CINEMA TICKETS

- The Cinema Tickets dataset, covering eight months in 2018, offers sales history and screening details from various cinemas. Ideal for predictive modeling, it supports forecasting, screening optimization, and ROI improvement. With anonymized locations, it aids decisions on cast, crew, and project planning.
- Emphasizing time series analysis, it provides insights for Cinema Clustering, sales forecasts, and movie genre recommendations. Evolving with additional movie data, it enhances strategic decision-making in the cinema industry.
- DOWNLOAD THE DATASET HERE

In [61]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import mean_absolute_error, mean_absolute_percentage_error, r2_score
warnings.filterwarnings('ignore')
```

In [62]:
```
# Load the dataset
df = pd.read_csv('cinemaTicket_Ref.csv')
df
```

Out[62]:

	film_code	cinema_code	total_sales	tickets_sold	tickets_out	show_time	occu_perc	ticket_price	ticket_use	capacity	date	mo
0	1492	304	3900000	26	0	4	4.26	150000.0	26	610.328638	2018-05-05	
1	1492	352	3360000	42	0	5	8.08	80000.0	42	519.801980	2018-05-05	
2	1492	489	2560000	32	0	4	20.00	80000.0	32	160.000000	2018-05-05	
3	1492	429	1200000	12	0	1	11.01	100000.0	12	108.991826	2018-05-05	
4	1492	524	1200000	15	0	3	16.67	80000.0	15	89.982004	2018-05-05	
...	...	...	...	...	...	...	...	...	...	...	...	
142519	1569	495	1320000	22	0	2	3.86	60000.0	22	569.948187	2018-11-04	
142520	1569	474	1200000	15	0	1	65.22	80000.0	15	22.999080	2018-11-04	
142521	1569	524	1060000	8	0	3	9.20	132500.0	8	86.956522	2018-11-04	
142522	1569	529	600000	5	0	2	5.00	120000.0	5	100.000000	2018-11-04	
142523	1569	486	250000	5	0	1	1.79	50000.0	5	279.329609	2018-11-04	

142524 rows × 14 columns

In [63]:
```
df.head()
```

Out[63]:

	film_code	cinema_code	total_sales	tickets_sold	tickets_out	show_time	occu_perc	ticket_price	ticket_use	capacity	date	month	c
0	1492	304	3900000	26	0	4	4.26	150000.0	26	610.328638	2018-05-05	5	
1	1492	352	3360000	42	0	5	8.08	80000.0	42	519.801980	2018-05-05	5	
2	1492	489	2560000	32	0	4	20.00	80000.0	32	160.000000	2018-05-05	5	
3	1492	429	1200000	12	0	1	11.01	100000.0	12	108.991826	2018-05-05	5	
4	1492	524	1200000	15	0	3	16.67	80000.0	15	89.982004	2018-05-05	5	

In [64]:
```
df.tail()
```

Out[64]:

	film_code	cinema_code	total_sales	tickets_sold	tickets_out	show_time	occu_perc	ticket_price	ticket_use	capacity	date	mo
142519	1569	495	1320000	22	0	2	3.86	60000.0	22	569.948187	2018-11-04	
142520	1569	474	1200000	15	0	1	65.22	80000.0	15	22.999080	2018-11-04	
142521	1569	524	1060000	8	0	3	9.20	132500.0	8	86.956522	2018-11-04	
142522	1569	529	600000	5	0	2	5.00	120000.0	5	100.000000	2018-11-04	
142523	1569	486	250000	5	0	1	1.79	50000.0	5	279.329609	2018-11-04	

In [65]:
```
df.columns
```

Out[65]:
```
Index(['film_code', 'cinema_code', 'total_sales', 'tickets_sold',
       'tickets_out', 'show_time', 'occu_perc', 'ticket_price', 'ticket_use',
       'capacity', 'date', 'month', 'quarter', 'day'],
      dtype='object')
```

In [66]:
```
#check shape of the datasets
print("Number of rows =",df.shape[0])
print("Number of column =",df.shape[1])
```

```
Number of rows = 142524
Number of column = 14
```

In [67]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 142524 entries, 0 to 142523
Data columns (total 14 columns):
 #   Column        Non-Null Count   Dtype  
---  ------        --------------   -----  
 0   film_code     142524 non-null  int64  
 1   cinema_code   142524 non-null  int64  
 2   total_sales   142524 non-null  int64  
 3   tickets_sold  142524 non-null  int64  
 4   tickets_out   142524 non-null  int64  
 5   show_time     142524 non-null  int64  
 6   occu_perc     142399 non-null  float64
 7   ticket_price  142524 non-null  float64
 8   ticket_use    142524 non-null  int64  
 9   capacity      142399 non-null  float64
 10  date          142524 non-null  object 
 11  month         142524 non-null  int64  
 12  quarter       142524 non-null  int64  
 13  day           142524 non-null  int64  
dtypes: float64(3), int64(10), object(1)
memory usage: 15.2+ MB
```

In [68]:
```
df.dtypes
```

Out[68]:
```
film_code         int64
cinema_code       int64
total_sales       int64
tickets_sold      int64
tickets_out       int64
show_time         int64
occu_perc       float64
ticket_price    float64
ticket_use        int64
capacity        float64
date             object
month             int64
quarter           int64
day               int64
dtype: object
```

In [69]:
```
#check null value of the datasets
pd.isnull(df).sum()
```

Out[69]:
```
film_code         0
cinema_code       0
total_sales       0
tickets_sold      0
tickets_out       0
show_time         0
occu_perc       125
ticket_price      0
ticket_use        0
capacity        125
date              0
month             0
quarter           0
day               0
dtype: int64
```

In [70]:
```
#Filling missing value
df['occu_perc']=df['occu_perc'].fillna(df['occu_perc'].mean())
df['capacity']=df['capacity'].fillna(df['capacity'].mean())
```

In [71]:
```
df.isnull().sum()
```

Out[71]:
```
film_code       0
cinema_code     0
total_sales     0
tickets_sold    0
tickets_out     0
show_time       0
occu_perc       0
ticket_price    0
ticket_use      0
capacity        0
date            0
month           0
quarter         0
day             0
dtype: int64
```

In [72]:
```
# Convert date column to datetime format
df['date'] = pd.to_datetime(df['date'])
df['year'] = df['date'].dt.year
df['month'] = df['date'].dt.month
df['day_of_week'] = df['date'].dt.dayofweek
df['is_weekend'] = df['day_of_week'].apply(lambda x: 1 if x >= 5 else 0)
```

In [73]:
```
df.head(4)
```

Out[73]:

	film_code	cinema_code	total_sales	tickets_sold	tickets_out	show_time	occu_perc	ticket_price	ticket_use	capacity	date	month
0	1492	304	3900000	26	0	4	4.26	150000.0	26	610.328638	2018-05-05	5
1	1492	352	3360000	42	0	5	8.08	80000.0	42	519.801980	2018-05-05	5
2	1492	489	2560000	32	0	4	20.00	80000.0	32	160.000000	2018-05-05	5
3	1492	429	1200000	12	0	1	11.01	100000.0	12	108.991826	2018-05-05	5

In [74]:
```
# Drop unnecessary columns
df = df.drop(['date', 'month', 'quarter', 'day'], axis=1)
```

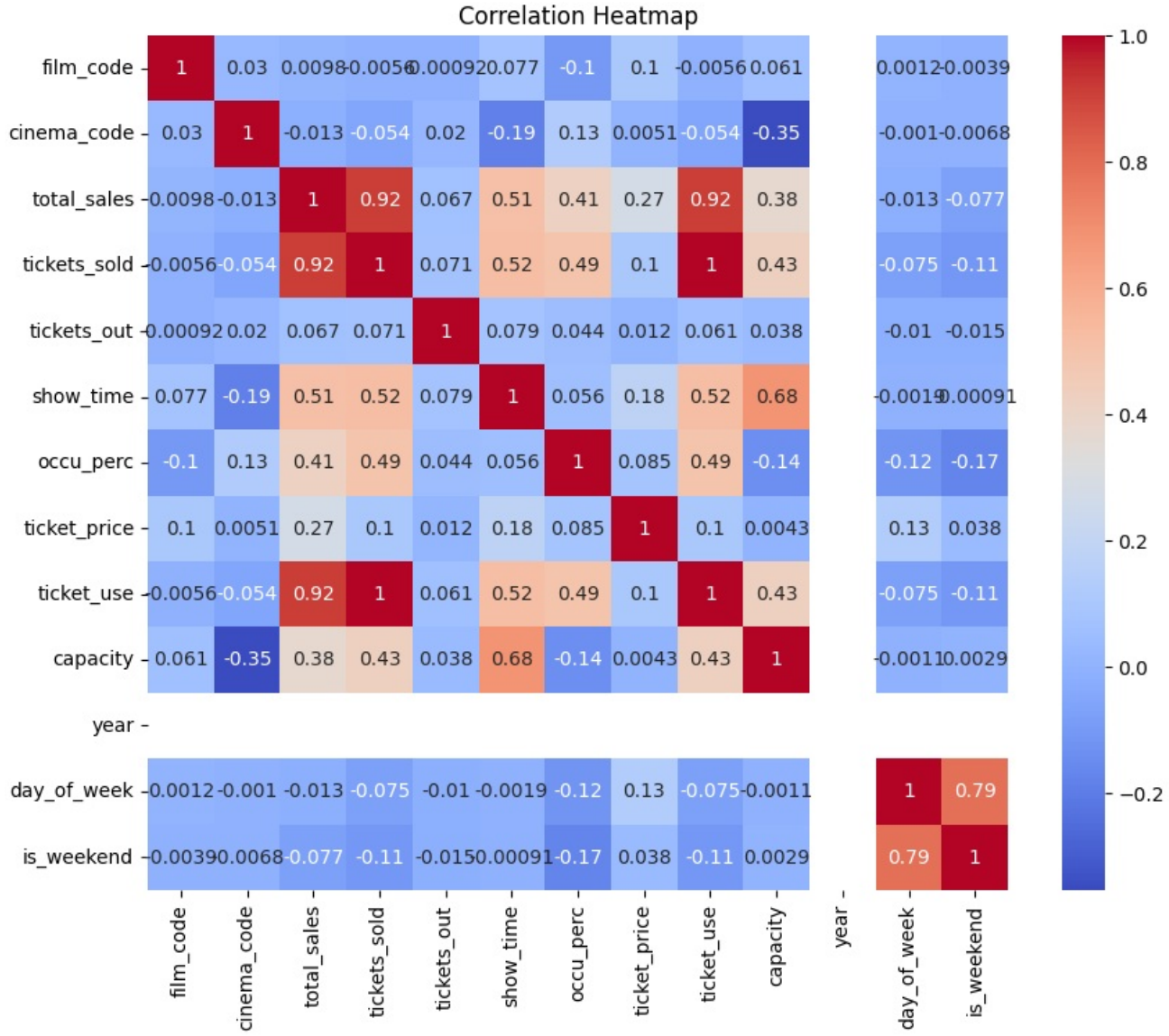
In [75]:
```
df.corr()
```

Out[75]:

	film_code	cinema_code	total_sales	tickets_sold	tickets_out	show_time	occu_perc	ticket_price	ticket_use	capacity	year
film_code	1.000000	0.029530	0.009802	-0.005588	-0.000920	0.076670	-0.104074	0.104820	-0.005582	0.061196	NaN
cinema_code	0.029530	1.000000	-0.012910	-0.053716	0.019534	-0.187125	0.125448	0.005079	-0.053957	-0.353136	NaN
total_sales	0.009802	-0.012910	1.000000	0.915176	0.066810	0.509812	0.413964	0.272450	0.915112	0.375722	NaN
tickets_sold	-0.005588	-0.053716	0.915176	1.000000	0.071498	0.522088	0.493403	0.103596	0.999946	0.425206	NaN
tickets_out	-0.000920	0.019534	0.066810	0.071498	1.000000	0.078860	0.044403	0.011916	0.061091	0.037596	NaN
show_time	0.076670	-0.187125	0.509812	0.522088	0.078860	1.000000	0.055603	0.176274	0.521625	0.681474	NaN
occu_perc	-0.104074	0.125448	0.413964	0.493403	0.044403	0.055603	1.000000	0.084631	0.493281	-0.141262	NaN
ticket_price	0.104820	0.005079	0.272450	0.103596	0.011916	0.176274	0.084631	1.000000	0.103544	0.004285	NaN
ticket_use	-0.005582	-0.053957	0.915112	0.999946	0.061091	0.521625	0.493281	0.103544	1.000000	0.425108	NaN
capacity	0.061196	-0.353136	0.375722	0.425206	0.037596	0.681474	-0.141262	0.004285	0.425108	1.000000	NaN
year	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
day_of_week	0.001211	-0.001010	-0.012581	-0.074746	-0.010408	-0.001857	-0.118716	0.132529	-0.074689	-0.001146	NaN
is_weekend	-0.003914	-0.006793	-0.076634	-0.106854	-0.014545	-0.000913	-0.170097	0.037524	-0.106776	0.002876	NaN

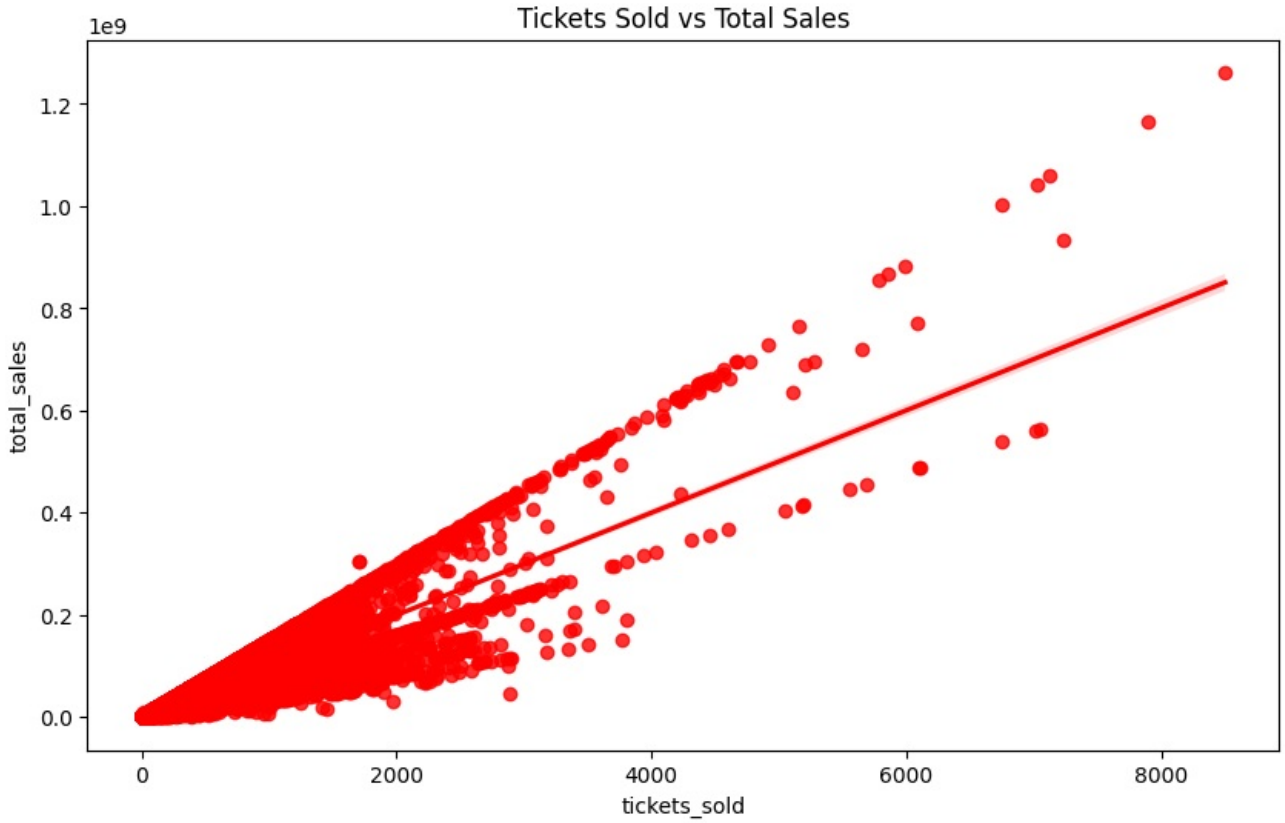
In [76]:

```
# Exploratory Data Analysis (EDA)
plt.figure(figsize=(10, 8))
sns.heatmap(df.corr(), annot=True, cmap='coolwarm')
plt.title('Correlation Heatmap')
plt.show()
```

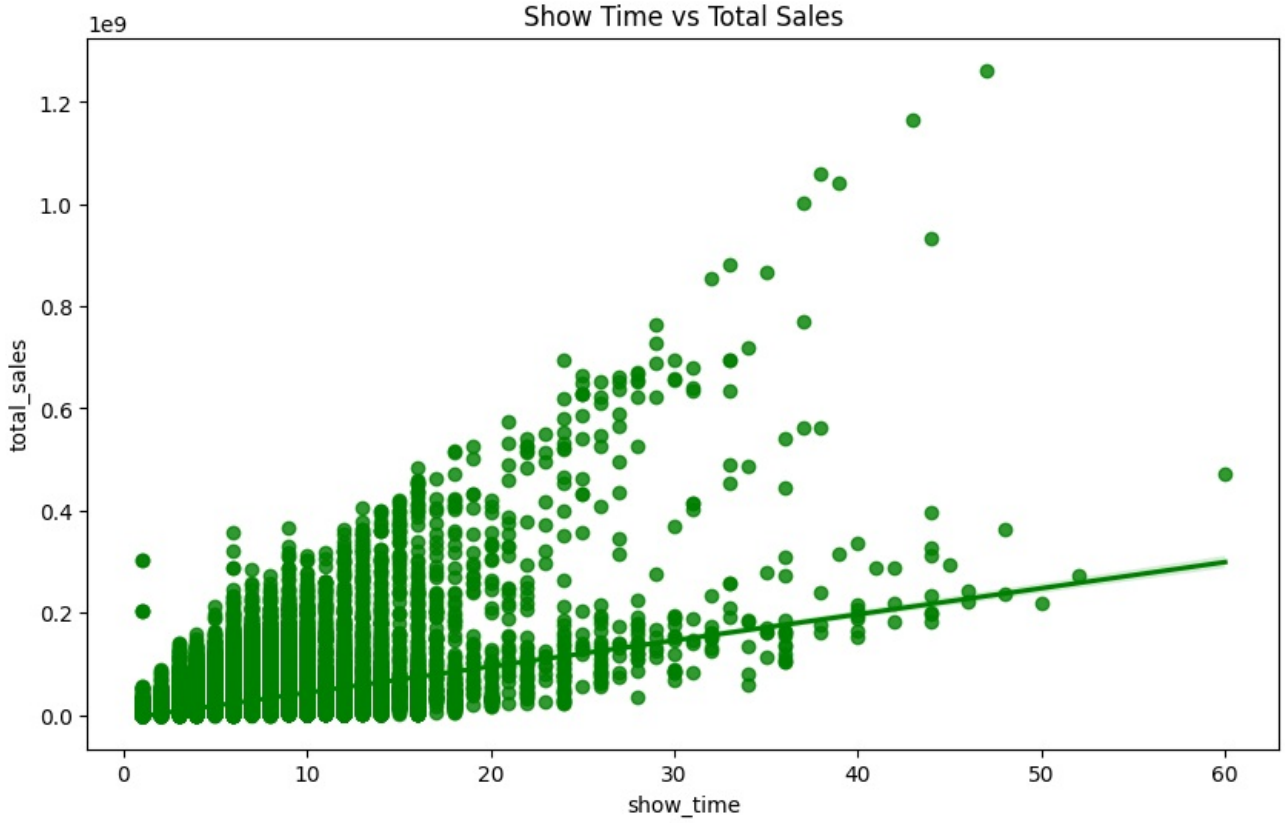


In [77]:

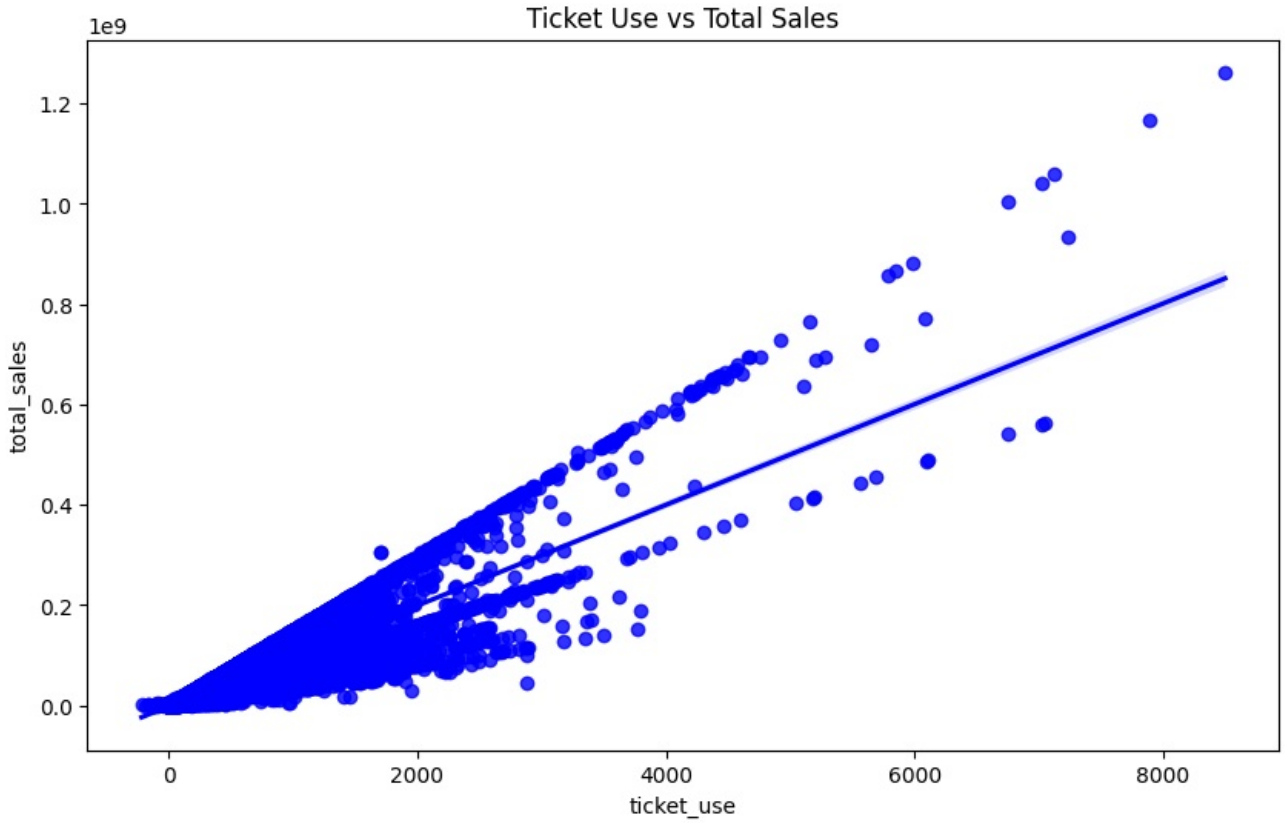
```
# Visualize key relationships
plt.figure(figsize=(10, 6))
sns.regplot(x='tickets_sold', y='total_sales', data=df, color='red')
plt.title('Tickets Sold vs Total Sales')
plt.show()
```



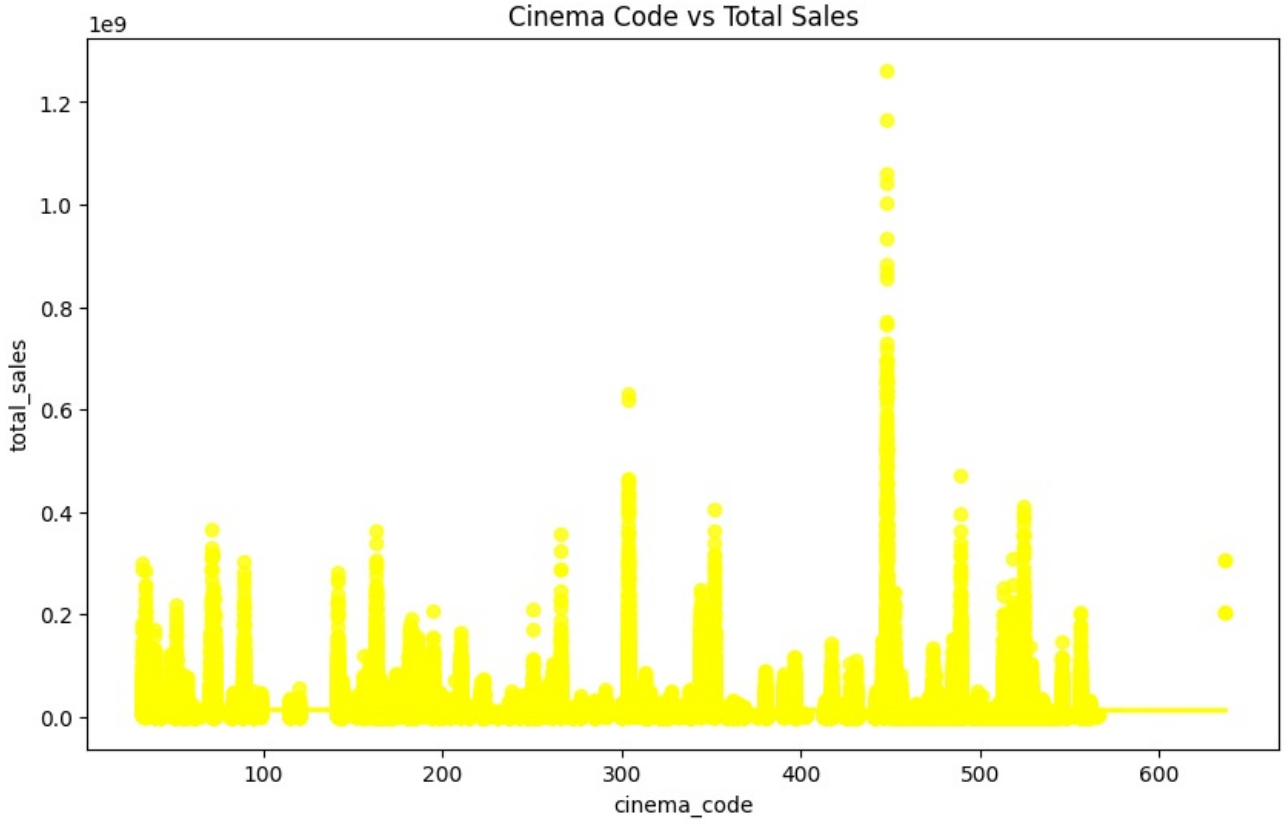
In [78]:
```
plt.figure(figsize=(10, 6))
sns.regplot(x='show_time', y='total_sales', data=df, color='green')
plt.title('Show Time vs Total Sales')
plt.show()
```



In [101…
```
plt.figure(figsize=(10, 6))
sns.regplot(x='ticket_use', y='total_sales', data=df, color='blue')
plt.title('Ticket Use vs Total Sales')
plt.show()
```



In [102…
```
plt.figure(figsize=(10, 6))
sns.regplot(x='cinema_code', y='total_sales', data=df, color='yellow')
plt.title('Cinema Code vs Total Sales')
plt.show()
```



In [100…
```
# Split data into features and target
X = df.drop(['total_sales'], axis=1)
y = df['total_sales']
```

In [82]:
```
X
```

Out[82]:

	film_code	cinema_code	tickets_sold	tickets_out	show_time	occu_perc	ticket_price	ticket_use	capacity	year	day_of_week	is
0	1492	304	26	0	4	4.26	150000.0	26	610.328638	2018	5	
1	1492	352	42	0	5	8.08	80000.0	42	519.801980	2018	5	
2	1492	489	32	0	4	20.00	80000.0	32	160.000000	2018	5	
3	1492	429	12	0	1	11.01	100000.0	12	108.991826	2018	5	
4	1492	524	15	0	3	16.67	80000.0	15	89.982004	2018	5	
...	...	...	...	...	...	...	...	...	...	...	...	
142519	1569	495	22	0	2	3.86	60000.0	22	569.948187	2018	6	
142520	1569	474	15	0	1	65.22	80000.0	15	22.999080	2018	6	
142521	1569	524	8	0	3	9.20	132500.0	8	86.956522	2018	6	
142522	1569	529	5	0	2	5.00	120000.0	5	100.000000	2018	6	
142523	1569	486	5	0	1	1.79	50000.0	5	279.329609	2018	6	

142524 rows × 12 columns

In [83]:
```
y
```

Out[83]:
```
0         3900000
1         3360000
2         2560000
3         1200000
4         1200000
           ...
142519    1320000
142520    1200000
142521    1060000
142522     600000
142523     250000
Name: total_sales, Length: 142524, dtype: int64
```

In [84]:
```
# Train-test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

In [85]:
```
X_train
```

Out[85]:

	film_code	cinema_code	tickets_sold	tickets_out	show_time	occu_perc	ticket_price	ticket_use	capacity	year	day_of_week
121375	1481	304	694	0	7	58.32	79193.083573	694	1189.986283	2018	1
32264	1551	170	15	0	10	0.30	70000.000000	15	5000.000000	2018	3
131582	1493	291	200	0	7	16.33	57000.000000	200	1224.739743	2018	5
94809	1572	546	10	0	3	5.13	150000.000000	10	194.931774	2018	6
88585	1554	243	110	0	4	14.71	60000.000000	110	747.790619	2018	4
...	...	...	...	...	...	...	...	...	...	...	..
73349	1483	237	39	0	5	2.29	50769.230769	39	1703.056769	2018	3
109259	1482	198	154	0	5	17.91	50519.480519	154	859.854830	2018	5
50057	1559	352	212	0	7	18.21	138867.924528	212	1164.195497	2018	4
5192	1497	489	519	0	16	55.76	80000.000000	519	930.774749	2018	1
128037	1493	88	13	0	3	6.77	50000.000000	13	192.023634	2018	0

114019 rows × 12 columns

In [86]:
```
X_test
```

Out[86]:

	film_code	cinema_code	tickets_sold	tickets_out	show_time	occu_perc	ticket_price	ticket_use	capacity	year	day_of_week
117965	1481	467	73	0	1	18.25	50000.000000	73	400.000000	2018	1
87638	1554	73	195	0	10	7.92	118769.230769	195	2462.121212	2018	2
57060	1484	141	105	0	6	15.15	110857.142857	105	693.069307	2018	2
8700	1498	277	5	0	1	1.37	100000.000000	5	364.963504	2018	5
94607	1572	196	22	0	2	11.89	86363.636364	22	185.029437	2018	5
...	...	...	...	...	...	...	...	...	...	...	...
73158	1483	338	9	0	1	4.23	62222.222222	9	212.765957	2018	0
97516	1564	504	7	0	4	0.87	40000.000000	7	804.597701	2018	3
4691	1497	73	155	0	4	21.89	98709.677419	155	708.085884	2018	3
124240	1493	390	123	0	4	86.51	74796.747967	123	142.180095	2018	2
2181	1497	546	13	0	3	6.67	150000.000000	13	194.902549	2018	2

28505 rows × 12 columns

In [87]:
```
y_train
```

Out[87]:
```
121375    54960000
32264      1050000
131582    11400000
94809      1500000
88585      6600000
            ...
73349      1980000
109259     7780000
50057     29440000
5192      41520000
128037      650000
Name: total_sales, Length: 114019, dtype: int64
```

In [88]:
```
y_test
```

Out[88]:
```
117965     3650000
87638     23160000
57060     11640000
8700        500000
94607      1900000
            ...
73158       560000
97516       280000
4691      15300000
124240     9200000
2181       1950000
Name: total_sales, Length: 28505, dtype: int64
```

In [89]:
```
# Linear Regression Model
lr = LinearRegression()
lr.fit(X_train, y_train)
y_pred_lr = lr.predict(X_test)
```

In [90]:
```
y_pred_lr
```

Out[90]:
```
array([  789768.77069313, 22585130.04241101, 13676695.10550301, ...,
       15825562.33718276,  4311567.51089288, 13900331.59332835])
```

In [91]:
```
y_test
```

Out[91]:
```
117965     3650000
87638     23160000
57060     11640000
8700        500000
94607      1900000
            ...
73158       560000
97516       280000
4691      15300000
124240     9200000
2181       1950000
Name: total_sales, Length: 28505, dtype: int64
```

In [92]:
```
# Decision Tree Regressor
dt = DecisionTreeRegressor(random_state=42)
dt.fit(X_train, y_train)
y_pred_dt = dt.predict(X_test)
```

In [93]:
```
y_pred_dt
```

Out[93]:
```
array([ 3650000., 23200000., 11760000., ..., 15400000.,  9200000.,
        1950000.])
```

In [94]:
```
y_test
```

Out[94]:
```
117965     3650000
87638     23160000
57060     11640000
8700        500000
94607      1900000
             ...
73158       560000
97516       280000
4691      15300000
124240     9200000
2181       1950000
Name: total_sales, Length: 28505, dtype: int64
```

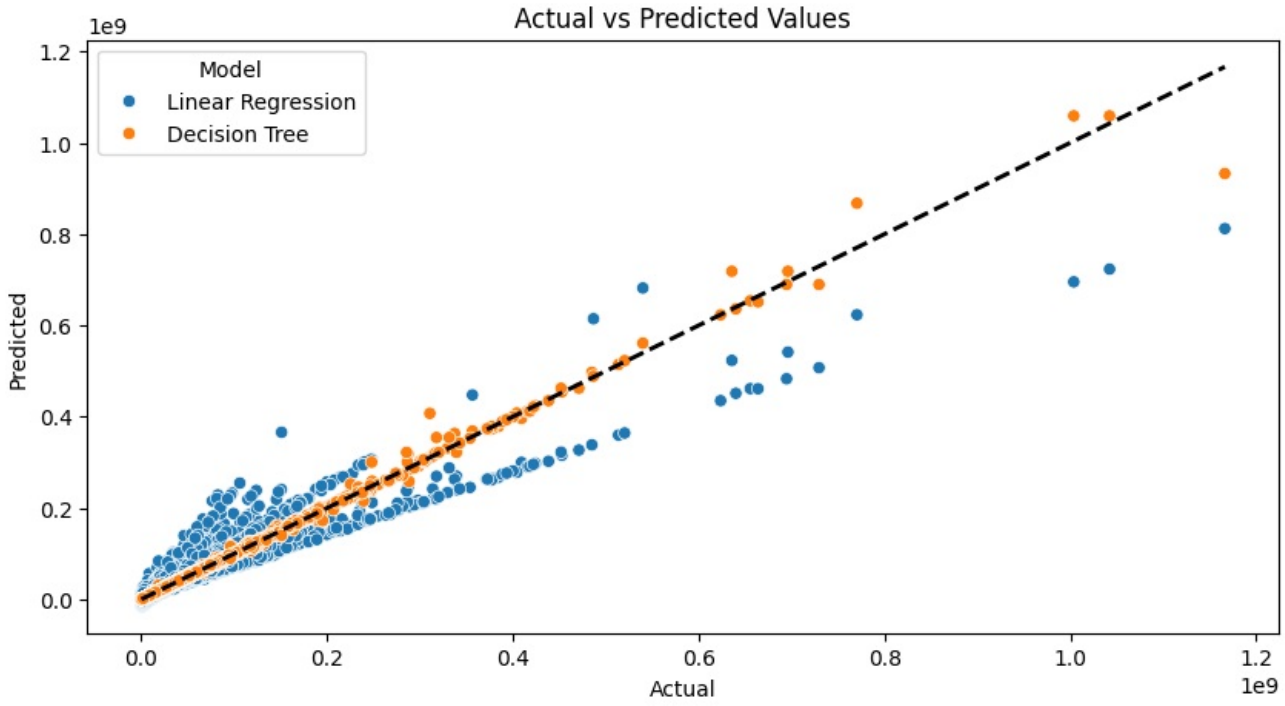
In [95]:
```
# Model Evaluation
models = {'Linear Regression': y_pred_lr, 'Decision Tree': y_pred_dt}
for model_name, y_pred in models.items():
    print(f"{model_name}:")
    print(f"  Mean Absolute Error: {mean_absolute_error(y_test, y_pred)}")
    print(f"  Mean Absolute Percentage Error: {mean_absolute_percentage_error(y_test, y_pred)}")
    print(f"  R2 Score: {r2_score(y_test, y_pred)}\n")
```

```
Linear Regression:
  Mean Absolute Error: 4853252.680095563
  Mean Absolute Percentage Error: 3.077354121965154
  R2 Score: 0.8826065213042851

Decision Tree:
  Mean Absolute Error: 130022.66525171022
  Mean Absolute Percentage Error: 0.0038536460082864016
  R2 Score: 0.9962886001432594
```

In [96]:
```
# Comparison of Actual vs Predicted Values
df_results_lr = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred_lr, 'Model': 'Linear Regression'})
df_results_dt = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred_dt, 'Model': 'Decision Tree'})
df_results = pd.concat([df_results_lr, df_results_dt])
```

In [97]:
```
plt.figure(figsize=(10, 5))
sns.scatterplot(data=df_results, x='Actual', y='Predicted', hue='Model')
plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2)
plt.title('Actual vs Predicted Values')
plt.show()
```



In [ ]:

In [ ]:

In [ ]:

In [ ]: