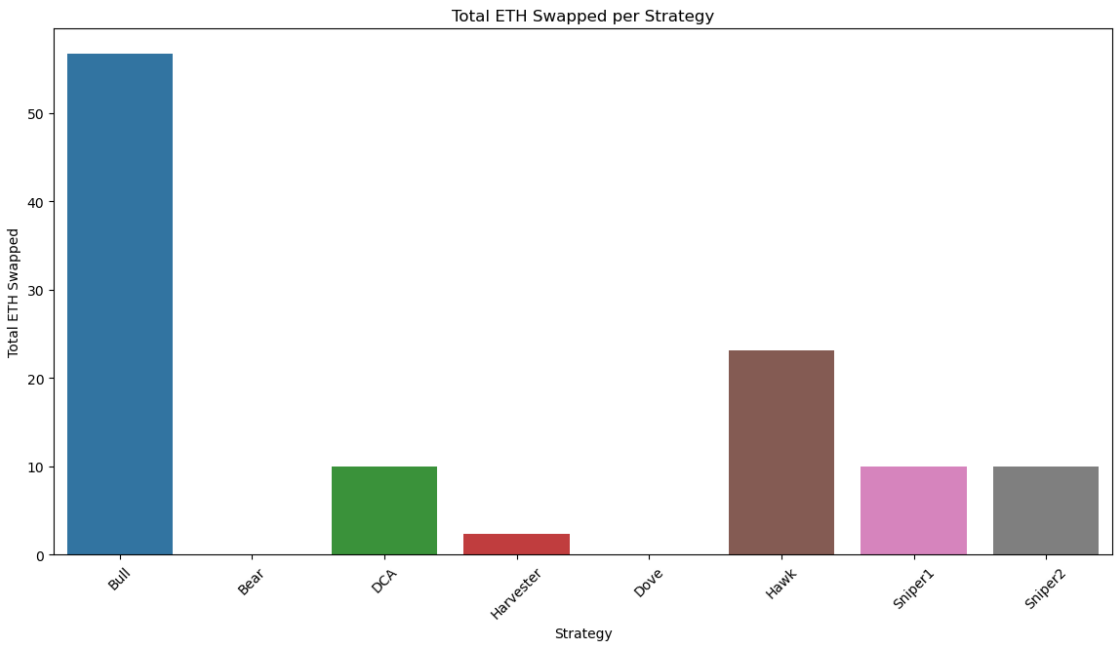
[1613]:
```
plot_final_tkn_balances_by_strategy(simulation_data['final_balances_table'])
```



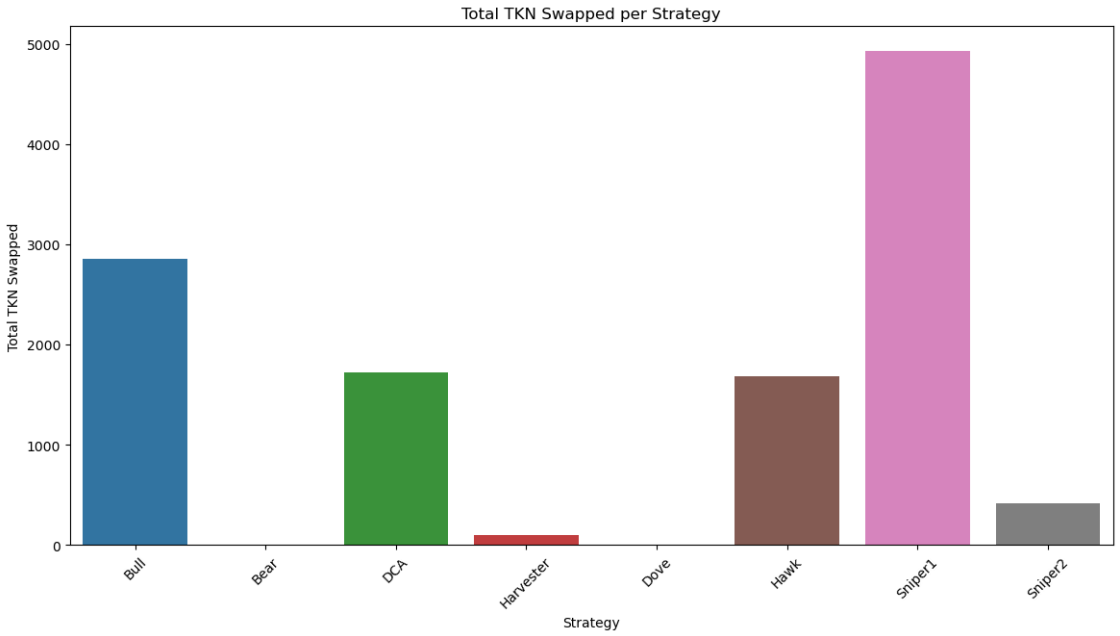
[1614]:
```
plot_final_usd_balances_by_strategy(simulation_data['final_balances_table'])
```



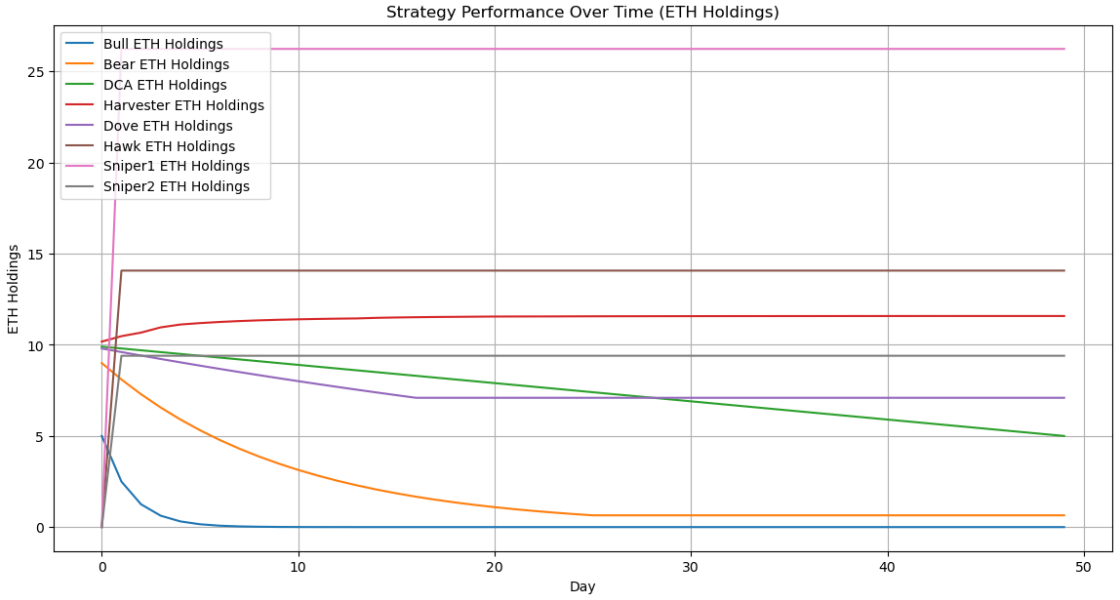
[1615]:
```
plot_total_eth_swapped_per_strategy(simulation_data['total_volume_swapped_table'])
```

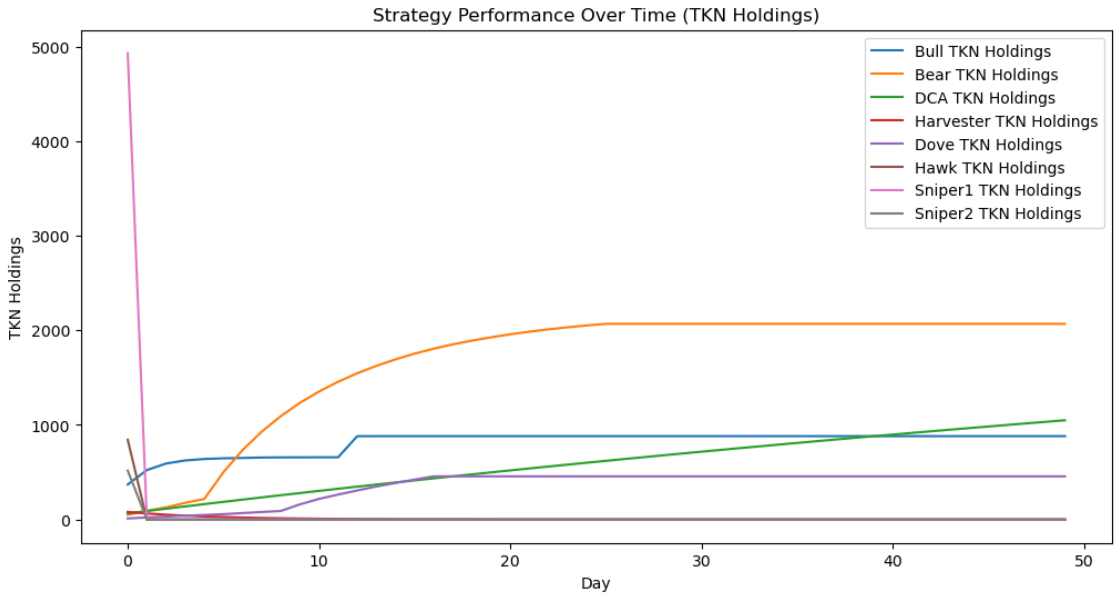


[1616]:
```
plot_total_tkn_swapped_per_strategy(simulation_data['total_volume_swapped_table'])
```

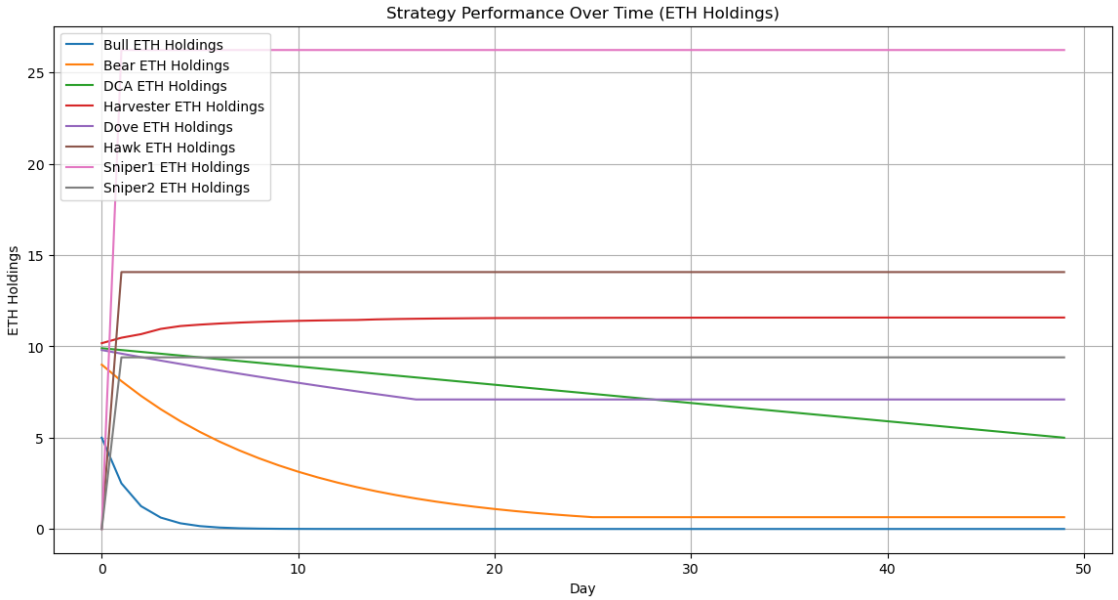


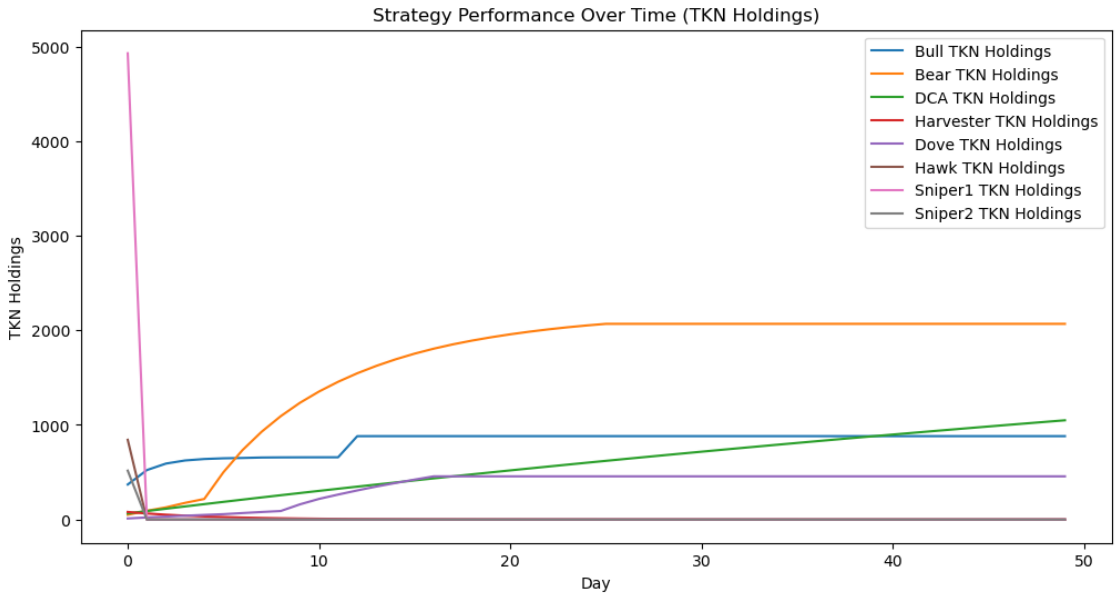
[1617]:
```
plot_strategy_performance_over_time(strategies, price_history)
```



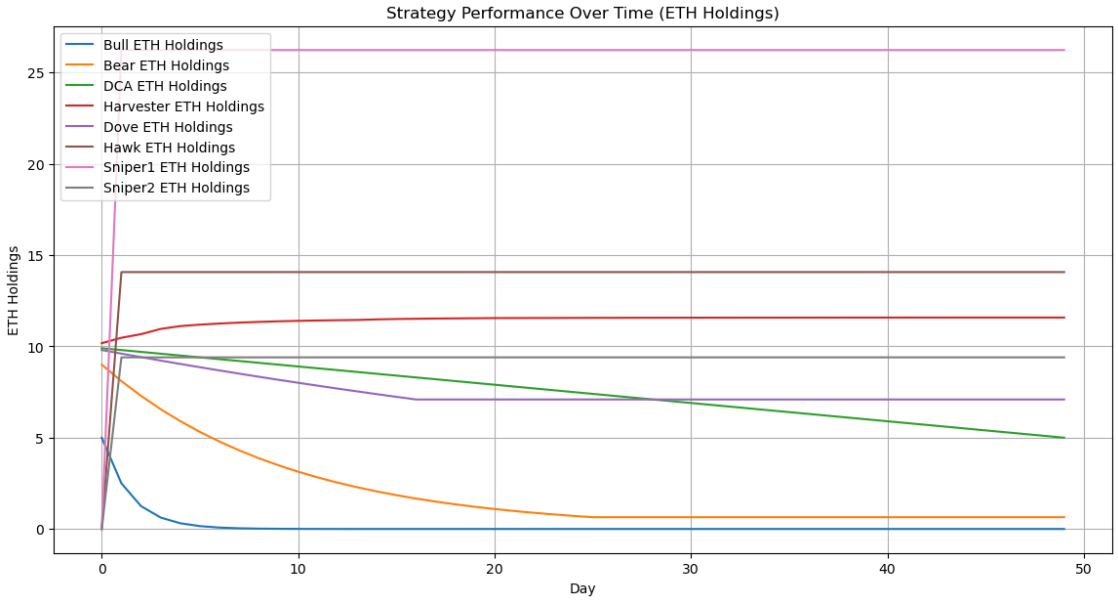


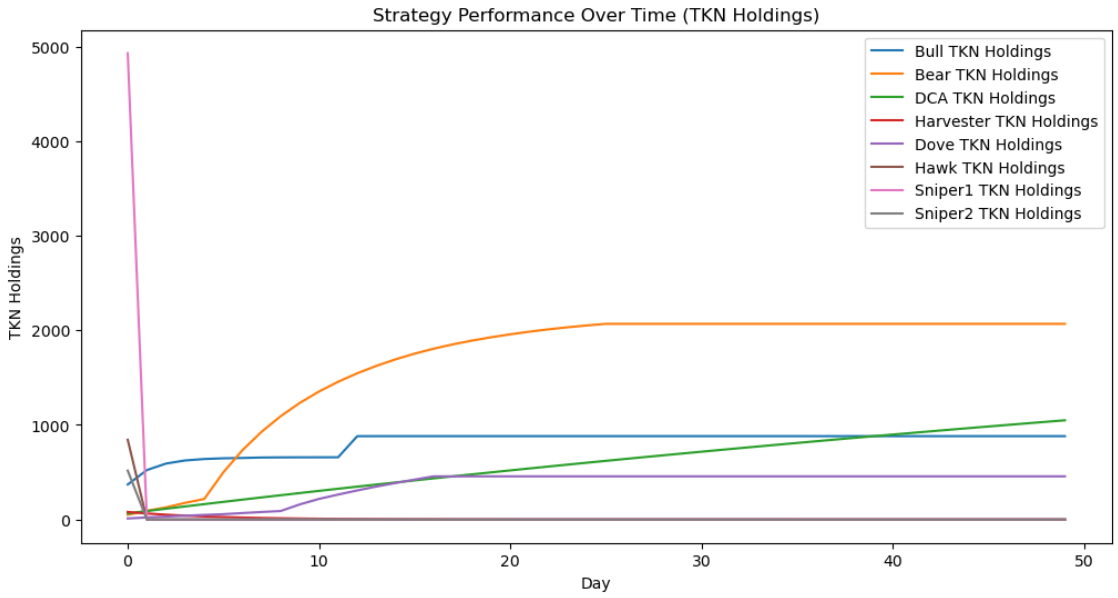
[1622]:
```
plot_strategy_performance_over_time(strategies, eth_price_history)
```



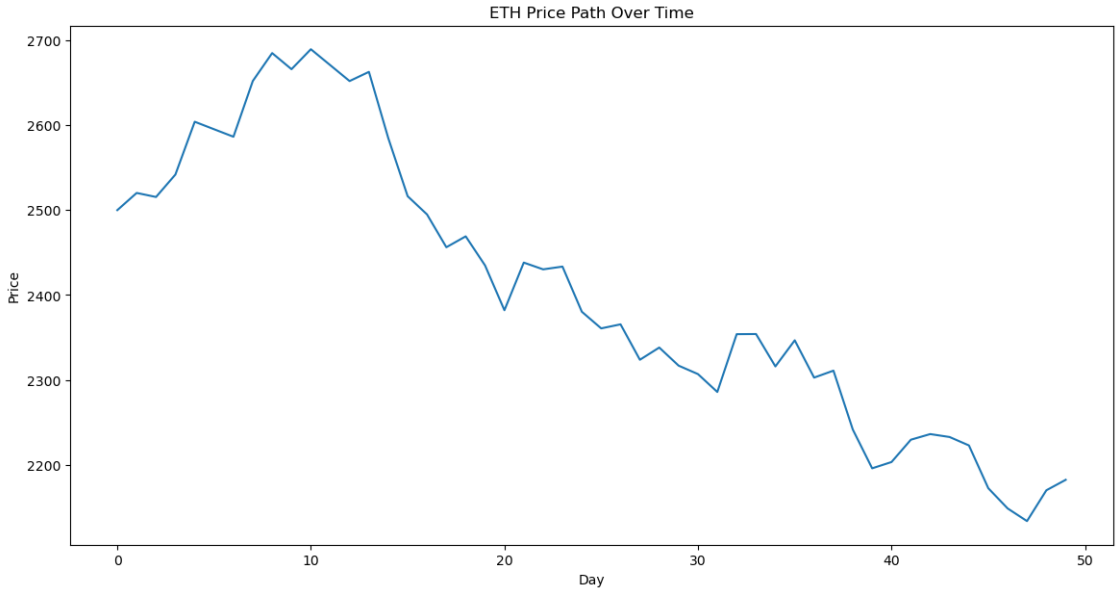


[1623]:
```
plot_strategy_performance_over_time(strategies, tkn_price_history)
```





[1618]:
```
plot_price_path(price_history)
```



[ ]:
```

```

[1619]:
```
'''
12. Strategy-Specific Logic Errors (Hardest Fix)
```

```
Issue: Some strategies may have logic that is not performing as intended,␣
 ↪leading to unexpected results or ineffective trades.
Correction: Conduct a thorough review of each strategy's logic to ensure it␣
 ↪aligns with its intended behavior. For example, Dove should consistently␣
 ↪manage its risk without depleting its ETH, and Bull should maintain a feasible␣
 ↪balance.
Likely Function(s):
Strategy.execute_trade()
Strategy.check_triggers()
Strategy.manage_balance()

lets get back to the strategies, lets treat this like a moba where we have a␣
 ↪goal, the goal of the game is to use a strategy to acquire USD via trading tkn␣
 ↪and eth. these are the gods/champions of the moba the strategies. each␣
 ↪strategy is nash equillibirum cannot change and they cannot be over tuned.

analyze these results as an expert defi strategist and data analyst to check for␣
 ↪accuracy of trades and results. this is a game theory experiment, where each␣
 ↪participants strategy is to acquire the most usd as possible based on a given␣
 ↪strategy in nash equillibrium. analyze the results are these results accurate?␣
 ↪the experiment is based on a univ2 initiation liquidity event. the strategies␣
 ↪are named by what they are supposed to do, are they accurately reflecting this?
 ↪ bear should be bear, bull should be bull, DCA should hold no token cause␣
 ↪using all eth to buy, harvester gets in early and then just harvests eth from␣
 ↪tkn, snipers go in early snipe supply and leave once in profit and dont␣
 ↪interact with tkn any more, sniper 2 is the same but lagging. evaluate the␣
 ↪strategies and the expected results. at this stage i want to complete the code␣
 ↪so we can focus on the results  but we are doing our last round of reviews to␣
 ↪check codes based on the results we have. what other errors based on the␣
 ↪results can you spot?


'''
```

[1619]: '\n12. Strategy-Specific Logic Errors (Hardest Fix)\nIssue: Some strategies may have logic that is not performing as intended, leading to unexpected results or ineffective trades.\nCorrection: Conduct a thorough review of each strategy's logic to ensure it aligns with its intended behavior. For example, Dove should consistently manage its risk without depleting its ETH, and Bull should maintain a feasible balance.\nLikely Function(s):\nStrategy.execute_trade()\nStrategy.check_triggers()\nStrategy.manage_balance()\n\nlets get back to the strategies, lets treat this like a moba where we have a goal, the goal of the game is to use a strategy to acquire USD via trading tkn and eth. these are the gods/champions of the moba the strategies. each strategy is nash equillibirum cannot change and they cannot be over tuned.\n\nanalyze these results as an expert defi strategist and data analyst to check for accuracy of trades and results. this is a game theory experiment, where each participants strategy is to acquire the most usd

```
as possible based on a given strategy in nash equillibrium. analyze the results
are these results accurate? the experiment is based on a univ2 initiation
liquidity event. the strategies are named by what they are supposed to do, are
they accurately reflecting this? bear should be bear, bull should be bull, DCA
should hold no token cause using all eth to buy, harvester gets in early and
then just harvests eth from tkn, snipers go in early snipe supply and leave once
in profit and dont interact with tkn any more, sniper 2 is the same but lagging.
evaluate the strategies and the expected results. at this stage i want to
complete the code so we can focus on the results  but we are doing our last
round of reviews to check codes based on the results we have. what other errors
based on the results can you spot?\n\n    \n'
```

[ ]:

[1620]:
```
'''

Sniper1 and Sniper2:

Expected Behavior:
Sniper1: Should enter early, acquire a large TKN position at a low price, and␣
 ↪exit once profitable.
Sniper2: Similar to Sniper1 but with a delayed entry.
Observed Behavior:
Sniper1 acquired a significant amount of TKN early but didn't sell it, leading␣
 ↪to a substantial unrealized loss.
Sniper2 entered the market late, bought a smaller amount of TKN, and also did␣
 ↪not sell.
Issues:
Neither sniper strategy sold their TKN positions after buying, which contradicts␣
 ↪the expectation that they would "snipe" and exit upon reaching a profitable␣
 ↪point.
Fix:
Code Adjustment: Implement a sell condition based on a predefined profit margin.␣
 ↪After acquiring TKN, the strategy should monitor the price and sell once a␣
 ↪profit threshold (e.g., 10-20% gain) is reached.
Bull:

Expected Behavior:
The Bull strategy should aggressively buy TKN as prices fluctuate, aiming to␣
 ↪accumulate as much TKN as possible. However, it should also be ready to sell␣
 ↪if the market trends upward, maximizing profits.
Observed Behavior:
The Bull strategy consistently bought TKN, but the final ROI is highly negative.
Issues:
The strategy overextended by depleting its ETH balance without considering␣
 ↪market trends, leading to a significant loss.
Fix:
```

```
Code Adjustment: Implement a strategy where Bull initially acquires TKN, but␣
↪with conditions to sell when a certain profit margin is reached, similar to␣
↪the Sniper strategy but with more aggressive entry points. Introduce risk␣
↪management controls to limit ETH expenditure, such as a cap on ETH usage per␣
↪trade or a mechanism to stop buying when prices trend downwards significantly.
Bear:

Expected Behavior:
The Bear strategy should be cautious, acquiring a small amount of TKN at␣
↪carefully chosen points (e.g., when volatility is low or prices are perceived␣
↪to be at a peak) and then selling it when bearish signals (e.g., moving␣
↪average crossover or significant price drops) appear.
Observed Behavior:
The Bear strategy didn't sell any TKN, resulting in no trades and a flat ROI.
Issues:
The strategy failed to acquire TKN, leading to no opportunities for selling or␣
↪executing a bearish strategy.
Fix:
Code Adjustment: Implement a strategy where Bear acquires a small amount of TKN␣
↪under specific conditions (e.g., low volatility) and sells when bearish␣
↪signals appear. This aligns with the opposite of the Bull strategy, where Bear␣
↪is defensive and looks to capitalize on downturns after acquiring TKN.
DCA (Dollar-Cost Averaging):

Expected Behavior:
The DCA strategy should periodically buy TKN in small increments to average the␣
↪purchase price over time.
Observed Behavior:
The DCA strategy accumulated TKN over time but ended with a negative ROI.
Issues:
The strategy correctly followed its DCA logic but suffered from buying during a␣
↪generally declining market.
Fix:
Strategy Enhancement: Consider adding an option to pause DCA purchases during␣
↪sharp market downturns to avoid consistently buying into losses. Implement a␣
↪trigger to resume DCA purchases when market conditions stabilize or improve.
Harvester:

Expected Behavior:
The Harvester should get in early, buy TKN, and then regularly sell small␣
↪portions as prices rise, harvesting profits.
Observed Behavior:
The Harvester strategy performed well with a positive ROI, reflecting expected␣
↪behavior.
Issues:
None.
```

```
Fix:
None Needed: The Harvester strategy is functioning correctly.
Dove:

Expected Behavior:
The Dove should be conservative, focusing on maintaining its initial capital.
Observed Behavior:
The Dove strategy made no trades, resulting in a flat ROI.
Issues:
None, as it preserved capital as expected.
Fix:
None Needed: The Dove strategy is functioning correctly.
Hawk:

Expected Behavior:
The Hawk should be aggressive, entering and exiting trades quickly to capitalize␣
↪on short-term price movements.
Observed Behavior:
The Hawk strategy performed exceptionally well, achieving a high ROI.
Issues:
None.
Fix:
None Needed: The Hawk strategy is functioning correctly.

'''
```

[1620]:
```
'\n\nSniper1 and Sniper2:\n\nExpected Behavior:\nSniper1: Should enter early,
acquire a large TKN position at a low price, and exit once profitable.\nSniper2:
Similar to Sniper1 but with a delayed entry.\nObserved Behavior:\nSniper1
acquired a significant amount of TKN early but didn\'t sell it, leading to a
substantial unrealized loss.\nSniper2 entered the market late, bought a smaller
amount of TKN, and also did not sell.\nIssues:\nNeither sniper strategy sold
their TKN positions after buying, which contradicts the expectation that they
would "snipe" and exit upon reaching a profitable point.\nFix:\nCode Adjustment:
Implement a sell condition based on a predefined profit margin. After acquiring
TKN, the strategy should monitor the price and sell once a profit threshold
(e.g., 10-20% gain) is reached.\nBull:\n\nExpected Behavior:\nThe Bull strategy
should aggressively buy TKN as prices fluctuate, aiming to accumulate as much
TKN as possible. However, it should also be ready to sell if the market trends
upward, maximizing profits.\nObserved Behavior:\nThe Bull strategy consistently
bought TKN, but the final ROI is highly negative.\nIssues:\nThe strategy
overextended by depleting its ETH balance without considering market trends,
leading to a significant loss.\nFix:\nCode Adjustment: Implement a strategy
where Bull initially acquires TKN, but with conditions to sell when a certain
profit margin is reached, similar to the Sniper strategy but with more
aggressive entry points. Introduce risk management controls to limit ETH
expenditure, such as a cap on ETH usage per trade or a mechanism to stop buying
```

```
when prices trend downwards significantly.\nBear:\n\nExpected Behavior:\nThe
Bear strategy should be cautious, acquiring a small amount of TKN at carefully
chosen points (e.g., when volatility is low or prices are perceived to be at a
peak) and then selling it when bearish signals (e.g., moving average crossover
or significant price drops) appear.\nObserved Behavior:\nThe Bear strategy
didn\'t sell any TKN, resulting in no trades and a flat ROI.\nIssues:\nThe
strategy failed to acquire TKN, leading to no opportunities for selling or
executing a bearish strategy.\nFix:\nCode Adjustment: Implement a strategy where
Bear acquires a small amount of TKN under specific conditions (e.g., low
volatility) and sells when bearish signals appear. This aligns with the opposite
of the Bull strategy, where Bear is defensive and looks to capitalize on
downturns after acquiring TKN.\nDCA (Dollar-Cost Averaging):\n\nExpected
Behavior:\nThe DCA strategy should periodically buy TKN in small increments to
average the purchase price over time.\nObserved Behavior:\nThe DCA strategy
accumulated TKN over time but ended with a negative ROI.\nIssues:\nThe strategy
correctly followed its DCA logic but suffered from buying during a generally
declining market.\nFix:\nStrategy Enhancement: Consider adding an option to
pause DCA purchases during sharp market downturns to avoid consistently buying
into losses. Implement a trigger to resume DCA purchases when market conditions
stabilize or improve.\nHarvester:\n\nExpected Behavior:\nThe Harvester should
get in early, buy TKN, and then regularly sell small portions as prices rise,
harvesting profits.\nObserved Behavior:\nThe Harvester strategy performed well
with a positive ROI, reflecting expected behavior.\nIssues:\nNone.\nFix:\nNone
Needed: The Harvester strategy is functioning correctly.\nDove:\n\nExpected
Behavior:\nThe Dove should be conservative, focusing on maintaining its initial
capital.\nObserved Behavior:\nThe Dove strategy made no trades, resulting in a
flat ROI.\nIssues:\nNone, as it preserved capital as expected.\nFix:\nNone
Needed: The Dove strategy is functioning correctly.\nHawk:\n\nExpected
Behavior:\nThe Hawk should be aggressive, entering and exiting trades quickly to
capitalize on short-term price movements.\nObserved Behavior:\nThe Hawk strategy
performed exceptionally well, achieving a high ROI.\nIssues:\nNone.\nFix:\nNone
Needed: The Hawk strategy is functioning correctly.\n\n'
```

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[1621]:
```
'''
```

```
It seems that the Sniper strategies might be sniping too much of the token (TKN)␣
 ↪supply early on, which could be contributing to the issue where they end up␣
 ↪holding large amounts of TKN instead of converting it back to ETH as intended.␣
 ↪Here's a deeper analysis of why this might be happening:

Potential Issues with the Sniper Strategies
Over-Accumulation of TKN:

The Sniper strategies might be buying a substantial amount of TKN when the pool␣
 ↪is still small, significantly impacting the pool's reserves. This large␣
 ↪purchase can cause a steep increase in the TKN price, which then makes it␣
 ↪difficult for the strategy to find a profitable exit point because the pool␣
 ↪dynamics have changed drastically.
Result: The Sniper strategies hold a significant amount of TKN, but due to the␣
 ↪large price impact and the nature of the constant product formula, the price␣
 ↪does not increase enough afterward to meet the profit targets, leaving the␣
 ↪strategies stuck with TKN.
Unrealistic Price Impact Assumptions:

The Uniswap-like price impact calculation in your TradingPool class might not be␣
 ↪accurately reflecting how large trades would affect the market in a real-world␣
 ↪scenario. If the price impact is underestimated, the strategies might think␣
 ↪they can buy and sell large amounts of TKN without much slippage, leading to␣
 ↪unrealistic trading behavior.
Impact: This could mean that after the Sniper strategies purchase a large␣
 ↪portion of TKN, they face a much worse market condition than anticipated,␣
 ↪making it hard to sell off their holdings at a profit.
Adjustments to Consider
Limit the Amount of TKN Purchased:

To prevent the Sniper strategies from over-accumulating TKN, you could introduce␣
 ↪a cap on the percentage of TKN they can purchase in a single transaction. This␣
 ↪would reduce the immediate price impact and help maintain a more stable market␣
 ↪for subsequent trades.
Implementation: Introduce logic in the Sniper1Strategy and Sniper2Strategy␣
 ↪classes to limit the amount of TKN purchased to, for example, 10-20% of the␣
 ↪pool's TKN reserves.
Refine Price Impact Calculation:

Improve the calculate_price_impact method to better simulate realistic market␣
 ↪conditions. This might involve introducing more sophisticated slippage models␣
 ↪or using a larger slippage factor to account for the impact of large trades␣
 ↪more accurately.
Example: Increase the slippage factor in the swap method or use a dynamic␣
 ↪slippage model that increases with trade size.
Adjust Profit Targets:
```

```
As previously discussed, lowering the profit targets might help the Sniper␣
 ↪strategies find more opportunities to exit their positions. Given that the␣
 ↪strategies might be dealing with less favorable market conditions after their␣
 ↪large initial buys, a smaller profit margin could be more achievable.
Suggestion: Consider lowering the target profit from 10% to something like 5% or␣
 ↪even 3%.
Simulate Real-World Liquidity:

If possible, model the liquidity pool more closely after real-world conditions,␣
 ↪where large purchases dramatically increase slippage, and subsequent price␣
 ↪movements are less predictable. This would give a more accurate reflection of␣
 ↪how such strategies would perform in practice.
Approach: Introduce liquidity shocks or external market influences in the␣
 ↪simulation to mimic real-world trading environments.
Conclusion
The issue with the Sniper strategies appears to be twofold: they are buying too␣
 ↪much of the supply at once, which is distorting the market, and the price␣
 ↪impact model may not be accurately representing the consequences of such large␣
 ↪trades. By limiting the amount of TKN purchased, refining the price impact␣
 ↪calculations, and adjusting profit targets, you can likely get the Sniper␣
 ↪strategies to behave more realistically and avoid ending up with large,␣
 ↪unsellable TKN holdings.

'''
```

[1621]: "\nIt seems that the Sniper strategies might be sniping too much of the token (TKN) supply early on, which could be contributing to the issue where they end up holding large amounts of TKN instead of converting it back to ETH as intended. Here's a deeper analysis of why this might be happening:\n\nPotential Issues with the Sniper Strategies\nOver-Accumulation of TKN:\n\nThe Sniper strategies might be buying a substantial amount of TKN when the pool is still small, significantly impacting the pool's reserves. This large purchase can cause a steep increase in the TKN price, which then makes it difficult for the strategy to find a profitable exit point because the pool dynamics have changed drastically.\nResult: The Sniper strategies hold a significant amount of TKN, but due to the large price impact and the nature of the constant product formula, the price does not increase enough afterward to meet the profit targets, leaving the strategies stuck with TKN.\nUnrealistic Price Impact Assumptions:\n\nThe Uniswap-like price impact calculation in your TradingPool class might not be accurately reflecting how large trades would affect the market in a real-world scenario. If the price impact is underestimated, the strategies might think they can buy and sell large amounts of TKN without much slippage, leading to unrealistic trading behavior.\nImpact: This could mean that after the Sniper strategies purchase a large portion of TKN, they face a much worse market condition than anticipated, making it hard to sell off their holdings at a profit.\nAdjustments to Consider\nLimit the Amount of TKN

Purchased:\n\nTo prevent the Sniper strategies from over-accumulating TKN, you could introduce a cap on the percentage of TKN they can purchase in a single transaction. This would reduce the immediate price impact and help maintain a more stable market for subsequent trades.\nImplementation: Introduce logic in the Sniper1Strategy and Sniper2Strategy classes to limit the amount of TKN purchased to, for example, 10-20% of the pool's TKN reserves.\nRefine Price Impact Calculation:\n\nImprove the calculate_price_impact method to better simulate realistic market conditions. This might involve introducing more sophisticated slippage models or using a larger slippage factor to account for the impact of large trades more accurately.\nExample: Increase the slippage factor in the swap method or use a dynamic slippage model that increases with trade size.\nAdjust Profit Targets:\n\nAs previously discussed, lowering the profit targets might help the Sniper strategies find more opportunities to exit their positions. Given that the strategies might be dealing with less favorable market conditions after their large initial buys, a smaller profit margin could be more achievable.\nSuggestion: Consider lowering the target profit from 10% to something like 5% or even 3%.\nSimulate Real-World Liquidity:\n\nIf possible, model the liquidity pool more closely after real-world conditions, where large purchases dramatically increase slippage, and subsequent price movements are less predictable. This would give a more accurate reflection of how such strategies would perform in practice.\nApproach: Introduce liquidity shocks or external market influences in the simulation to mimic real-world trading environments.\nConclusion\nThe issue with the Sniper strategies appears to be twofold: they are buying too much of the supply at once, which is distorting the market, and the price impact model may not be accurately representing the consequences of such large trades. By limiting the amount of TKN purchased, refining the price impact calculations, and adjusting profit targets, you can likely get the Sniper strategies to behave more realistically and avoid ending up with large, unsellable TKN holdings.\n\n"

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]:

[ ]: