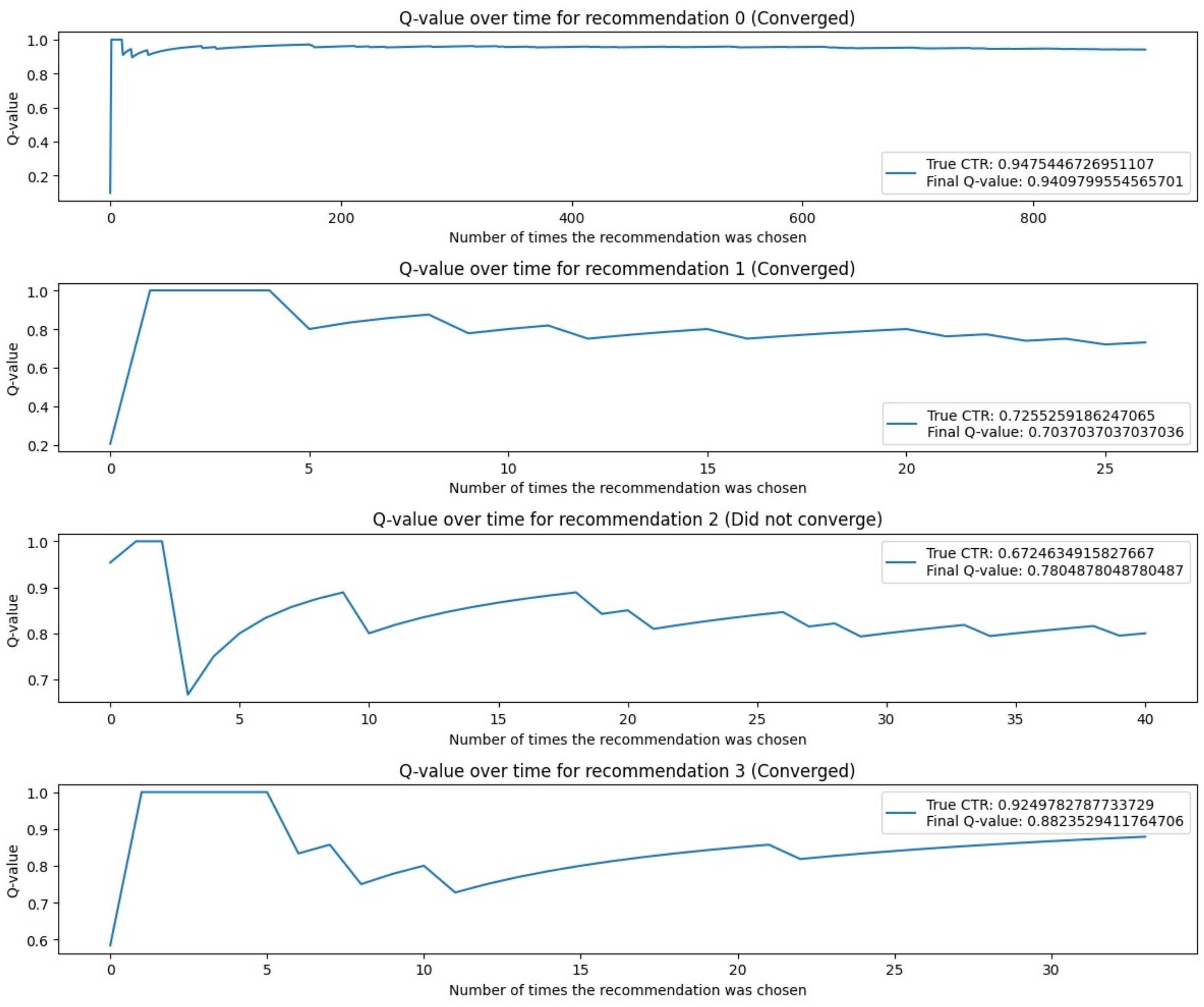
CIA - 1

Name: Purusharth Malik

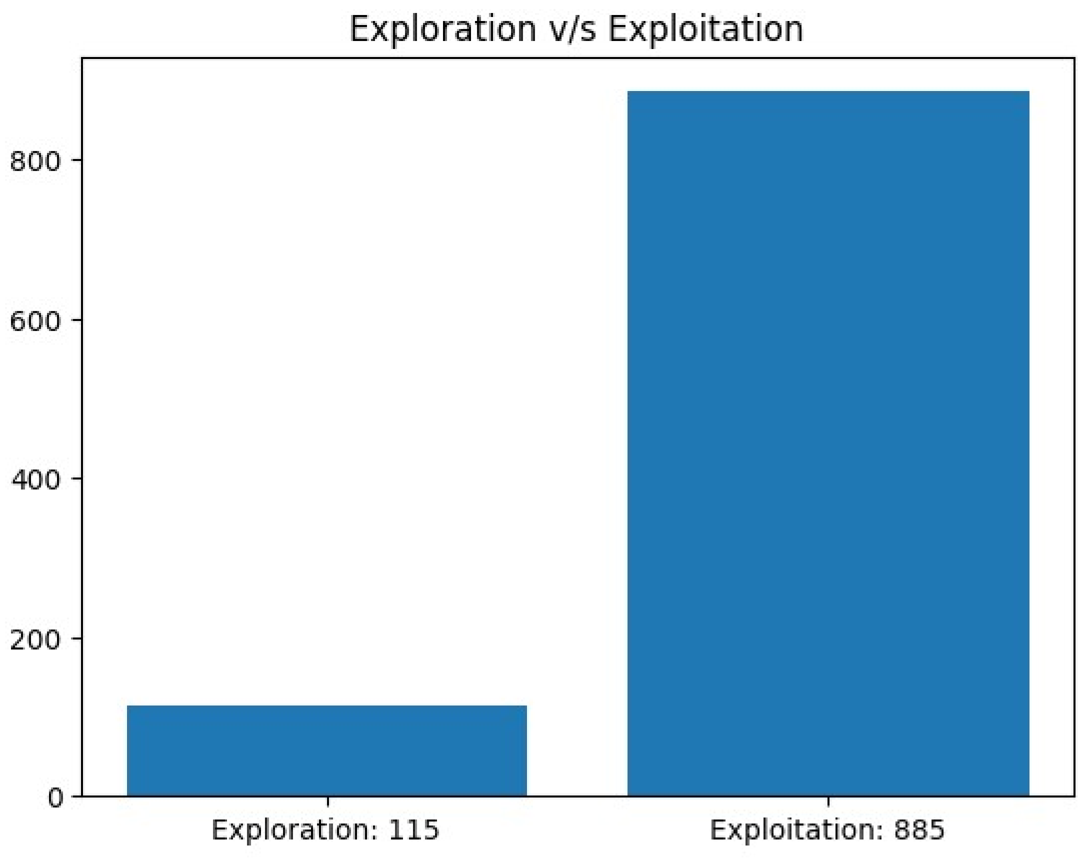
Registration No.: 2348542

In [65]:
```
import numpy as np
import matplotlib.pyplot as plt

# Creating a class for the bandit
class Bandit:
    def __init__(self, n_arms, epsilon):
        self.n_arms = n_arms
        self.epsilon = epsilon
        self.q_value = np.random.rand(n_arms)
        self.count = np.zeros(n_arms)
        self.values_over_time = {
                                    0: [],
                                    1: [],
                                    2: [],
                                    3: []
                                }
        self.exploration = 0

    # Function to select an arm using epsilon-greedy
    def selection(self):
        if np.random.random() < self.epsilon:
            self.exploration += 1
            return np.random.randint(self.n_arms)
        else:
            return np.argmax(self.q_value)

    # Updating the Q-value
    def update(self, arm, reward):
        self.count[arm] += 1
        self.values_over_time[arm].append(self.q_value[arm])
        self.q_value[arm] += (reward - self.q_value[arm]) / self.count[arm]

arms = 4
true_ctr = np.random.random(4)
epsilon = 0.1
n = 1000

# Creating an object for the bandit
obj = Bandit(arms, epsilon)
total_clicks = 0

# Deploying the algorithm
for i in range(n):
    # Choosing an arm
    chosen_arm = obj.selection()
    # Reward will be binary in nature
    reward = np.random.binomial(1, true_ctr[chosen_arm])
    # Updating the estimated value
    obj.update(chosen_arm, reward)
    total_clicks += reward
```

```
# Plotting the q-values
fig, axs = plt.subplots(4, figsize=(12, 10))
for i in range(arms):
    q = obj.values_over_time[i]
    axs[i].plot(range(len(q)), q,
                label=f"True CTR: {true_ctr[i]}\nFinal Q-value: {obj.q_value[i]}")
    # Checking for convergence
    if abs(true_ctr[i] - obj.q_value[i]) < 0.05:
        axs[i].set_title(f"Q-value over time for recommendation {i} (Converged)")
    else:
        axs[i].set_title(f"Q-value over time for recommendation {i} (Did not converge)")
    axs[i].set(xlabel="Number of times the recommendation was chosen",
               ylabel="Q-value")
    axs[i].legend()
    plt.tight_layout()
plt.show()

print(f"Total Clicks after {n} recommendations: {total_clicks}")
for i in range(arms):
    print(f"Recommendation {i}:\nChosen: {int(obj.count[i])} times\nTrue CTR: {true_ctr[i]}\nCli

# Plotting exploration v/s exploitation
plt.bar([f'Exploration: {obj.exploration}', f'Exploitation: {n - obj.exploration}'],
        [obj.exploration, n-obj.exploration])
plt.title("Exploration v/s Exploitation")
plt.show()
```



```
Total Clicks after 1000 recommendations: 926
Recommendation 0:
Chosen: 898 times
True CTR: 0.9475446726951107
Clicks: 0.9409799554565701


Recommendation 1:
Chosen: 27 times
True CTR: 0.7255259186247065
Clicks: 0.7037037037037036


Recommendation 2:
Chosen: 41 times
True CTR: 0.6724634915827667
Clicks: 0.7804878048780487


Recommendation 3:
Chosen: 34 times
True CTR: 0.9249782787733729
Clicks: 0.8823529411764706
```



End