In [2]:
```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

# Load the datasets
logs = pd.read_csv(r'C:\Users\junai\OneDrive - Middlesex University\Applied Data Ar
scores = pd.read_csv(r'C:\Users\junai\OneDrive - Middlesex University\Applied Data
```

In [3]:
```python
print (logs.head(5))       #check the first 5 data of the dataset
```

```
  StudentId             Time         Type                   Action
0      72af  28/05/23, 10:51  User report  Grade user report viewed
1      72af  28/05/23, 10:51       System           Course viewed
2      c426  27/05/23, 15:53       System           Course viewed
3      0326  26/05/23, 22:22       System           Course viewed
4      8b7a  26/05/23, 21:52       System           Course viewed
```

In [4]:
```python
print (scores.head(5))
```

```
  StudentId Grade
0      c426   2nd
1      8de3   2nd
2      d969   2nd
3      6d29   1st
4      1dd9   1st
```

In [5]:
```python
print (logs.tail(5))          #check the bottom 5 data of the dataset
```

```
      StudentId             Time    Type                Action
83202      e2e7  12/09/22, 21:30  System         Course viewed
83203      e2e7  12/09/22, 21:17     URL  Course module viewed
83204      e2e7  12/09/22, 21:16  System         Course viewed
83205      e2e7  12/09/22, 21:16  System         Course viewed
83206      e2e7  12/09/22, 21:15  System         Course viewed
```

In [6]:
```python
print (scores.tail(5))
```

```
    StudentId Grade
100      9673   3rd
101      5867   3rd
102      8976   2nd
103      56fe  Fail
104      1d56   2nd
```

In [7]:
```python
# Data Exploration
# Summary Statistics
logs.describe()
# method generates a DataFrame that contains various statistical metrics for each r
```

Out[7]:

	StudentId	Time	Type	Action
count	83207	83207	83207	83207
unique	115	23377	17	47
top	d3e2	12/10/22, 14:52	Quiz	Course viewed
freq	1979	200	28418	25951

In [8]:
```python
scores.describe()
```

Out[8]:

	StudentId	Grade
count	105	105
unique	105	4
top	c426	3rd
freq	1	36

In [9]:
```
# Data Distribution
print(scores['Grade'].value_counts())         #To count the occurrences of values
```

```
Grade
3rd     36
2nd     35
Fail    18
1st     16
Name: count, dtype: int64
```

In [ ]:
```

```

In [10]:
```
# Missing Values
print(logs.isnull().sum())

```

```
StudentId    0
Time         0
Type         0
Action       0
dtype: int64
```

In [11]:
```
logs.isna()    # Returns a DataFrame or Series of boolean values,
#where True indicates a null value else False indicates no Null values
```

Out[11]:

	StudentId	Time	Type	Action
0	False	False	False	False
1	False	False	False	False
2	False	False	False	False
3	False	False	False	False
4	False	False	False	False
...	...	...	...	...
83202	False	False	False	False
83203	False	False	False	False
83204	False	False	False	False
83205	False	False	False	False
83206	False	False	False	False

83207 rows × 4 columns

In [12]:
```
print(scores.isnull().sum())
```

```
StudentId    0
Grade        0
dtype: int64
```

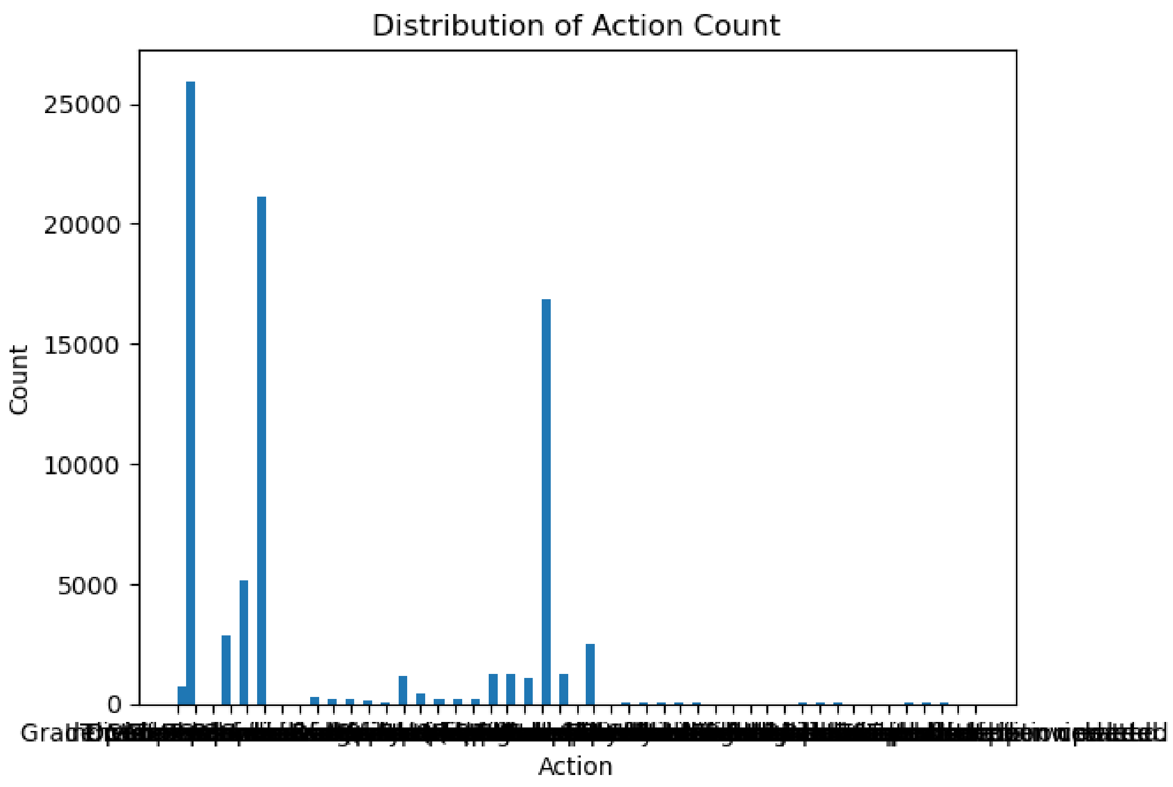
In [13]:
```
scores.isna()                              #Returns a DataFrame or Series of boolean va
```

Out[13]:

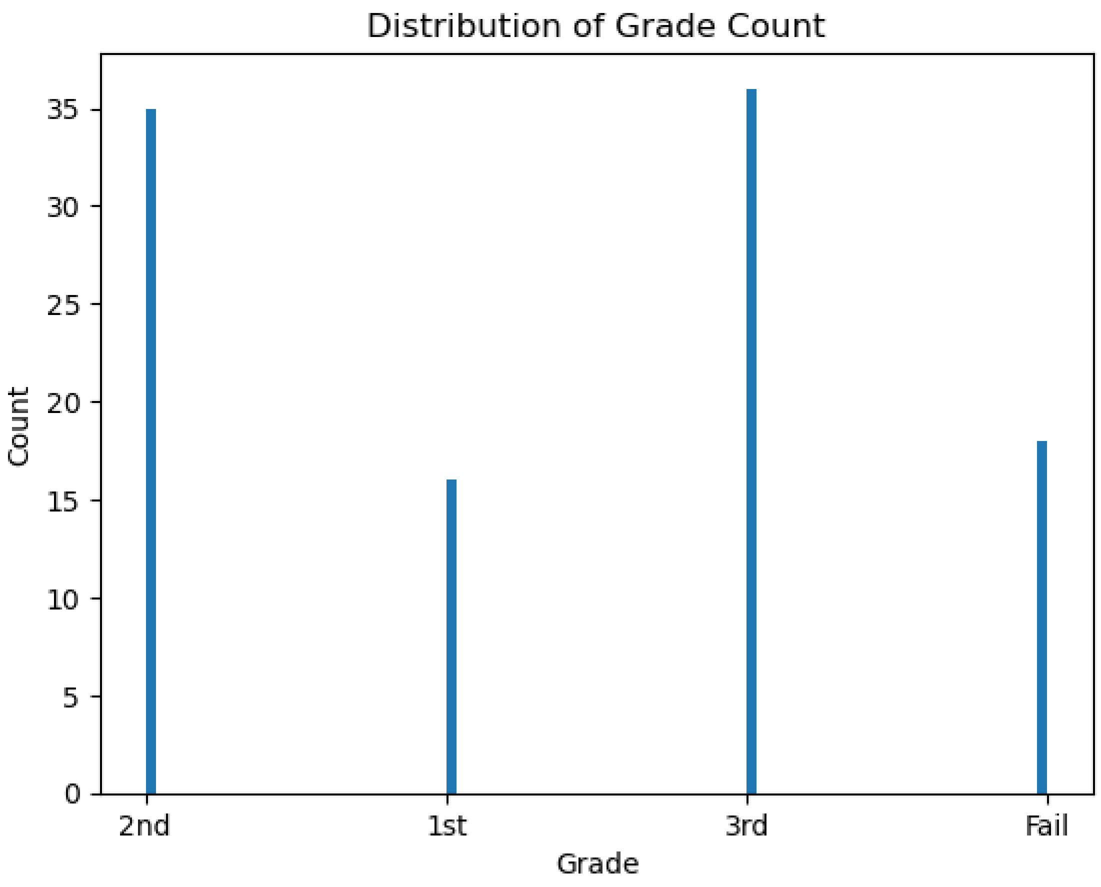
	StudentId	Grade
0	False	False
1	False	False
2	False	False
3	False	False
4	False	False
...	...	...
100	False	False
101	False	False
102	False	False
103	False	False
104	False	False

105 rows × 2 columns

In [44]:
```
# Plot histogram for a numerical feature (e.g., Action count)
plt.hist(logs['Action'], bins=90)     #takes a numerical array or series as its firs
                                     #bins=20 argument specifies that the range of v
plt.xlabel('Action')     #sets the label for the x-axis of the histogram
plt.ylabel('Count')      #sets the label for the y-axis, indicating that the values
plt.title('Distribution of Action Count')
plt.show()
```



In [43]:
```
# Plot histogram for a numerical feature (e.g., Grade)
plt.hist(scores['Grade'], bins=90)    # takes a numerical array or series as its fi
                                             #bins=20 argument specifies that th
plt.xlabel('Grade')     #sets the label for the x-axis of the histogram
plt.ylabel('Count')             #sets the label for the y-axis, indicating that the
plt.title('Distribution of Grade Count')
plt.show()
```



In [18]:
```
# Feature Engineering

# Time-Based Features
logs['Time'] = pd.to_datetime(logs['Time'])
#likely contains string representations of timestamps, into actual datetime objects
logs['DayOfWeek'] = logs['Time'].dt.dayofweek
#line extracts the day of the week from the 'Time' column. eg: Mon=0 & Sun=6
logs['HourOfDay'] = logs['Time'].dt.hour
#line extracts the hour component from the 'Time' column
```

```
C:\Users\junai\AppData\Local\Temp\ipykernel_21864\392715146.py:3: UserWarning: Cou
ld not infer format, so each element will be parsed individually, falling back to
`dateutil`. To ensure parsing is consistent and as-expected, please specify a form
at.
  logs['Time'] = pd.to_datetime(logs['Time'])
```

In [33]:
```
# Engagement Features

interaction_counts = logs.groupby('StudentId').size().reset_index(name='Interactior
#logs.groupby('studentid')subsequent operations will be applied separately for each
#This function calculates the number of records (or interactions) for each group of
#This resets the index of the resulting DataFrame and renames the calculated size c

time_spent = logs.groupby('StudentId')['Time'].apply(lambda x: (x.max() - x.min()).
#x.max() - x.min() calculates the time difference between the latest and earliest t
```

In [34]:
```
# Action-Specific Features
action_types = logs['Type'].unique()    # retrieves the unique values from the 'Typ
for action_type in action_types:        # retrieves the unique values from the 'Typ
    logs[f'Action_{action_type}'] = logs['Type'].apply(lambda x: 1 if x == action_t
    #line creates a new binary column in the 'logs' DataFrame
#checks if the 'Type' matches the current 'action_type'. If it does, it assigns a v


action_type_counts = logs.groupby('StudentId')[[f'Action_{action_type}' for action_
#generates a list of column names corresponding to the action-specific features
 #This sums up the binary values (1 or 0) for each action type within each group
```

In [35]:
```
# Merge engineered features with scores dataset
features = pd.merge(scores, interaction_counts, on='StudentId', how='left')
#line merges the 'scores' DataFrame with the 'interaction_counts' DataFrame based o
#how='left' argument specifies a left join, meaning that all the rows from the 'sco
#and matching rows from the 'interaction_counts' DataFrame will be merged based on

features = pd.merge(features, time_spent, on='StudentId', how='left')
#resulting DataFrame now includes the total time spent feature for each student

features = pd.merge(features, action_type_counts, on='StudentId', how='left')
#merges the 'features' DataFrame with the 'action_type_counts' DataFrame based on t
#The resulting DataFrame now includes the action-specific count features for each s
```

In [36]:
```
# Handle missing values if any
features.fillna(0, inplace=True)
#used to fill missing (NaN) values in the DataFrame with a specified value, in this
```

In [37]:
```
# Save the engineered features to a new CSV file
features.to_csv(r"C:\Users\junai\OneDrive - Middlesex University\Applied Data Analy
```

In [38]:
```
Check = pd.read_csv(r"C:\Users\junai\OneDrive - Middlesex University\Applied Data A
```

In [45]:
```
print (Check.head())
```

```
  StudentId Grade  InteractionCount  TotalTimeSpent  Action_User report  \
0      c426   2nd               374    16638.233333                   0   
1      8de3   2nd               295    13748.650000                   0   
2      d969   2nd               356    15862.383333                  13   
3      6d29   1st               194    15862.350000                   4   
4      1dd9   1st               261    15843.950000                   3   

   Action_System  Action_Open Grader  Action_Turnitin Assignment 2  \
0            145                   0                            82   
1             74                   0                            49   
2            112                   0                            23   
3             29                   0                            21   
4             64                   0                            35   

   Action_Kaltura Video Resource  Action_Quiz  ...  Action_Forum  \
0                              8           95  ...             7   
1                             26           85  ...             1   
2                             46          112  ...             3   
3                              0          132  ...             0   
4                              0          148  ...             0   

   Action_Scheduler  Action_Folder  Action_File  Action_Page  Action_URL  \
0                 0             18           12            1           1   
1                 0             46            8            0           2   
2                 0             23           12            0           1   
3                 0              4            0            0           0   
4                 0              7            2            0           0   

   Action_Assignment  Action_Overview report  Action_File submissions  \
0                  4                       0                        0   
1                  2                       0                        0   
2                  0                       8                        0   
3                  0                       1                        0   
4                  0                       2                        0   

   Action_User tours  
0                  0  
1                  2  
2                  3  
3                  3  
4                  0  

[5 rows x 21 columns]
```

In [46]:
```
Check.describe()
```

Out[46]:

	InteractionCount	TotalTimeSpent	Action_User report	Action_System	Action_Open Grader	Action_Turniti Assignment
count	105.000000	105.000000	105.000000	105.000000	105.000000	105.00000
mean	716.771429	15649.094127	6.190476	240.761905	0.066667	28.00952
std	453.099693	1165.778261	10.190583	169.978648	0.347150	22.88747
min	116.000000	9022.833333	0.000000	19.000000	0.000000	0.00000
25%	320.000000	15190.966667	0.000000	109.000000	0.000000	13.00000
50%	671.000000	15862.100000	2.000000	212.000000	0.000000	22.00000
75%	1028.000000	16570.133333	7.000000	354.000000	0.000000	38.00000
max	1979.000000	16638.233333	48.000000	814.000000	3.000000	132.00000

In [ ]: