```
from google.colab import files
upload=files.upload()
```

Choose Files anime.csv

- **anime.csv**(text/csv 36463 bytes, last modified: 4/29/2025 - 100% done

Start coding or generate with AI.

```
import pandas as pd
df=pd.read_csv("anime.csv")
df
```

	anime_id	name	genre	type	episodes	rating	members
0	32281	Kimi no Na wa.	Drama, Romance, School, Supernatural	Movie	1	9.37	200630
1	5114	Fullmetal Alchemist: Brotherhood	Action, Adventure, Drama, Fantasy, Magic, Mili...	TV	64	9.26	793665
2	28977	Gintama°	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.25	114262
3	9253	Steins;Gate	Sci-Fi, Thriller	TV	24	9.17	673572
4	9969	Gintama'	Action, Comedy, Historical, Parody, Samurai, S...	TV	51	9.16	151266
...	...	...	...	...	...	...	...
12289	9316	Toushindai My Lover: Minami tai Mecha-Minami	Hentai	OVA	1	4.15	211
12290	5543	Under World	Hentai	OVA	1	4.28	183
12291	5621	Violence Gekiga David no Hoshi	Hentai	OVA	4	4.88	219
12292	6133	Violence Gekiga Shin David no Hoshi: Inma Dens...	Hentai	OVA	1	4.98	175
12293	26081	Yasuji no Pornorama: Yacchimae!!	Hentai	Movie	1	5.46	142

12294 rows × 7 columns

Next steps: Generate code with df | View recommended plots | New interactive sheet

Generate Using dataframe: df suggest a plot Close

```
df.shape
```

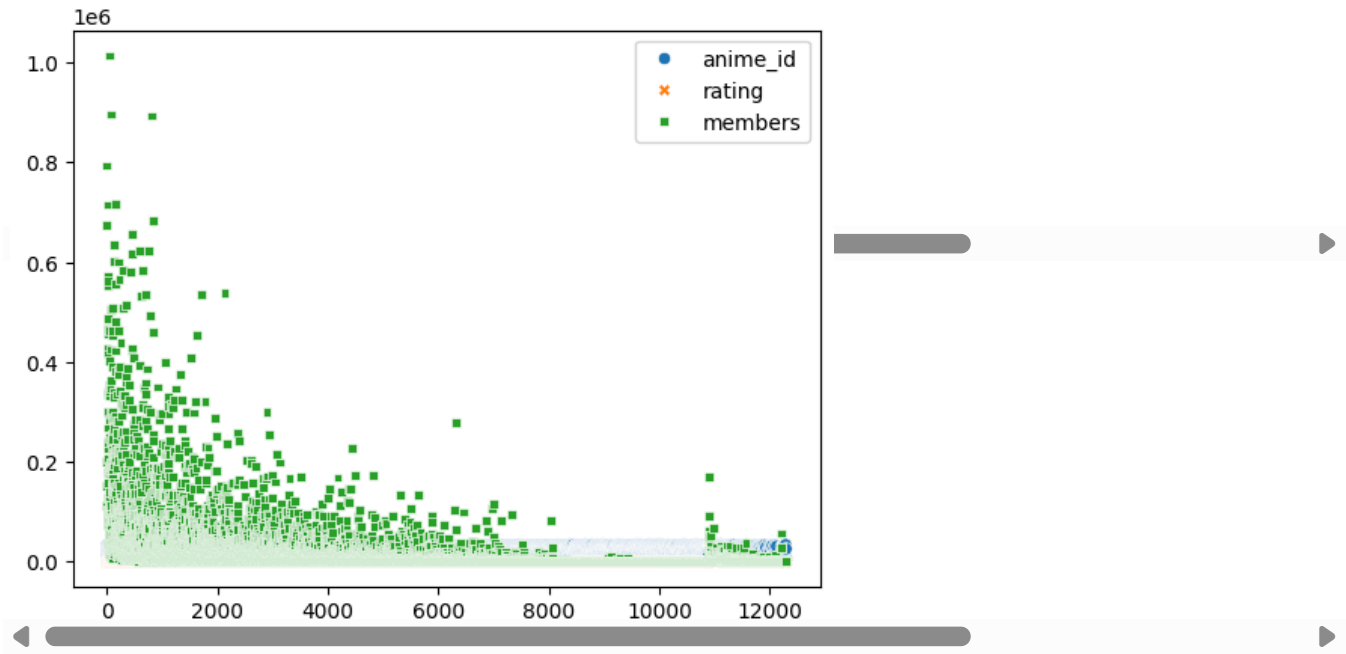
```
(12294, 7)
```

Generate randomly select 5 items from a list Close

```
df.isnull().sum()
```

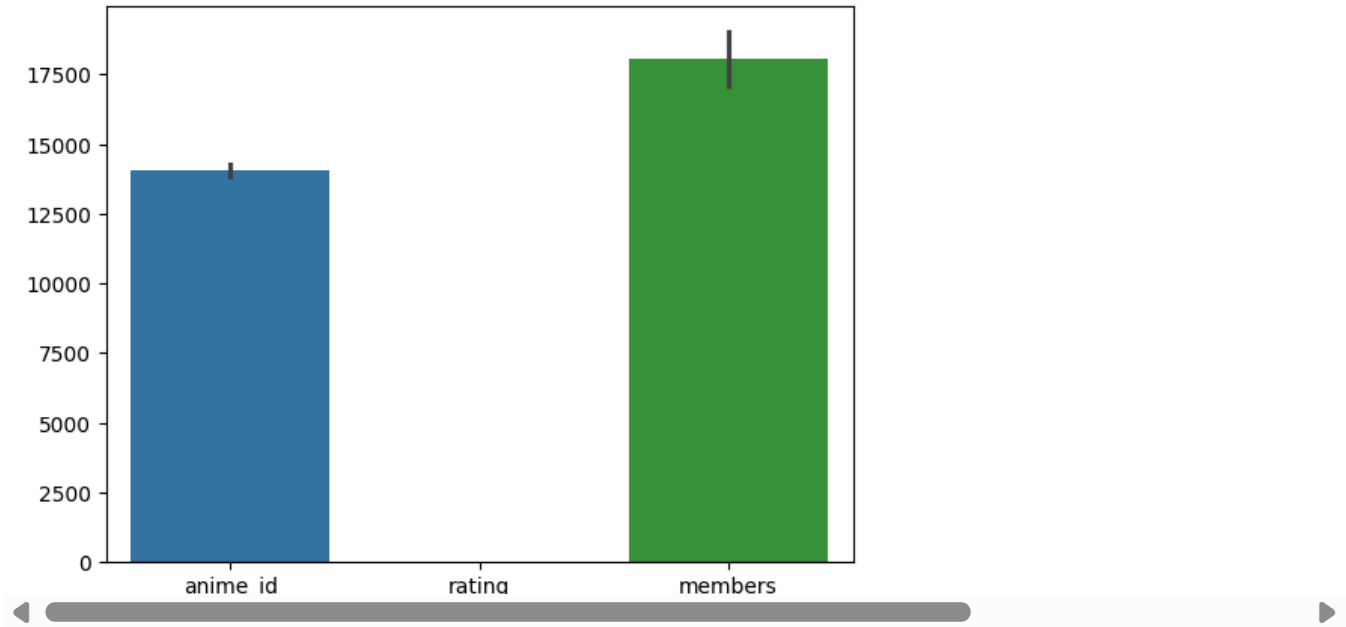
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.scatterplot(df)
```

```
<Axes: genre>      62
```




```
import seaborn as sns
import matplotlib.pyplot as plt
sns.barplot(df)
```

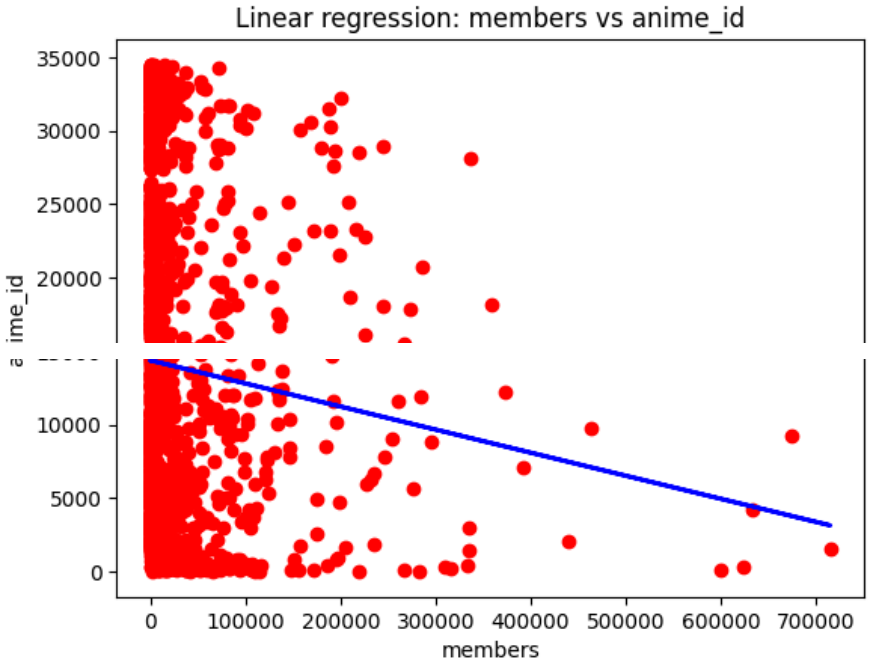
```
<Axes: >
```




```
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
# Load the diabetes dataset (replace 'diabetes.csv' with the actual path if needed)
df = pd.read_csv('anime.csv')
# Assuming 'BMI' is a column in your dataset and you want to predict 'Glucose'
X = df[['members']]  # Features (BMI in this case)
y = df['anime_id']  # Target variable (Glucose)
# Split the data into training and testing sets
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)  # Adjust test_size and rand
# Create and train a Linear Regression model
model = LinearRegression()
model.fit(x_train, y_train)
```

```
# Make predictions on the test set
y_pred = model.predict(x_test)
# Now you can plot the results
plt.scatter(x_test, y_test, color='red', label='Actual')
plt.plot(x_test, y_pred, color='blue', linewidth=2, label='Predicted')
plt.xlabel('members')
plt.ylabel('anime_id')
plt.title('Linear regression: members vs anime_id')
```

```
Text(0.5, 1.0, 'Linear regression: members vs anime_id')
```




```
print(df.columns)
```

```
Index(['anime_id', 'name', 'genre', 'type', 'episodes', 'rating', 'members'], dtype='object')
```

```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
import matplotlib.pyplot as plt

# Assuming 'rating' is your target variable
y = df['rating']  # Replace 'rating' with your actual target column name

# ... (rest of your code) ...
```

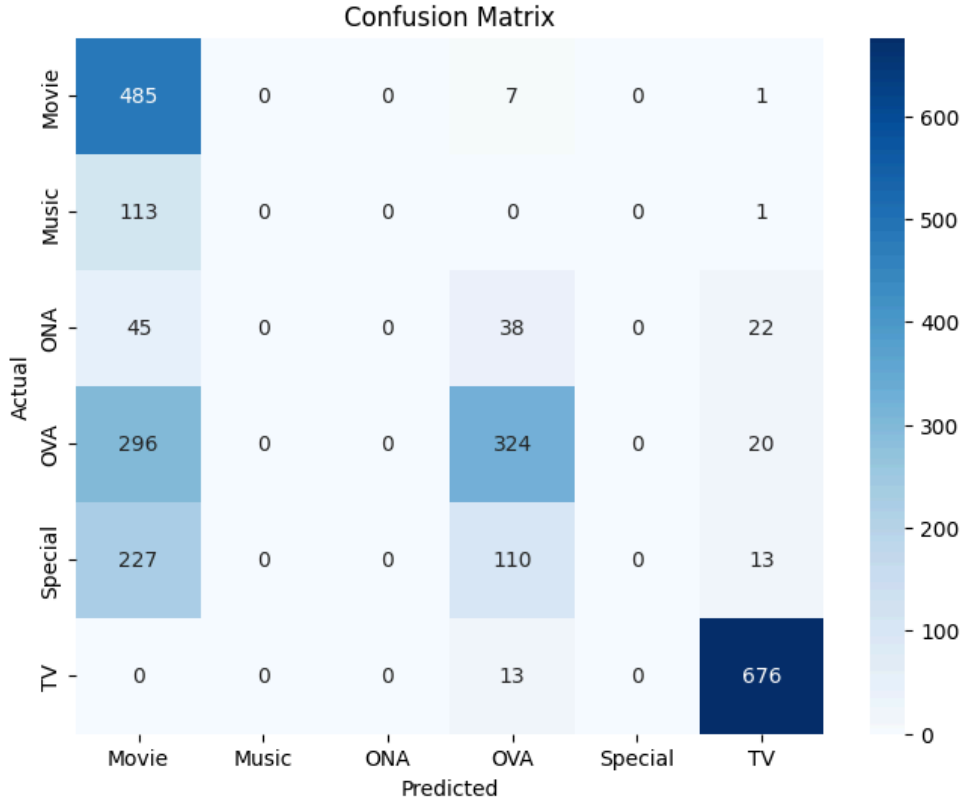
```
['anime_id', 'name', 'genre', 'type', 'episodes', 'rating', 'members']
```

```
['anime_id', 'name', 'genre', 'type', 'episodes', 'rating', 'members']
```

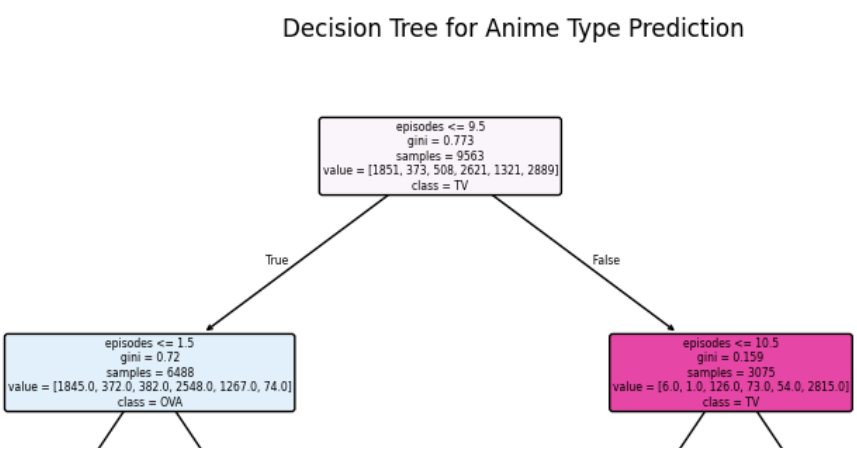
```
import pandas as pd
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split

# Use 'episodes' as feature, 'type' as target
X = df[['episodes']].copy()
y = df['type']

# Convert episodes to numeric
X['episodes'] = pd.to_numeric(X['episodes'], errors='coerce')
```

```
# Drop rows with NaNs in X or y
valid_rows = X['episodes'].notna() & y.notna()
X = X[valid_rows]
y = y[valid_rows]

# Split data
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train model
clf = DecisionTreeClassifier(max_depth=3)
clf.fit(X_train, y_train)

# Predict
predictions = clf.predict(X_test)



from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
import seaborn as sns
import matplotlib.pyplot as plt



# 1. Accuracy
print("Accuracy:", accuracy_score(y_test, predictions))

# 2. Classification Report
print("\nClassification Report:")
print(classification_report(y_test, predictions))

# 3. Confusion Matrix
conf_matrix = confusion_matrix(y_test, predictions, labels=clf.classes_)

# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt='d', xticklabels=clf.classes_, yticklabels=clf.classes_, cmap='Blues')
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix')
plt.show()
```

```
Accuracy: 0.6210790464240903

Classification Report:
              precision    recall  f1-score   support

       Movie       0.42      0.98      0.58       493
       Music       0.00      0.00      0.00       114
         ONA       0.00      0.00      0.00       105
         OVA       0.66      0.51      0.57       640
     Special       0.00      0.00      0.00       350
          TV       0.92      0.98      0.95       689

    accuracy                           0.62      2391
   macro avg       0.33      0.41      0.35      2391
weighted avg       0.53      0.62      0.55      2391

/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision i
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision i
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.11/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision i
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
```




```
from sklearn.tree import plot_tree


# Visualize the decision tree
plt.figure(figsize=(12, 8))
plot_tree(clf, feature_names=['episodes'], class_names=clf.classes_, filled=True, rounded=True)
plt.title("Decision Tree for Anime Type Prediction")
plt.show()
```


Decision Tree for Anime Type Prediction
episodes <= 9.5
gini = 0.773
samples = 9563
value = [1851, 373, 508, 2621, 1321, 2889]
class = TV
True
False
episodes <= 1.5
gini = 0.72
samples = 6488
value = [1845.0, 372.0, 382.0, 2548.0, 1267.0, 74.0]
class = OVA
episodes <= 10.5
gini = 0.159
samples = 3075
value = [6.0, 1.0, 126.0, 73.0, 54.0, 2815.0]
class = TV