In [1]:
```python
import pandas as pd
import numpy as np
```

In [2]:
```python
data = pd.read_csv("Video_Games_Sales_as_at_22_Dec_2016.csv")
data
```

Out[2]:

	Name	Platform	Year_of_Release	Genre	Publisher	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales	Critic_Score	Critic_Count	User_Score	User_Count	Developer	Rating
0	Wii Sports	Wii	2006.0	Sports	Nintendo	41.36	28.96	3.77	8.45	82.53	76.0	51.0	8.0	322.0	Nintendo	E
1	Super Mario Bros.	NES	1985.0	Platform	Nintendo	29.08	3.58	6.81	0.77	40.24	NaN	NaN	NaN	NaN	NaN	NaN
2	Mario Kart Wii	Wii	2008.0	Racing	Nintendo	15.68	12.76	3.79	3.29	35.52	82.0	73.0	8.3	709.0	Nintendo	E
3	Wii Sports Resort	Wii	2009.0	Sports	Nintendo	15.61	10.93	3.28	2.95	32.77	80.0	73.0	8.0	192.0	Nintendo	E
4	Pokemon Red/Pokemon Blue	GB	1996.0	Role-Playing	Nintendo	11.27	8.89	10.22	1.00	31.37	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
16714	Samurai Warriors: Sanada Maru	PS3	2016.0	Action	Tecmo Koei	0.00	0.00	0.01	0.00	0.01	NaN	NaN	NaN	NaN	NaN	NaN
16715	LMA Manager 2007	X360	2006.0	Sports	Codemasters	0.00	0.01	0.00	0.00	0.01	NaN	NaN	NaN	NaN	NaN	NaN
16716	Haitaka no Psychedelica	PSV	2016.0	Adventure	Idea Factory	0.00	0.00	0.01	0.00	0.01	NaN	NaN	NaN	NaN	NaN	NaN
16717	Spirits & Spells	GBA	2003.0	Platform	Wanadoo	0.01	0.00	0.00	0.00	0.01	NaN	NaN	NaN	NaN	NaN	NaN
16718	Winning Post 8 2016	PSV	2016.0	Simulation	Tecmo Koei	0.00	0.00	0.01	0.00	0.01	NaN	NaN	NaN	NaN	NaN	NaN

16719 rows × 16 columns

In [3]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16719 entries, 0 to 16718
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Name             16717 non-null  object 
 1   Platform         16719 non-null  object 
 2   Year_of_Release  16450 non-null  float64
 3   Genre            16717 non-null  object 
 4   Publisher        16665 non-null  object 
 5   NA_Sales         16719 non-null  float64
 6   EU_Sales         16719 non-null  float64
 7   JP_Sales         16719 non-null  float64
 8   Other_Sales      16719 non-null  float64
 9   Global_Sales     16719 non-null  float64
 10  Critic_Score     8137 non-null   float64
 11  Critic_Count     8137 non-null   float64
 12  User_Score       7590 non-null   float64
 13  User_Count       7590 non-null   float64
 14  Developer        10096 non-null  object 
 15  Rating           9950 non-null   object 
dtypes: float64(10), object(6)
memory usage: 2.0+ MB
```

In [4]:
```python
data.shape
```

Out[4]:
```
(16719, 16)
```

In [5]:
```python
data.describe()
```

Out[5]:

	Year_of_Release	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Global_Sales	Critic_Score	Critic_Count	User_Score	User_Count
count	16450.000000	16719.000000	16719.000000	16719.000000	16719.000000	16719.000000	8137.000000	8137.000000	7590.000000	7590.000000
mean	2006.487356	0.263330	0.145025	0.077602	0.047332	0.533543	68.967679	26.360821	7.125046	162.229908
std	5.878995	0.813514	0.503283	0.308818	0.186710	1.547935	13.938165	18.980495	1.500006	561.282326
min	1980.000000	0.000000	0.000000	0.000000	0.000000	0.010000	13.000000	3.000000	0.000000	4.000000
25%	2003.000000	0.000000	0.000000	0.000000	0.000000	0.060000	60.000000	12.000000	6.400000	10.000000
50%	2007.000000	0.080000	0.020000	0.000000	0.010000	0.170000	71.000000	21.000000	7.500000	24.000000
75%	2010.000000	0.240000	0.110000	0.040000	0.030000	0.470000	79.000000	36.000000	8.200000	81.000000
max	2020.000000	41.360000	28.960000	10.220000	10.570000	82.530000	98.000000	113.000000	9.700000	10665.000000

In [7]:
```python
data.fillna(0, inplace=True)
```

In [8]:
```python
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 16719 entries, 0 to 16718
Data columns (total 16 columns):
 #   Column           Non-Null Count  Dtype  
---  ------           --------------  -----  
 0   Name             16719 non-null  object 
 1   Platform         16719 non-null  object 
 2   Year_of_Release  16719 non-null  float64
 3   Genre            16719 non-null  object 
 4   Publisher        16719 non-null  object 
 5   NA_Sales         16719 non-null  float64
 6   EU_Sales         16719 non-null  float64
 7   JP_Sales         16719 non-null  float64
 8   Other_Sales      16719 non-null  float64
 9   Global_Sales     16719 non-null  float64
 10  Critic_Score     16719 non-null  float64
 11  Critic_Count     16719 non-null  float64
 12  User_Score       16719 non-null  float64
 13  User_Count       16719 non-null  float64
 14  Developer        16719 non-null  object 
 15  Rating           16719 non-null  object 
dtypes: float64(10), object(6)
memory usage: 2.0+ MB
```

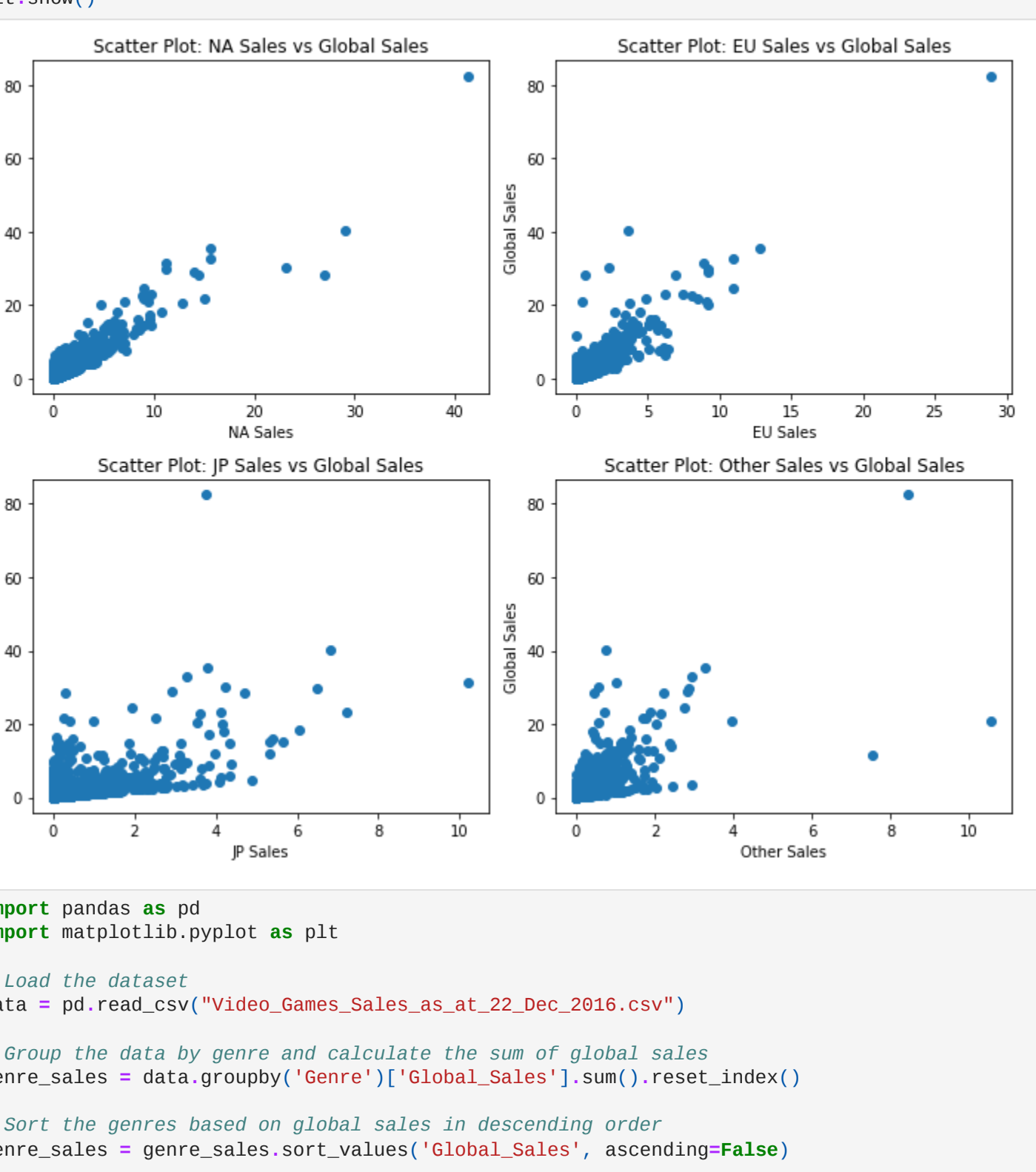
In [9]:
```python
data['Platform'].unique()
```

Out[9]:
```
array(['Wii', 'NES', 'GB', 'DS', 'X360', 'PS3', 'PS2', 'SNES', 'GBA',
       'PS4', '3DS', 'N64', 'PS', 'XB', 'PC', '2600', 'PSP', 'XOne',
       'WiiU', 'GC', 'GEN', 'DC', 'PSV', 'SAT', 'SCD', 'WS', 'NG', 'TG16',
       '3DO', 'GG', 'PCFX'], dtype=object)
```

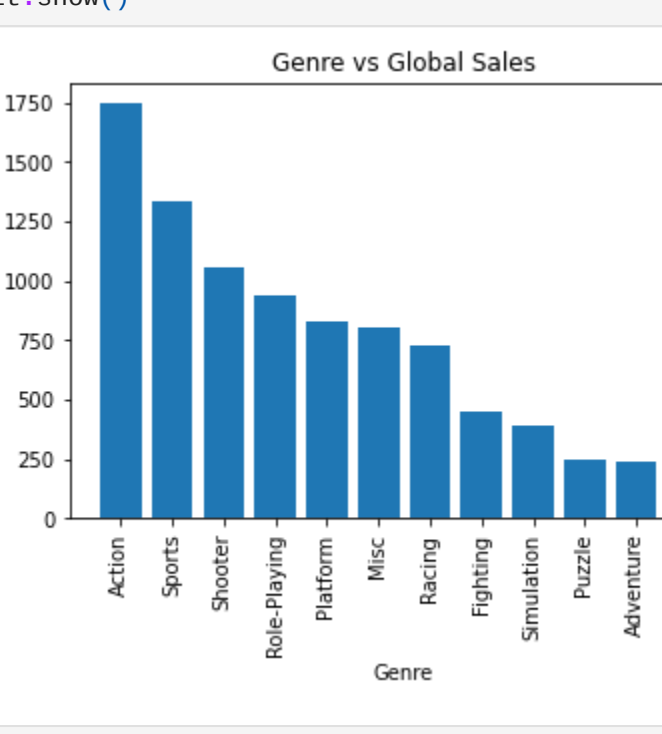
In [10]:
```python
data['Genre'].unique()
```

Out[10]:
```
array(['Sports', 'Platform', 'Racing', 'Role-Playing', 'Puzzle', 'Misc',
       'Shooter', 'Simulation', 'Action', 'Fighting', 'Adventure',
       'Strategy', 0], dtype=object)
```

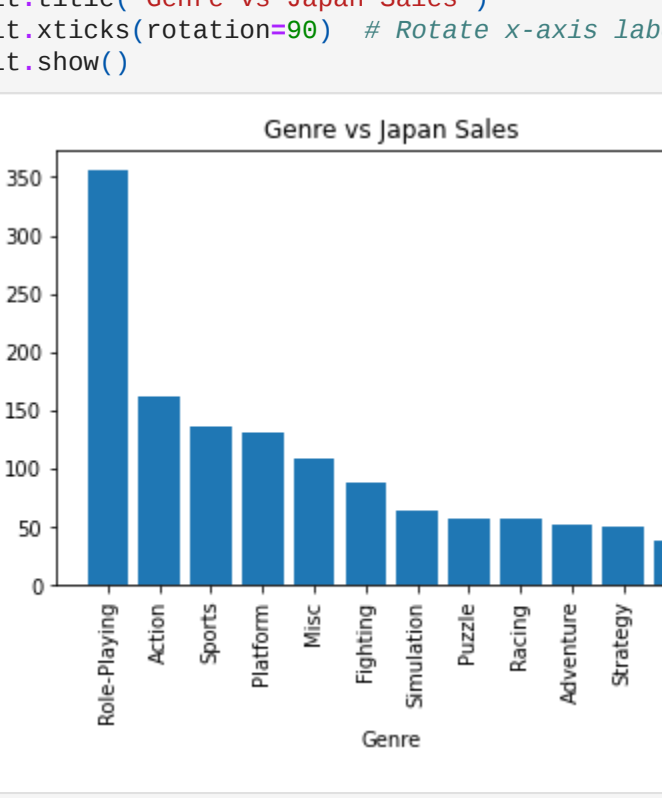
In [13]:
```python
import pandas as pd
import matplotlib.pyplot as plt

# Select the columns for the scatter plots
x_na_sales = data['NA_Sales']
y_na_sales = data['Global_Sales']

x_eu_sales = data['EU_Sales']
y_eu_sales = data['Global_Sales']

x_jp_sales = data['JP_Sales']
y_jp_sales = data['Global_Sales']

x_other_sales = data['Other_Sales']
y_other_sales = data['Global_Sales']

# Plot the scatter plots
plt.figure(figsize=(10, 8))

plt.subplot(221)
plt.scatter(x_na_sales, y_na_sales)
plt.xlabel('NA Sales')
plt.ylabel('Global Sales')
plt.title('Scatter Plot: NA Sales vs Global Sales')

plt.subplot(222)
plt.scatter(x_eu_sales, y_eu_sales)
plt.xlabel('EU Sales')
plt.ylabel('Global Sales')
plt.title('Scatter Plot: EU Sales vs Global Sales')

plt.subplot(223)
plt.scatter(x_jp_sales, y_jp_sales)
plt.xlabel('JP Sales')
plt.ylabel('Global Sales')
plt.title('Scatter Plot: JP Sales vs Global Sales')

plt.subplot(224)
plt.scatter(x_other_sales, y_other_sales)
plt.xlabel('Other Sales')
plt.ylabel('Global Sales')
plt.title('Scatter Plot: Other Sales vs Global Sales')

plt.tight_layout()
plt.show()
```



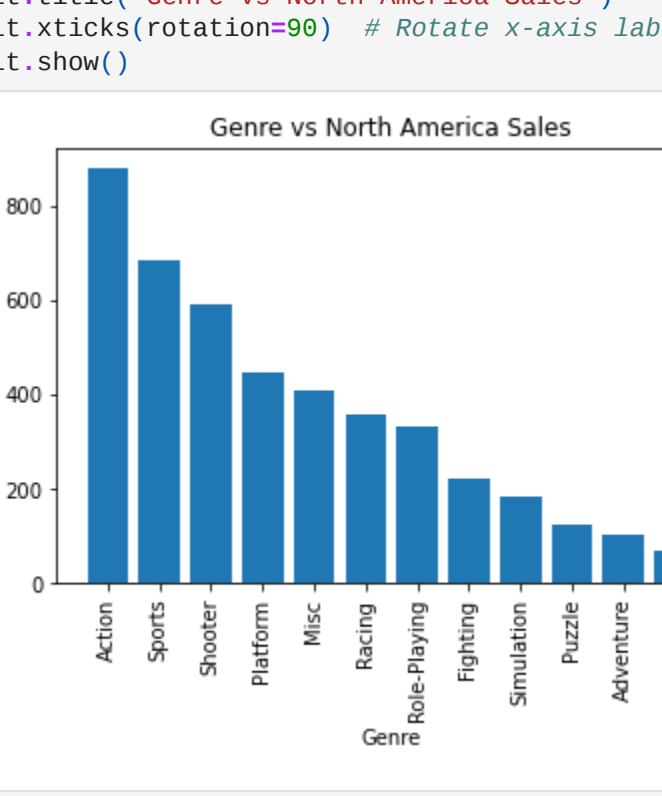
In [14]:
```python
import pandas as pd
import matplotlib.pyplot as plt

# Load the dataset
data = pd.read_csv("Video_Games_Sales_as_at_22_Dec_2016.csv")

# Group the data by genre and calculate the sum of global sales
genre_sales = data.groupby('Genre')['Global_Sales'].sum().reset_index()

# Sort the genres based on global sales in descending order
genre_sales = genre_sales.sort_values('Global_Sales', ascending=False)

# Plot the bar chart
plt.bar(genre_sales['Genre'], genre_sales['Global_Sales'])
plt.xlabel('Genre')
plt.ylabel('Global Sales')
plt.title('Genre vs Global Sales')
plt.xticks(rotation=90)  # Rotate x-axis labels for better visibility
plt.show()
```



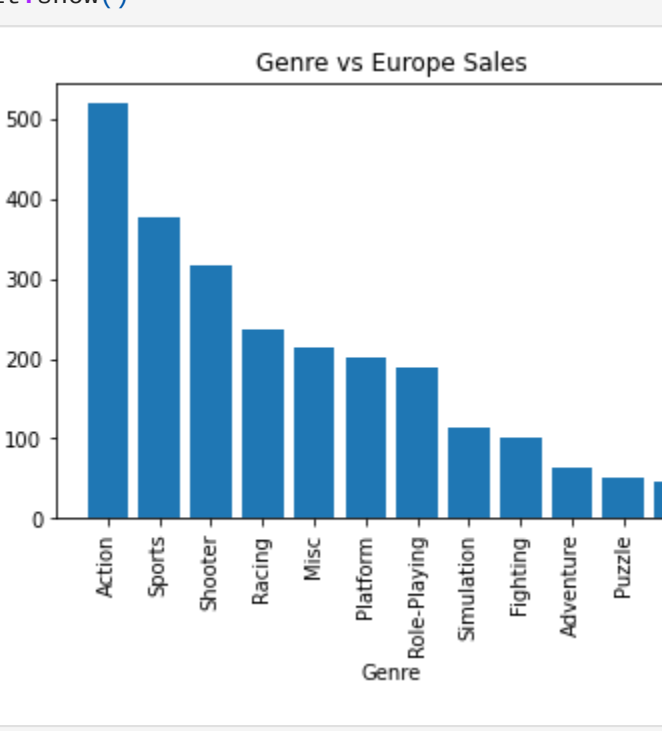
In [15]:
```python
import pandas as pd
import matplotlib.pyplot as plt

# Group the data by genre and calculate the sum of global sales
genre_sales = data.groupby('Genre')['JP_Sales'].sum().reset_index()

# Sort the genres based on global sales in descending order
genre_sales = genre_sales.sort_values('JP_Sales', ascending=False)

# Plot the bar chart
plt.bar(genre_sales['Genre'], genre_sales['JP_Sales'])
plt.xlabel('Genre')
plt.ylabel('Japan Sales')
plt.title('Genre vs Japan Sales')
plt.xticks(rotation=90)  # Rotate x-axis labels for better visibility
plt.show()
```



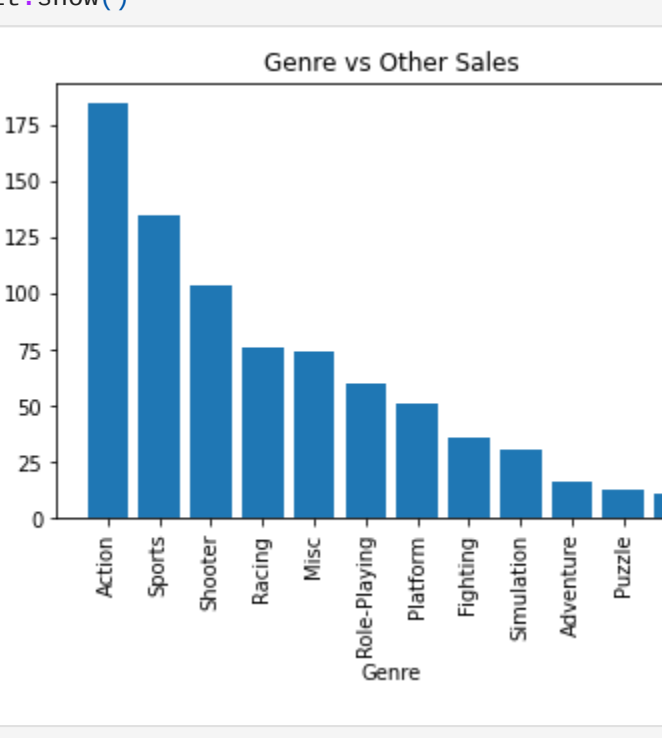
In [16]:
```python
# Group the data by genre and calculate the sum of global sales
genre_sales = data.groupby('Genre')['NA_Sales'].sum().reset_index()

# Sort the genres based on global sales in descending order
genre_sales = genre_sales.sort_values('NA_Sales', ascending=False)

# Plot the bar chart
plt.bar(genre_sales['Genre'], genre_sales['NA_Sales'])
plt.xlabel('Genre')
plt.ylabel('North America Sales')
plt.title('Genre vs North America Sales')
plt.xticks(rotation=90)  # Rotate x-axis labels for better visibility
plt.show()
```



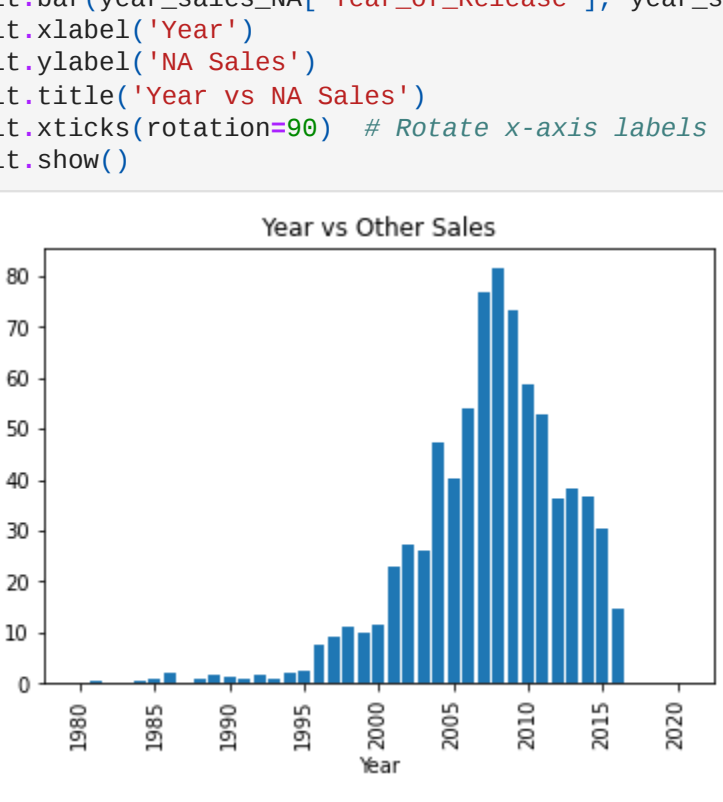
In [17]:
```python
# Group the data by genre and calculate the sum of global sales
genre_sales = data.groupby('Genre')['EU_Sales'].sum().reset_index()

# Sort the genres based on global sales in descending order
genre_sales = genre_sales.sort_values('EU_Sales', ascending=False)

# Plot the bar chart
plt.bar(genre_sales['Genre'], genre_sales['EU_Sales'])
plt.xlabel('Genre')
plt.ylabel('Europe Sales')
plt.title('Genre vs Europe Sales')
plt.xticks(rotation=90)  # Rotate x-axis labels for better visibility
plt.show()
```

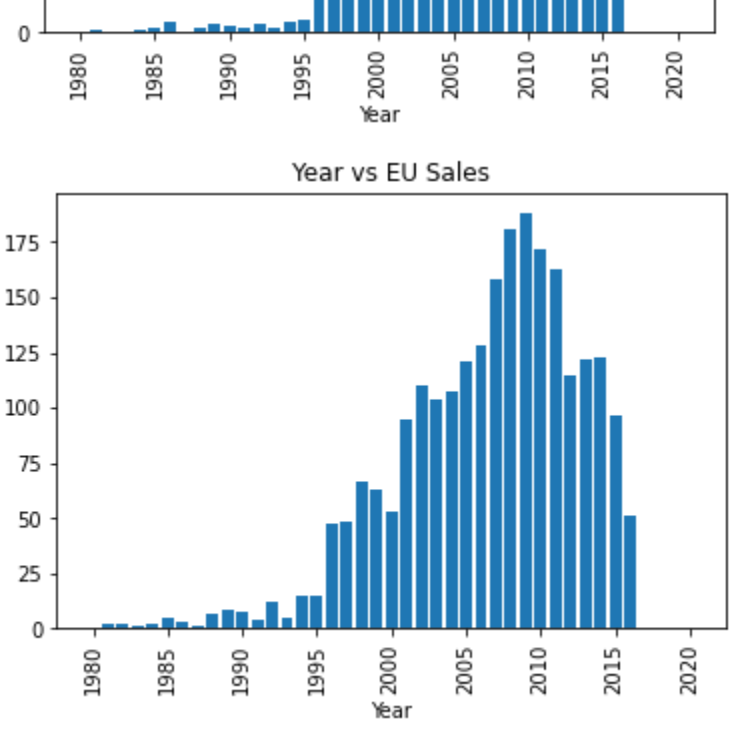


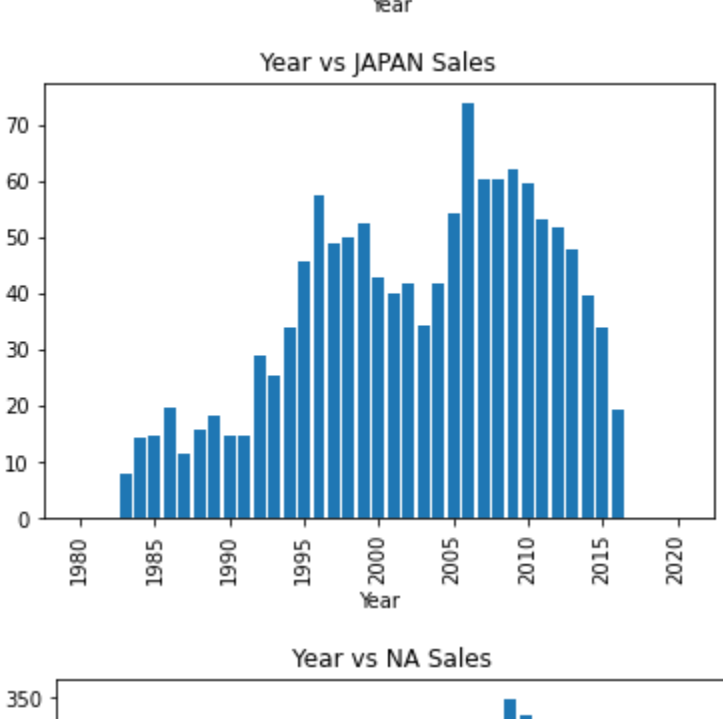
In [18]:
```python
# Group the data by genre and calculate the sum of global sales
genre_sales = data.groupby('Genre')['Other_Sales'].sum().reset_index()

# Sort the genres based on global sales in descending order
genre_sales = genre_sales.sort_values('Other_Sales', ascending=False)

# Plot the bar chart
plt.bar(genre_sales['Genre'], genre_sales['Other_Sales'])
plt.xlabel('Genre')
plt.ylabel('Other Sales')
plt.title('Genre vs Other Sales')
plt.xticks(rotation=90)  # Rotate x-axis labels for better visibility
plt.show()
```

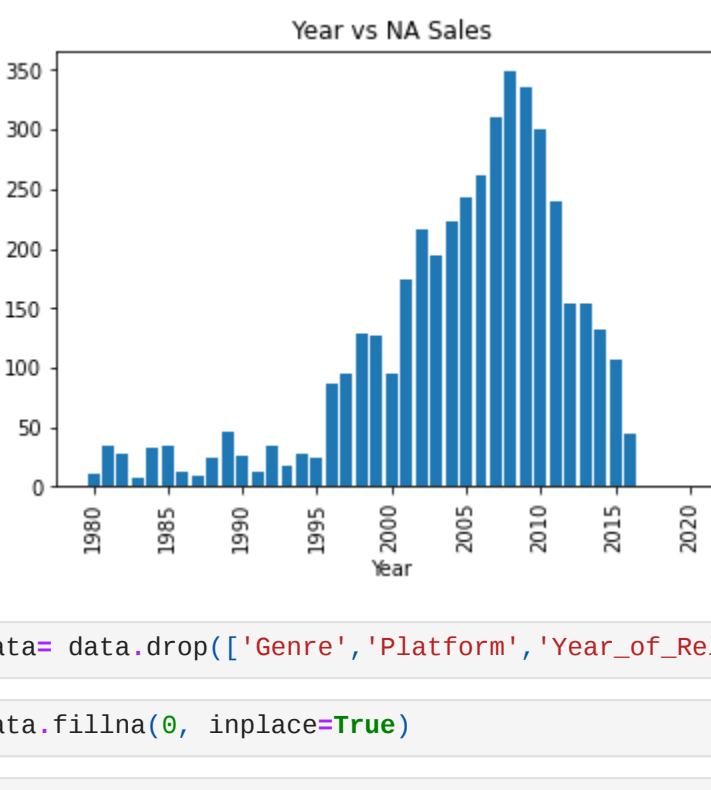


In [19]:
```python
# Group the data by year and calculate the sum of other sales
year_sales_others = data.groupby('Year_of_Release')['Other_Sales'].sum().reset_index()
# Sort the year based on other sales in descending order
year_sales_others = year_sales_others.sort_values('Other_Sales', ascending=False)


# Group the data by year and calculate the sum of EUROPE sales
year_sales_EU = data.groupby('Year_of_Release')['EU_Sales'].sum().reset_index()
# Sort the year based on other sales in descending order
year_sales_EU = year_sales_EU.sort_values('EU_Sales', ascending=False)


# Group the data by year and calculate the sum of JAPAN sales
year_sales_JP = data.groupby('Year_of_Release')['JP_Sales'].sum().reset_index()
# Sort the year based on other sales in descending order
year_sales_JP = year_sales_JP.sort_values('JP_Sales', ascending=False)


# Group the data by year and calculate the sum of NORTH AMERICA sales
year_sales_NA = data.groupby('Year_of_Release')['NA_Sales'].sum().reset_index()
# Sort the year based on other sales in descending order
year_sales_NA = year_sales_NA.sort_values('NA_Sales', ascending=False)



# Plot the bar chart
plt.bar(year_sales_others['Year_of_Release'], year_sales_others['Other_Sales'])
plt.xlabel('Year')
plt.ylabel('Other Sales')
plt.title('Year vs Other Sales')
plt.xticks(rotation=90)  # Rotate x-axis labels for better visibility
plt.show()


# Plot the bar chart
plt.bar(year_sales_EU['Year_of_Release'], year_sales_EU['EU_Sales'])
plt.xlabel('Year')
plt.ylabel('EU Sales')
plt.title('Year vs EU Sales')
plt.xticks(rotation=90)  # Rotate x-axis labels for better visibility
plt.show()


# Plot the bar chart
plt.bar(year_sales_JP['Year_of_Release'], year_sales_JP['JP_Sales'])
plt.xlabel('Year')
plt.ylabel('JAPAN Sales')
plt.title('Year vs JAPAN Sales')
plt.xticks(rotation=90)  # Rotate x-axis labels for better visibility
plt.show()


# Plot the bar chart
plt.bar(year_sales_NA['Year_of_Release'], year_sales_NA['NA_Sales'])
plt.xlabel('Year')
plt.ylabel('NA Sales')
plt.title('Year vs NA Sales')
plt.xticks(rotation=90)  # Rotate x-axis labels for better visibility
plt.show()
```







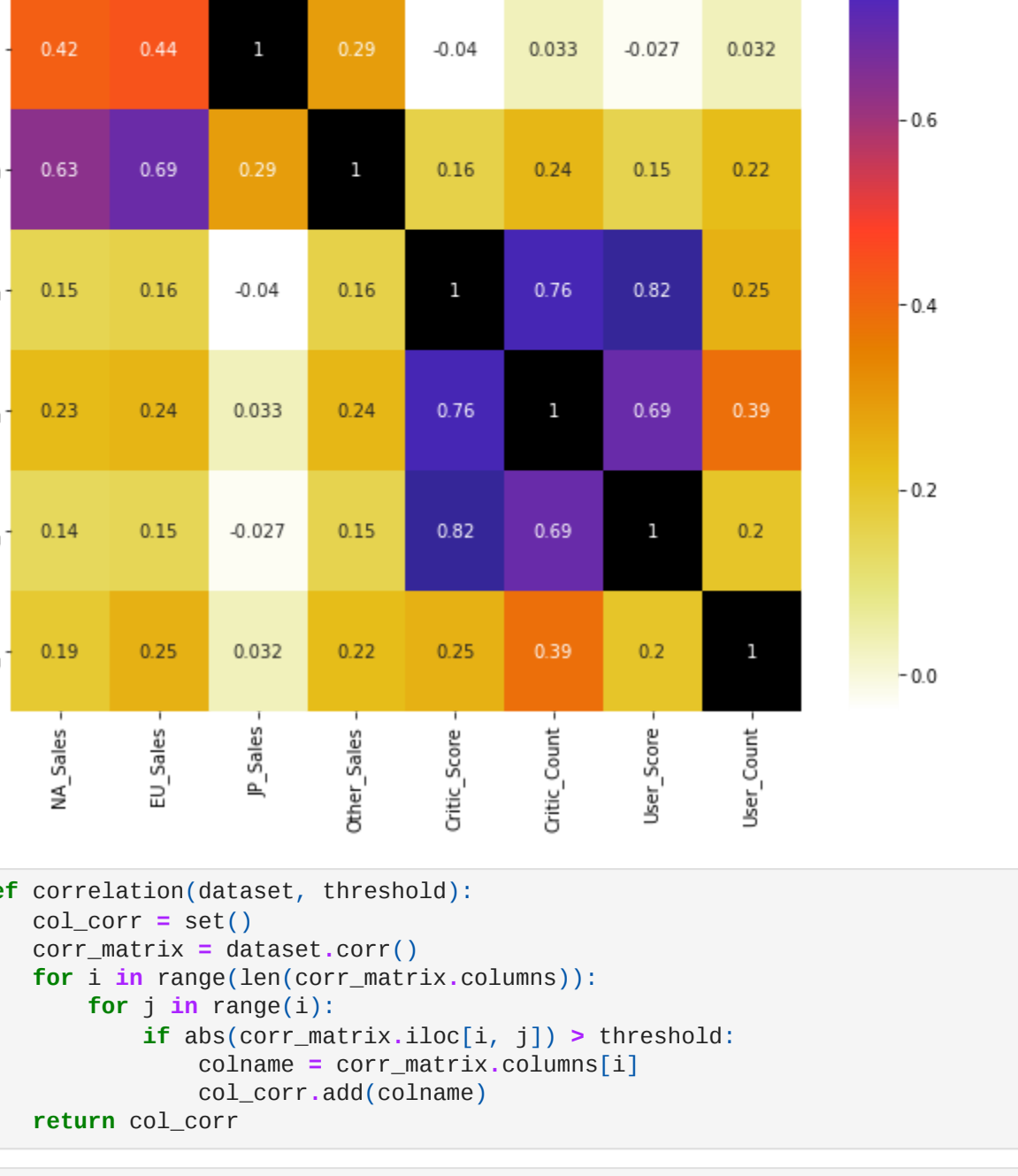


In [20]:
```python
data= data.drop(['Genre','Platform','Year_of_Release','Name','Publisher','Developer','Rating'],axis=1)
```

In [31]:
```python
data.fillna(0, inplace=True)
```

In [32]:
```python
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
y = data['Global_Sales']
#Load X Variables into a Pandas Dataframe with columns
X = data.drop(['Global_Sales'], axis = 1)
#y is dependent variable and X is independent variable.
X_train, X_test, y_train, y_test = train_test_split(X, y,random_state=100,test_size=0.3)
```

In [33]:
```python
cor = X_train.corr()
plt.figure(figsize=(10,10))
sns.heatmap(cor, cmap=plt.cm.CMRmap_r,annot=True)
plt.show()
```



In [34]:
```python
def correlation(dataset, threshold):
    col_corr = set()
    corr_matrix = dataset.corr()
    for i in range(len(corr_matrix.columns)):
        for j in range(i):
            if abs(corr_matrix.iloc[i, j]) > threshold:
                colname = corr_matrix.columns[i]
                col_corr.add(colname)
    return col_corr
```

In [35]:
```python
corr_features = correlation(X_train, 0.7)
corr_features
```

Out[35]:
```
{'Critic_Count', 'EU_Sales', 'User_Score'}
```

In [36]:
```python
X_train.drop(corr_features,axis=1)
X_test.drop(corr_features,axis=1)
X_train
```

Out[36]:

	NA_Sales	EU_Sales	JP_Sales	Other_Sales	Critic_Score	Critic_Count	User_Score	User_Count
5306	0.13	0.18	0.00	0.04	48.0	50.0	5.0	53.0
1162	1.15	0.36	0.04	0.04	96.0	16.0	9.2	568.0
13175	0.05	0.00	0.00	0.00	60.0	25.0	5.8	9.0
669	1.74	0.45	0.00	0.18	42.0	10.0	5.5	16.0
16495	0.00	0.01	0.00	0.00	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...
16304	0.00	0.00	0.01	0.00	0.0	0.0	0.0	0.0
79	1.75	3.47	2.49	0.67	68.0	42.0	7.4	54.0
12119	0.06	0.00	0.00	0.00	65.0	16.0	6.3	4.0
14147	0.00	0.00	0.03	0.00	0.0	0.0	0.0	0.0
5640	0.27	0.03	0.00	0.03	79.0	80.0	7.3	117.0

11703 rows × 8 columns

In [37]:
```python
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
y_train = label_encoder.fit_transform(y_train)
```

In [38]:
```python
#Chi-square
#Perform chi2 test
from sklearn.feature_selection import chi2
#Calculating Fscore and p value
f_p_values=chi2(X_train,y_train)
f_p_values
```

Out[38]:
```
(array([2.87376486e+04, 2.07428781e+04, 1.02933661e+04, 9.22909098e+03,
        6.63691920e+04, 8.48249698e+04, 7.14273374e+03, 9.80031538e+06]),
 array([0., 0., 0., 0., 0., 0., 0., 0.]))
```

In [39]:
```python
import pandas as pd
p_values=pd.Series(f_p_values[1])
p_values.index=X_train.columns
p_values
```

Out[39]:
```
NA_Sales        0.0
EU_Sales        0.0
JP_Sales        0.0
Other_Sales     0.0
Critic_Score    0.0
Critic_Count    0.0
User_Score      0.0
User_Count      0.0
dtype: float64
```

In [40]:
```python
p_values.sort_index(ascending=False)
```

Out[40]:
```
User_Score      0.0
User_Count      0.0
Other_Sales     0.0
NA_Sales        0.0
JP_Sales        0.0
EU_Sales        0.0
Critic_Score    0.0
Critic_Count    0.0
dtype: float64
```

In [41]:
```python
#Mutual information gain
#Importing mutual information gain
from sklearn.feature_selection import mutual_info_classif
# determine the mutual information
mutual_info = mutual_info_classif(X_train, y_train)
mutual_info
```

Out[41]:
```
array([1.07060838, 0.63653717, 0.54841337, 0.65351435, 0.0642521 ,
       0.06211576, 0.04514415, 0.0562902 ])
```

In [42]:
```python
#Representing in list form
mutual_info = pd.Series(mutual_info)
mutual_info.index = X_train.columns
mutual_info.sort_values(ascending=False)
```

Out[42]:
```
NA_Sales        1.070608
Other_Sales     0.653514
EU_Sales        0.636537
JP_Sales        0.548413
Critic_Score    0.064252
Critic_Count    0.062116
User_Count      0.056290
User_Score      0.045144
dtype: float64
```

In [43]:
```python
mutual_info.sort_values(ascending=False).plot.bar(figsize=(20, 8))
```

Out[43]:
```
<AxesSubplot:>
```

In [44]:
```python
columns = ['NA_Sales', 'Other_Sales', 'EU_Sales','JP_Sales','Critic_Score','Critic_Count']
X = data[columns]
Y = data['Global_Sales']
```

In [45]:
```python
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random_state=2)
print(X.shape, X_train.shape, X_test.shape)
```

```
(16719, 6) (13375, 6) (3344, 6)
```

In [46]:
```python
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import accuracy_score
```

In [47]:
```python
from sklearn.linear_model import LinearRegression

# Create and fit the linear regression model
model = LinearRegression()
model.fit(X_train, Y_train)

# Calculate the R-squared score for training data
train_score = model.score(X_train, Y_train)
print("Training score:", train_score)

# Calculate the R-squared score for testing data
test_score = model.score(X_test, Y_test)
print("Testing score:", test_score)
```

```
Training score: 0.9999893386950928
Testing score: 0.9999848436769464
```

In [48]:
```python
y_pred = model.predict(X_test)
```

In [49]:
```python
#Mean Squared Error (MSE)
mse = mean_squared_error(Y_test, y_pred)
print("Mean Squared Error:", mse)
```

```
Mean Squared Error: 2.6647849820630276e-05
```

In [50]:
```python
#Mean Absolute Error (MAE)
from sklearn.metrics import mean_absolute_error

# Assuming you have the true target values in y_true and the predicted values in y_pred
mae = mean_absolute_error(Y_test, y_pred)
print("Mean Absolute Error:", mae)
```

```
Mean Absolute Error: 0.0028306184298071547
```

In [51]:
```python
from sklearn.metrics import r2_score

# Assuming you have the true target values in y_true and the predicted values in y_pred
r2 = r2_score(Y_test, y_pred)
print("R-squared:", r2)
```

```
R-squared: 0.9999848436769464
```

In [ ]: