```
print('Module 2-DESCRIPTIVE ANALYTICS ')
```

```
Module 2-DESCRIPTIVE ANALYTICS
```

Frequency distributions – Outliers –Interpreting distributions – Graphs – Describing variability – Interquartile range – Variability for qualitative and ranked data - Normal distributions – Z scores –correlation – scatter plots – Regression – regression line – least squares regression line – standard error of estimate – Interpretation of r2 – Multiple regression equations .

Frequency Distributions

A frequency distribution is a summary of how often different values occur in a dataset. It helps analyze categorical and numerical data by grouping values into ranges or categories.

For categorical data: It counts the occurrences of each unique value.

For numerical data: It groups values into bins (ranges) and counts occurrences.

Dataset used : df = sns.load_dataset("titanic")

```
# Import necessary libraries
import seaborn as sns
import pandas as pd

# Load Titanic dataset
df = sns.load_dataset("titanic")

# Frequency distribution for categorical column 'class'
freq_class = df['class'].value_counts()
print("Frequency Distribution of Passenger Classes:\n", freq_class)

# Frequency distribution for numerical column 'age' (grouped into bins)
freq_age = pd.cut(df['age'].dropna(), bins=5).value_counts()
print("\nFrequency Distribution of Age Groups:\n", freq_age)
```

```
Frequency Distribution of Passenger Classes:
 class
Third     491
First     216
Second    184
Name: count, dtype: int64

Frequency Distribution of Age Groups:
 age
(16.336, 32.252]    346
(32.252, 48.168]    188
(0.34, 16.336]      100
(48.168, 64.084]     69
(64.084, 80.0]       11
Name: count, dtype: int64
```

Double-click (or enter) to edit

Outliers

An outlier is a value that is significantly different from other observations in a dataset.

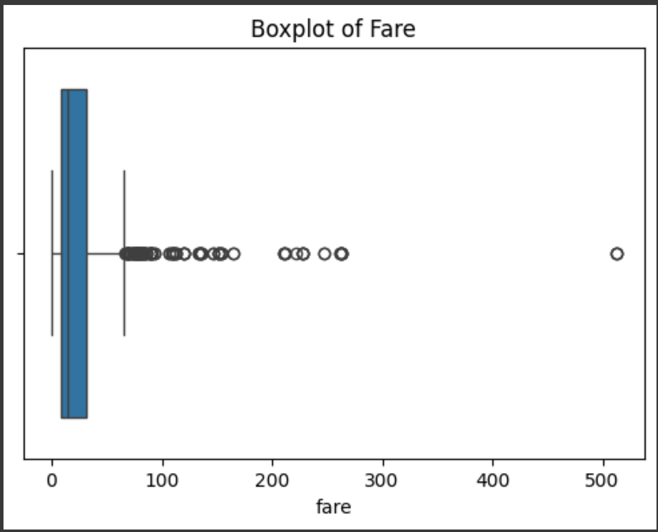
Causes of Outliers: Data entry errors, measurement variability, or genuine rare events.

Effect of Outliers: Can distort statistical models and affect accuracy.

Detection Methods: Boxplots, Z-scores, and Interquartile Range (IQR).

```
import matplotlib.pyplot as plt

# Boxplot to visualize outliers in 'fare'
plt.figure(figsize=(6,4))
sns.boxplot(x=df['fare'])
plt.title("Boxplot of Fare")
plt.show()

# Identifying outliers using IQR
Q1 = df['fare'].quantile(0.25)
Q3 = df['fare'].quantile(0.75)
IQR = Q3 - Q1

outliers = df[(df['fare'] < (Q1 - 1.5 * IQR)) | (df['fare'] > (Q3 + 1.5 * IQR))]
```

```
print("Outliers in Fare Column:\n", outliers[['fare']])
```




```
Outliers in Fare Column:
          fare
1     71.2833
27   263.0000
31   146.5208
34    82.1708
52    76.7292
..        ...
846   69.5500
849   89.1042
856  164.8667
863   69.5500
879   83.1583

[116 rows x 1 columns]
```

Interpreting Distributions:

A distribution describes how data points are spread.

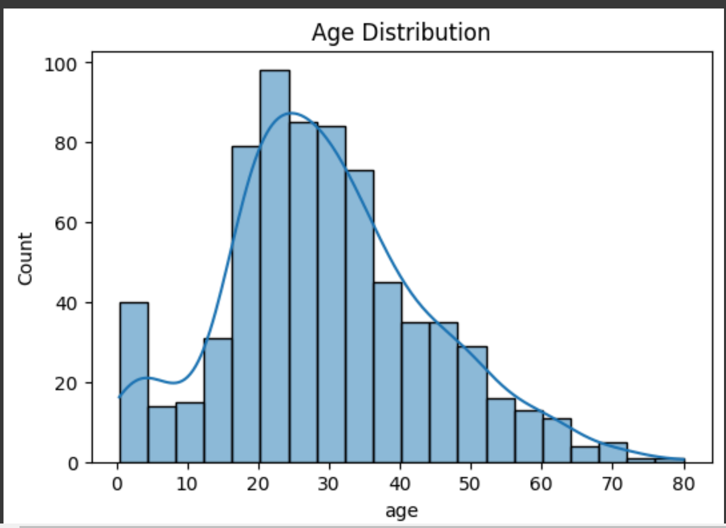
Left-Skewed: More values on the right.

Right-Skewed: More values on the left.

Normal Distribution: Bell-shaped curve.

Output Explanation : A boxplot showing extreme values in Fare and a list of outlier row

```
# Histogram to visualize distribution of 'age'
plt.figure(figsize=(6,4))
sns.histplot(df['age'].dropna(), bins=20, kde=True)
plt.title("Age Distribution")
plt.show()
```




Output explanation : A histogram showing the age distribution with a KDE (Kernel Density Estimation) curve.

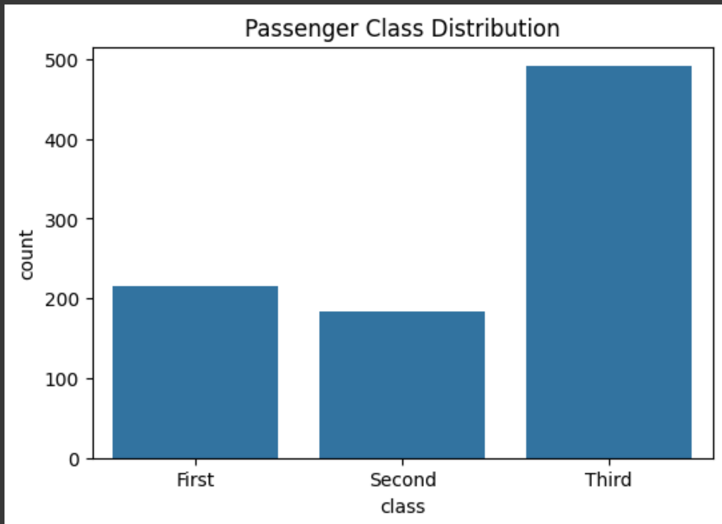
Graphs (Visual Representations):

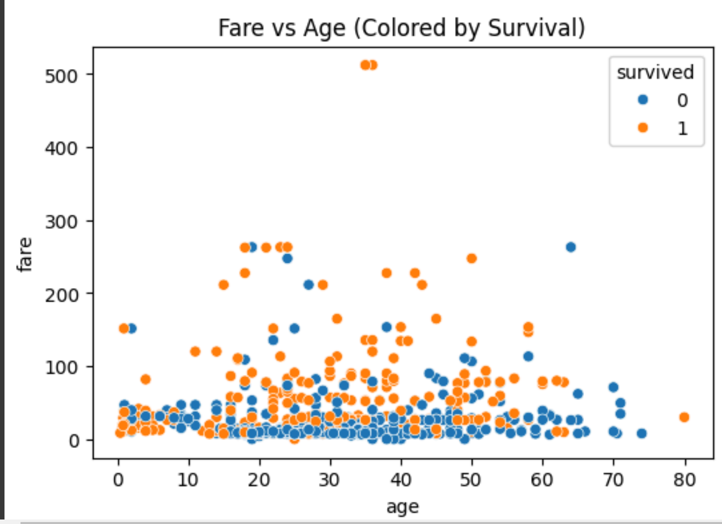
Graphs help visualize relationships and distributions in data.

Bar charts: Show categorical data.

Scatter plots: Show numerical relationships.

```
# Bar chart for class distribution
plt.figure(figsize=(6,4))
sns.countplot(x=df['class'])
plt.title("Passenger Class Distribution")
plt.show()

# Scatter plot for Fare vs Age
plt.figure(figsize=(6,4))
sns.scatterplot(x=df['age'], y=df['fare'], hue=df['survived'])
plt.title("Fare vs Age (Colored by Survival)")
plt.show()
```




Describing Variability (Range, IQR, Standard Deviation)

Variability describes how spread out data values are.

Range: Difference between max & min.

Standard Deviation: Average distance from the mean.

Interquartile Range (IQR): Spread of the middle 50% of data

```
# Compute range
age_range = df['age'].max() - df['age'].min()
print("Age Range:", age_range)

# Compute standard deviation
std_age = df['age'].std()
print("Standard Deviation of Age:", std_age)
```

```
# Compute Interquartile Range (IQR)
Q1 = df['age'].quantile(0.25)
Q3 = df['age'].quantile(0.75)
iqr = Q3 - Q1
print("Interquartile Range (IQR) of Age:", iqr)
```

```
Age Range: 79.58
Standard Deviation of Age: 14.526497332334044
Interquartile Range (IQR) of Age: 17.875
```

Variability of qualitative and ranked data

Variability measures the spread or dispersion of data. While numerical data uses range, standard deviation, and IQR, categorical (qualitative) and ranked (ordinal) data require different measures:

Qualitative Data Variability:

Mode: Most frequent category.

Diversity Index: Measures the proportion of categories.

Ranked Data Variability (Ordinal Data):

Median & Percentiles: To measure the central spread.

Interquartile Range (IQR): Used for ranked data

```
# Mode for categorical column
mode_class = df['class'].mode()[0]
print("Most Frequent Passenger Class (Mode):", mode_class)

# Diversity index: Proportion of each class
class_proportions = df['class'].value_counts(normalize=True)
print("\nDiversity Index (Proportion of Classes):\n", class_proportions)

# Interquartile Range (IQR) for ranked data (Pclass: 1, 2, 3)
iqr_pclass = df['pclass'].quantile(0.75) - df['pclass'].quantile(0.25)
print("\nInterquartile Range (IQR) for Passenger Class:", iqr_pclass)
```

```
Most Frequent Passenger Class (Mode): Third

Diversity Index (Proportion of Classes):
 class
Third     0.551066
First     0.242424
Second    0.206510
Name: proportion, dtype: float64

Interquartile Range (IQR) for Passenger Class: 1.0
```

Normal Distributions & Z-Scores:

Normal Distribution: Data follows a symmetrical bell-curve.

Z-Score: Measures how many standard deviations a data point is from the mean.

```
from scipy.stats import zscore

# Compute Z-scores
df['age_zscore'] = zscore(df['age'].dropna())
print("Z-Scores for Age:\n", df[['age', 'age_zscore']].head())
```

```
Z-Scores for Age:
    age  age_zscore
0  22.0   -0.530377
1  38.0    0.571831
2  26.0   -0.254825
3  35.0    0.365167
4  35.0    0.365167
```

Correlation & Scatter Plots:

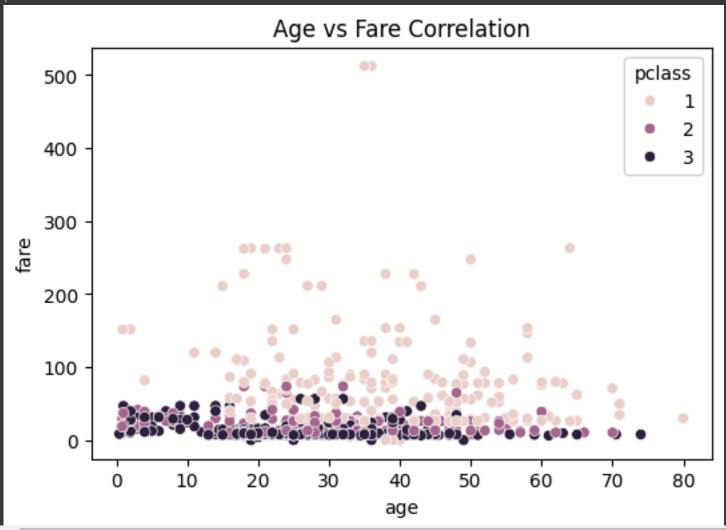
Correlation (r): Measures the strength of relationships between variables (-1 to 1).

Scatter Plot: Visualizes relationships between variables.

```
# Compute correlation matrix
corr_matrix = df[['age', 'fare', 'pclass']].corr()
print("Correlation Matrix:\n", corr_matrix)

# Scatter plot for correlation
plt.figure(figsize=(6,4))
```

```
sns.scatterplot(x=df['age'], y=df['fare'], hue=df['pclass'])
plt.title("Age vs Fare Correlation")
plt.show()
```

```
Correlation Matrix:
             age      fare    pclass
age     1.000000  0.096067 -0.369226
fare    0.096067  1.000000 -0.549500
pclass -0.369226 -0.549500  1.000000
```




Regression (Simple & Multiple):

Simple Regression: One independent variable predicts a dependent variable.

Multiple Regression: Multiple independent variables predict an outcome.

```
#Simple Regression
from sklearn.linear_model import LinearRegression

# Prepare data
df_clean = df.dropna(subset=['age', 'fare'])
X = df_clean[['age']]
y = df_clean['fare']

# Train Linear Regression model
model = LinearRegression()
model.fit(X, y)

# Predict fares
predicted_fare = model.predict(X[:5])

print("Predicted Fares for first 5 passengers:", predicted_fare)
```

```
Predicted Fares for first 5 passengers: [32.00010245 37.59952136 33.39995717 36.54963031 36.54963031]
```

Regression line A regression line is a straight line that best fits the relationship between independent (X) and dependent (Y) variables in linear regression.

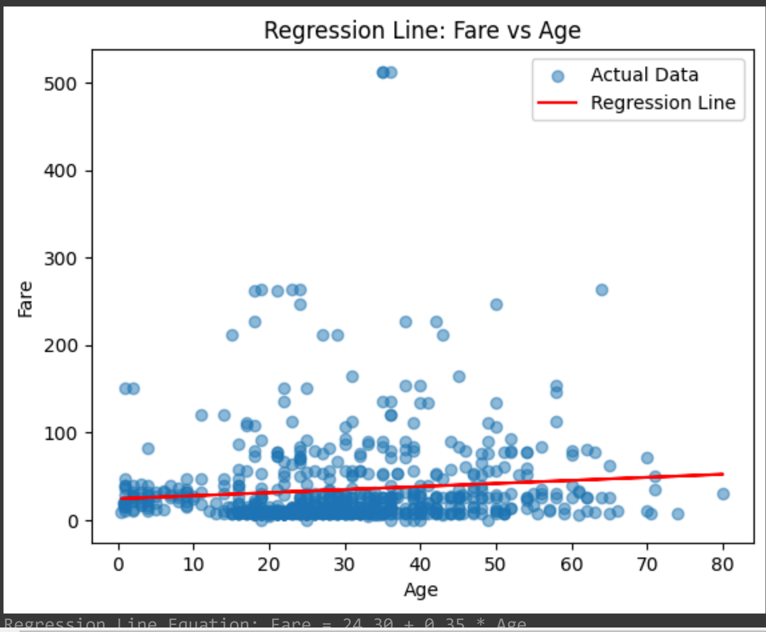
Formula of Regression Line: Y=a+bX

a = Intercept (where the line crosses Y-axis)

b = Slope (rate of change)

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

# Clean dataset
df_clean = df.dropna(subset=['age', 'fare'])

# Independent (X) and Dependent (Y) variable
X = df_clean[['age']]
y = df_clean['fare']

# Train Linear Regression Model
model = LinearRegression()
model.fit(X, y)

# Get line equation parameters
```

```
intercept = model.intercept_
slope = model.coef_[0]

# Generate predictions
predicted_fares = model.predict(X)

# Plot regression line
plt.scatter(df_clean['age'], df_clean['fare'], label="Actual Data", alpha=0.5)
plt.plot(df_clean['age'], predicted_fares, color='red', label="Regression Line")
plt.xlabel("Age")
plt.ylabel("Fare")
plt.title("Regression Line: Fare vs Age")
plt.legend()
plt.show()

print(f"Regression Line Equation: Fare = {intercept:.2f} + {slope:.2f} * Age")
```




```
Regression Line Equation: Fare = 24.30 + 0.35 * Age
```

Output Explanation : A scatter plot with a red regression line showing the relationship between Age and Fare.

Least Squares Regression Line Definition & Explanation The Least Squares Regression Line (LSRL) minimizes the sum of squared differences between actual values and predicted values.

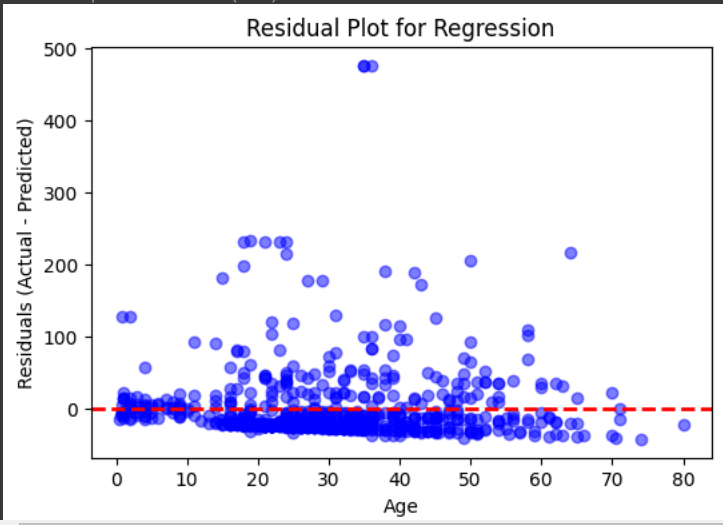
It finds the best-fit line by reducing the Sum of Squared Errors (SSE)

$$SSE = \sum (y_i - \hat{y}_i)^2$$

where $y_i$ is actual value and $\hat{y}_i$ is predicted value.

```
# Compute residuals (actual - predicted)
residuals = df_clean['fare'] - predicted_fares

# Compute Sum of Squared Errors (SSE)
SSE = np.sum(residuals**2)

print("Sum of Squared Errors (SSE):", SSE)

# Plot residuals
plt.figure(figsize=(6,4))
plt.scatter(df_clean['age'], residuals, color='blue', alpha=0.5)
plt.axhline(y=0, color='red', linestyle='--', linewidth=2)
plt.xlabel("Age")
plt.ylabel("Residuals (Actual - Predicted)")
plt.title("Residual Plot for Regression")
plt.show()
```

```
Sum of Squared Errors (SSE): 1978267.427006883
```




Standard Error of Estimate (SEE)

The Standard Error of Estimate (SEE) measures the accuracy of predictions from a regression model.

$$SEE = \sqrt{\frac{SSE}{n-2}}$$

where **n** is the number of observations.

Lower SEE -> Best Fit

```
n = len(df_clean)  # Number of observations
SEE = np.sqrt(SSE / (n - 2))
print("Standard Error of Estimate (SEE):", SEE)
```

```
Standard Error of Estimate (SEE): 52.711151451748975
```

Output explanation : (A lower value indicates a better model fit.)

Interpretation of r2

r2 measures how much of the variance in the dependent variable (Y) is explained by the independent variable (X).

$$R^2 = 1 - \frac{SSE}{SST}$$

where:

- $SST$ = Total Sum of Squares (total variance in Y)
- $SSE$ = Sum of Squared Errors (unexplained variance)

✔ **Interpretation:**

- $R^2 = 1$ → Perfect model (explains 100% variance)
- $R^2 = 0$ → Model explains nothing

```
# Compute Total Sum of Squares (SST)
SST = np.sum((df_clean['fare'] - df_clean['fare'].mean())**2)

# Compute R-squared
R_squared = 1 - (SSE / SST)
print("Coefficient of Determination (R^2):", R_squared)
```

```
Coefficient of Determination (R^2): 0.009228809267447624
```

```
# Multiple Regression (Age, Pclass -> Fare)
X_multi = df_clean[['age', 'pclass']]
y_multi = df_clean['fare']

# Train model
multi_model = LinearRegression()
multi_model.fit(X_multi, y_multi)

# Predict
pred_multi = multi_model.predict(X_multi[:5])
print("Predicted Fares (Multiple Regression):", pred_multi)
```

```
Predicted Fares (Multiple Regression): [ 9.27869203 77.78353111  7.44722704 79.15712986  3.32643081]
```

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.

Start coding or generate with AI.