In [4]:
```
import pandas as pd

# Load the Excel file
df = pd.read_excel(r"C:\Users\ASUS\Downloads\Synthetic Agent Data (1).xlsx")

# Display the first few rows to understand the structure
df.head()
```

Out[4]:

	Unnamed: 0	Agent 0	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5	Agent 6	Agent 7	Agent 8	...	Age 11
0	0	1	2	1	0	1	1	0	0	0	...	
1	1	9	10	8	2	15	7	0	2	2	...	
2	2	9	7	11	3	10	7	0	1	3	...	
3	3	11	14	13	0	8	9	0	0	3	...	
4	4	11	13	7	1	7	8	1	0	4	...	

5 rows × 1201 columns

In [5]:
```
#Dropping the 'unnamed column: 0' and removing the column from the original
df.drop(columns = 'Unnamed: 0', axis=1, inplace=True)
```

In [6]:
```
df
```

Out[6]:

	Agent 0	Agent 1	Agent 2	Agent 3	Agent 4	Agent 5	Agent 6	Agent 7	Agent 8	Agent 9	...	Agent 1190
0	1	2	1	0	1	1	0	0	0	2	...	0
1	9	10	8	2	15	7	0	2	2	11	...	0
2	9	7	11	3	10	7	0	1	3	11	...	3
3	11	14	13	0	8	9	0	0	3	14	...	1
4	11	13	7	1	7	8	1	0	4	13	...	2
...	...	...	...	...	...	...	...	...	...	...	...	...
79	8	8	7	1	10	8	0	1	2	8	...	0
80	7	11	7	1	7	7	3	2	3	8	...	0
81	11	7	9	1	13	10	0	1	3	8	...	1
82	9	9	11	0	7	7	0	0	2	7	...	2
83	2	2	2	0	2	2	0	0	0	1	...	0

84 rows × 1200 columns

In [10]:
```
#Question 1 #What are the mean, median, maximum, and minimum number of tasks comple
#What is the standard deviation of their number of tasks completed?
#Please also calculate the same statistics (mean, median, max, and minimum) for the

# Calculate statistics for each agent
agent_stats = df.describe().transpose()
agent_stats
```

Out[10]:

	count	mean	std	min	25%	50%	75%	max
Agent 0	84.0	6.940476	3.816108	1.0	2.0	7.0	10.00	14.0
Agent 1	84.0	6.702381	3.867628	1.0	2.0	7.0	9.00	15.0
Agent 2	84.0	7.000000	4.032996	1.0	2.0	7.5	9.00	15.0
Agent 3	84.0	0.535714	0.870486	0.0	0.0	0.0	1.00	3.0
Agent 4	84.0	7.297619	4.236331	1.0	2.0	8.0	10.00	15.0
...	...	...	...	...	...	...	...	...
Agent 1195	84.0	0.309524	0.559177	0.0	0.0	0.0	1.00	2.0
Agent 1196	84.0	2.083333	1.716036	0.0	0.0	2.0	3.00	6.0
Agent 1197	84.0	1.714286	1.923673	0.0	0.0	1.0	3.00	8.0
Agent 1198	84.0	7.285714	4.249937	1.0	2.0	7.0	10.25	15.0
Agent 1199	84.0	0.595238	0.879746	0.0	0.0	0.0	1.00	3.0

1200 rows × 8 columns

In [12]:
```
# Calculate statistics for all agents combined
combined_stats = {
    'mean': df.values.mean(),
    'median': pd.Series(df.values.flatten()).median(),
    'max': df.values.max(),
    'min': df.values.min(),
    'std': df.values.std()
}
combined_stats
```

Out[12]:
```
{'mean': 3.5428571428571427,
 'median': 1.0,
 'max': 24,
 'min': 0,
 'std': 4.230781543835055}
```

In [14]:
```
#Question 2: Not all agents are the same, some are driven by decidedly different fa
#From the data, what can you say about the different types of agents?


from sklearn.cluster import KMeans

# Prepare data for clustering (e.g., use mean and std of tasks completed)
#X = df.describe().transpose()[['mean', 'std']]

# Perform K-means clustering
kmeans = KMeans(n_clusters=3)
agent_stats['Cluster'] = kmeans.fit_predict(agent_stats)
```

```
# Add cluster labels to the data
print(agent_stats)
```

```
C:\Users\ASUS\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:870: FutureWarn
ing: The default value of `n_init` will change from 10 to 'auto' in 1.4. Set the val
ue of `n_init` explicitly to suppress the warning
  warnings.warn(
C:\Users\ASUS\anaconda3\Lib\site-packages\sklearn\cluster\_kmeans.py:1382: UserWarni
ng: KMeans is known to have a memory leak on Windows with MKL, when there are less c
hunks than available threads. You can avoid it by setting the environment variable O
MP_NUM_THREADS=5.
  warnings.warn(
```

	count	mean	std	min	25%	50%	75%	max	Cluster
Agent 0	84.0	6.940476	3.816108	1.0	2.0	7.0	10.00	14.0	1
Agent 1	84.0	6.702381	3.867628	1.0	2.0	7.0	9.00	15.0	1
Agent 2	84.0	7.000000	4.032996	1.0	2.0	7.5	9.00	15.0	1
Agent 3	84.0	0.535714	0.870486	0.0	0.0	0.0	1.00	3.0	0
Agent 4	84.0	7.297619	4.236331	1.0	2.0	8.0	10.00	15.0	1
...	...	...	...	...	...	...	...	...	...
Agent 1195	84.0	0.309524	0.559177	0.0	0.0	0.0	1.00	2.0	0
Agent 1196	84.0	2.083333	1.716036	0.0	0.0	2.0	3.00	6.0	2
Agent 1197	84.0	1.714286	1.923673	0.0	0.0	1.0	3.00	8.0	2
Agent 1198	84.0	7.285714	4.249937	1.0	2.0	7.0	10.25	15.0	1
Agent 1199	84.0	0.595238	0.879746	0.0	0.0	0.0	1.00	3.0	0

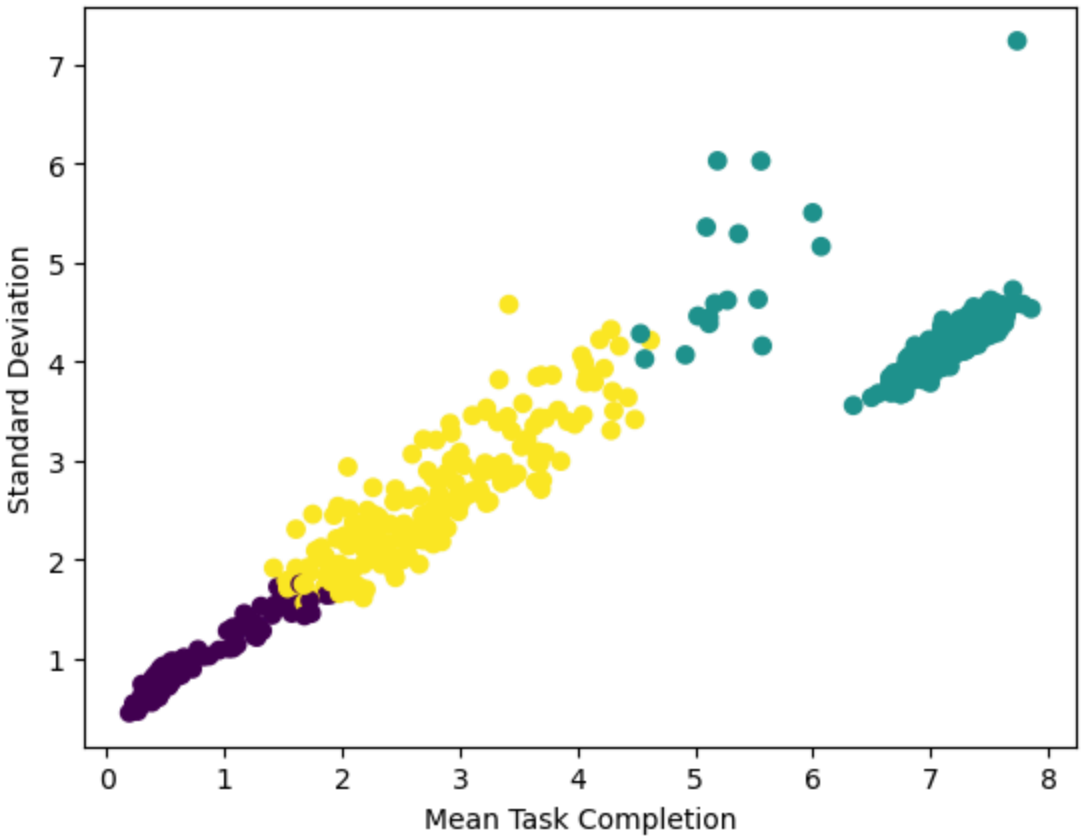
[1200 rows x 9 columns]

In [18]:
```
# Cluster Summary
cluster_summary = agent_stats.groupby('Cluster').mean()
print(cluster_summary)
```

Cluster	count	mean	std	min	25%	50%	75%	max
0	84.0	0.515381	0.811575	0.000000	0.000000	0.072519	1.017653	3.074427
1	84.0	7.093262	4.183311	0.965164	1.929816	7.404713	9.887807	14.948770
2	84.0	2.765198	2.606393	0.000000	0.000000	2.473404	4.498670	8.952128

In [16]:
```
#Use matplotlib to plot the clusters and interpret the different agent personas
import matplotlib.pyplot as plt

plt.scatter(agent_stats['mean'], agent_stats['std'], c=agent_stats['Cluster'])
plt.xlabel('Mean Task Completion')
plt.ylabel('Standard Deviation')
plt.show()
```



In [ ]:
```
#For each cluster, summarize the characteristics as their persona:
     #High Performers: Cluster 2 with High mean, low standard deviation indicates c
    #Inconsistent Performers: Cluster 1 with High standard deviation might represen
    #Low Performers: Cluster 0 with Low mean and low standard deviation might indic

#What contributed to their different persona?
   #Experience and Skill Level: More experienced agents often perform better due to
   #Work Environment: The work environment, including the quality of tools, support
                      #A supportive and well-equipped environment can boost product
   #Personal Factors: Individual differences such as health, stress levels, and per
   #Incentives and Rewards: Agents who are aware of and are motivated by the incent
   #incentives align with their goals and aspirations.
```

In [ ]:
```
#Question 3: We would like to design a compensation structure for the agents that s
            #but are not quite the best yet. Really, we're trying to motivate them
            #We'd like the total average compensation for a task to be at most 300K
            #We'd like every task to have a minimum payment of at least 175KSH so a

#Please design an incentive structure for agents in the form of a per task rate and
#You can create a single flat bonus, multiple tiers of incentives, or anything that
#Your solution will be judged based on how well it met the financial criteria, your
#and the ease of which the incentives can be explained to agents.
```

In [32]:
```
# Define base rate and tiers

BASE_RATE = 250  # Base payment per task
BONUS_TIERS = {  # Weekly bonus structure
    'Tier 1': (20, 30, 500),  # Range: 20-30 tasks, Bonus: 500 KSH
    'Tier 2': (31, 40, 1000),  # Range: 31-40 tasks, Bonus: 1000 KSH
```

```
    'Tier 3': (41, float('inf'), 2000)  # 41+ tasks, Bonus: 2000 KSH
}

def calculate_bonus(tasks_completed):
    """
    This function returns the bonus based on tasks completed.
    """
    for tier, (min_tasks, max_tasks, bonus) in BONUS_TIERS.items():
        if min_tasks <= tasks_completed <= max_tasks:
            return bonus
    return 0  # No bonus if tasks are below 20

def calculate_total_compensation(tasks_completed):
    """
    This function calculates the total compensation including base pay and bonus.
    """
    # Base pay is simply the number of tasks completed times the base rate
    base_pay = tasks_completed * BASE_RATE

    # Calculate bonus
    bonus = calculate_bonus(tasks_completed)

    # Total compensation is base pay + bonus
    total_compensation = base_pay + bonus

    return total_compensation

def calculate_average_pay_per_task(tasks_completed):
    """
    This function calculates the average pay per task.
    """
    if tasks_completed == 0:
        return 0  # Avoid division by zero

    # Total compensation
    total_compensation = calculate_total_compensation(tasks_completed)

    # Average pay per task
    average_pay_per_task = total_compensation / tasks_completed

    return average_pay_per_task

# Example usage:
tasks_completed = int(input("Enter the number of tasks completed: "))

# Calculate total compensation and average pay per task
total_compensation = calculate_total_compensation(tasks_completed)
average_pay = calculate_average_pay_per_task(tasks_completed)

print(f"Total Compensation: {total_compensation} KSH")
print(f"Average Pay Per Task: {average_pay:.2f} KSH")
```

```
Total Compensation: 5500 KSH
Average Pay Per Task: 275.00 KSH
```

In [ ]:

In [ ]: