In [13]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

In [14]:
```

df = pd.read_csv(r'C:\Users\kriti\OneDrive\Desktop\UM\Entertainer full
```

In [15]:
```
df.head()
```

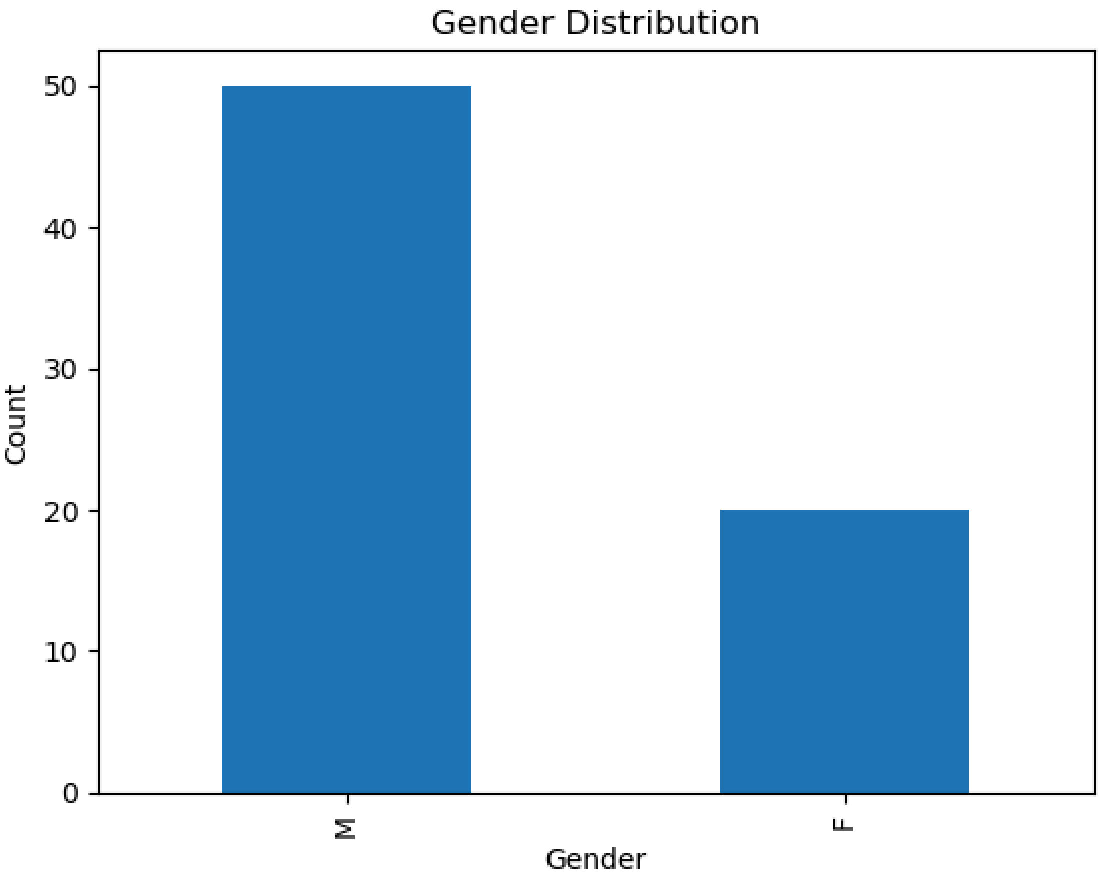
Out[15]:

	Entertainer	Gender	Year of Last Major Work (arguable)	Year of Death	Birth Year	Year of Breakthrough/#1 Hit/Award Nomination	Breakthrough Name	Oscar
0	Adele	F	2016	NaN	1988	2008	19	
1	Angelina Jolie	F	2016	NaN	1975	1999	Girl, Interrupted	
2	Aretha Franklin	F	2014	NaN	1942	1967	I Never Loved a Man (The Way I Love You)	
3	Bette Davis	F	1989	1989.0	1908	1934	Of Human Bondage	
4	Betty White	F	2016	NaN	1922	1952	Life with Elilzabeth	

In [16]:
```
def clean_data(df):
    # Drop rows with missing values
    df_cleaned = df.dropna()
    return df_cleaned

```

In [ ]:
```

```

```
In [17]: # Gender Distribution
         gender_counts = df['Gender'].value_counts()
         gender_counts.plot(kind='bar', title='Gender Distribution')
         plt.xlabel('Gender')
         plt.ylabel('Count')
         plt.show()
```



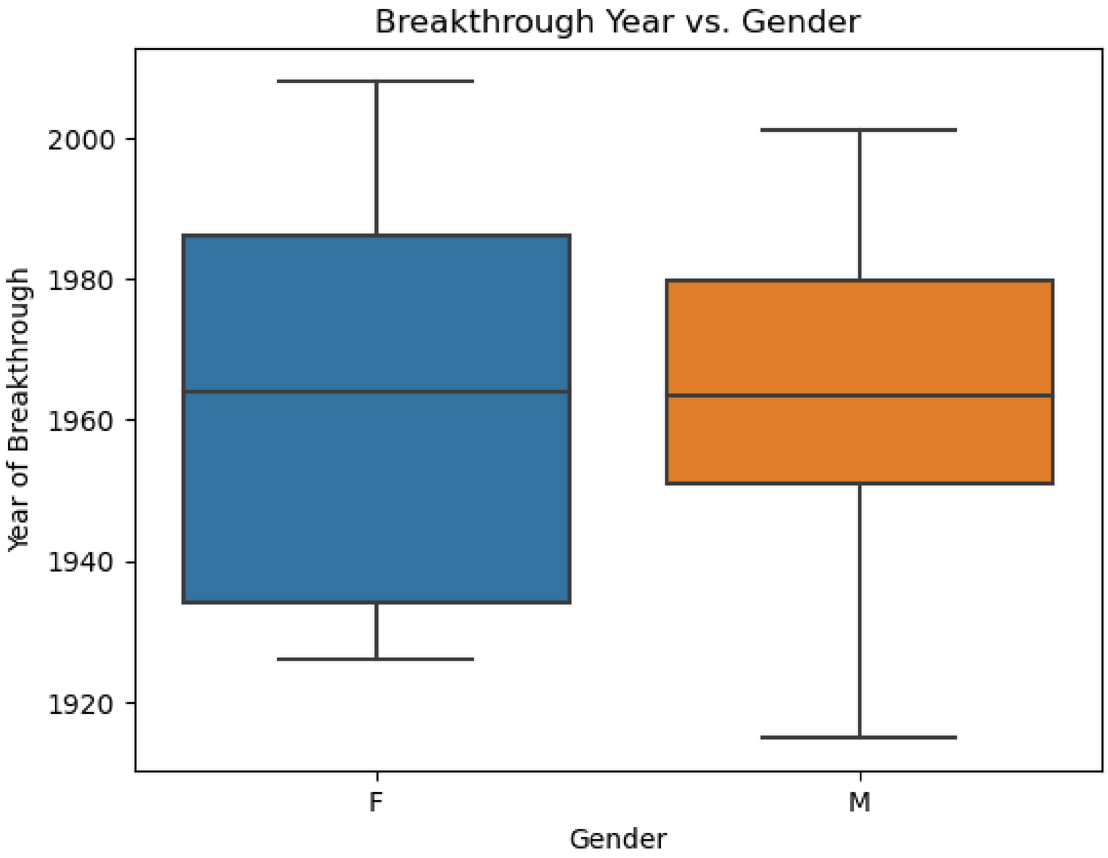
In [18]:

```python
# Calculate age from birth year and visualize age distribution
df['Age'] = pd.Timestamp.now().year - df['Birth Year'].dt.year
plt.hist(df['Age'], bins=20)
plt.title('Age Distribution')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
Cell In[18], line 2
      1 # Calculate age from birth year and visualize age distribution
----> 2 df['Age'] = pd.Timestamp.now().year - df['Birth Year'].dt.year
      3 plt.hist(df['Age'], bins=20)
      4 plt.title('Age Distribution')

File E:\Anaconda\Lib\site-packages\pandas\core\generic.py:5902, in NDFrame.__getattr__(self, name)
   5895 if (
   5896     name not in self._internal_names_set
   5897     and name not in self._metadata
   5898     and name not in self._accessors
   5899     and self._info_axis._can_hold_identifiers_and_holds_name(name)
   5900 ):
   5901     return self[name]
-> 5902 return object.__getattribute__(self, name)

File E:\Anaconda\Lib\site-packages\pandas\core\accessor.py:182, in CachedAccessor.__get__(self, obj, cls)
    179 if obj is None:
    180     # we're accessing the attribute of the class, i.e., Dataset.geo
    181     return self._accessor
--> 182 accessor_obj = self._accessor(obj)
    183 # Replace the property with the accessor object. Inspired by:
    184 # https://www.pydanny.com/cached-property.html (https://www.pydanny.com/cached-property.html)
    185 # We need to use object.__setattr__ because we overwrite __setattr__ on
    186 # NDFrame
    187 object.__setattr__(obj, self._name, accessor_obj)

File E:\Anaconda\Lib\site-packages\pandas\core\indexes\accessors.py:512, in CombinedDatetimelikeProperties.__new__(cls, data)
    509 elif is_period_dtype(data.dtype):
    510     return PeriodProperties(data, orig)
--> 512 raise AttributeError("Can only use .dt accessor with datetimelike values")

AttributeError: Can only use .dt accessor with datetimelike values
```

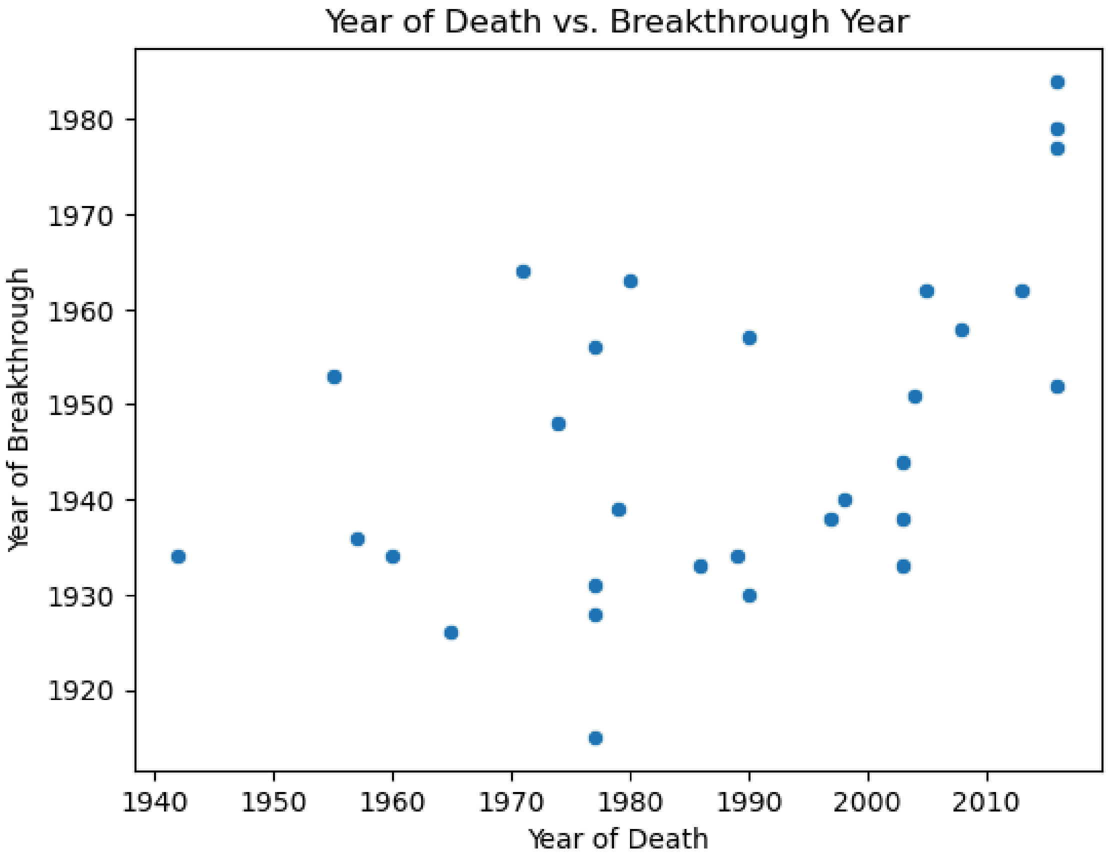
In [19]:

```
# Breakthrough Year vs. Gender
sns.boxplot(data=df, x='Gender', y='Year of Breakthrough/#1 Hit/Award N
plt.title('Breakthrough Year vs. Gender')
plt.xlabel('Gender')
plt.ylabel('Year of Breakthrough')
plt.show()
```

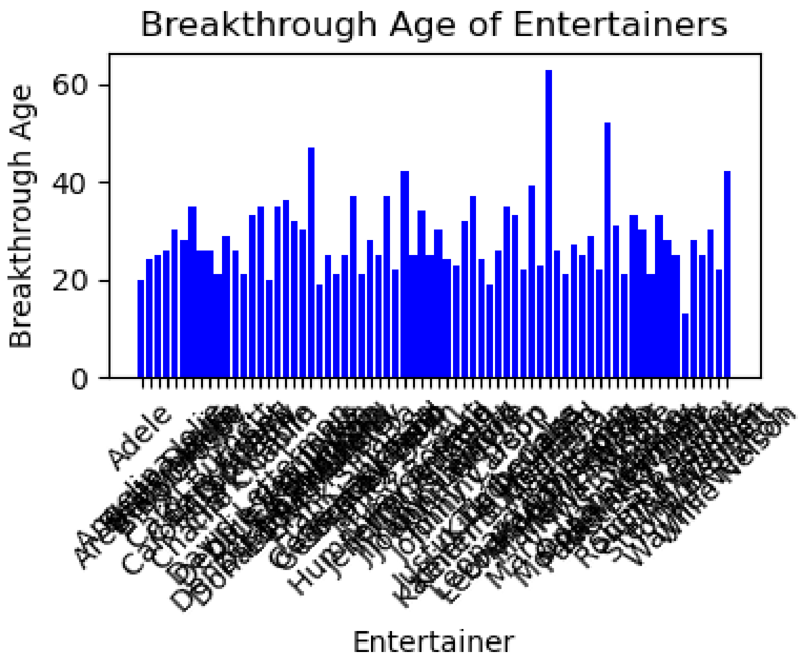


In [20]:

```
# Year of Death vs. Breakthrough Year
sns.scatterplot(data=df, x='Year of Death', y='Year of Breakthrough/#1
plt.title('Year of Death vs. Breakthrough Year')
plt.xlabel('Year of Death')
plt.ylabel('Year of Breakthrough')
plt.show()
```



In [21]:
```
import pandas as pd
import matplotlib.pyplot as plt


# Plotting
plt.figure(figsize=(4, 2))
plt.bar(df['Entertainer'], df['Breakthrough Age'], color='blue')
plt.xlabel('Entertainer')
plt.ylabel('Breakthrough Age')
plt.title('Breakthrough Age of Entertainers')
plt.xticks(rotation=45)
plt.show()

plt.figure(figsize=(10, 6))
plt.bar(df['Entertainer'], df['Year of Breakthrough'], color='green')
plt.xlabel('Entertainer')
plt.ylabel('Year of Breakthrough')
plt.title('Year of Breakthrough for Entertainers')
plt.xticks(rotation=100)
plt.show()
```



```
---------------------------------------------------------------------------
-----
KeyError                                  Traceback (most recent call
last)
File E:\Anaconda\Lib\site-packages\pandas\core\indexes\base.py:3802, i
n Index.get_loc(self, key, method, tolerance)
   3801 try:
-> 3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:

File E:\Anaconda\Lib\site-packages\pandas\_libs\index.pyx:138, in pand
as._libs.index.IndexEngine.get_loc()

File E:\Anaconda\Lib\site-packages\pandas\_libs\index.pyx:165, in pand
as._libs.index.IndexEngine.get_loc()

File pandas\_libs\hashtable_class_helper.pxi:5745, in pandas._libs.has
htable.PyObjectHashTable.get_item()

File pandas\_libs\hashtable_class_helper.pxi:5753, in pandas._libs.has
htable.PyObjectHashTable.get_item()

KeyError: 'Year of Breakthrough'

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call
last)
Cell In[21], line 15
     12 plt.show()
     14 plt.figure(figsize=(10, 6))
---> 15 plt.bar(df['Entertainer'], df['Year of Breakthrough'], color
='green')
     16 plt.xlabel('Entertainer')
     17 plt.ylabel('Year of Breakthrough')

File E:\Anaconda\Lib\site-packages\pandas\core\frame.py:3807, in DataF
rame.__getitem__(self, key)
   3805 if self.columns.nlevels > 1:
   3806     return self._getitem_multilevel(key)
-> 3807 indexer = self.columns.get_loc(key)
   3808 if is_integer(indexer):
   3809     indexer = [indexer]

File E:\Anaconda\Lib\site-packages\pandas\core\indexes\base.py:3804, i
n Index.get_loc(self, key, method, tolerance)
   3802     return self._engine.get_loc(casted_key)
   3803 except KeyError as err:
-> 3804     raise KeyError(key) from err
   3805 except TypeError:
   3806     # If we have a listlike key, _check_indexing_error will ra
ise
   3807     #  InvalidIndexError. Otherwise we fall through and re-rai
se
   3808     #  the TypeError.
   3809     self._check_indexing_error(key)
```

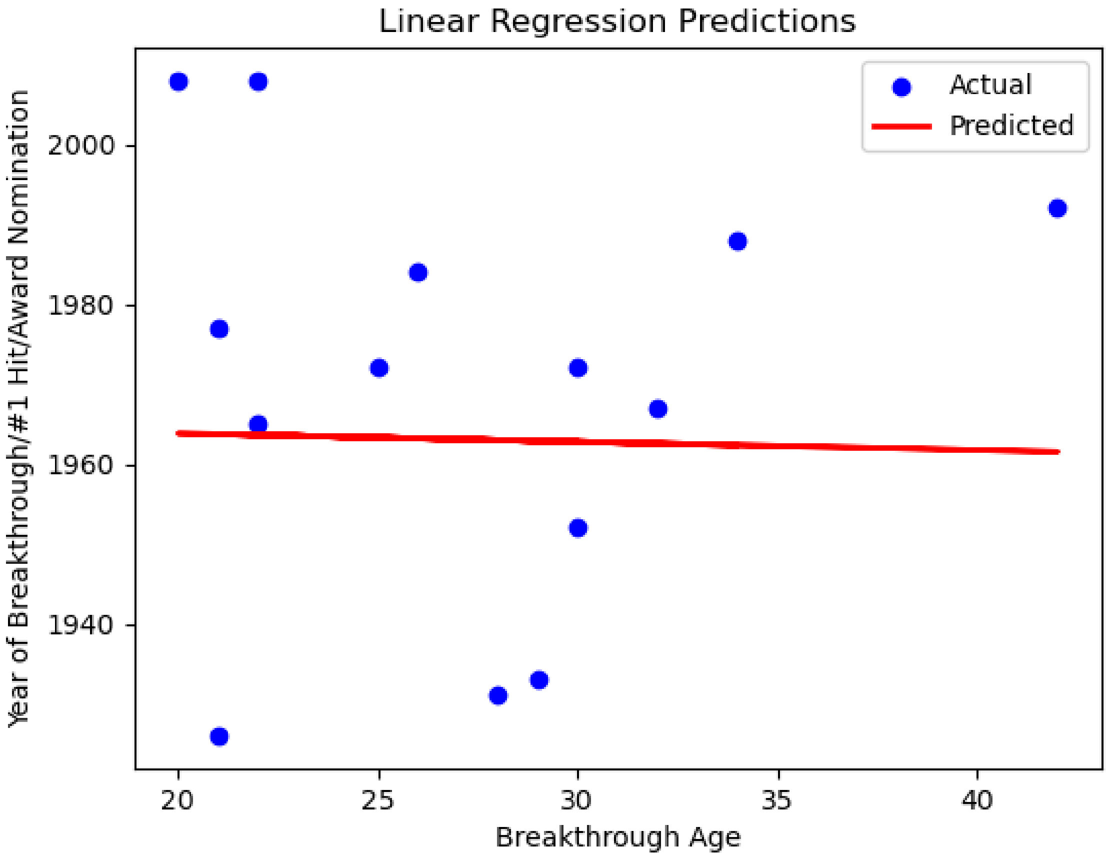
```
KeyError: 'Year of Breakthrough'
```

```
<Figure size 1000x600 with 0 Axes>
```

In [22]:

```
# Import necessary libraries
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score




X = df[['Breakthrough Age']]  # Features
y = df['Year of Breakthrough/#1 Hit/Award Nomination']  # Target

# Split the data into training and testing sets (e.g., 80% training and
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2

# Create a linear regression model
model = LinearRegression()

# Fit the model on the training data
model.fit(X_train, y_train)

# Make predictions on the testing data
y_pred = model.predict(X_test)

# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Print the model's performance metrics
print(f'Mean Squared Error (MSE): {mse}')
print(f'Coefficient of Determination (R^2): {r2}')

# Visualize the predicted vs. actual values (optional)
import matplotlib.pyplot as plt

plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.plot(X_test, y_pred, color='red', linewidth=2, label='Predicted')
plt.xlabel('Breakthrough Age')
plt.ylabel('Year of Breakthrough/#1 Hit/Award Nomination')
plt.legend()
plt.title('Linear Regression Predictions')
plt.show()
```

```
Mean Squared Error (MSE): 696.8716758192621
Coefficient of Determination (R^2): -0.06844535197615209
```


Linear Regression Predictions
Actual
Predicted
Year of Breakthrough/#1 Hit/Award Nomination
2000
1980
1960
1940
20
25
30
35
40
Breakthrough Age

```
In [ ]:
```

```
In [ ]:
```