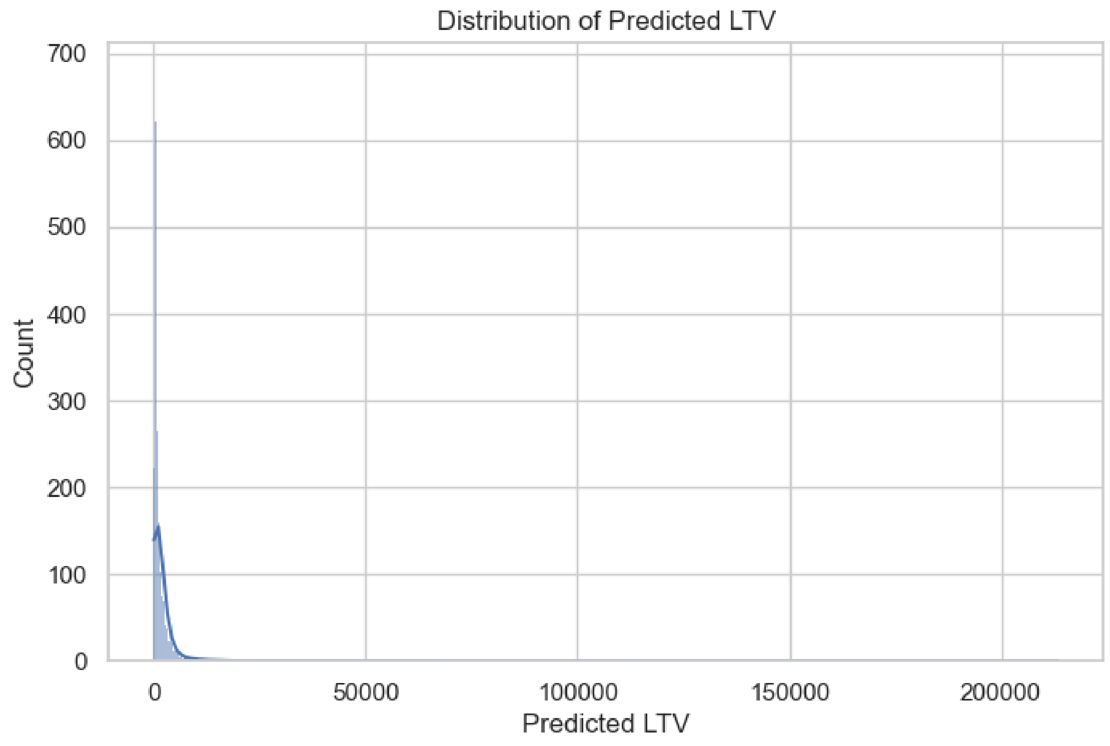
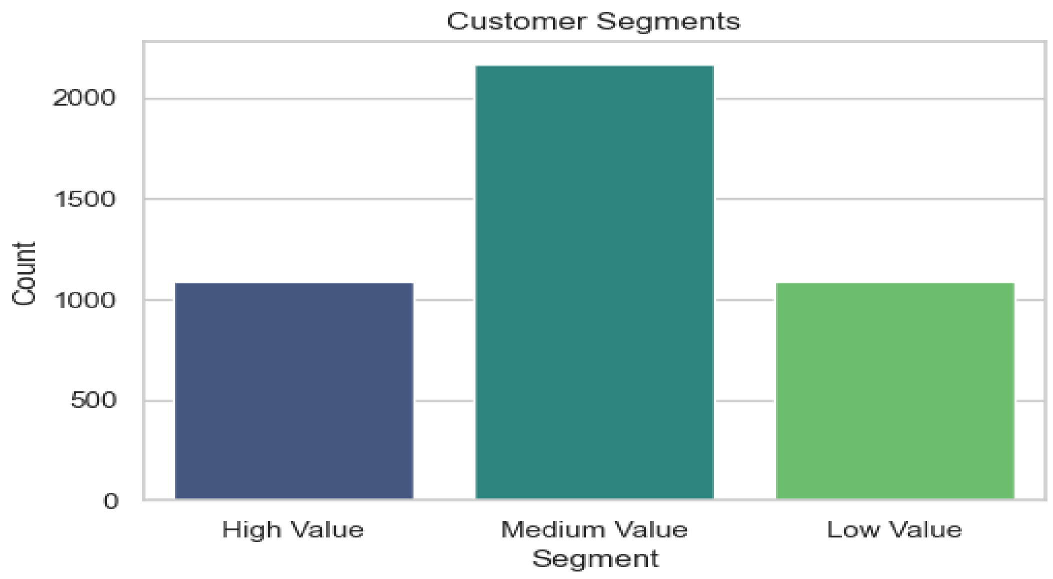
```python
# Step 1: Import Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from datetime import datetime

from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Step 2: Load Dataset
# Use correct encoding to avoid UnicodeDecodeError
df = pd.read_csv(
    r"C:\Users\Balambiga2910\Desktop\data anal intern\ecommerce data.csv",
    encoding='ISO-8859-1'
)

# Step 3: Preprocessing
# Remove rows with missing CustomerID
df = df.dropna(subset=['CustomerID'])

# Remove rows with negative Quantity (returns)
df = df[df['Quantity'] > 0]

# Create 'Amount' column
df['Amount'] = df['Quantity'] * df['UnitPrice']

# Rename for consistency
df = df.rename(columns={'InvoiceNo': 'OrderID', 'InvoiceDate': 'OrderDate'})

# Convert date column
df['OrderDate'] = pd.to_datetime(df['OrderDate'])

# Step 4: Feature Engineering
snapshot_date = df['OrderDate'].max() + pd.Timedelta(days=1)

customer_data = df.groupby('CustomerID').agg({
    'OrderDate': [lambda x: (snapshot_date - x.max()).days, 'count'],
    'Amount': ['mean', 'sum']
})

customer_data.columns = ['Recency', 'Frequency', 'AOV', 'Revenue']
customer_data = customer_data.reset_index()

# Step 5: Feature Selection
X = customer_data[['Recency', 'Frequency', 'AOV']]
y = customer_data['Revenue']
```

```
# Step 6: Train-Test Split
X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.2, random_state=42
)

# Step 7: Train the Model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Step 8: Predictions & Evaluation
y_pred = model.predict(X_test)
mae = mean_absolute_error(y_test, y_pred)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))

print(f"MAE: {mae:.2f}")
print(f"RMSE: {rmse:.2f}")

# Step 9: Predict LTV for All Customers
customer_data['Predicted_LTV'] = model.predict(X)

# Step 10: Segment Customers
def segment_customer(ltv):
    if ltv >= customer_data['Predicted_LTV'].quantile(0.75):
        return 'High Value'
    elif ltv >= customer_data['Predicted_LTV'].quantile(0.25):
        return 'Medium Value'
    else:
        return 'Low Value'

customer_data['Segment'] =
customer_data['Predicted_LTV'].apply(segment_customer)

# Step 11: Save Outputs
customer_data.to_csv("final_ltv_predictions.csv", index=False)

# Step 12: Visualization
sns.set(style="whitegrid")

plt.figure(figsize=(8, 5))
sns.histplot(customer_data['Predicted_LTV'], kde=True)
plt.title("Distribution of Predicted LTV")
plt.xlabel("Predicted LTV")
plt.show()

plt.figure(figsize=(6, 4))
sns.countplot(x='Segment', data=customer_data, palette='viridis')
plt.title("Customer Segments")
plt.xlabel("Segment")
plt.ylabel("Count")
plt.show()
```

```
MAE: 1131.54
RMSE: 5915.53
```


Distribution of Predicted LTV
Count
Predicted LTV


Customer Segments
Count
Segment
High Value
Medium Value
Low Value