1. Install the scikit-learn Library

In [3]:
```
!pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\asus\anaconda3\newanaconda3\lib\site-packages (1.4.2)
Requirement already satisfied: numpy>=1.19.5 in c:\users\asus\anaconda3\newanaconda3\lib\site-packages (from scikit-learn) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in c:\users\asus\anaconda3\newanaconda3\lib\site-packages (from scikit-learn) (1.13.1)
Requirement already satisfied: joblib>=1.2.0 in c:\users\asus\anaconda3\newanaconda3\lib\site-packages (from scikit-learn) (1.4.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\asus\anaconda3\newanaconda3\lib\site-packages (from scikit-learn) (2.2.0)
```

2. Import the Libraries

In [8]:
```
import pandas as pd
import numpy as np
import os
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns
# Create path
path = r'C:\Users\Asus\Music\achievement 6 project'

# Load the Titanic dataset
data = pd.read_csv(os.path.join(path, 'Data', 'tested.csv'), index_col=False)

# Display the first few rows to get an overview of the data
data.head()
```

Out[8]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	0	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	1	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	0	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	0	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	1	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

3. Clean our Data

In [11]:
```
# Check for missing values in the dataset
print(data.isnull().sum())
```

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

In [13]:
```
# Handle Missing Values
# Fill missing values in 'Age' with the median age
data['Age'] = data['Age'].fillna(data['Age'].median())

# Check if 'Ticket' and 'Cabin' columns exist before dropping them
columns_to_drop = ['Ticket', 'Cabin']
existing_columns_to_drop = [col for col in columns_to_drop if col in data.columns]

# Drop columns with too many missing values or irrelevant columns
data = data.drop(existing_columns_to_drop, axis=1)

# Drop rows with missing values in essential columns
data = data.dropna(subset=['Fare', 'Embarked'])
```

In [15]:
```
# Check for Duplicates
# Check for duplicate rows
print(data.duplicated().sum())
```

```
0
```

In [17]:
```
# Drop duplicate rows if any
data = data.drop_duplicates()
```

In [19]:
```
# Check for Consistency
# Check the unique values in 'Sex' and 'Embarked'
print(data['Sex'].value_counts())
print(data['Embarked'].value_counts())
```

```
Sex
male      265
female    152
Name: count, dtype: int64
Embarked
S    269
C    102
Q     46
Name: count, dtype: int64
```

In [21]:
```
# Convert 'Sex' and 'Embarked' columns to lowercase (if needed)
data['Sex'] = data['Sex'].str.lower()
data['Embarked'] = data['Embarked'].str.upper()
```

In [23]:
```
#  Review the Cleaned Data
# Display the first few rows of the cleaned data
print(data.head())

# Display the data types and non-null counts
print(data.info())
```

```
   PassengerId  Survived  Pclass  \
0          892         0       3
1          893         1       3
2          894         0       2
3          895         0       3
4          896         1       3

                                           Name     Sex   Age  SibSp  Parch  \
0                              Kelly, Mr. James    male  34.5      0      0
1              Wilkes, Mrs. James (Ellen Needs)  female  47.0      1      0
2                     Myles, Mr. Thomas Francis    male  62.0      0      0
3                              Wirz, Mr. Albert    male  27.0      0      0
4  Hirvonen, Mrs. Alexander (Helga E Lindqvist)  female  22.0      1      1

      Fare Embarked
0   7.8292        Q
1   7.0000        S
2   9.6875        Q
3   8.6625        S
4  12.2875        S
<class 'pandas.core.frame.DataFrame'>
Index: 417 entries, 0 to 417
Data columns (total 10 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   PassengerId  417 non-null    int64
 1   Survived     417 non-null    int64
 2   Pclass       417 non-null    int64
 3   Name         417 non-null    object
 4   Sex          417 non-null    object
 5   Age          417 non-null    float64
 6   SibSp        417 non-null    int64
 7   Parch        417 non-null    int64
 8   Fare         417 non-null    float64
 9   Embarked     417 non-null    object
dtypes: float64(2), int64(5), object(3)
memory usage: 35.8+ KB
None
```

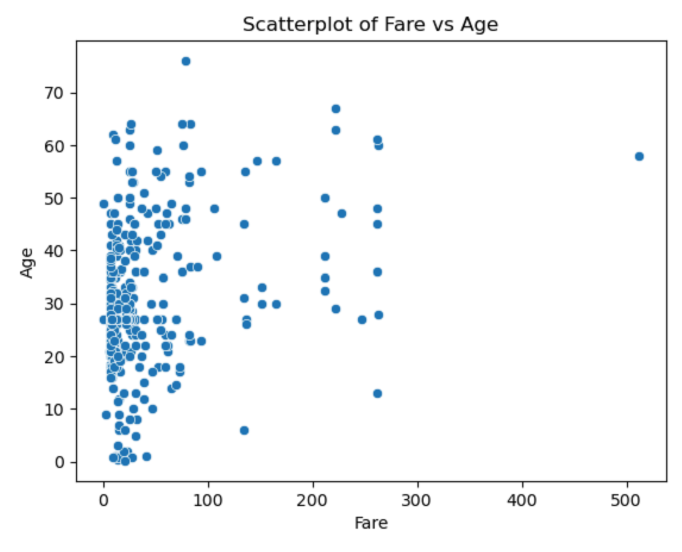
In [25]:
```
# Perform Descriptive Statistical Analysis
# Display descriptive statistics
print(data.describe(include='all'))
```

```
        PassengerId    Survived      Pclass              Name   Sex  \
count    417.000000  417.000000  417.000000               417   417
unique          NaN         NaN         NaN               417     2
top             NaN         NaN         NaN  Kelly, Mr. James  male
freq            NaN         NaN         NaN                 1   265
mean    1100.635492    0.364508    2.263789               NaN   NaN
std      120.923774    0.481870    0.842077               NaN   NaN
min      892.000000    0.000000    1.000000               NaN   NaN
25%      996.000000    0.000000    1.000000               NaN   NaN
50%     1101.000000    0.000000    3.000000               NaN   NaN
75%     1205.000000    1.000000    3.000000               NaN   NaN
max     1309.000000    1.000000    3.000000               NaN   NaN

               Age       SibSp       Parch        Fare Embarked
count   417.000000  417.000000  417.000000  417.000000      417
unique         NaN         NaN         NaN         NaN        3
top            NaN         NaN         NaN         NaN        S
freq           NaN         NaN         NaN         NaN      269
mean     29.525180    0.448441    0.393285   35.627188      NaN
std      12.628258    0.897568    0.982419   55.907576      NaN
min       0.170000    0.000000    0.000000    0.000000      NaN
25%      23.000000    0.000000    0.000000    7.895800      NaN
50%      27.000000    0.000000    0.000000   14.454200      NaN
75%      35.000000    1.000000    0.000000   31.500000      NaN
max      76.000000    8.000000    9.000000  512.329200      NaN
```

4. Explore Your Data Visually

Let's create a scatterplot to explore the relationship between two variables. For instance, you can look at how Fare (independent variable) relates to Age (dependent variable)

In [29]:
```
sns.scatterplot(x='Fare', y='Age', data=data)
plt.title('Scatterplot of Fare vs Age')
plt.show()
```



5. State of our Hypothesis

Hypothesis

There is a linear relationship between the fare paid by passengers and their age.

6. Reshape the Variables

Convert Fare (X) and Age (y) into NumPy arrays for modeling:

In [34]:
```
X = data['Fare'].values.reshape(-1, 1)
y = data['Age'].values.reshape(-1, 1)
```

7. Split the Data

Split the data into training and test sets:

In [39]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

8. Run a Linear Regression

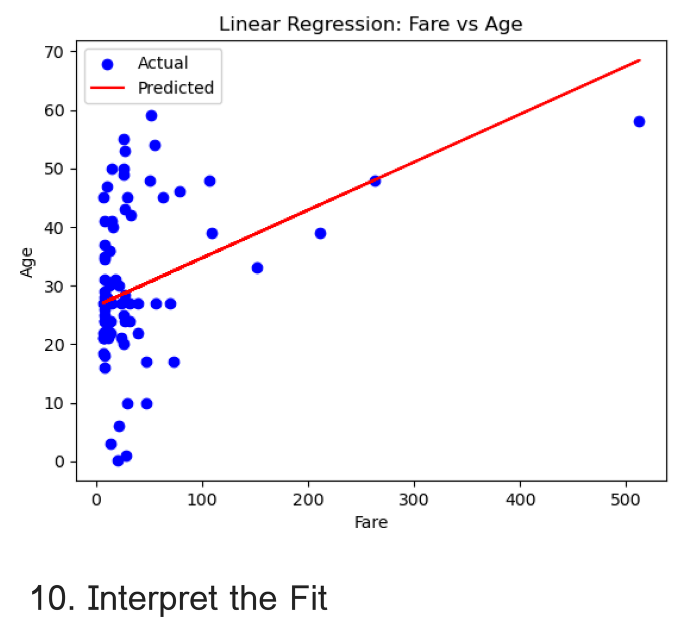
Fit the linear regression model to the training set and make predictions on the test set:

In [42]:
```
# Initialize the model
model = LinearRegression()

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

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

9. Create a Plot Showing the Regression Line

Plot the regression line against the test data:

In [45]:
```
plt.scatter(X_test, y_test, color='blue', label='Actual')
plt.plot(X_test, y_pred, color='red', label='Predicted')
plt.title('Linear Regression: Fare vs Age')
plt.xlabel('Fare')
plt.ylabel('Age')
plt.legend()
plt.show()
```



10. Interpret the Fit

The regression line in the plot does not fit the data well. The scatter of actual data points around the line is quite dispersed, indicating that there is not a strong linear relationship between Fare and Age. Most of the data points are clustered near the lower end of the Fare axis, and the large spread of points away from the regression line suggests that other factors, aside from Age, are likely influencing Fare. The outlier with a high fare and an average age further shows that the model may not adequately capture the relationship between these variables.

11. Check Model Performance

Calculate the Mean Squared Error (MSE) and R2 score to evaluate the model's performance:

In [50]:
```
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

print(f'Mean Squared Error: {mse}')
print(f'R2 Score: {r2}')
```

```
Mean Squared Error: 143.64374599072613
R2 Score: 0.12521718804752469
```

12. Compare Predicted and Actual Values

Create a DataFrame to compare the predicted y values with the actual y values:

In [53]:
```
comparison = pd.DataFrame({'Actual': y_test.flatten(), 'Predicted': y_pred.flatten()})
print(comparison.head())
```

```
   Actual  Predicted
0    36.0  27.489381
1    22.0  27.128257
2    28.0  27.128257
3    25.0  28.621233
4    24.0  27.126209
```

13. reflections I have on the impact of possible data bias

Model Performance on the Test Set

The linear regression model was evaluated using the test set to predict the `Age` of passengers based on the `Fare` they paid. Here are the key insights:

1. **Model Performance Metrics:**
   - **Mean Squared Error (MSE):** 143.64
   - **R2 Score:** 0.125

   The MSE indicates that there is a considerable average squared difference between the actual and predicted values, which suggests that the model's predictions are not very accurate. The R2 score of 0.125 means that the model explains only about 12.5% of the variance in the `Age` based on `Fare`, highlighting that the linear model has a weak explanatory power in this context.

2. **Visual Analysis:**
   - The scatterplot and regression line show a poor fit, with the regression line failing to capture the true relationship between `Fare` and `Age`. The points are dispersed around the line, indicating that the model does not accurately represent the relationship between these variables.

3. **Comparison of Predicted and Actual Values:**
   - The DataFrame comparing predicted and actual values reveals significant deviations, with predicted ages not aligning closely with the actual ages. This discrepancy further suggests that the model is not performing well.

Reflections on Data Bias

1. **Potential Data Bias:**
   - **Sampling Bias:** The dataset used for training and testing might not be representative of the broader population, which could skew the model's performance. Ensuring that the dataset is representative is crucial for accurate predictions.
   - **Feature Relevance:** The model uses only `Fare` to predict `Age`, which may not capture the full complexity of the relationship. Other features, such as `Pclass`, `Sex`, and `Embarked`, could provide additional context and improve the model's performance.
   - **Outliers:** Outliers in the dataset can significantly impact model performance. For example, passengers with unusually high fares and average ages may distort the regression line. Handling outliers or using robust regression techniques could improve model accuracy.
   - **Data Quality:** The presence of missing or inconsistent data can affect the model. Although missing values were addressed, ongoing data quality checks are essential to ensure reliable predictions.
2. **Future Improvements:**
   - **Feature Engineering:** Including additional relevant features or creating interaction terms might better capture the relationship between `Fare` and `Age`.
   - **Model Selection:** Exploring more complex models or machine learning techniques might improve performance if a linear regression model proves insufficient.
   - **Data Exploration:** Conducting a deeper exploratory data analysis to understand relationships between variables and potential biases can lead to better modeling strategies.

In summary, while the linear regression model provides some insights, its performance on the test set indicates that it may not be the most suitable model for predicting `Age` based on `Fare`. Addressing potential biases and exploring additional features or models could enhance prediction accuracy and overall model performance.

In [ ]: