```
# Comprehensive Data Engineering and Pipelining for TikTok Dataset

# Install necessary libraries
!pip install pandas numpy matplotlib seaborn

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import MinMaxScaler, LabelEncoder

# Load the Dataset
FILE_PATH = '/content/tiktok.csv'  # Dataset path
df = pd.read_csv(FILE_PATH)

# 1. Define Modular Functions for Pipelining

def remove_duplicates(data):
    """Remove duplicate rows."""
    return data.drop_duplicates()

def clean_missing_values(data):
    """Fill missing values for numerical and categorical data."""
    data = data.copy()
    for col in data.select_dtypes(include=['float64', 'int64']).columns:
        data.loc[:, col] = data[col].fillna(data[col].mean())
    for col in data.select_dtypes(include=['object']).columns:
        data.loc[:, col] = data[col].fillna(data[col].mode()[0])
    return data

def normalize_numerical(data):
    """Normalize numerical columns using MinMaxScaler."""
    data = data.copy()
    scaler = MinMaxScaler()
    numerical_cols = data.select_dtypes(include=['float64', 'int64']).columns
    data.loc[:, numerical_cols] = scaler.fit_transform(data[numerical_cols])
    return data

def encode_categorical(data):
    """Encode categorical columns using LabelEncoder."""
    data = data.copy()
    for col in data.select_dtypes(include=['object']).columns:
        le = LabelEncoder()
        data.loc[:, col] = le.fit_transform(data[col])
    return data

def create_tempo_bins(data, column_name='tempo'):
    """Create bins for tempo column."""
    if column_name in data.columns:
        data = data.copy()
        data['tempo_bins'] = pd.cut(
            data[column_name],
            bins=[0, 90, 120, 150, 200],
            labels=['Slow', 'Moderate', 'Fast', 'Very Fast']
        )
    return data

# 2. Apply Pipelining
print("Applying data engineering pipeline...")
processed_df = (
    df.pipe(remove_duplicates)
      .pipe(clean_missing_values)
      .pipe(normalize_numerical)
      .pipe(encode_categorical)
      .pipe(create_tempo_bins)
)

# 3. Exploratory Data Analysis (EDA)
print("Performing EDA...")

# Exclude non-numeric columns like 'tempo_bins' for correlation
correlation_df = processed_df.select_dtypes(include=['float64', 'int64'])

# Correlation Heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_df.corr(), annot=True, cmap='coolwarm')
```

```
plt.title("Correlation Heatmap")
plt.savefig('correlation_heatmap.png')
plt.show()

# Histograms for Numerical Columns
numerical_cols = correlation_df.columns
for col in numerical_cols:
    plt.figure()
    sns.histplot(processed_df[col], bins=20, kde=True)
    plt.title(f"Distribution of {col}")
    plt.xlabel(col)
    plt.ylabel("Frequency")
    plt.savefig(f'histogram_{col}.png')
    plt.show()

# Boxplots for Numerical Columns
for col in numerical_cols:
    plt.figure()
    sns.boxplot(x=processed_df[col])
    plt.title(f"Boxplot of {col}")
    plt.savefig(f'boxplot_{col}.png')
    plt.show()

# 4. Save Processed Dataset
PROCESSED_DATA_PATH = '/content/processed_tiktok_data.csv'
processed_df.to_csv(PROCESSED_DATA_PATH, index=False)
print(f"Processed dataset saved to {PROCESSED_DATA_PATH}")

# 5. Insights
print("Generating insights...")
# Basic statistics
print(f"Dataset contains {processed_df.shape[0]} rows and {processed_df.shape[1]} columns.")
for col in numerical_cols:
    print(f"Column '{col}': Mean={processed_df[col].mean():.2f}, Median={processed_df[col].median():.2f}, Std Dev={processed_df[col].std():.2

# Correlation Analysis
strong_correlation = correlation_df.corr().unstack().sort_values(ascending=False)
strong_correlation = strong_correlation[(strong_correlation != 1.0) & (strong_correlation.abs() > 0.7)]
if not strong_correlation.empty:
    print("Strong correlations detected:")
    for index, value in strong_correlation.items():
        print(f"{index}: {value:.2f}")
else:
    print("No strong correlations found.")

print("Data engineering, pipelining, and analysis process complete.")
```

```
equirement already satisfied: pandas in /usr/local/lib/python3.10/dist-packages (2.2.2)
equirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (1.26.4)
equirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-packages (3.8.0)
equirement already satisfied: seaborn in /usr/local/lib/python3.10/dist-packages (0.13.2)
equirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.10/dist-packages (from pandas) (2.8.2)
equirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
equirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.10/dist-packages (from pandas) (2024.2)
equirement already satisfied: contourpy>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.3.1)
equirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (0.12.1)
equirement already satisfied: fonttools>=4.22.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (4.55.3)
equirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (1.4.7)
equirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (24.2)
equirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (11.0.0)
equirement already satisfied: pyparsing>=2.3.1 in /usr/local/lib/python3.10/dist-packages (from matplotlib) (3.2.0)
equirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-packages (from python-dateutil>=2.8.2->pandas) (1.17.0)
oplying data engineering pipeline...
erforming EDA...
ipython-input-6-a77997da71e7>:36: FutureWarning: Setting an item of incompatible dtype is deprecated and will raise in a future error
 data.loc[:, numerical_cols] = scaler.fit_transform(data[numerical_cols])
ipython-input-6-a77997da71e7>:36: FutureWarning: Setting an item of incompatible dtype is deprecated and will raise in a future error
 data.loc[:, numerical_cols] = scaler.fit_transform(data[numerical_cols])
ipython-input-6-a77997da71e7>:36: FutureWarning: Setting an item of incompatible dtype is deprecated and will raise in a future error
 data.loc[:, numerical_cols] = scaler.fit_transform(data[numerical_cols])
```