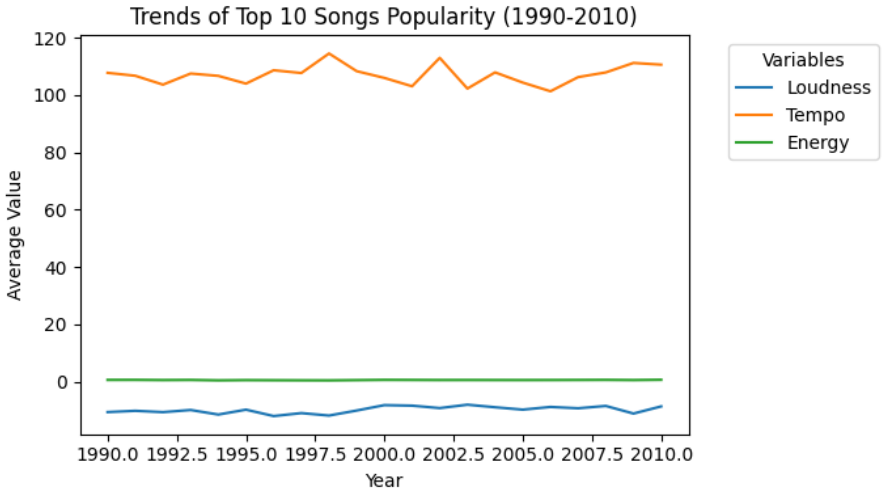
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

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

Choose files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.
Saving songs.csv to songs (1).csv

```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

# Load the CSV file into a DataFrame
df = pd.read_csv('songs.csv',encoding="latin-1")

# Filter for Top 10 songs
top_10_songs = df[df['Top10'] == 1]

# Group data by year and calculate aggregated statistics
aggregated_data = top_10_songs.groupby('year').mean().reset_index()

# # Set up the figure and axis
plt.figure(figsize=(7,4))

# # Plot trends for key variables over the years
sns.lineplot(x='year', y='loudness', data=aggregated_data, label='Loudness')
sns.lineplot(x='year', y='tempo', data=aggregated_data, label='Tempo')
sns.lineplot(x='year', y='energy', data=aggregated_data, label='Energy')

# # Customize the chart
plt.title('Trends of Top 10 Songs Popularity (1990-2010)')
plt.xlabel('Year')
plt.ylabel('Average Value')
plt.legend(title='Variables', bbox_to_anchor=(1.05, 1), loc='upper left')

# # Show the plot
plt.tight_layout()
plt.show()
```

```
<ipython-input-43-855476d63aae>:12: FutureWarning: The default value of numeric_only in
  aggregated_data = top_10_songs.groupby('year').mean().reset_index()
```



```
# Remove non-numeric variables
numeric_df = df[['timesignature', 'loudness', 'tempo', 'key', 'energy', 'pitch', 'Top10']]

# Calculate correlation matrix
correlation_matrix = numeric_df.corr()

# Extract correlation values for 'Top10' variable
top10_correlation = correlation_matrix['Top10']

# Display the correlation values
print("Correlation of 'Top10' with other variables:")
print(top10_correlation)
```

```
Correlation of 'Top10' with other variables:
timesignature    0.042368
loudness        -0.087649
tempo           -0.002545
key              0.029125
energy          -0.116992
pitch           -0.137622
Top10            1.000000
Name: Top10, dtype: float64
```

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

numeric_df = df[['timesignature', 'loudness', 'tempo', 'key', 'energy', 'pitch', 'Top10','year']]

# Split data into training (uptill year 2009) and test data (year 2010)
train_data = numeric_df[numeric_df['year'] < 2010]
test_data = numeric_df[numeric_df['year'] == 2010]

# Define features (X) and target variable (y) for training and test sets
X_train = train_data.drop('Top10', axis=1)
y_train = train_data['Top10']
X_test = test_data.drop('Top10', axis=1)
y_test = test_data['Top10']

# Train a logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)

# Test the model using the test data
y_pred = model.predict(X_test)


print("y_test:",y_test)
print("y_pred:",y_pred)
```

```
y_test: 0      0
1      0
2      0
3      0
4      0
      ..
368    0
369    0
370    0
371    0
372    1
Name: Top10, Length: 373, dtype: int64
y_pred: [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0]
```

```
# # Evaluate the model
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

# # Display results
print(f"Accuracy: {accuracy:.2f}")
print("\nConfusion Matrix:")
print(conf_matrix)
print("\nClassification Report:")
print(classification_rep)
```

```
Accuracy: 0.84

Confusion Matrix:
[[314   0]
 [ 59   0]]

Classification Report:
              precision    recall  f1-score   support

           0       0.84      1.00      0.91       314
           1       0.00      0.00      0.00        59

    accuracy                           0.84       373
   macro avg       0.42      0.50      0.46       373
weighted avg       0.71      0.84      0.77       373

/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-d
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-d
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning: Precision and F-score are ill-d
  _warn_prf(average, modifier, msg_start, len(result))
```